Novel Method for NKS Search in Multidimensional Dataset Using Advance Promish & Ranking Function

[1]Shilpa Thakare, [2]Prof. Shinde Jayashri
[1]PG Student, [2]Asst. Professor, [1,2]Department of Computer Engineering, Late G. N. Sapkal College of Engineering, Nashik, Maharashtra, India.
[1]shilpa.thakare30@gmail.com , [2]jv.shinde@rediffmail.com

Abstract We Consider multi-dimensional dataset where each data point has set of keyword in feature space allows for the development of new tools to query and explore these multidimensional dataset. In this paper, we study nearest keyword set Queries on text reach multidimensional dataset. We propose a novel method called ProMiSH(Projection and Multiscale Hashing) that uses random projection and hash-based index structure. Our experimental result shows that ProMiSH has Speedup over state-of-art-tree-based techniques. Keyword-based search in text-rich multi-dimensional datasets facilitates many novel applications and tools. In this paper, we consider objects that are tagged with keywords and are embedded in a vector space. For these datasets, we study queries that ask for the tightest groups of points satisfying a given set of keywords.

Keywords: NKS, Multi-Dimensional, data, Indexing, Hashing.

I. INTRODUCTION

In Existing techniques using tree based indexes suggest possible solution to NKS queries on multi-dimensional dataset, the performance of these algorithms decline sharply with the increase of size or dimensionality in dataset. Therefore there is need for an efficient algorithm that scales with dataset dimension, and yield practical query efficiency on large datasets. Multi-dimensional dataset. An NKS query is set of user-provide keywords, and result of the query may include k-sets of data points each of which contains all the query keywords and forms one of the top-k tightest cluster in the multi-dimensional space. In this paper We study nearest keyword set queries on text-rich multi-dimensional datasets. An NKS query is set of user provided keywords, and the result of the query may include k sets of data points each of which contains all the query keywords and forms one of the top-k tightest Cluster in the multi-dimensional space.

Following are some applications of NKS queries,

1)NKS queries can also reveal geographic patterns. GIS can characterize a region by a high-dimensional set of attributes, such as pressure, humidity, and soil types Meanwhile, these regions can also be tagged with information such as diseases. An epidemiologist can formulate NKS queries to discover patterns by finding a set of similar regions with all the diseases of her interest

Propose Pro MiSH(short for Projection and Multi-scale Hashing) to enable fast processing for NKS Queries. In Particular, developed an exact ProMiSH(refer to as ProMiSH-E) that always retrieves the optimal top-k results and an approximate ProMiSH(referred to as ProMiSH-A) that is more efficient in term of time and space ,and is able to obtained near-optimal results in practice. ProMiSH-E uses a set of hash tables and inverted indexes to perform a localized search. The hashing technique is inspired by Locality Sensitive Hashing (LSH), which is state-of-the-art method for nearest neighbor search in high-dimensional spaces. Unlike

LHS-based method that allow only approximate search with probabilistic guarantees, the index structure in ProMiSH-E supports accurate search. ProMiSH-E creates hash tables at multiple bin-widths, called index levels. A single round of search in a Hash table yield subset of points that contain query results, and ProMiSH-E explores each subset using a fast pruning based algorithm ProMiSH-A is an approximate variation of ProMiSH-E for better time and space efficiency. using this algorithm evaluate the performance of ProMiSH on both real and synthetic datasets and employ state-of-art VbR*-Tree and CoSKQ as baselines. The empirical results reval that ProMiSH Consistently outperforms the baseline algorithms and ProMiSH-A

II. LITRATURE SURVEY

A variety of related queries have been studied in literature on text-rich spatial datasets. Locating Mapped Resources in Web 2.0 [9] in this technique locating geographical resources and proposed an efficient tag-centric query processing strategy. find set of nearest co-related objects which together match the query tags. Felipe et al. [1] present an efficient method to answer top-k special keyword queries. To do so, we introduce an indexing structure called IR2-Tree(Information Retrieval R-Tree)which combines an R-Tree with superimposed text signatures. Author present an algorithms that construct and maintain an IR2-Tree and use it to answer top-k spatial keyword queries. its show superior performance and excellent scalability.IR2Tree to rank object from spatial dataset based on combination of their distances to the query location and relevance of their text description to the query keyword. Top-k spatial keyword queries which is based on tight integration of data structure and algorithm used in special database search and information retrieval R-Tree(IR2-Tree)which is structure based on the R-Tree at query time and incremental algorithm is employed that uses IR2-Tree which is structure based on the R-Tree at query time and incremental algorithms.is employed that uses IR2-Tree which is structure based on the R-Tree at query time and incremental algorithm. other related queries include aggregate nearest keyword search in spatial database [4] ,in this query retrieves k objects from Q with minimum sum of distances to it's nearest point in D such that each nearest point matches at least one query keyword for processing this query several algorithm proposed using IR2-Tree as index structure.

Another track of related works deal with m-closest keyword queries[2].In[2],bR*-Tree is developed based on R*-tree[3] that stores bitmaps and minimum bounding rectangles(MBRs) of keywords in every node along with points MBRs.bR-Tree also suffers from a high storage cost; therefore Zang et al Modified bR*-Tree to create virtual bR*-tree in memory at run time. Virtual bR*-tree is created from a pre-stored r*-Tree, which indexes all the points, and an inverted index which stored keyword information and path from the root node in R*-Tree for each point. Both bR*-Tree and virtual bR*-Tree shares similar performance weaknesses as bR*-Tree.

Tree-based indexes, such as M-tree[5] ,is proposed to organize and search large dataset from generic. M-Tree always balanced several heuristic split alternatives are considered and experimentally evaluated. This M-Tree have been extensively investigated for nearest neighbor search in high dimensional spaces. this index fails to scale to dimensions greater than 10 because of the curse of dimensionality. Random projection with hashing [6][7][8] has comes to be the state-of-the-art method for nearest neighbor search in high dimensional dataset. Jon M. Kleinberg[8] Develop new approach to the nearest-neighbor problem, based on a method for combining randomly chosen one dimensional projections of the underlying point set. Two algorithms are introduce in this first for finding epsilon-approximate nearest neighbors and second epsilon-approximate nearest-neighbor algorithm with near-linear storage and query time improves asymptotically linear search in all dimensions. Aristides Gionis [6] examine a novel scheme for approximate similarity search based on hashing. the basic idea is to hash the points from the database so as to ensure that the probability of collision is much higher for objects that are close to each other than for those that are far

apart. the method gives significant improvement in running time over other methods for searching in high dimensional spaces based on hierarchical tree de-composition. This scheme scales well even for relatively large number of dimensions(more than 50). previous technique[6] solve this problem efficiently only for the approximate case Accurate and efficient Near neighbor Search in High Dimensional Spaces [7] In this are design to solve r-near neighbor queries for a fixed query range or for set of query ranges with probabilistic guarantees .and then extend for nearest neighbor queries. Vishwakarma Singh introduce novel indexing and querying scheme called Spatial Intersection and Metric Pruning(SIMP) Empirical study of this method on three real datasets having dimensions between 32 to 256 and size up to 10 million show a superior performance of SIMP over LSH.

III. SYSTEM ARCHITECTURE

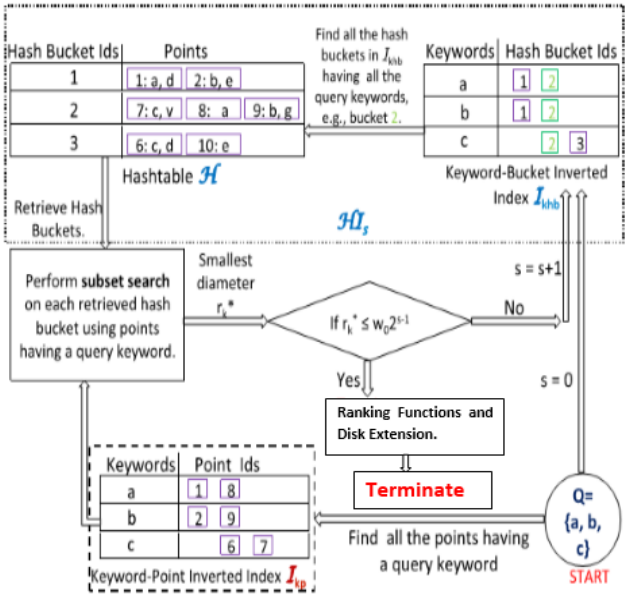

Fig. 1 System Architecture

Modules Information:

a) Search Algorithm Module

ProMiSH referred to as ProMiSH-A. We start with the algorithm description of ProMiSH-A, and then analyze its approximation quality. ProMiSH-E highly depends on an efficient search algorithm that finds top- k results from a subset of data points.

b) HI Construction Module

It consists of multiple hash tables and inverted indexes referred to as HI. HI is controlled by three parameters:

(1) Index level(L)

HI at all the index level then it performs a search in the complete dataset D.

(2) Number of random unit vectors(m)

We consider its projection space as a segment [0, pMax] and partition the segment into $2^{(L-s+1)}$ +1 overlapping bins, where each bin has width and is equally overlapped with two other bins. We conduct the projection space partition on all the m random unit vectors.

(3) Hash table size(B)

A given a dictionary V and hash table $H^{(s)}$, we create the inverted index $I^{(s)}_{khb}$. In this inverted index, keys are still keywords. HI with one pair of hash table and inverted index shown in the dotted rectangle.

c) Ranking functions

In the future, we plan to explore other scoring schemes for ranking the result sets. In one scheme, we may assign weights to the keywords of a point by using techniques like tf-idf. Then, each group of points can be scored based on distance between points and weights of keywords. Furthermore, the criteria of a result containing all the keywords can be relaxed to generate results having only a subset of the query keywords.

d) Disk extension

We plan to explore the extension of ProMiSH to disk. ProMiSH-E sequentially reads only required buckets from Ikp to find points containing at least one query keyword. Therefore, Ikp can be stored on disk using a directory-file structure. We can create a directory for Ikp. Each bucket of Ikp will be stored in a separate file named after its key in the directory. Moreover, ProMiSH-E sequentially probes HI data structures starting at the smallest scale to generate the candidate point ids for the subset search, and it reads only required buckets from the hash table and the inverted index of

a HI structure. Therefore, all the hash tables and the inverted indexes of HI can again be stored using a similar directory file structure as Ikp, and all the points in the dataset can be indexed into a B+-Tree [36] using their ids and stored on the disk. In this way, subset search can retrieve the points from the disk using B+-Tree for exploring the final set of results.

IV. CONCLUSION

The problem of top-k nearest keyword set search in multi-dimensional datasets. We proposed a novel index called ProMiSH based on random projections and hashing. Based on this index, we developed ProMiSH-E that finds an optimal subset of points and ProMiSH-A that searches near-optimal results with better efficiency. Our empirical results show that ProMiSH is faster than state-of-the-art tree-based techniques, with multiple orders of magnitude performance improvement. Moreover, our techniques scale well with both real and synthetic datasets.

ACKNOWLEDGMENT

It gives us great pleasure in presenting the preliminary project report on "Novel method for nks search in multidimensional dataset using advance promish and ranking function".

I would like to take this opportunity to thank my internal guide Prof. J. V. Shinde for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful.

REFERENCES

[1] I. De Felipe, V. Hristidis, and N. Rishe, "*Keyword search on spatial databases*," in ICDE, 2008, pp. 656–665.

[2] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa,"*Keyword search in spatial databases: Towards searching by document*," in ICDE, 2009, pp. 688–699.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "*The R*-tree: An fficient and robust access method for points and rectangles*," in SIGMOD, 1990, pp. 322–331.

[4] Z. Li, H. Xu, Y. Lu, and A. Qian, "*Aggregate nearest keyword search in spatial databases*," in Asia-Pacific Web Conference, 2010.

[5] P. Ciaccia, M. Patella, and P. Zezula, "*M-tree: An efficient access method for similarity search in metric spaces*," in VLDB, 1997.

[6] A. Gionis, P. Indyk, and R. Motwani, "*Similarity search in high dimensions via hashing*," in VLDB, 1999, pp. 518–529.

[7] V. Singh and A. K. Singh, "*Simp: accurate and efficient near neighbor search in high dimensional spaces*," in EDBT, 2012, pp. 492–503.

[8] J. M. Kleinberg, "*Two algorithms for nearest-neighbor search in high dimensions*," in STOC, 1997, pp. 599–608.

[9] D. Zhang, B. C. Ooi, and A. K. H. Tung, "*Locating mapped resources in web 2.0*," in ICDE, 2010, pp. 521–532.

[10] V. Singh, S. Venkatesha, and A. K. Singh, "*Geo-clustering of images with missinggeotags*," in GRC, 2010, pp. 420–425.

