EFFICIENTLY SEARCHING NEAREST NEIGHBOR IN DOCUMENTS USING KEYWORDS

Sonal S. Kasare[1], Anup Bongale[2]

[1]Lecturer, [2]Asst. Prof., Computer Engineering, Dr. D. Y. Patil School of Engg, Maharashtra, India,
sonalkasare@gmail.com, anup.dypcoe@gmail.com

Abstract

Conservative spatial queries, such as range search and nearest neighbor reclamation, involve only conditions on objects' numerical properties. Today, many modern applications call for novel forms of queries that aim to find objects satisfying both a spatial predicate, and a predicate on their associated texts. For example, instead of considering all the restaurants, a nearest neighbor query would instead ask for the restaurant that is the closest among those whose menus contain "steak, spaghetti, brandy" all at the same time. Currently the best solution to such queries is based on the InformationRetrieval2-tree, which, has a few deficiencies that seriously impact its efficiency. Motivated by this, there is a development of a new access method called the spatial inverted index that extends the conventional inverted index to cope with multidimensional data, and comes with algorithms that can answer nearest neighbor queries with keywords in real time. As verified by experiments, the proposed techniques outperform the InformationRetrieval2-tree in query response time significantly, often by a factor of orders of magnitude

Keywords: *Information retrieval, spatial index, keyword search*

----------***----------

1. INTRODUCTION

Keyword search in document performed with various approaches ranked retrieval results, clustering search results & identifying the nearest neighbor Keyword search on xml document categorized as two different approaches one is Keyword search on xml document which can be performed by ranking the searched results based on match or the answer to keyword & finding the nearest neighbor of keyword by using GST or by Xpath Query. In paper [2] the problem of returning clustered results for keyword search on documents the core of the semantics is the conceptually related relationship between keyword matches, which is based on the conceptual relationship between nodes in trees. Then, we propose a new clustering methodology for search results, which clusters results according to the way they match the given query.

A spatial database manages multidimensional objects (such as points, rectangles, etc.), and provides fast access to those objects based on different selection criteria. The importance of spatial databases is reflected by the convenience of modeling entities of reality in a geometric manner. For example, locations of restaurants, hotels, hospitals and so on are often represented as points in a map, while larger extents such as parks, lakes, and landscapes often as a combination of rectangles. Many functionalities of a spatial database are useful in various ways in specific contexts. For instance, in a geography information system, range search can be deployed to find all restaurants in a certain area, while nearest neighbor retrieval can discover the restaurant closest to a given address.

Today, the widespread use of search engines has made it realistic to write spatial queries in a brand new way. Conventionally, queries focus on objects' geometric properties only, such as whether a point is in a rectangle, or how close two points are from each other. We have seen some modern applications that call for the ability to select objects based on both of their geometric coordinates and their associated texts. For example, it would be fairly useful if a search engine can be used to find the nearest restaurant that offers "steak, spaghetti, and brandy" all at the same time. Note that this is not the "globally" nearest restaurant (which would have been returned by a traditional nearest neighbor query), but the nearest restaurant among only those providing all the demanded foods and drinks.

There are easy ways to support queries that combine spatial and text features. For example, for the above query, we could first fetch all the restaurants whose menus contain the set of keywords {steak, spaghetti, brandy}, and then from the retrieved restaurants, find the nearest one. Similarly, one could also do it reversely by targeting first the spatial conditions – browse all the restaurants in ascending order of their distances to the query point until encountering one whose menu has all the keywords. The major drawback of these straightforward approaches is that they will fail to provide real time answers on difficult inputs. A typical example is that the real nearest neighbor lies quite far away from the query point, while all the closer neighbors are missing at least one of the query keywords. Spatial queries with keywords have not been extensively explored. In the past years, the community has

sparked enthusiasm in studying keyword search in relational databases. It is until recently that attention was diverted to multidimensional data.

The boom of Internet has given rise to an ever increasing amount of text data associated with multiple dimensions (attributes), for example, customer reviews in shopping websites (e.g., Amazon) are always associated with attributes like price, model, and rate. A traditional OLAP data cube can be naturally extended to summarize and navigate structured data together with unstructured text data. Such a cube model is called text cube [1]. A cell in the text cube aggregates a set of documents/items with matching attribute values in a subset of dimensions. Keyword query, one of the most popular and easy-to-use ways to retrieve useful information from a collection of plain documents, is being extended to RDBMSs to retrieve information from text-rich attributes [2], [3]. Given a set of keywords, existing methods aim to find relevant items or joins of items (e.g., linked by foreign keys) that contain all or some of the keywords.

Traditional IR techniques can be used to rank documents according to the relevance. In a large text database, however, the number of relevant documents to a query could be large, and a user has to spend much time reading them. If a document is associated with attribute information, in a data cube model (a multidimensional space induced by the attributes), e.g., the text cube, a cell aggregates the documents with matching values in a subset of attributes. Such a collection of documents is associated with each cell, corresponding to an object that can be directly recommended to the user for the given query. This paper studies the problem of keyword-based top-k search in text cube, i.e., given a keyword query, find the top-k most relevant cells in a text cube. When users want to retrieve information from a text cube using keyword question, believe that relevant cells, rather than relevant documents, are preferred as the answers, because: 1) relevant cells are easy for users to browse; and 2) relevant cells provide users insights about the relationship between the values of relational attributes and the text data.

2. PROBLEM DEFINITION

1. Implement k nearest neighbor search algorithm using mat lab for given data set and to find out closest point from give query also analyze the result fetch time and accuracy.

2. Implement the Inverted index algorithm by extending point the k nearest neighbor and forming R tree to find closest point from given set of query and also analyze the result against time and result accuracy.

3. LITERATURE SURVEY

In the paper ‘fast nearest neighbor search with keywords’, there are methods like spatial index, inverted index, nearest neighbor search. The first method spatial index is used for creating indices because there is huge amount of data need to be stored for searching that data stored in the form of xml documents. If the data storage created in the form of indices then space required is less also time needed for searching the keyword is less.

Second method is inverted index. The inverted index data structure is a central component of a typical search engine indexing algorithm. A goal of a search engine performance is to optimize the speed of the query: find the documents where word occurs. Once an index is developed, which provisions lists of words per document; it is next inverted to develop an inverted index. Querying the index would require sequential iteration through each document and to each word to verify a matching document. The time memory and processing property to execute such a query are not always theoretically realistic. Instead of listing the words per article in the index, the inverted index data structure is developed which lists the documents per word. The inverted index produced, the query can now be determined by jumping to the word id in the inverted index. These were effectively inverted indexes with a small amount of supplementary explanation that required a implausible amount of attempt to produce.

Third method is nearest neighbor search. Nearest neighbor search (NNS), also identified as closeness search, parallel search is an optimization problem for finding closest points in metric spaces. In the paper ‘Efficient Keyword-Based Search for Top-K Cells in Text Cube’ methods used are inverted-index one-scan, document sorted-scan, bottom-up dynamic programming, and search-space ordering. In the top k cells, there is a searching of nearest key to the query. Cubes forms clusters of single unique group which shows its identity. Method like inverted index used for giving index rather than providing whole data which can be space consuming.

4. COMPARITIVE STUDY

Table -1: Comparison of Depth first search and inverted index

Parameters\Methods	Depth first search	Inverted index
Space	More	Less
Time complexity	O(n+m)	O(n)
Working	Slow	Efficient

Methods used before is depth first search which required more space because it needs to travel through left and right trees respectively. Rather in inverted indexing there is less space required as there is use of indices. Next is time complexity, for DFS its O(n+m) (n for edges and m for nodes). While in inverted index there is time required is O(n) that is for single index. Also DFS works slowly and inverted indexing is efficient to use.

CONCLUSIONS

By studying above methods the main motto is searching a relevant keyword with meaningful information with minimum time with valid results. I can conclude with this there should be efficient method to identify the keyword, grouping those results by using any one of clustering methods. The major drawbacks in paper [1] are if two nodes have same keyword then how to differentiate the keyword with the searched one. In paper [2] there can be come up with indexing the cluster results will be more if cluster is dynamically growing, or we can form clusters of clusters.

REFERENCES

[1]. S. Agrawal S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In Proc. Of International Conference on Data Engineering (ICDE), pages 5–16, 2002.

[2]. N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In Proc. of ACM Management of Data (SIGMOD), pages 322–331, 1990.

[3]. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In Proc. of International Conference on Data Engineering (ICDE), pages 431–440, 2002.

[4]. X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu. Spatial keyword querying. In ER, pages 16–29, 2012. X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. PVLDB, 3(1):373–384, 2010.

[5]. X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In Proc. of ACM Management of Data (SIGMOD), pages 373–384, 2011.

[6]. B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The bloomier filter: an efficient data structure for static support lookup tables. In Proc. of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 30–39, 2004.

[7]. Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In Proc. of ACM Management of Data (SIGMOD), pages 277–288, 2006.

[8]. E. Chu, A. Baid, X. Chai, A. Doan, and J. Naughton. Combining keyword search and forms for ad hoc querying of databases. In Proc. of ACM Management of Data (SIGMOD), 2009.

[9]. G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. PVLDB, 2(1):337–348, 2009.

[10]. C. Faloutsos and S. Christodoulakis. Signature files: An access method for documents and its analytical performance evaluation. ACM Transactions on Information Systems (TOIS), 2(4):267–288, 1984.

[11]. I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In Proc. of International Conference on Data Engineering (ICDE), pages 656–665, 2008.

[12]. R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatialkeyword (SK) queries in geographic information retrieval (GIR) systems. In Proc. of Scientific and Statistical Database Management (SSDBM), 2007.

[13]. G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. ACM Transactions on Database Systems (TODS), 24(2):265–318, 1999.

[14]. V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In Proc. of Very Large Data Bases (VLDB), pages 670–681, 2002.

[15]. I. Kamel and C. Faloutsos. Hilbert R-tree: An improved r-tree using fractals. In Proc. of Very Large Data Bases (VLDB), pages 500–509, 1994.