International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Quality Preference Spatial Approximate String Search

Joslin T.J

Department of Computer Science and Engineering, Malabar College of Engineering and Technology, Thrissur, India

Abstract: ***This work deals with the approximate string search in large spatial databases. A spatial preference query ranks objects based on the qualities of features in their spatial neighborhood. Specifically, investigate range queries augmented with a string similarity search predicate in road networks. And dub this query the spatial approximate string (SAS) query. The min-wise signature for an index node u keeps a concise representation of the union of q-grams from strings under the sub-tree of u. Analyze the pruning functionality of such signatures based on the set resemblance between the query string and the q-grams from the sub-trees of index nodes. For queries on road networks, using a novel exact method, RSASSOL, which significantly outperforms the baseline algorithm in practice. The RSASSOL combines the q-gram based inverted lists and the reference nodes based pruning. Extensive experiments on large real data sets demonstrate the efficiency and effectiveness of our approaches.***

Keywords: approximate string search, range query, road network, spatial databases, quality preference search

1. Introduction

Keyword search over a large amount of data is an important operation in a wide range of domains. In practice, keyword search for retrieving approximate string matches is required [4], [6], [9], [11], [17], [18], [22]. Since exact match is a special case of approximate string match, it is clear that keyword search by approximate string matches has a much larger pool of applications. Approximate string search is necessary when users have a fuzzy search condition, or a spelling error when submitting the query, or the strings in the database contain some degree of uncertainty or error. In the context of spatial databases, approximate string search could be combined with any type of spatial queries. In this work, focus on range queries and dub such queries as Spatial Approximate String (SAS) queries. Denote SAS queries in Euclidean space as (ESAS) queries. Similarly, extends SAS queries to road networks (referred as RSAS queries).

Object ranking is a popular retrieval task in various applications. In spatial databases, ranking is often associated to nearest neighbor (NN) retrieval. Given a query location, we are interested in retrieving the set of nearest objects to it that qualify a condition (e.g., restaurants). Assuming that the set of interesting objects is indexed by an R-tree, we can apply distance bounds and traverse the index in a branch-and-bound fashion to obtain the answer. 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.). Here study an interesting type of preference based approximate string queries, which select the best spatial location with respect to the quality of facilities in its spatial neighborhood.

A straightforward solution to any SAS query is to use any existing techniques for answering the spatial component of an SAS query and verify the approximate string match predicate either in post-processing or on the intermediate results of the spatial search. We refer to them as the spatial solution. For RSAS queries, the baseline spatial solution is based on the Dijkstra's algorithm. Given a query point q, the query range radius r, and a string predicate, we expand from q on the road network using the Dijkstra algorithm until we reach the points distance r away from q and verify the string predicate either in a post-processing step or on the intermediate results of the expansion. We denote this approach as the Dijkstra solution. Its performance degrades quickly when the query range enlarges and/or the data on the network increases. This motivates us to find a novel method to avoid the unnecessary road network expansions, by combining the prunings from both the spatial and the string predicates simultaneously.

A straightforward solution in both ESAS and RSAS queries is to build a string matching index and evaluate only the string predicate, completely ignoring the spatial component of the query. After all similar strings are retrieved, points that do not satisfy the spatial predicate are pruned in a post-processing step. We dub this the string solution. First, the string solution suffers the same scalability and performance issues (by ignoring one dimension of the search) as the spatial solution. Second, we want to enable the efficient processing of standard spatial queries (such as nearest neighbor queries, etc.) while being able to answer SAS queries additionally in existing spatial databases, i.e., a spatial-oriented solution is preferred in practice in spatial databases.

Another interesting problem is the selectivity estimation for SAS queries. The goal is to accurately estimate the size of the results for an SAS query with cost significantly smaller than that of actually executing the query itself. Selectivity estimation is very important for query optimization purposes and data analysis and has been studied extensively in database research for a variety of approximate string queries and spatial range queries [1].

Computing edit distance exactly is a costly operation. Several techniques have been proposed for identifying candidate strings within a small edit distance from a query string fast. All of them are based on q-grams and a q-gram counting argument. For a string σ, its q-grams are produced by sliding a window of length q over the characters of σ. To deal with the special case at the beginning and the end of σ,

that have fewer than q characters, one may introduce special characters, such as "#" and "$", which are not in _. This helps conceptually extend σ by prefixing it with q − 1 occurrences of "#" and suffixing it with q − 1 occurrences of "$". Hence, each q-gram for the string σ has exactly q characters. These q-grams are used for searching.

2. Problem Formulation

Formally, a spatial database P contains points with strings. Each point in P may be associated with one or more strings. For brevity and without loss of generality, here we assume that each point in P has one associated string. A data set P with N points is the following set:$\{(p_1, \sigma_1), . , (p_N, \sigma_N)\}$. Different points may contain duplicate strings. In the sequel, when the context is clear, we simply use a point p_i to denote both its geometric coordinates and its associated string.

A spatial approximate string (SAS) query Q consists of two parts: the spatial predicate Q_r and the string predicate Q_s. In this paper concentrating on using range queries as the spatial predicate. In the Euclidean space, Q_r is defined by a query rectangle r; in road networks, it is specified by a query point q and a radius r. In both cases, the string predicate Q_s is defined by a string σ and an edit distance threshold τ.

Let the set $A_r = \{p_x | p_x \in P \wedge p_x$ is contained in r$\}$ if P is in the Euclidean space; or, $A_r = \{p_x | p_x \in P \wedge d(q, p_x) \leq r\}$ if P is in a road network and d(q, p) is the network distance between two points q and p.

Let the set $A_s = \{p_x | p_x \in P \wedge \varepsilon(\sigma, \sigma_x) \leq \tau\}$.

We define the SAS query as follows: An SAS query Q: (Q_r, Q_s) retrieves the set of points $A = A_r \cap A_s$. The problem of selectivity estimation for an SAS query Q is to efficient and accurately estimate the size |A| of the query answer. We use σ_p to denote the associated string of a point p.

3. Proposed Work

The main objective of the paper is to rank the objects based on the features in their spatial neighbourhood. In this paper, a study of an interesting type of preference based spatial approximate string queries are made, which select the best spatial location with respect to the quality of facilities in its spatial neighborhood. Spatial database systems manage large collection of geographic entities, which apart from spatial attributes contain non-spatial values like size, type, price etc. In this paper, a study of an interesting type of preference queries is made, which selects the best spatial object with respect to the quality of features in its spatial neighbourhood.

Given a set D of interesting objects, a 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 (facilities or services) in its spatial neighbourhood. The user wishes to find a food facility that may be hotel or restaurants also with different types of transport facility can input these purposes as spatial query. For each hotel 'P' will defined in terms of (i) the maximum quality for each feature in the neighbourhood region of the particular position 'P' and (ii) the aggregation of those user requirements.

Here propose, (i)spatial ranking, which ranks the objects according to their distance from a reference point (ii)non-spatial ranking, which ranks the objects by aggregating all the non-spatial values(size, price, type), (iii) neighbour retrieval, In spatial database, ranking is often associated to nearest neighbour (NN) retrieval. Given a query location, we are interested in retrieving the set of nearest objects to it that qualify a condition (example: restaurants).Assuming that the set of interesting objects is indexed by an R-tree, then apply distance bounds and traverse the index in a branch and bound fashion to obtain the answer, (iv)spatial query evaluation on R-tree, which is the most popular spatial access method ,which indexes Minimum Bounding Rectangles (MBR'S) of objects. R-tree can efficiently process main spatial query types, including spatial range queries, nearest neighbour queries, and spatial joins. The spatial preference query integrates these four types of ranking in an intuitive way.

In this case, since the locations of points are constrained by the road network and represented by the edge holding the point and the distance offset to the edge end, the MHR-tree is not applicable in this context. In order to handle large scale datasets, we adopt a disk-based road network storage framework and develop external-memory algorithms. Partition a road network G = {V,E} into m edge-disjoint subgraphs G1,G2, . . . ,Gm, where m is a user parameter, and build one string index (FilterTree) for strings in each subgraph. We also select a small subset V_R of nodes from V as reference nodes: they are used to prune candidate points/nodes whose distances to the query point q are out of the query range r.

Conceptually, our RSAS query framework consists of five steps .Given a query,

step 1.find all subgraphs that intersect with the query range.

step 2.use the FilterTrees of these subgraphs to retrieve the points whose strings are potentially similar to the query string.

step 3. prune away some of these candidate points by calculating the lower and upper bounds of their distances to the query point, using V_R.

step 4. prune away some candidate points using the exact edit distance between the query string and strings of remaining candidates

step 5.for the remaining candidate points, we check their exact distances to the query point and return those with distances within r.

We dub this algorithm RSASSOL. We use $d(o_1, o_2)$ to denote the network distance of two objects o_1, o_2 (where an object can be a network vertex, or a point on the network).

Algorithm: RSASSOL (network G, Qr = (q, r), Qs =(σ , τ))

1 Find the set X of ids of all subgraphs intersecting (q, r);
2 Set $A = \varnothing$, $A_C = \varnothing$

3 for *each subgraph id* $i \in X$ **do**
4 Find all point ids in G_C whose associated strings σ' may satisfy $\varepsilon(\sigma', \sigma) \leq \tau$ using FilterTree$_i$, and insert them into A_C
5 for *every point* $p_i \in A_C$ **do**
6 calculate $d^+(p_i, q)$ and $d^-(p_i, q)$
7 if $d^+(p_i, q) \leq r$ **then**
8 if $\varepsilon(\sigma_i, \sigma) \leq \tau$ **then**
9 move p_i from A_c to A
10 else
11 delete p_i from A_c
12 else
13 if $d^-(p_i, q) > r$ **then**
14 delete p_i from A_c
15 for *every point* $p_i \in A_c$ **do**
16 if $\varepsilon(\sigma_i, \sigma) > \tau$ **then**
17 delete p_i from A_c;
18 Use algorithm to find all points p's in A_c such that $d(p, q) \leq r$, push them to A
19 Return A.

3.1 Query processing

First, we find all subgraphs that intersect with the query range. We employ the Dijkstra's algorithm to traverse nodes in G (note that we ignore points on G), starting from the query point q. Whenever this traversal meets the first node of a new subgraph, we examine that subgraph for further exploration. The algorithm terminates when we reach the boundary of the query range (defined by the distance r to q). For each subgraph G_i to be examined, we use the approximate string search over G_i's FilterTree as the next pruning step, to find points from G_i that may share similar strings to the query string. Then we further prune the candidate points using the spatial predicate, by computing lower and upper bounds on their distances to q using V_R. Given a candidate point p on an edge $e = (n_i, n_j)$, the shortest path from p to a reference node n_r must pass through either n_i or n_j. Thus, the network distance $d(p, n_r) = \min(d(p, n_i) + d(n_i, n_r), d(p, nj) + d(n_j, n_r))$. We compute $d(p, n_r)$ on the fly rather than explicitly storing the distance between a point and a reference node since the number of points is much larger than the number of the nodes in G. By doing so, we avoid significant space blowup. We can also compute $d(q, n_r)$ in a similar way. Given $d(p, n_r)$ and $d(q, n_r)$ for every $n_r \in V_R$, we then obtain the distance lower and upper bounds between p and q using the triangle inequality. Each reference node yields such a pair of lower and upper bounds. We take the maximum (minimum) value from the lower (upper) bounds of all reference nodes as the final lower (upper) bound of $d(p, q)$, denoted as $d^-(p, q)$ and $d^+(p, q)$ respectively. If $d^+(p, q) \leq r$, we know for sure p satisfies the spatial predicate and we only need to check the exact edit distance as the last measure; if $d^-(p, q) > r$, we can safely remove p; otherwise, we need to check both the exact edit distance and compute $d(p, q)$ to complete the verification on p. After the pruning by $d^-(p, q)$ and $d^+(p, q)$, we compute the exact edit distances on the remaining candidate points in A_C and prune away points whose edit distances to the query string σ are larger than τ. Note that for a point p satisfying $d^+(p, q) \leq r$ and the exact edit distance threshold, it has already been removed from A_C and pushed to A before this step. For all other remaining candidates A_C, we only need to compute the exact network distances between them and q to complete the algorithm. The naive solution is to apply the algorithm for every $p \in A_C$ and q to find their shortest path. However, this can be prohibitive when $|A_C|$ is still large. Next, we introduce an improvement to the shortest path algorithm, which computes multiple shortest paths, within the query range, simultaneously at once between a single source point s and multiple destination points $\{t_1, \ldots, t_m\}$.

3.2 Selectivity estimation of RSAS queries

The selectivity estimation of range queries on road networks is a much harder problem than its counterpart in the Euclidean space. Several methods were proposed in [15], [25]. However, they are only able to estimate the number of nodes and edges in the range. None can be efficiently adapted to estimate the number of points in the range. One naive solution is to treat points as nodes in the network by introducing more edges. This clearly increases the space consumption significantly (and affects the efficiency) since the number of points is typically much larger than the number of existing nodes.

4. Literature Survey

The most popular spatial access method is the R-tree [3], which indexes minimum bounding rectangles (MBRs) of objects. R-trees can efficiently process main spatial query types, including spatial range queries, nearest neighbor queries, and spatial joins. Given a spatial region W, a spatial range query retrieves from D the objects that intersect W. The IR2-tree was proposed in [13] to perform exact keyword search with kNN queries in spatial databases.

The IR2-tree cannot support spatial approximate string searches, neither their selectivity estimation was addressed therein. Two other relevant studies appear in [7], [12] where ranking queries that combine both the spatial and text relevance to the query object were investigated. Another related work appears in [2] where the LBAKtree was proposed to answer location-based approximate keyword queries which are similar to our definition of spatial approximate string queries in the Euclidean space. The basic idea in the LBAK-tree is to augment a tree-based spatial index (such as an R-tree) with q-grams of sub tree nodes to support edit-distance based approximate string/keyword searches. The LBAK-tree returns exact answers for the ESAS queries, and the MHR-tree returns approximate answers. That said, for ESAS queries, the LBAK-tree should be adopted when exact answers are required; when space consumption must be small and approximate solutions are acceptable, the MHR-tree is the candidate.

RSAS queries and selectivity estimation of SAS queries have been explored before in the literature [3]. Approximate

string search alone has been extensively studied in the literature [4], [5], [6], [8], [9], [11], [14], [17], [18], [19],[21], [22], [23]. These works generally assume a similarity function to quantify the closeness between two strings. There are a variety of these functions such as edit distance and Jaccard. Many approaches leverage the concept of q-grams. q-gram based pruning for edit distance that has been used extensively in the field [14], [19], [21], [24]. Improvements to the q-grams based pruning has also been proposed, such as vgrams[22], where instead of having a fixed length for all grams variable length grams were introduced, or the two-level q-gram inverted index [16].

The problem in our paper is different: we want to search in a collection (unordered set) of strings to find those similar to a single query string ("selection query"). Our effort for selectivity estimation in ESAS queries is also related to selectivity estimation for spatial range queries [1],[15]. Several methods for selectivity estimation for range queries on road networks were proposed in [3], [20]. However, they are only able to estimate the number of nodes and edges in the range (not the number of points residing on the network in the range). Here we extend these techniques for points and combine them with string predicate. Also, they deal with a single feature dataset whereas our queries consider multiple feature datasets.

5. Conclusion

This paper presents a comprehensive study for spatial approximate string queries in both the Euclidean space and road networks. We use the edit distance as the similarity measurement for the string predicate and focus on the range queries as the spatial predicate. We also address the problem of query selectivity estimation for queries in the Euclidean space. We study an interesting type of preference based approximate string queries, which select the best spatial location with respect to the quality of facilities in its spatial neighborhood.

6. Future Work

Future work include examining spatial approximate sub-string queries, designing methods that are more update friendly and solving the selectivity estimation problem for RSAS queries.

7. Acknowledgement

I would like to express my immense gratitude to Asst. Professor Ms. Mrudula K..P, for all her guidance and encouragement throughout this research work. She has been always there providing sufficient support with her expertise in this area. I would also like to thank my examiners for their precious comments and suggestions.

References

[1] S. Acharya, V. Poosala, and S. Ramaswamy. Selectivity estimation inspatial databases. In *SIGMOD*, pages 13–24, 1999.

[2] S. Alsubaiee, A. Behm, and C. Li. Supporting location-basedapproximate-keyword queries. In *GIS*, pages 61–70, 2010.

[3] Feifei Li, Bin Yao, Mingwang Tang, MariosHadjieleftheriou. Spatial Approximate String Search. In IEEE Transactions on volume 25 issue 6, 2013.

[4] Arasu, S. Chaudhuri, K. Ganjam, and R. Kaushik. Incorporating string transformations in record matching. In *SIGMOD*, pages 1231– 1234, 2008.

[5] G. Li, J. Feng, and C. Li. Supporting search-as-you-type using sql in databases. *TKDE*, To Appear, 2011.

[6] S. Sahinalp, M. Tasan, J. Macker, and Z. Ozsoyoglu. Distance based indexing for string proximity search. In *ICDE*, pages 125–136, 2003.

[7] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. *Proc. VLDB Endow.*, 3:373–384, 2010.

[8] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin. An efficient filterfor approximate membership checking. In *SIGMOD*, pages 805–818, 2008.

[9] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust andefficient fuzzy match for online data cleaning. In *SIGMOD*, pages 313–324, 2003.

[10] S. Chaudhuri, V. Ganti, and L. Gravano. Selectivity estimation for string predicates: Overcoming the underestimation problem. In *ICDE*, pages 227–238, 2004.

[11] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator forsimilarity joins in data cleaning. In *ICDE*, pages 5–16, 2006.

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

[13] D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatialdatabases. In *ICDE*, pages 656–665, 2008.

[14] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB*, pages 491–500, 2001.

[15] D. Gunopulos, G. Kollios, J. Tsotras, and C. Domeniconi. Selectivityestimators for multidimensional range queries over real attributes. *TheVLDB Journal*, 14(2):137–154, 2005.

[16] M.-S. Kim, K.-Y. Whang, J.-G. Lee, and M.-J. Lee. n-gram/2l: a spaceand time efficient two-level n-gram inverted index structure. In *VLDB*, pages 325–336, 2005.

[17] N. Koudas, A. Marathe, and D. Srivastava. Flexible string matching against large databases in practice. In *VLDB*, pages 1078–1086, 2004.

[18] C. Li, J. Lu, and Y. Lu. Efficient merging and filtering algorithms forapproximate string searches. In *ICDE*, pages 257–266, 2008.

[19] E. Sutinen and J. Tarhio. On using q-gram locations in approximate string matching. In *ESA*, pages 327–340, 1995.

[20] E. Tiakas, A. N. Papadopoulos, A. Nanopoulos, and Y. Manolopoulos. Node and edge selectivity estimation for range queries in spatial networks. *Inf. Syst.*, 34:328–352, 2009.

[21] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theor. Comput. Sci.*, 92(1):191–211, 1992.
[22] X. Yang, B. Wang, and C. Li. Cost-based variable-length-gram selectionfor string collections to support approximate queries efficiently. In*SIGMOD*, pages 353–364, 2008.
[23] H. V. Jagadish, R. T. Ng, and D. Srivastava. Substring selectivity estimation. In *PODS*, pages 249–260, 1999.
[24] L. Jin and C. Li. Selectivity estimation for fuzzy string predicates in large data sets. In *VLDB*, pages 397–408, 2005.
[25] L. Jin, C. Li, and R. Vernica. Sepia: estimating selectivities ofapproximate string predicates in large databases. *The VLDB Journal*, 17(5):1213–1229, 2008.
[26] D. Zhang, B. C. Ooi, and A. Tung. Locating mapped resources in web 2.0. In *ICDE*, pages 521–532, 2010.

Author Profile

Joslin T.J was born in Thrissur, Kerala in 1990. She completed bachelor degree in Information Technology from Jyothi Engineering College, Thrissur in the academic year 2008-2012. She is pursuing Master degree in computer science and engineering in Calicut University, Malabar College of Engineering and Technology, Thrissur in the academic year 2012-2014.