ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Optimal Keyword Cover Search

P.Anuradha
M.Tech(SE) Student
Department of IT
Geethanjali College of Engineering & Technology.

K.Srinivas
Professor
Department of CSE
Geethanjali College of Engineering & Technology.

ABSTRACT

It is common that the objects in a spatial database (e.g., restaurants/hotels) are associated with keyword(s) to indicate their businesses/services/features. An interesting problem known as Closest Keywords search is to query objects, called keyword cover, which together cover a set of query keywords and have the minimum inter-objects distance. In recent years, we observe the increasing availability and importance of keyword rating in object evaluation for the better decision making. This motivates us to investigate a generic version of Closest Keywords search called Best Keyword Cover which considers inter-objects distance as well as the keyword rating of objects. The baseline algorithm is inspired by the methods of Closest Keywords search which is based on exhaustively combining objects from different query keywords to generate candidate keyword covers. When the number of query keywords increases, the performance of the baseline algorithm drops dramatically as a result of massive candidate keyword covers generated. To attack this drawback, this work proposes a much more scalable algorithm called keyword nearest neighbor expansion (keyword-NNE). Compared to the baseline algorithm, keyword-NNE algorithm significantly reduces the number of candidate keyword covers generated. The in-depth analysis and extensive experiments on real data sets have justified the superiority of our keyword-NNE algorithm.

INTRODUCTION

Now a days, use of mobile computing increases. Inspired by the mobile computing, the spatial keywords search problem has attracted much attention recently because of location-based services and wide availability of extensive digital maps and satellite imagery. So the number of users using the location based services has been also increased to large extend. Spatial objects indicates the information such as its business/services/features which are associated to keyword In spatial database, each tuple represents a spatial object. The main idea behind the spatial keywords search is to identify spatial object(s) which are associated with keywords relevant to a set of query keywords which are close to each other and/or close to the query location. This problem has unique value in various Applications because users‟‟ requirements are often expressed as multiple keywords. In existing, spatial keyword search problem have been studied because of the value of the special keyword search in practice. This paper investigates a generic version of mock query, called Best Keyword Cover (BKC) query, which considers inter-objects distance as well as keyword rating. It is motivated by the observation of increasing availability and importance of keyword rating in decision making. Millions of businesses/services/features around the world have been rated by customers through online business review sites such as Yelp, City search, ZAGAT and Dining, etc. For example, a restaurant is rated 65 out of 100 (ZAGAT.com) and a hotel is rated 3.9 out of 5 (hotels.com). According to a survey in 2013 conducted by Dimensional Research (dimensionalresearch.com), an overwhelming 90 percent of respondents claimed that buying decisions are influenced by online business review/rating. Due to the consideration of keyword rating, the solution of BKC query can be very different from that of mCK query).

This work develops two BKC query processing algorithms, baseline and keyword-NNE. The baseline algorithm is inspired by the mCK query processing methods Both the baseline algorithm and keyword-NNE algorithm are supported by indexing the objects with an R*-tree like index, called KRR*-tree. In the baseline algorithm, the idea is to combine nodes in higher hierarchical levels of KRR*-trees to generate candidate keyword covers. Then, the most promising candidate is assessed in priority by combining their child nodes to generate new candidates. Even though BKC query can be effectively resolved, when the number of query keywords increases, the performance drops dramatically as a result of massive candidate keyword covers generated. To overcome this critical drawback, we developed much scalable keyword nearest neighbour expansion (keyword-NNE) algorithm which applies a different strategy. Keyword-NNE selects one query keyword as principal query keyword. The objects associated with the principal query keyword are principal objects. For each principal object, the local best solution (known as local best keyword cover (lbkc)) is computed. Among them, the lbkc with the highest evaluation is the solution of BKC query. Given a principal object, its lbkc can be identified by simply retrieving a few nearby and highly rated objects in each non-principal query keyword (two-four objects in average as illustrated in experiments). Compared to the baseline algorithm, the number of candidate keyword covers generated in keyword-NNE algorithm is significantly reduced. The in-depth analysis reveals that the number of candidate keyword covers further processed in keyword-NNE algorithm is optimal, and each keyword candidate cover processing generates much less new candidate keyword covers than that in the baseline algorithm.

Baseline Algorithm:

The baseline algorithm is inspired by the mCK query processing methods. For mCK query processing, the method in browses index in top-down manner while the method in 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 over the method in 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 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. 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 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.

$$O.score = score(A, B).$$
$$A = \max_{N_i, N_j \in O} dist(N_i, N_j)$$
$$B = \min_{N \in O}(N.maxrating),$$

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 O', O' consists of objects $\{o_{k1}, \ldots, o_{kn}\}$ and O consists of nodes $\{N_{k1}, \ldots, N_{kn}\}$. If o_{ki} is under N_{ki} in KRR$^*_{ki}$-tree for $1 \leq i \leq n$, it is true that $O'.score \leq O.score$.*

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

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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 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 $bkc$;
```

```
Algorithm 2. $Depth\_First\_Tree\_Browsing\ (H, T, can, bkc)$
```

EXISTING SYSTEM:

- 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.
- The approaches proposed by Cong et al. and Li et al. employ a hybrid index that augments nodes in non-leaf nodes of an R/R*-tree with inverted indexes.
- 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.

DISADVANTAGES OF EXISTING SYSTEM:

- When the number of query keywords increases, the performance drops dramatically as a result of massive candidate keyword covers generated.
- 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.

PROPOSED SYSTEM:

- This paper investigates a generic version of mCK query, called Best Keyword Cover (BKC) query, which considers inter-objects distance as well as keyword rating. It is motivated by the observation of increasing availability and importance of keyword rating in decision making. Millions of businesses/services/features around the world have been rated by customers through online business review sites such as Yelp, Citysearch, ZAGAT and Dianping, etc.
- This work develops two BKC query processing algorithms, baseline and keyword-NNE. The baseline algorithm is inspired by the mCK query processing methods. Both the baseline algorithm and keyword-NNE algorithm are supported by indexing the objects with an R*-tree like index, called KRR*-tree.

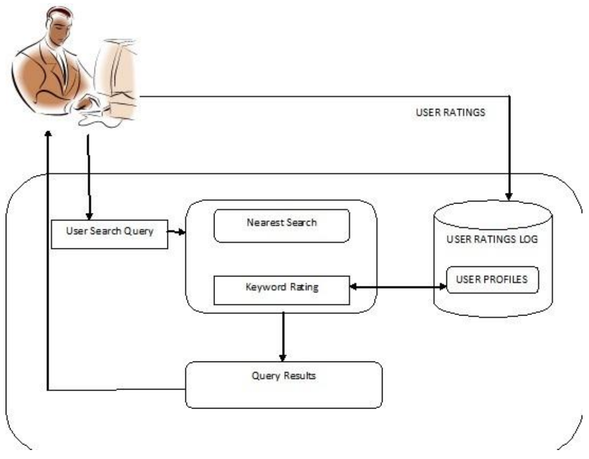
ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

- We developed much scalable keyword nearest neighbor expansion (keyword-NNE) algorithm which applies a different strategy. Keyword-NNE selects one query keyword as principal query keyword. The objects associated with the principal query keyword are principal objects. For each principal object, the local best solution (known as local best keyword cover lbkc) is computed. Among them, the lbkc with the highest evaluation is the solution of BKC query. Given a principal object, its lbkc can be identified by simply retrieving a few nearby and highly rated objects in each non-principal query keyword (two-four objects in average as illustrated in experiments).

ADVANTAGES OF PROPOSED SYSTEM:

- Compared to the baseline algorithm, the number of candidate keyword covers generated in keyword-NNE algorithm is significantly reduced. The in-depth analysis reveals that the number of candidate keyword covers further processed in keyword-NNE algorithm is optimal, and each keyword candidate cover processing generates much less new candidate keyword covers than that in the baseline algorithm.
- The proposed keyword-NNE algorithm applies a different processing strategy, i.e., searching local best solution for each object in a certain query keyword. As a consequence, the number of candidate keyword covers generated is significantly reduced.
- The analysis reveals that the number of candidate keyword covers which need to be further processed inkeyword-NNE algorithm is optimal and processing each keyword candidate cover typically generates much less new candidate keyword covers in keyword-NNE algorithm than in the baseline algorithm.

SYSTEM ARCHITECTURE:



IMPLEMENTATION MODULES:

1. Indexing Keyword Ratings
2. Keyword nearest Neighbor Expansion
3. LBKC Computation
4. Weighted Average of Keyword Ratings

MODULESDESCSRIPTION:

Indexing Keyword Ratings

A single tree structure is used to index objects of different keywords. The single tree can be extended with an additional dimension to index keyword rating. A single tree structure suits the situation that most keywords are query keywords. For the above mentioned example, all keywords, i.e., “hotel”, “restaurant” and “bar”, are query keywords. However, it is more frequent that only a small fraction of keywords are query keywords. For example in the experiments, only less than 5 percent keywords are query keywords. In this situation, a single tree is poor to approximate the spatial relationship between objects of few specific keywords. Therefore, multiple KRR*-trees are used in this work, each for one keyword.1 The KRR*-tree for keyword ki is denoted as KRR*ki-tree. Given an object, the rating of an associated keyword is typically the mean of ratings given by a number of customers for a period of time. The change does happen but slowly. Even though dramatic change occurs, the KRR*-tree is updated in the standard way of R*-tree update.

Keyword nearest Neighbor Expansion

Using the baseline algorithm, BKC query can be effectively resolved. However, it is based on exhaustively combining objects (or their MBRs). Even though pruning techniques have been explored, it has been observed that the performance drops dramatically, when the number of query keywords increases, because of the fast increase of candidate keyword covers generated. This motivates us to develop a different algorithm called keyword nearest neighbor expansion. We focus on a particular query keyword, called principal query keyword. The objects associated with the principal query keyword are called principal objects.

The goal of the interface is to provide point of interest information (static and dynamic ones) with, at least, a location, some mandatory's attributes and optional details (description,…). In order to provide that information, the component that implements the interface uses the map database information to locate and display point of interest (POI) or to select POI as route waypoint and favorite. This component not only provides search functionalities for the local database but also a way to connect external search engine to this component and enhance the search criteria and the list of results It also proposes a solution to get custom POIs (not part of the local map database) or to dynamically update content and description of local POI. This is achieved by specifying and providing interfaces to:

- Select POIs from one of their attributes (e.g., Category, Name,…)
- Retrieve POI attributes (e.g., Location and Description)
- Get dynamic content for a given POI.
- Add custom POI to the map display
- Import new POIs and POIs categories from local file.

LBKC Computation

Given a spatial database, each object may be associated with one or multiple keywords. Without loss of generality, the object with multiple keywords are transformed to multiple objects located at the same location, each with a distinct single keyword.When further processing a candidate keyword cover, keyword-NNE algorithm typically generates much less new candidate keyword covers compared to BF-baseline algorithm. Since the number of candidate keyword covers further processed in keyword-NNE algorithm is optimal the number of keyword covers generated in BF-baseline algorithm is much more than that in keyword- NNE algorithm. In turn, we conclude that the number of keyword covers generated in baseline algorithm is much more than that in keyword-NNE algorithm. This conclusion is independent of the principal query keyword since the analysis does not apply any constraint on the selection strategy of principal query keyword.

Weighted Average of Keyword Ratings

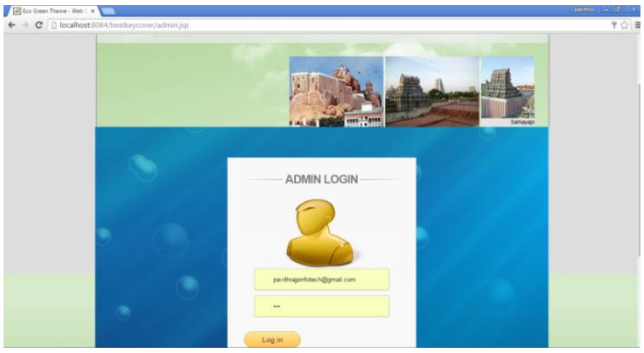
In keyword-NNE algorithm, the best-first browsing strategy is applied like BF-baseline but large memory requirement is avoided. For the better explanation, we can imagine all candidate keyword covers generated in BF-baseline algorithm are grouped into independent groups. Each group is associated with one principal node (or object). That is, the candidate keyword covers fall in the same group if they have the same principal node (or object).

When further processing a candidate keyword cover, keyword-NNE algorithm typically generates much less new candidate keyword covers compared to BF-baseline algorithm. Since the number of candidate keyword covers further processed in keyword-NNE algorithm is optimal, the number of keyword covers generated in BF-baseline algorithm is much more than that in keyword-NNE algorithm. In turn, we conclude that the number of keyword covers generated in baseline algorithm is much more than that in keyword-NNE algorithm. This conclusion is independent of the principal query keyword since the analysis does not apply any constraint on the selection strategy of principal query keyword.

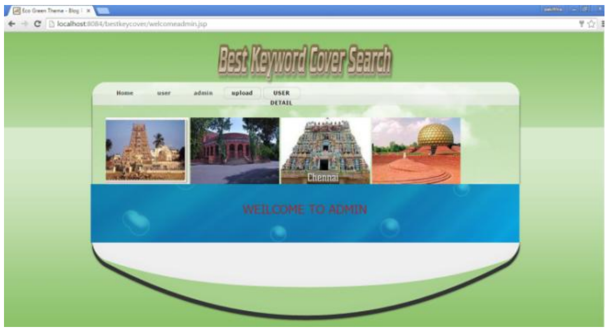
ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

SCREEN SHOTS

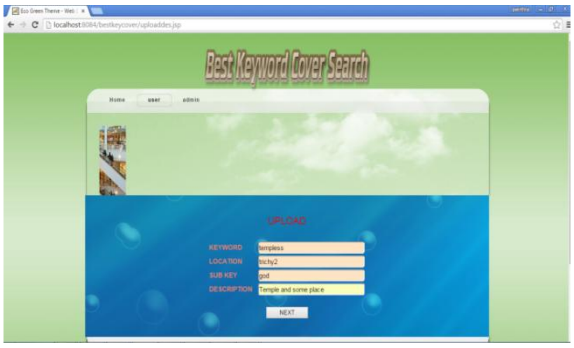
Admin login:



2.Admin Home:

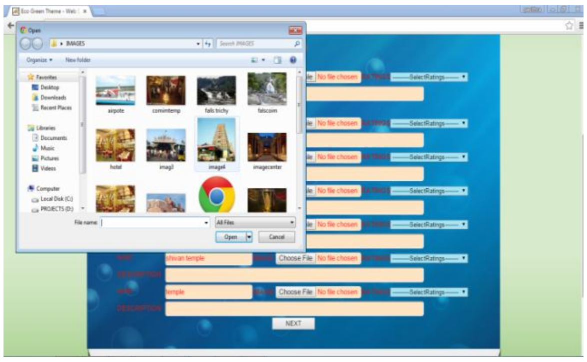


UserDetails:

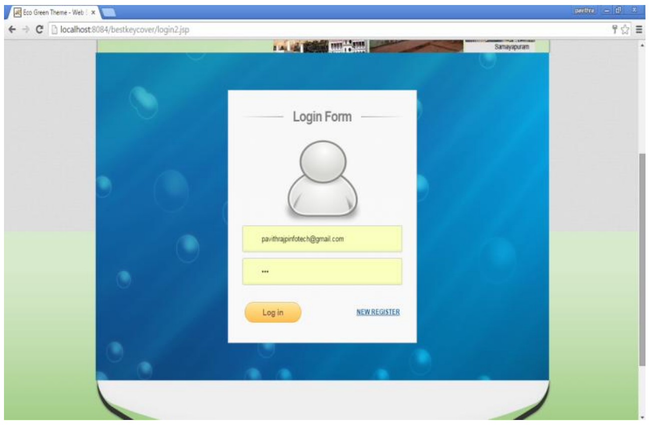


KEYWORD: templess
LOCATION: trichy2
SUB KEY: god
DESCRIPTION: Temple and some place

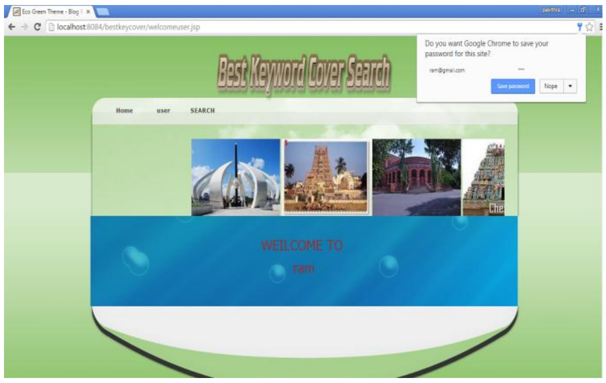
Upload Images:



User Login:



UserHome:



CONCLUSION

Compared to the most relevant mCK query, BKC query provides an additional dimension to support more sensible decision making. The introduced baseline algorithm is inspired by the methods for processing mCK query.

The baseline algorithm generates a large number of candidate keyword covers which leads to dramatic performance drop when more query keywords are given. The proposed keyword- NNE algorithm applies a different processing strategy, i.e., searching local best solution for each object in a certain query keyword. As a consequence, the number of candidate keyword covers generated is significantly reduced. The analysis reveals that the number of candidate keyword covers which need to be further processed in.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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. 18th ACM 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.

[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. Collaborative computing,”INFOCOM 2008, pp. 1211-1219.