International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.2., Issue.6, 2014

RESEARCH ARTICLE

ISSN: 2321-7758

TOP-K RANKING SPATIAL QUERIES OVER FILTERING DATA

CH.RAGA MADHURI[1], B.NARASMIHA SWAMY[2]
[1]M.Tech in CSE Dept , PVP Siddhartha Institute of Technology
Kanuru,Vijayawada, ANDHRA PRADESH, INDIA
[2]Senior Assistant Professor in CSE Dept, PVP Siddhartha Institute of Technology
Kanuru,Vijayawada, ANDHRA PRADESH, INDIA

Article Received: 25/11/2014 | Article Revised on: 03/12/2014 | Article Accepted on:04/12/2014

CH.RAGA MADURI

ABSTRACT

A spatial inclination question positions items focused around the characteristics of gimmicks in their spatial neighborhood. For instance, utilizing a land office database of pads for lease, a client may need to rank the pads concerning the fittingness of their area, characterized in the wake of amassing the characteristics of different peculiarities (e.g., restaurants, bistros, healing center, market, and so on.) inside their spatial neighborhood. Such an area idea can be determined by the client by means of diverse capacities. At the same time they are getting to administrations just non sifting philosophy. In this paper we propose to create Top-k ideal item classification can be characterized through effective procedure.Our test results show effective separating principles with semantic information representation.

Keywords: Query processing, spatial databases, top-k spatial preference query.

INTRODUCTION

Spatial database systems manage large collections of geographic entities, which apart from spatial attributes contain non-spatial information (e.g., name, size, type, price, etc.). In this paper, we study an interesting type of preference queries, which select the best spatial location with respect to the quality of facilities in its spatial neighborhood [1]. Given a set D of interesting objects (e.g., candidate locations), a top-k spatial preference query retrieves the k objects in D with the highest scores. The score of an object is defined by the quality of features (e.g., facilities or services) in its spatial neighborhood. As a motivating example, consider a real estate agency office that holds a database with available flats for lease. Here “feature” refers to a class of objects in a spatial map such as specific facilities or services

With the popularization of geotagging information, there has been an increasing number of Web information systems specialized in providing interesting results through location-based queries. However, most of the existing systems are limited to plain spatial queries that return the objects present in a given region of the space.

International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in
Vol.2., Issue.6, 2014

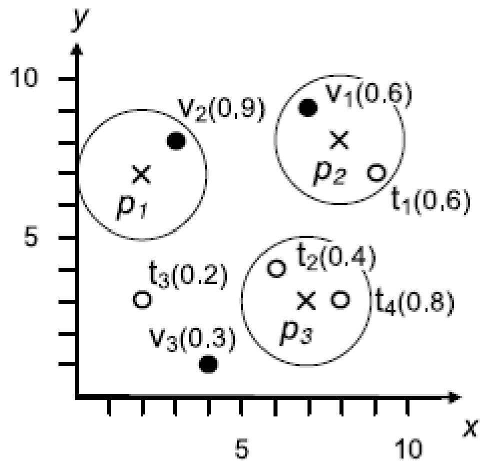

Figure 1: Spatial area containing data and feature objects.

For example, Figure 1 presents a spatial area containing data objects p (hotels) together with feature objects t (restaurants) and v (caf´es) with their respective scores (e.g. rating). Consider a tourist interested in hotels with good restaurants and caf´es in their spatial neighborhood. The tourist specifies a spatial constraint (in the figure depicted as a range around each hotel) to restrict the distance of the eligible feature objects for each hotel [2]. Thus, if the tourist

wants to rank the hotels based on the score of restaurants, the top-1 hotel is p3(0.8) whose score 0.8 is determined by t4. However, if the tourist wants to rank the hotels based on caf´es, the top-1 hotel is p1(0.9) determined by v2.

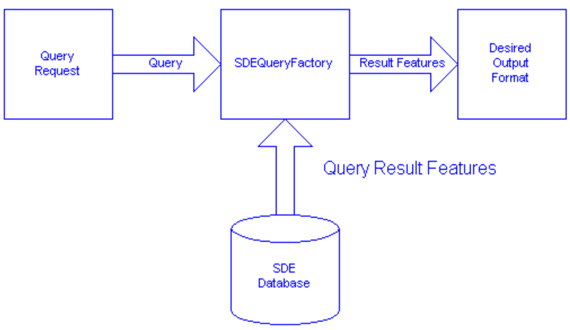

Figure 2: Interoatability of the query processing in spatial query processing.

In this paper, we propose a novel approach for processing spatial preference queries efficiently. The main difference compared to traditional top-k queries is that the score of each data object is defined by the feature objects that satisfy a spatial constraint (for example range constraint). Therefore, pairs of data and feature objects need to be examined to determine the score of an object [3]. Our approach relies on mapping of pairs of data and feature objects to a distance-score space, which in turn allows us to identify the minimal subset of pairs that is sufficient to answer all spatial preference queries. Capitalizing on the materialization of this subset of pairs, we present an efficient algorithm that improves query processing performance by avoiding examining the spatial neighborhood of data objects during query execution.

I. RELATED WORK

Several approaches have been proposed for ranking spatial data objects. The reverse nearest neighbor (RNN) query was first proposed by Korn and Muthukrishnan. Then, Xia *et al.* studied the problem of retrieving the top-k most influential spatial objects, where the score of each spatial data object p is defined as the sum of the scores of all feature objects that have p as their nearest neighbor. Yang *et al.* studied the problem of finding an optimal location.

The main difference compared to is that the optimal location can be any point in the data space and not necessarily an object of the dataset, while the score is computed in a similar way to. The aforementioned approaches define the score of a spatial data object p based on the scores of feature objects that have p as their nearest neighbor and are limited to a single feature set. Differently, Yiu *et al.* first considered computing the score of a data object p based on feature objects in its spatial neighborhood from multiple feature sets. To this end, three different spatial scores were defined: range, nearest neighbor, and influence score; and different algorithms were developed to compute top-k spatial preference queries for these scores.

To solve this type of queries, many of the researchers proposed several methods. In 2004, F. Ilyas et.al. introduced a new rank-join algorithm that made use of the individual orders of its inputs to produce join results ordered on a user-specified scoring function. They had experimentally evaluated their proposed rank join operators and analyze its performance. In 2006, Rank-aware query optimization framework by Ihab F. Ilyas et.al. fully integrated the rank-join operators into relational query engines and shown the performance of the proposed framework. In 2007 Yiu et.al., proposed the Branch and bound (BB) and Feature join algorithm (FJ) that rank objects based on the qualities of features. They proved that their proposed work is better than simple and Group probing algorithms with real and synthetic data. The top-k queries produce ordered result by using some calculated score. Generally, users are interested in top-k join result. For this, the top-k queries require joins to produce top-k result. The relational processors should not process the ranking queries with join efficiently.

BACKGROUND WORK

Definitions and Index Structures

Let Fc be a feature data set, in which each feature object s 2 Fc is associated with a quality !ðsÞ and a spatial point.

We assume that the domain of !ðsÞ is the interval ½0; 1_. As an example, the quality !ðsÞ of a restaurant s can be obtained from a ratings provider. Let D be an object data set, where each object p 2 D is a spatial point [4]. In other words, D is the set of interesting points (e.g., hotel locations) considered by the user.

Probing Algorithms

We first introduce a brute-force solution that computes the score of every point p 2 D in order to obtain the query results [5]. Then, we propose a group evaluation technique that computes the scores of multiple points concurrently.

Algorithm 1 Simple Probing Algorithm (SP)

```
    algorithm SP(Node N)
1:  for each entry e ∈ N do
2:      if N is non-leaf then
3:          read the child node N' pointed by e;
4:          SP(N');
5:      else
6:          for c:=1 to m do
7:              if τ+(e) > γ then          ▷ upper bound score
8:                  compute τc(e) using tree Fc; update τ+(e);
9:          if τ(e) > γ then
10:             update Wk (and γ) by e;
```

Algorithm 1 is a pseudo code of the simple probing (SP) algorithm, which retrieves the query results by computing the score of every object point. The algorithm uses two global variables: Wk is a min-heap for managing the top-k results and represents the top-k score so far (i.e., lowest score in Wk). Initially, the algorithm is invoked at the root node of the object tree (i.e., N ¼ D:root). The procedure is recursively applied (at Line 4) on tree nodes until a leaf node is accessed.

Upper Bound Score Computation

It remains to clarify how the (upper bound) scores T cðeÞ of nonleaf entries (within the same node N) can be computed concurrently. Our goal is to compute these upper bound scores such that . the bonds are computed with low I/O cost, and . the bonds are reasonably tight, in order to facilitate effective pruning. To achieve this, we utilize only level-1 entries (i.e., lowest level nonleaf entries) in Fc 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 Fc cannot provide tight bounds.

PROPOSED WORK

A Spatial preference query, ranks the spatial objects based on quality of its neighbor facilities. For instance a tourist might retrieve a sorted list of hotels based on the facilities around that (e.g. restaurant, hospital , market, etc.). Assume that p is our point of interest (e.g. a hotel) and we have *m* type of facilities(e.g. restaurant means m=1 and park means m=2). Then assume that *n m f* is *n-th* facility from type *m (e.g.* Restaurant *A [6])*. First we retrieve a list of candidates for *P* according to Table 1. Table 1 shows how one of the methods choose the primary candidates.

Table 1 Candidate Selection Criteria

Method	
Nearest Neighbor	Min(d, f_m^n))
Range Score	D(p, f_m^n)<R
Influence Score	All

As we can see, Nearest Neighbor, from each type *m* retrieves *n-th* element of that (*n m f*) which has the minimum distance with p. Range score retrieves a list of items which have at least distance(*d*) of pre-defined *R* with *P*. Influence score retrieves all the items for further computation. Afterwards, We define Score of point *P* according to the following equation:

{ } 1 *m Ci m m p*

Ci S = Σ *Agg w* ×α (1)

Where, *Agg* denotes the aggregation function which can be maximum or sum. *w* is equal to the weight or quality of item(e.g. hotel with 5 star can have weight of 5 and hotel with one star can have weight of 1) and *i* is an index of retrieved candidates. α is influence function which is equal to 1 for Nearest Neighbor and

Range score and is equal to the equation 2 for Influence score. ( ) *R f p d im* , 2− α = (2) Where *d* denotes the distance between point P and facility *i* of category *m*. and *R* is a pre-defined radius. Then the result of Top-K spatial preference query is a sorted list of Sp for all point of interests (*P*).

BB* Algorithm: In order to allocate the location process present in the processing of operations as follows:

Algorithm 2 Optimized Group Range Score algorithm

```
    algorithm Optimized_Group_Range(Trees 𝓕1, 𝓕2, ⋯ , 𝓕m,
    Set V, Value ε, Value γ)
1:  for c:=1 to m do
2:      Hc:=new max-heap (with quality score as key);
3:      insert 𝓕c.root into Hc;
4:      μc:=1;
5:      for each entry p ∈ V do
6:          τc(p):=0;
7:  α:=1;                          ▷ ID of the current feature tree
8:  while |V| > 0 and there exists a non-empty heap Hc do
9:      deheap an entry e from Hα;
10:     μα:=ω(e);                      ▷ update threshold
11:     if ∀p ∈ V, mindist(p, e) > ε then
12:         continue at Line 8;
13:     for each p ∈ V do              ▷ prune unqualified points
14:         if (∑_{c=1}^{m} max{μc, τc(p)}) ≤ γ then
15:             remove p from V;
16:     read the child node CN pointed to by e;
17:     for each entry e′ of CN do
18:         if CN is a non-leaf node then
19:             if ∃p ∈ V, mindist(p, e′) ≤ ε then
20:                 insert e′ into Hα;
21:         else                       ▷ update component scores
22:             for each p ∈ V such that dist(p, e′) ≤ ε do
23:                 τα(p):=max{τα(p), ω(e′)};
24:     α:=next (round-robin) value where Hα is not empty;
25: for each entry p ∈ V do
26:     τ(p):=∑_{c=1}^{m} τc(p);
```

The process of the above algorithm will perform efficient and effective processing operations in spatial ranking specifications with consistent data processing.

The pseudo-code for registering the scores of articles proficiently from the peculiarity trees F1;f2; ;Fm. The set V contains objects whose scores need to be registered. alludes to the separation edge of the extent score, and speaks to the best score discovered in this way. For each one peculiarity tree Fc, we utilize a max-load Hc to navigate the entrances of Fc in dropping request of their quality qualities. The foundation of Fc is initially embedded into Hc [7]. The variable c keeps up the upper bound nature of passages in the tree that will be gone to. We then instate every part score c(p) of each article p 2 V to 0. At Line 7, the variable stays informed concerning the ID of the current peculiarity tree being transformed. The circle at Line 8 is utilized to process the scores for the focuses in the set V . We then deheap a section e from the current pile H. The property of the max-store ensures that the quality estimation of any future entrance d-eheaped from H is at generally !(e). In this way, the bound c is overhauled to !(e). At Lines 11–12, we prune the entrance e on the off chance that its separation from each one item point p 2 V is bigger than . In the event that e is not pruned, we register the tight upper headed score ?(p) for every p 2 V (by Equation 4); the item p is expelled from V if ?(p) (Lines 13–15). Next, we get to the tyke hub indicated by e, and inspect every passage e0 in the hub [8]. A nonleaf section e0 is embedded into the load H on the off chance that its base separation from some p 2 V is inside (Lines 18–20); though a leaf entrance e0 is utilized to overhaul the segment score (p) for any p 2 V inside separation from e0 (Lines 22–23). At Line 24, we apply the round robin method to discover the following

esteem such that the stack H is not unfilled. The circle at Line 8 rehashes while V is not void and there exists a non-exhaust stack Hc. At the end, the calculation determines the accurate scores for the remaining purpose.

EXPERIMENTAL RESUTLS

In this section, we evaluate our proposed algorithm (*SFA*) and we compare SFA against the algorithms developed by Yiu *et al.* denoted as *GP*, *BB*, *BB**, and

FJ. All algorithms were implemented in Java and executed on a PC with 3GHz Dual Core AMD Processor with 2GB RAM. The datasets were indexed by an R-tree (aR-tree for [16, 17]) with block size of 4KB. We used an LRU memory buffer with a fixed size of 0.2% of the size of the total number of objects stored in O and Fi. We report the average values of 20 experiments, and in each experiment we recreate all datasets and indexes to factor out the effects of randomization. In all experiments, we measured the total execution time (referred to as response time) and number of I/Os. All charts are plotted using a logarithmic scale on the y-axis.

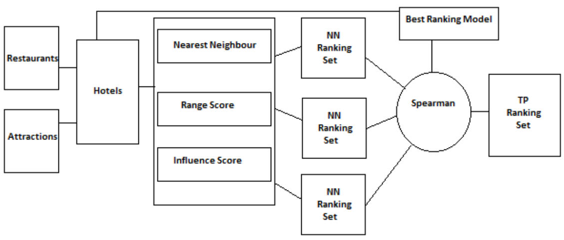

Figure 3: Experimental setup progression.

Experimental Settings

We conduct experiments using both synthetic and real datasets. First, we perform experiments using uniform distribution (UN) for the spatial locations of data and feature objects and for the score of the feature objects (within the range [0, 1]). We also generate a synthetic dataset (CN) that resembles the real world: (1) there exist multiple city centers (centroids) with higher occurrences of data objects, (2) there exists a higher probability of finding feature objects nearby the city centers (centroids) [9]. Appendix C.1 provides a detailed description of CN including a plot of a generated dataset. We use the synthetic dataset (CN) as our default dataset. By default, the non-spatial score of the feature objects is a uniformly generated value within the range [0, 1].

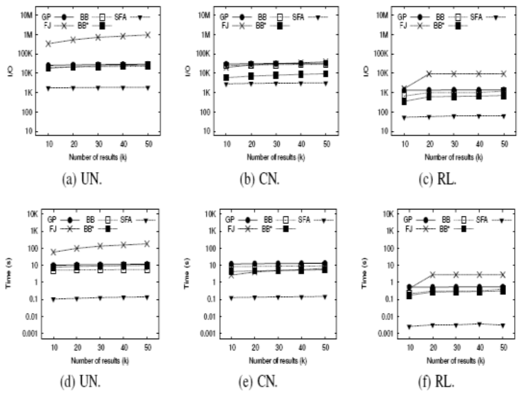

Figure 4: Effect of different data distributions {UN,CN,RL} on I/O and response time (range score).

Query Processing Performance

Range Score. In Figure 4, we use our default setup and study the number of I/Os and the response time for all datasets, while varying k. Figure 4 presents the I/O cost using the UN dataset. The performance of GP is stable because it always computes the score of all data objects [10]. FJ requires a much higher number of I/Os, as it needs to access many leaf entries of the feature R-trees in order to report the correct top-k result set. The branch-and-bound algorithms (BB and BB*) perform slightly better than GP for this setup. However, SFA results in one order of magnitude fewer I/Os than the best of its competitors. We plot the number of I/Os for the CN dataset. BB* performs better than GP, BB, and FJ due to the employed pruning [11]. However, SFA reduces even further the number of required I/Os compared to BB* and scales better than BB* for increasing value of k. In I/O cost for the real dataset (RL) is presented. Again, SFA outperforms all other algorithms (in terms of I/Os) by at least one order of magnitude. This experiment indicates that SFA performs efficiently for a wide range of different datasets [12]. Depict the response time for the same experimental setups respectively

CONCLUSION

In this paper, we present a novel approach for boosting the performance of top-k spatial preference query processing. At the heart of our framework lies a mapping of pairs of data and feature objects to a distance-score space, which enables us to identify the minimal subset of pairs necessary to answer any ranked spatial preference query. By materializing this subset of pairs, we present efficient algorithms for query processing that result in improved performance. Furthermore, we describe an efficient algorithm for materialization and elaborate on useful properties that reduce the cost of maintenance. Our experimental evaluation demonstrates that our approach reduces I/Os and response time by more than one order of magnitude compared to the state-of-the-art algorithms in most of the setups.

REFERENCES

[1]. Extracting Spatial Association Rules from the Maximum Frequent Itemsets Based on Boolean.Matrix Junming Chen, Guangfa Lin, Zhihai Yang

[2]. C. B¨ohm, B. C. Ooi, C. Plant, and Y. Yan. Efficiently processing continuous k-nn queries on data streams. In Proc. of Int. Conf. on Data Engineering (ICDE), pages 156–165, 2007.

[3]. S. B¨orzs¨onyi, D. Kossmann, and K. Stocker. The skyline operator. In Proc. of Int. Conf. on Data Engineering (ICDE), page 421430, 2001.

[4]. S. Chaudhuri, N. N. Dalvi, and R. Kaushik. Robust cardinality and cost estimation for skyline operator. In Proc. of Int. Conf. on Data Engineering (ICDE), page 64, 2006.

[5]. Y. Du, D. Zhang, and T. Xia. The Optimal-Location query. In Proc. of the Int. Symposium on Spatial and Temporal Databases (SSTD), pages 163–180, 2005.

[6]. https://developers.google.com/maps/

[7]. http://developer.yahoo.com/maps/

[8]. C. Spearman, The Proof and Measurement of Association between Two Things, The American Journal of Psychology, Vol. 15, No. 1 (Jan., 1904), pp. 72-101.

[9]. Yiu. M. L, Dai.X,Mamoulis.N, Vaitis.M. “Ranking Spatial data by quality preferences”.IEEE Transaction on Knowledge and Data Engineering,Vol.23, 3, pp.433 – 446, 2011.

[10]. Yong Zhang, Lizhu Zhou, Jun Chen. “Layered r-tree: an index structure for three dimensional points and lines”. International Archives of Photogrammetry and Remote Sensing, Vol. XXXIII,B4.

[11]. Antonio Leopoldo, Corral Liria. Algorithms for the Processing of Spatial Queries using R-trees. The Closest Pairs Query and its Application on Spatial Databases Almeria, January 2002

[12]. Yunjunga gao,“Optimal-location-selection query processing in spatial database”knowledge and data engineering, august 2009.

[13]. http://www.ijetae.com/files/Volume2Issue11/IJETAE_1112_117.pdf

[14]. http://www.vldb.org/pvldb/vol4/p93-rocha-junior.pdf

[15]. http://www.iosrjen.org/Papers/vol2_issue11(part-2)/B021120817.pdf