

# SEARCHING BEST LOCATION TECHNIQUE USING MACHINE ALGORITHM

Mrs. Amruta Harshal Salvi

Research Scholar, Department of Computer Science and Information Technology

S.P.Hegshetye College of Arts Commerce and Science Ratnagiri

**Abstract:**-The objects in a spatial database (e.g., restaurants or hotels) are related with keyword(s) to show their businesses or services. A novel issue known as Closest Keywords search is to query objects, called keyword cover, which have an arrangement of query keywords and have the base between objects remove. Presently a days, observe expanding accessibility and significance of keyword rating in object evaluation for the better decision making. This is inspiring us to examine a nonexclusive form of Closest Keywords search called Best Keyword Cover which considers inter-objects distance and the keyword rating of objects. The standards-based algorithm is the most similar keyword detection method based on many elements included in various keyword queries to create candidate keyword coverage. As the number of keywords in the query increases, many keywords are created, which degrades the performance of the base algorithm. This unfavorable solution suggests another important algorithm for the keyword nearest neighbor expansion (keyword-NNE). Comparatively the baseline algorithm, keyword-NNE algorithm significantly reduces the number of candidate keyword covers generated. This is in-depth analysis and extensive experiments on real data sets have

justified the superiority of our keyword-NNE algorithm. The contribution work in this project is the Point of interest (POI) recommendation is to provide personalized recommendations of places, such as restaurants and hotels.

**Keywords:** Best Keyword Cover, Keywords, Keyword Cover, Keyword Rating, Point Of Interests, Spatial Database

## I. INTRODUCTION

An expanding number of applications require the proficient execution of nearest neighbor (NN) queries constrained by the properties of the spatial objects. Due to the fame of keyword search, particularly on the Internet, many of these applications allow the user to provide a list of keywords that the spatial objects (henceforth referred to simply as objects) should contain, in their description or other attribute. For example, online yellow pages allow users to specify an address and a set of keywords, and return businesses whose description contains these keywords, ordered by their distance to the specified address location. As another example, real estate web sites allow users to search for properties with specific keywords in their description and rank them according to their distance from a specified location.

We call such queries spatial keyword queries. A spatial keyword query consists of a query area and a set of keywords. The answer is a list of objects ranked according to a combination of their distance to the query area and the relevance of their text description to the query keywords. A simple yet popular variant, which is used in our running example, is the distance-first spatial keyword query, where objects are ranked by distance and keywords are applied as a conjunctive filter to eliminate objects that do not contain them. Which is our running example, displays a dataset of fictitious hotels with their spatial coordinates and a set of descriptive attributes (name, amenities)? An example of a spatial keyword query is “find the nearest hotels to point that contain keywords internet and pool”. The top result of this query is the hotel object. Unfortunately there is no efficient support for top-k spatial keyword queries, where a prefix of the results list is required. Rather, momentum frameworks utilize specially appointed mixes of nearest neighbor (NN) and keyword search methods to handle the issue.. For example, an R-Tree is used to discover the nearest neighbors and for each neighbor an inverted index is used to check if the query keywords are contained. We demonstrate that such two-phase approaches are inefficient.

Goals:

The main goal is to rank the methods, so the system will report on the binary comparisons that allowed us to determine the ordering of the four methods (excluding redundant comparisons). The current goals are to allow explicit queries, and to rank document results with the objective of maximizing the coverage of all the in the spatial database, while minimizing redundancy in a short list of the best keyword search.

When the number of query keywords increases, the performance of the baseline algorithm drops dramatically as a result of massive candidate keyword covers generated. The Point of interest (POI) recommendation is to provide personalized recommendations of places, such as restaurants and hotels.

Objectives:

1. The keyword closest to the keyword cover search to the best keyword and related to the spatial database on the web application considering security parameters and maintaining integrity.
2. Search result should be ranked by the best keyword and closest cover keyword according to some criteria.
3. To reduce the communication cost.

## II. LITERATURE SURVEY

**IRTree: An Efficient Index For Geographic Document Search:** Given a geographic query that is made of query keywords and a location, a geographic search engine retrieves documents that are the most textually and spatially important to the query keywords and the area, respectively, and ranks the retrieved documents according to their joint textual and spatial relevance's to the query. The absence of an efficient index that can concurrently handle both the textual and spatial conditions of the documents makes existing geographic search engines inefficient in answering geographic queries. In [1] paper, we propose an efficient index, called IR-tree, that together with a top-k document search algorithm facilitates four major tasks in document searches, namely, 1) spatial filtering, 2) textual filtering, 3) relevance computation, and 4) document ranking in a completely coordinated way. Also, IR-tree allows

searches to adopt different weights on textual and spatial relevance of documents at the runtime and thus caters for a wide variety of applications. A set of comprehensive experiments over an extensive variety of situations has been led and the investigation results demonstrate that IR-tree beats the cutting edge approaches for geographic report looks.

**Retrieving Top-K Prestige-Based Relevant Spatial Web Objects:** The location-aware keyword query returns ranked objects that are near a query location and that have textual descriptions that match query keywords. This query occurs inherently in many types of mobile and traditional web services and applications, e.g., Yellow Pages and Maps services. Previous work considers the potential results of such a query as being independent when ranking them. However, a relevant result object with nearby objects that are also relevant to the query is likely to be preferable over a relevant object without relevant nearby objects. The paper [2] proposes the concept of prestige-based significance to capture both the textual importance of an object to a query and the impacts of adjacent items. Based on this, a new type of query, the Location-aware top-k Prestige-based Text retrieval (LkPT) query, is proposed that retrieves the top-k spatial web objects ranked according to both prestige-based relevance and location proximity. We propose two algorithms that compute LkPT queries. Exact investigations with certifiable spatial information exhibit that LkPT queries are more effective in retrieving web objects than a previous approach that does not consider the effects of nearby objects; and they show that the proposed algorithms are scalable and outperform baseline approach significantly.

**Efficient Retrieval of The Top-K Most Relevant Spatial Web Objects:** The regular Internet is getting a geo-spatial measurement. Web documents are being geo-tagged, and geo-referenced objects such as points of interest are being illustrated with descriptive text documents. The resulting fusion of geo-location and documents enables a new kind of top-k query that takes into account both location proximity and text relevancy. To our knowledge, only naive techniques exist that is capable of computing a general web information retrieval query while also taking location into account. This [3] paper proposes a new indexing framework for location aware top-k text retrieval. The framework leverages the inverted file for text retrieval and the R-tree for spatial proximity querying. Several indexing approaches are explored within the framework. The framework encompasses algorithms that utilize the proposed indexes for computing the top-k query, thus taking into account both text relevancy and location proximity to prune the search space. Results of empirical studies with an implementation of the framework demonstrate that the paper's proposal offers scalability and is capable of excellent performance.

**Location-Aware Type Ahead Search on Spatial Databases: Semantics And Efficiency:** Users often search spatial databases like yellow page data using keywords to and businesses near their current location. Such searches are increasingly being performed from mobile devices. Typing the entire query is cumbersome and prone to errors, especially from mobile phones. We address this problem by introducing type-ahead search functionality on spatial databases. Like keyword search on spatial data, type-ahead search needs to be location-aware, i.e., with

every letter being typed, it needs to return spatial objects whose names (or descriptions) are valid completions of the query string typed so far, and which rank highest in terms of proximity to the user's location and other static scores. Existing solutions for type-ahead search cannot be used directly as they are not location-aware. We show that a straight-forward combination of existing techniques for performing type-ahead search with those for performing proximity search perform poorly. This paper [4] proposes a formal model for query processing cost and develops novel techniques that optimize that cost. Our empirical evaluations on real and synthetic datasets demonstrate the effectiveness of our techniques. To the best of our knowledge, this is the first work on location-aware type-ahead search.

### **Locating Mapped Resources in Web 2.0:**

Mapping mashups are emerging Web 2.0 applications in which data objects such as blogs, photos and videos from different sources are combined and marked in a map using APIs that are released by online mapping solutions such as Google and Yahoo Maps. These objects are typically associated with a set of tags capturing the embedded semantic and a set of coordinates indicating their geographical locations. Traditional web resource searching strategies are not effective in such an environment due to the lack of the gazetteer context in the tags. Instead, a better alternative approach is to locate an object by tag matching. However, the number of tags associated with each object is typically small, making it difficult for an object to capture the complete semantics in the query objects. In [5] paper, we focus on the fundamental application of locating geographical resources and propose an efficient tag centric query

processing strategy. In particular, we aim to find a set of nearest co-located objects which together match the query tags. Given the fact that there could be large number of data objects and tags, we develop an efficient search algorithm that can scale up in terms of the number of objects and tags. Further, to ensure that the results are relevant, we also propose a geographical context sensitive geo-t f-id f ranking mechanism. Our experiments on synthetic data sets demonstrate its scalability while the experiments using the real life data set confirm its practicality.

The 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 similarity between documents is 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.

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.

For example, a tourist who plans to visit a city may have particular shopping, dining and accommodation needs. It is desirable that all these needs can be satisfied without long distance traveling. Due to the remarkable value in practice, several variants of spatial keyword search problem have been studied. The works aim to find a number of individual objects, each of which is close to a query location and the associated keywords (or called document) are very relevant to a set of query keywords (or called query document).

Disadvantages of existing system :

1. The problem has unique value in various applications because users' requirements are often expressed as multiple keywords. For example, a tourist who plans to visit a city may have particular shopping, dining and accommodation needs. It is desirable that all these needs can be satisfied without long distance traveling.
2. 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.
3. The problem in searching result which does not have a set of query keywords and the minimum inter-objects distance.

### III. SYSTEM OVERVIEW

To overcome the critical drawback, 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 (2-4 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.

Advantages of proposed system:

- I. 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.
- II. 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.
- III. The analysis reveals that the number of candidate keyword covers which need to be further

processed in keyword-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.

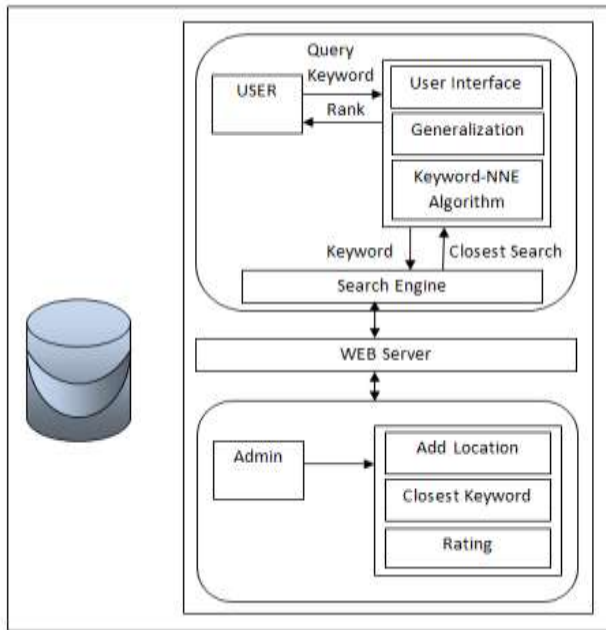


Fig.1. Block Diagram of Proposed System

The Fig. 1 illustrates idea about system architecture. The query consisting of a query location and a set of query keywords. Each recovered item is connected with keywords important to the query keywords and is near the query area. The comparability between reports is connected to quantify the importance between two arrangements of keywords. Since it is likely no individual article is connected with all query keywords, some different works mean to recover various items which together cover all query keywords. System helps to find major issues like: 1) cover all query keywords, 2) have minimum inter-objects distance and 3) are close to a query location.

The objective of the interface is to give point of interest data (static and element ones) with, no less

than, an area, some compulsory's qualities and open slight elements (description).In request to give those data, the segment that executes the interface utilizes the guide database data to find and show point of interest (POI) or to choose a POI as course waypoint and top pick.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.

#### IV. MATHEMATICAL MODEL

##### 1) BaseLine algorithm:

**Input:** user given query keywords, the root node of (kRR\* trees root)

**Output :** Best Keyword cover(bkc)

//Step1 to 4 generate candidate function

**Step1:** Calculate distance between child nodes by using Euclidean distance algorithm

**Step2:** After, count the minimum records which given max\_rating to the child nodes.

**Step3:** Combine child\_nodes distance and records which generate keyword covers and store it into heap.

**Step4:** The candidate with the highest score in heap is selected and its child nodes are combined using Generate Candidate function to generate more candidates.

// Step 5 to 15 depth-first tree browsing function

**Step5:** After check **while** the heap is not empty **do**

**Step6:** Select the highest score candidate and pass to the depth-first tree browsing function

**Step7:** **if** candidate is the leaf node **then**

**Step8:** Obj S = objects in candidate

**Step9:** bkc'=the keyword cover with highest score given in S.

**Step10:** **if** bkc.score<bkc'.score **then**

**Step11:**  $bkc = bkc'$   
**Step12:** else  
**Step13:** new\_can = Generate Candidate function  
**Step14:** candidate = new\_can // Replace in heap  
**Step15:** Again Select the highest score candidate  
 AND pass to the depth- first tree browsing  
**Step16:** foreach candidate in heap do  
**Step17:** if candidate.score  $\leq$  bkc.score then  
**Step18:** remove candidate from heap  
**Step19:** return bkc;

## 2) Keyword-NNE Algorithm

**Input:** A set of query keywords T, a spatial database D

**Output:** Best Keyword Cover

**Step1:**  $bkc.score = 0$ ;  
**Step2:**  $k =$  select the principal query keyword from T;  
**Step3:** H = child nodes of the root in KRR\*k-tree;  
**Step4:** foreach node in KRR\*k-tree  $N_k \in H$  do  
**Step5:** Compute local best keyword cover score,  $lbkc_{N_k.score}$ ;  
**Step6:**  $H.head = N_k \in H$  with  $\max_{N_k \in H} lbkc_{N_k.score}$ ;  
**Step7:** while  $H \neq \emptyset$  ; do  
**Step8:** while H.head is a node in KRR\*k-tree do  
**Step9:** N = child nodes of H.head;  
**Step10:** foreach  $N_k$  in N do  
**Step11:** Compute  $lbkc_{N_k.score}$   
**Step12:** Insert  $N_k$  into H;  
**Step13:** Remove H.head from H;  
**Step14:**  $H.head = N_k \in H$   
**Step15:**  $ok = H.head$ ;  
**Step16:** Compute  $lbkc_{ok.score}$  ;  
**Step17:** if  $bkc.score < lbkc_{ok.score}$  then  
**Step18:**  $bkc = lbkc_{ok}$ ;  
**Step19:** foreach  $N_k$  in H do

**Step20:** if  $lbkc_{N_k.score} \leq bkc.score$  then

**Step21:** Remove  $N_k$  from H;

**Step22:** return bkc;

## V. SYSTEM ANALYSIS

The number of query keywords m has significant impact to query processing efficiency. In this test, m is changed from 2 to 9 when  $\alpha = 0.4$ . Each BKC query is generated by randomly selecting m keyword from all keywords as the query keywords. For each setting, we generate and perform 100 BKC queries, and the averaged results are reported. Fig. 2 shows the number of candidate keyword covers generated for BKC query processing. When m increases, the number of candidate keyword covers generated increases dramatically in the baseline algorithm. In contrast, keyword-NNE algorithm shows much better scalability.

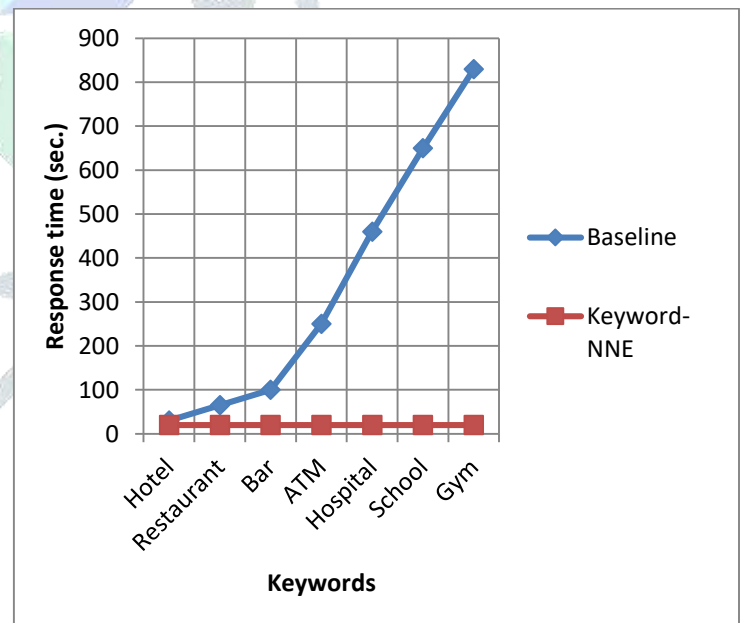


Fig.2 Response time versus m ( $\alpha = 0.4$ )

Table I Response time versus m ( $\alpha = 0.4$ )

Keywords	Baseline	Keyword-NNE
Hotel	30	20
Restaurant	65	20
Bar	100	20
ATM	250	20
Hospital	460	20
School	650	20
Gym	830	20

## VI. CONCLUSION

Differentiated to the mCK(object closest keyword) query, BKC query provides more sensible decision making. The introduced baseline algorithm follows by the methods for processing mCK query. The baseline algorithm requires for generation of candidate keyword. The proposed keyword-NNE algorithm used for searching purpose. It helps for different processing strategy, i.e., searching local best solution for each object in a certain query keyword. It ranks the result of user query on the basis of thee evidence that is rating, review, rank. So it will help to retrieve exact or relevant search for user query. In this project, the Point of interest (POI) recommendation is to provide personalized recommendations of places, such as restaurants and hotels to the users. As future work, a keyword cover of keyword that is the word related to that keyword, and cover keyword is called to be the best keyword for the search finds valuable search and ranking, without interrupting the conversation flow, thus ensuring the usability of our system. In the future, this will be tested with human users of the system within real-life meetings.

## REFERENCES:-

- [1] 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.
- [2] 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.
- [3] 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.
- [4] S. B. Roy and K. Chakrabarti, "Location-aware type ahead search on spatial databases: emantics and efficiency," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2011, pp. 361–37.
- [5] 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.