INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

APPROACHING NEAREST NEIGHBORS WITH KEYWORDS IN EFFICIENT MANNER

RASIKA A. DUGANE[1], PROF. A.B. RAUT[2]

1. CSE, Dept, HVPM, Amravati.
2. Ass. Professor CSE Dept, HVPM, Amravati.

Accepted Date: 05/03/2015; Published Date: 01/05/2015

Abstract: Nearest Neighbor Search (NNS) also known as Proximity Search or closest point search is an optimization Problem for finding closest points. Conventional spatial queries, such as range search and nearest neighbor retrieval, involve only conditions on objects' geometric 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 sample case, rather considering all the bookstalls, a query of nearest neighbor would rather ask for the bookstall that is the nearest among those whose catalog contain "Drama, scientific, comics" all at the same place**.** We develop a new access method called the spatial inverted index that extends the conventional Iindex to cope with multidimensional data, and comes with algorithms that can answer nearest neighbor queries with keywords in real time.

Keywords: Nearest Neighbor Search, Keyword Search, Spatial Index

Corresponding Author: MS. RASHMI SHRIVASTAVA

Access Online On:

www.ijpret.com

How to Cite This Article:

Rashmi Shrivastava, IJPRET, 2015; Volume 3 (9): 732-737

PAPER-QR CODE

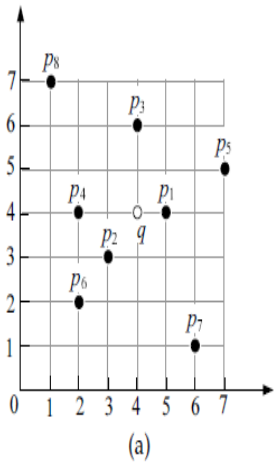
INTRODUCTION

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.

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. Current systems use ad-hoc combinations of nearest neighbor (NN) and keyword search techniques to tackle the problem. For instance, an R-Tree is used to find the nearest neighbors and for each neighbor an inverted index is used to check if the query keywords are contained.Spatial Inverted Index method successfully incorporates point coordinates into a conventional inverted index with small extra space, owing to a delicate compact storage scheme. Meanwhile, an SI-index preserves the spatial locality of data points, and comes with an R-tree built on every inverted list at little space overhead.

2. PROBLEM DEFINITION

Let P be a set of multidimensional points. As our goal is to combine keyword search with the existing location finding services on facilities such as hospitals, restaurants, hotels, etc., we will focus on dimensionality 2, but our technique can be extended to arbitrary dimensionalities with no technical obstacle.



p	W_p
p_1	$\{a, b\}$
p_2	$\{b, d\}$
p_3	$\{d\}$
p_4	$\{a, e\}$
p_5	$\{c, e\}$
p_6	$\{c, d, e\}$
p_7	$\{b, e\}$
p_8	$\{c, d\}$

(b)

Fig 1 (a) shows the locations of points and (b) gives their associated texts

We will assume that the points in P have integer coordinates, such that each coordinate ranges in [0, t], where t is a large integer. This is not as restrictive as it may seem, because even if one would like to insist on real-valued coordinates, the set of different coordinates represent able under a space limit is still finite and enumerable; therefore, we could as well convert everything to integers with proper scaling.

3. INVERTED INDEX

Inverted indexes (I-index) have proved to be an effective access method for keyword-based document retrieval. In the spatial context, nothing prevents us from treating the text description Wp of a point p as a document, and then, building an I-index. Figure 3 illustrates the index for the dataset of Figure 1.

Each word in the vocabulary has an inverted list, enumerating the ids of the points that have the word in their documents. Note that the list of each word maintains a sorted order of point ids, which provides considerable convenience in query processing by allowing an efficient merge step. For example, assume that we want to find the points that have words c and d. This is essentially to compute the intersection of the two words' inverted lists. As both lists are sorted in the same order, we can do so by merging them, whose I/O and CPU times are both linear to the total length of the lists. Recall that, in NN processing with IR2-tree, a point retrieved from the index must be verified (i.e., having its text description loaded and checked). Verification is also necessary with I-index, but for

word	*inverted list*
a	p_1 p_4
b	p_1 p_2 p_7
c	p_5 p_6 p_8
d	p_2 p_3 p_6 p_8
e	p_4 p_5 p_6 p_7

Fig 2. Example of an Inverted Index

Exactly opposite reason. For IR2-tree, verification is because we do not have the detailed texts of a point, while for I-index, it is because we do not have the coordinates. Specifically, given an NN query q with keyword set Wq, the query algorithm of I-index first retrieves (by merging) the set Pq of all points that have all the keywords of Wq , and then, performs |Pq | random I/Os to get the coordinates of each point in Pq in order to evaluate its distance to q. According to the experiments of [12], when Wq has only a single word, the performance of I-index is very bad, which is expected because everything in the inverted list of that word must be verified. Interestingly, as the size of Wq increases, the performance gap between I index and IR2-tree keeps narrowing such that I-index even starts to outperform IR2-tree at |Wq | = 4. This is not as surprising as it may seem. As |Wq | grows large, not many objects need to be verified because the number of objects carrying all the query keywords drops rapidly. On the other hand, at this point an advantage of I index starts to pay off. That is, scanning an inverted list is relatively cheap because it involves only sequential I/Os1, as opposed to the random nature of accessing the nodes of an IR2-tree.

Given the texts

T [0] = "it is what it is"

T [1] = "what is it"

T [2] = "it is a banana"

We have the following inverted file index (where the integers in the set notation brackets refer to the indexes (or keys) of the text symbols, T[0], T[1] etc.): "a" : {2}

"banana" : {2}

"is" : {0, 1, 2}

"it" : {0, 1, 2}

"what" : {0, 1}

A term search for the terms "what", "is" and "it" would give the set {0, 1} ∩{0, 1, 2} ∩{0, 1, 2} = {0, 1} With the same texts, we get the following full inverted index, where the pairs are document numbers and local word numbers. Like the document numbers, local word numbers also begin with zero. So, "banana": {(2, 3)} means the word "banana" is in the third document (T[2]), and it is the fourth word in that document (position 3).

"a" : {(2, 2)}

"banana" : {(2, 3)}

"is" : {(0, 1), (0, 4), (1, 1), (2, 1)}

"it" : {(0, 0), (0, 3), (1, 2), (2, 0)}

"what" : {(0, 2), (1, 0)}

If we run a phrase search for "what is it" we get hits for all the words in both document 0 and 1. But the terms occur consecutively only in document 1.Signature file in general refers to a hashing-based framework, whose instantiation is known as superimposed coding (SC), which is shown to be more effective than other instantiations. It is designed to perform membership tests: determine whether a query word w exists in a set W of words. SC is conservative, in the sense that if it says “no”, then w is definitely not in W. If, on the other hand, SC returns “yes”, the true answer can be either way, in which case the whole W must be scanned to avoid a falsehit. In the context, SC works in the same way as the classic technique of bloom filter. In preprocessing, it builds a bit signature of length l from W by hashing each word in W to a string of l bits, and then taking the disjunction of all bit strings. To illustrate, denote by h(w) the bit string of a word w. First, all the l bits of h(w) are initialized to 0. Then, SC repeats the following m times: randomly choose a bit and set it to 1. Very importantly, randomization must use w a sits seed to ensure that the same w always ends up with an identical h(w). A spatial keyword query consists of a query area and a set of keywords shown in below figure. 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. Furthermore, the m choices are mutually independent, and may even

happen to be the same bit. The concrete values of l and m affect the space cost and false hit probability, as will be discussed later. Gives an example to illustrate the above process, assuming l = 5 and m = 2. For example, in the bit string h (a) of a, the 3rd and 5th (counting from left) bits are set to 1. As mentioned earlier, the bit signature of a set W of words simply ORs the bit strings of all the members of W. For instance, the signature of a set {a, b} equals 01101, while that of {b, d} equals 01111.

word	*hashed bit string*
a	00101
b	01001
c	00011
d	00110
e	10010

Fig 3. Example of bit string Computation with l=5, m=2

4. CONCLUSION

We have seen many applications vocation for a search engine that's ready to expeditiously support novel forms of abstraction queries that area unit integrated with keyword search. The existing solutions to such queries either incur prohibitive space consumption or are unable to give real time answers. During this paper, we've got remedied the state of affairs by developing associate degree access methodology called the abstraction inverted index (SI-index). Not solely that then SI-index is fairly area economical, however conjointly it's the ability to perform keyword-augmented nearest neighbor research in time that's at the order of dozens of milliseconds.

5. 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.

5. X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. PVLDB, 3(1):373–384, 2010.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

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

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

16. 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.

17. J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. In Proc. of ACM Management of Data (SIGMOD), pages 349–360, 2011.

18. S. Stiassny. mathematical analysis of various superimposed coding methods. Am. Doc., 11(2):155–169, 1960.

19. J. S. Vitter. Algorithms and data structures for external memory. Foundation and Trends in Theoretical Computer Science, 2(4):305–474,2006.

20. D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In Proc. of International Conference on Data Engineering (ICDE), pages 688–699, 2009.

21. Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma. Hybrid index structures for location-based web search. In Proc. of Conference on Information and Knowledge Management (CIKM), pages 155–162, 2005.