Quality based Ordering Approach for Spatial Data

Linu Paulose
ACM Student member
Department of CSE, VJCET
Ernakulam, Kerala, India

Sindhu Jose
Department of CSE, VJCET
Ernakulam, Kerala, India

ABSTRACT

Object ranking is a popular retrieval task that is concerned with the ranking of objects in context of a given user query. In relational databases, tuples are ranked using an aggregate score function on their attribute values. Spatial databases manage large collections of geographic entities, where ranking is often associated to nearest neighbor (NN) retrieval. A spatial preference query ranks objects based on the quality of features in their spatial neighborhood. For example, using a real estate agency database of flats for lease, a customer may want to rank the flats with respect to the appropriateness of their location, defined after aggregating the qualities of other features (e.g., restaurants, hospital etc.) within their spatial neighborhood. A neighborhood concept can be specified by the user via different functions. It can be an explicit circular region within a given distance from the object (range score function) or a region obtained by assigning higher weights to the features based on their proximity to the object (influence score function). A wide range of location based applications rely upon spatial preference queries. One of the existing strategies for processing the spatial preference query is brute force, which is not quite adaptable since it is computationally inefficient and it is worthy only for small data inputs. In the proposed system, indexing techniques and query processing algorithms are presented for efficient processing of the top-k spatial preference queries.

General Terms

Spatial database, Query Processing, Ranking

Keywords

Spatial Data, Quality, Features

1. INTRODUCTION

Spatial databases store data that is related to objects in space. In addition to spatial data it may contain non-spatial information. A top-k spatial preference query return ranked set of k best data objects based on the quality of feature objects in their spatial neighborhood. Quality may be subjective and query parametric.

There are two basic ways for ranking objects-spatial ranking and non-spatial ranking. Spatial ranking orders the objects according to their distance from a reference point whereas in non-spatial ranking objects are ordered by an aggregate function on their non-spatial values. A spatial preference query combines both spatial and non-spatial ranking.

Fig 1 illustrates three locations (p_1, p_2, p_3) of user interest and two feature sets (v and t) which are within the spatial neighborhood of the location. Feature points are labeled with quality values which are obtained from rating providers. The qualities are normalized to values in [0, 1].

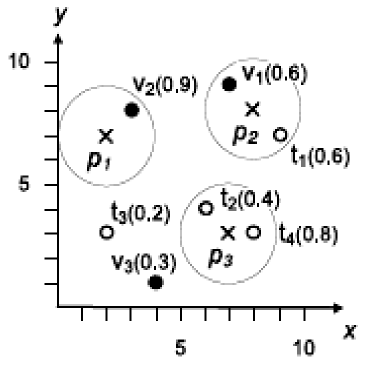

Fig: 1 Spatial area containing data and feature objects

Consider an example. A user wishes to retrieve the top-k flats from a spatial database maintained by a real estate agency, i.e., the user wants to retrieve the top-k flats which are near-to a high quality restaurant and a high quality hospital. The spatial neighborhood can be specified by the user to restrict the distance of the eligible feature objects (in the figure depicted as a range around each hotel). Thus, if the user wants to rank the flats based on the score of t (let it be restaurants), the top-1 flat is p_3 (0.8) whose score 0.8 is determined by t_4. However, if the user wants to rank the flats based on v (let it be cafes), the top-1 flat is p_1 (0.9) determined by v_2. Finally, if the user is interested in restaurants and cafes (e.g. summing the scores), the top-1 flat is p_2 (1.2).

In general, the score of a location p is obtained by aggregating the maximum quality of each feature in the spatial neighborhood of p. The semantics of the aggregate function is relevant to the user's query. The SUM function attempts to balance the overall qualities of all features. The MIN function that the top result has reasonably high qualities in all features. The MAX function, is used to optimize the quality in a particular feature, but not necessarily all of them.

Top-k spatial preference queries comprise a useful tool for a wide range of location based applications. But, processing this query is quite complex. Because it may require computing the scores of all data objects and select the top-k. This method seems to be expensive in terms of large input data sets. The proposed system aim at minimizing the i/o accesses to the data and feature objects while being also computationally efficient. To effectively prune the search space, the technique specified in the proposed system compute the upper score bounds for the objects indexed by spatial partitioning access methods.

It is not always possible to use multidimensional indexes for top-k retrieval. First, such indexes break down in high-dimensional spaces[10]. Second, top-k queries may involve an arbitrary set of user-specified attributes (e.g., size and price) from possible ones (e.g., size, price, distance to the beach, number of bedrooms, floor, etc.) and indexes may not be available for all possible attribute combinations (i.e., they are too expensive to create and maintain). Third, information for different rankings to be combined (i.e., for different attributes) could appear in different databases (in a distributed database scenario) and unified indexes may not exist for them. Solutions for top-k queries [2], [8] focus on the efficient merging of object rankings that may arrive from different (distributed) sources. Their motivation is to minimize the number of accesses to the input rankings until the objects with the top-k aggregate scores have been identified. To achieve this, upper and lower bounds for the objects seen so far are maintained while scanning the sorted lists.

2. BACKROUND KNOWLEDGE

Object ranking (ordering) is a popular retrieval task in various applications. The tuples in relational databases are ordered using an aggregate function on their attribute values. Consider a real estate agency database maintaining information regarding flats available for rent. A customer wishes to view the top ten flats with the largest sizes and lowest prices. Here, the score of each flat is expressed by the sum of two attributes: size and price. Ranking (ordering) in spatial databases is often associated to nearest neighbor (NN) retrieval. NN query returns the set of nearest objects to a given query location that qualify a condition(e.g., restaurants). If the set of interesting objects is indexed using an R-tree [3], then the index can be traversed in a branch and bound fashion to obtain the answer [4].

2.1 Spatial Query Evaluation on R-trees

Several approaches have been proposed for ranking spatial data. In order to handle spatial data efficiently, an effective indexing mechanism is required. One of the most popular spatial access methods is the R-tree [3] which indexes minimum bounding rectangles (MBRs) of spatial objects.

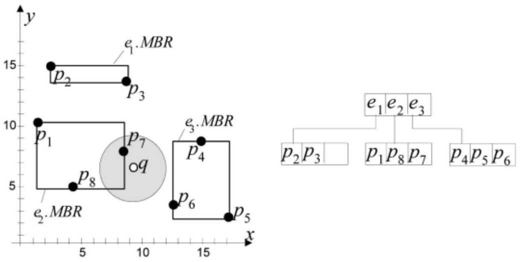

Fig 2 R-tree index structure

Fig 2 shows a set of interesting points D= {p_1…p_8} indexed using an R-tree. A spatial range query returns the objects in D that intersect the given query region W. For example, consider a range query that asks for all objects within the shaded area in Fig. 2. Starting from the root of the tree, the query is processed by recursively following entries, having MBRs that intersect the query region. For instance, e_1 does not intersect the query region, thus the subtree pointed by e_1 cannot contain any query result. In contrast, e_2 is followed by the algorithm and the points in the corresponding node are examined recursively to find the query result p_7.

A variant of an R-tree is the aR-tree [14]. Here each non-leaf entry develops an aggregate value (MAX) for some attribute measures in its subtree. For instance, a MAX aR-tree can be constructed over the point set given in Fig 2, if the entries e_1, e_2, e_3 contain the maximum measure values of sets {p_2, p_3}, {p_1, p_8, p_7}, {p_4, p_5, p_6}, respectively. Assume that the measure values of p_4, p_5, p_6 are 0.2, 0.1, and 0.4, respectively. Then the aggregate measure augmented in e_3 would be max {0.2, 0.1, 0.4} =0.4.

Given a feature data set F and a multidimensional region R, the range top-k query selects the tuples (from F) within the region R and returns only those with the k highest qualities. Hong et al. [11] indexed the data set by a MAX aR-tree and developed an efficient tree traversal algorithm to answer the query. Instead of finding the best k qualities from F in a specified region, range score query considers multiple spatial regions based on the points from the object data set D, and attempts to find out the best k regions (based on scores derived from multiple feature data sets F_c).

2.2 Feature Based Spatial Queries

Xia et al. [5] solved the problem of finding top-k sites (e.g., restaurants) based on their influence on feature points. As an example, Fig.3a shows a set of sites (white points), a set of features (black points with weights), such that each line links a feature point to its nearest site. The influence of a site p_i is defined by the sum of weights of feature points having p_i as their closest site. For instance, the score of p_1 is 0.9 + 0.5 = 1.4. Similarly, the scores of p_2 and p_3 are 1.5 and 1.2, respectively. Hence, p_2 is returned as the top-1 influential site.

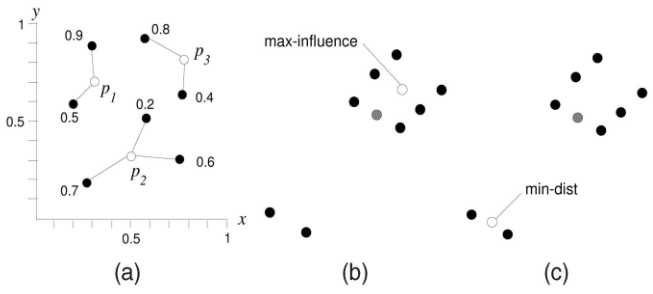

Fig 3 Influential sites and optimal location queries (a) Top-k influential (b) Max-Influence (c) Min-distance

Related to top-k influential sites query are the optimal location queries in [6], [7]. The goal is to find the location in space (not chosen from a specific set of sites) that minimizes an objective function. In Figs. 3b and 3c, feature points and existing sites are shown as black and gray points, respectively. Assume that all feature points have the same quality. The maximum influence optimal location query finds the location (to insert to the existing set of sites) with the maximum influence, whereas the minimum distance optimal location query searches for the location that minimizes the average distance from each feature point to its nearest site. The optimal locations for both queries are marked as white points in Figs. 3b and 3c, respectively.

The techniques proposed in [5], [6], [7] are specific to the particular query types and cannot be extended for top-k spatial preference queries. Also, they deal with only a single-feature data set

3. PRELIMINARIES

Let F_c be a feature data set, in which each feature object s ε F_c is associated with a quality w(s) and a spatial point. It is assumed that the domain of w(s) is in the interval [0, 1]. As an example, the quality w(s) of a restaurant s can be obtained from a ratings provider.

Let D be an object data set, where each object p ε D is a spatial point. In other words, D is the set of interesting points (e.g., hotel locations) considered by the user.

Given an object data set D and m feature data sets $F_1, F_2, \dots, F_m$, the top-k spatial preference query retrieves the k points in D with the highest score. Here, the score of an object point p ε D is defined as

$$\pi^{\theta}(p) = AGG\{\pi_c^{\theta}(p) \mid c \in [1, m]\}$$

where AGG is an aggregate function and $\pi^{\theta}{}_c(p)$ is the (cth) component score of p with respect to the neighborhood condition θ and the (cth) feature data set Fc. Typical examples of the aggregate function AGG are SUM, MIN, MAX.

The component score function, $\pi_c^{\theta}(p)$ taken is the range score function. The range score, π_c^{rng} (p) is taken as the maximum quality w(s) of points s ε F_c that are within a given parameter distance ε from p, or 0 if no such point exists.

$$\pi_c^{rng}(p) = \max(\{w(s) \mid s \in F_c \wedge dist(p,s) \le \varepsilon\} \cup \{0\})$$

The object data set D is indexed by an R-tree and each feature data set F_c is indexed using separate aR-tree. The reason for indexing different feature data sets by separate aR-trees is that: 1) A user queries for only few features (e.g., restaurants and cafes) out of all possible features (e.g., restaurants, cafes, hospital, market, etc.) and 2) Different users may consider different subsets of features.

4. PROPOSED APPROACH

4.1 Query Processing

A brute force approach for processing the top-k spatial preference query computes the score of every point p ε D in order to obtain the query results. Brute-force approach is expensive as it examines all objects in D and computes their component scores. The algorithm proposed here can significantly reduce the number of objects to be examined. The key idea is to compute, for non-leaf entries e in the object tree D, an upper bound T(e) of the score for any point p in the subtree of e. If T < γ, then need not access the subtree of e, thus numerous score computations can be saved.

Algorithm1 is a pseudocode of the proposed algorithm (branch and bound BB), based on this idea. BB is called with N being the root node of D. If N is a nonleaf node, Lines 3-5 compute the scores T (e) for nonleaf entries e concurrently. T(e) is an upper bound score for any point in the subtree of e. With the component scores $T_c(e)$ known so far, derive $T_{+}(e)$, an upper bound of T(e). If $T_+(e) \le \gamma$, then the subtree of e cannot contain better results than those in Wk and it is removed from the set V. In order to btain points with high scores early, sort the entries in descending order of T(e) before invoking the above procedure recursively on the child nodes pointed by the entries in V . If N is a leaf node, compute the scores for all points of N concurrently and then update the set W_k of the top-k results. Since both W_k and γ are global variables, their values are updated during recursive call of the BB.

Algorithm1.Branch-and-Bound Algorithm

```
Wk: = new min-heap of size k (initially empty);
                              - kth score in Wk
γ :=0

algorithm BB(Node N )
1: V :={e|e ε N};
2: If N is nonleaf then
3:  for c: = 1 to m do
4:     compute $T_c(e)$ for all e ε V concurrently;
5:     remove entries e in V such that $T_+(e) \le \gamma$;
6:  sort entries e ε V in descending order of T (e);
7:  for each entry e ε V such that T(e) > γ do
8:     read the child node N ′ pointed by e;
9:     BB (N’);
10: else
11:  for c: = 1 to m do
12:     compute $\pi_c(e)$ for all e ε V concurrently;
13:     remove entries e in V such that $\pi_+(e) \le \gamma$;
14:  update Wk (and γ) by entries in V
```

Upper bound scores $T_c(e)$ of nonleaf entries (within the same node N) can be computed concurrently (at Line 4). Upper bound score is to be computed such that 1) the bounds are computed with low I/O cost, and 2) the bounds are reasonably tight, in order to facilitate effective pruning.To achieve this, only level-1 entries (i.e., lowest level nonleaf entries) in F_c are utilized for deriving upper bound scores because: 1) there are much fewer level-1 entries than leaf entries (i.e., points), and 2) high-level entries in F_c cannot provide tight bounds.

5. IMPLEMENTATION DETAILS

The proposed algorithm was implemented in Java. The object data set is indexed using an R-tree. Two feature data sets were taken. Each of them indexed using separate aR-tree. For each data set the coordinates of points are random values uniformly and independently generated. The domain of the quality attribute is normalized to the unit interval [0, 1]. The aggregate function used is SUM.

6. CONCLUSION

Top-k spatial preference queries, provides a novel type of ranking for spatial objects based on qualities of features in their neighborhood. But processing these queries is quite complex. The brute force approach computes the scores of every data object by querying on feature data sets. This method is expensive for large input data sets. The alternative technique, branch and bound algorithm, aims at minimizing the i/o accesses to the object set. It derives upper bound scores for nonleaf entries in the object tree, and prunes those that cannot lead to better results. BB is scalable to large data sets and it is a robust algorithm with respect to various parameters.

7. REFERENCES

[1] M.L. Yiu, X. Dai, N. Mamoulis, and M. Vaitis, “Top-k Spatial Preference Queries,” Proc. IEEE Int’l Conf. Data Eng. (ICDE),

[2] N. Bruno, L. Gravano, and A. Marian, “Evaluating Top-k Queries over Web-Accessible Databases,” Proc. IEEE Int’l Conf. Data Eng.(ICDE), 2002Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[3] A. Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching,” Proc. ACM SIGMOD, 1984.

[4] G.R. Hjaltason and H. Samet, "Distance Browsing in Spatial Databases," ACM Trans. Database Systems, vol. 24, no. 2, pp. 265-318, 1999

[5] T. Xia, D. Zhang, E. Kanoulas, and Y. Du, "On Computing Top-t Most Influential Spatial Sites," Proc. 31st Int'l Conf. Very Large Data Bases (VLDB), 2005

[6] T. Xia, D. Zhang, E. Kanoulas, and Y. Du, "On Computing Top-t Most Influential Spatial Sites," Proc. 31st Int'l Conf. Very Large Data Bases (VLDB), 2005

[7] D. Zhang, Y. Du, T. Xia, and Y. Tao, "Progessive Computation of The Min-Dist Optimal-Location Query," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB),2006

[8] N. Mamoulis, M.L. Yiu, K.H. Cheng, and D.W. Cheung, "Efficient Top-k Aggregation of Ranked Inputs," ACM Trans. Database Systems, vol. 32, no. 3, p. 19, 2007

[9] Joao B. Rochar,Akrivi Vlachou, Christos Doulkeridis, and Kjetil "Efficient Processing of Topk Spatial Preference Queries" Proceedings of the VLDB Endowment Volume 4 Issue 2, November 2010

[10] K.S. Beyer, J. Goldstein, R.Ramakrishnan, and U. Shaft, "When is 'Nearest Neighbor' Meaningful?" Proc. Seventh Int'l Conf. Database Theory (ICDT), 1999

[11] S. Hong, B. Moon, and S. Lee, "Efficient Execution of Range Top-k Queries in Aggregate R-Trees," IEICE Trans. Information and Systems, vol. 88-D, no. 11, pp. 2544-2554,2005.

[12] Y. Chen and J.M. Patel, "Efficient Evaluation of All-Nearest- Neighbor Queries," Proc. IEEE Int'l Conf. Data Eng.(ICDE),2007.

[13] P.G.Y. Kumar and R. Janardan, "Efficient Algorithms for Reverse Proximity Query Problems," Proc. 16th ACM Int'l Conf. Advances in Geographic Information Systems (GIS), 2008.

[14] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao, "Efficient OLAP Operations in Spatial Data Warehouses," Proc. Int'l Symp. Spatial and Temporal Databases(SSTD),2001