MHR-Tree Based Spatial Approximate String Search

Ms. Ashvini Shingare[1] Ms. Nitu Pariyal[2]
[1]Student [2]Assistant Professor
[1,2]MGM College of engineering, Nanded, Maharashtra, India

Abstract— Keyword search in a large amount of data is very essential factor in data mining. This project focuses on keyword search, called as spatial approximate string search (SASS).A main limitation of the approximate search is that it only supports exact keyword search in a large amount of data. Keyword search for retrieving estimated string matches is required. 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. In this work, we focus on range queries and name such queries as Spatial Approximate String (SAS) queries. (i.e. ESAS, RSAS). We have to efficiently find that string in the collection that is similar to a query string. Such type of query is called an "approximate string search".

Key words: MHR, Spatial Approximate String (SAS), String Search

I. Introduction

Suppose given a collection of strings, we have to efficiently find that string in the collection that is similar to a query string. Such type of query is called an "approximate string search." This kind of problem is of great interests to a variety of application like.

A. *Spell Checker*

A spell checker needs to find words similar to the words by searching in its dictionary those are possibly mistyped words. Thus, for each word that is not in the dictionary, we need to find potentially matched candidates to recommend.

B. *Data Cleaning*

Information from different data sources often has different inconsistencies. There are little different formats for the same real-world entity could be represented in. There could also be errors in the original data introduced in the data collection process. Thus, data cleaning needs to find from a collection of entities those similar to a given entity.

Main motivation of my research is that:

1) Exact search:

In exact search, we will type it any word means it will search in spatial database for exact word in present or not, if it is present means, it will return the exact information, it is based on word while user can type.

2) Distance:

In this module, we will calculate the distance between two points in different units in exact manner by using road network.

We have studied many research articles and methods presented over the problem of approximate string search, however most this methods suffered from the accuracy and speed. There are two primary problems in research on approximate string search :(1) How to build a model that can archive both high accuracy and efficiency, and(2) How to develop a data structure and algorithm that can facilitate efficient retrieval of the top k candidates.

The objective of spatial approximate string search is shown in figure1.2, is to find the approximate string search in large spatial databases. A key issue of spatial approximate string queries is defining similarity between to string by using following two spatial approximate queries i.e. ESAS and RSAS query.

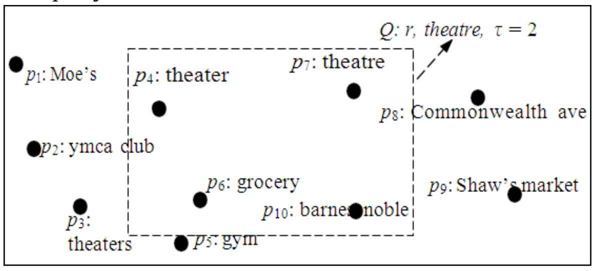


Fig. 1: Example of ESAS query

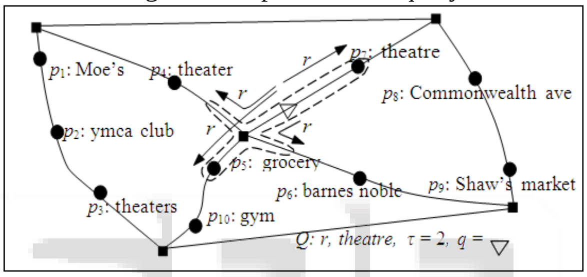


Fig. 2: Example of RSAS query

An example for the approximate string match range query is shown in Figure 1& 2, enhancing a common scenario in location- based services: find all objects within a spatial range r that have a description that is similar to 'theatre'. Where the distance between two points is the length of their shortest path.

The edit distance metric is often adopted. Particularly, given strings σ1 and σ2, the edit distance between σ1 and σ2, denoted as ε(σ1, σ2), is defined as the minimum number of edit operations required to transform one string into the other. The edit operations refer to an insertion, deletion, or substitution of a single character. Clearly, ε is symmetric, i.e., ε(σ1, σ2) = ε(σ2, σ1). For example, let σ1 ='theatre' and σ2 ='theater', then ε(σ1, σ2) = 2, by substituting the first 'r' with 'e' and the second 'e' with 'r'. We do not consider the generalized edit distance in which the transposition operator (i.e., swapping two characters in a string while keeping others fixed) is also included. The standard method for computing ε(σ1, σ2) is a dynamic programming formulation. For two strings with lengths n1 and n2 respectively, it has a complexity of O(n1n2). That said, given the edit distance threshold τ = 2 the answer to the ESAS query in Figure 3.1 is {p4, p7}.

II. Problem Formulation

Formally, a spatial database P contains points with strings. Each point in P may be connected with one or more strings. We simply assume that each point in P has ω number of strings. Hence, a data set P with N points is the following set: {(p1, σ1,1, . . . , σ1,ω), . . . , (pN , σN,1, . . . , σN,ω)}. Different points may include duplicate strings. When the context is clear, we simply use a point pi to denote both its geometric coordinates and its associated strings.

A spatial approximate string (ESAS) query Q consists of two parts: the spatial query Qr and the string query Qs. The spatial query specifies spatial predicate and predicate parameters. In this paper we think on the classical range queries as the spatial predicates. In the Euclidean space, Qr is simply defined by query rectangle r, which is a rectangle for an ESAS query. The string query is defined by one or more query strings and their associated edit distance thresholds. The approximate string search has been studied separately in the database society. The main challenge for ESAS problem is that the basic solution of simply integrating approximate edit distance assessment techniques into a normal R-tree is expensive and im-practical. We propose the MHR-tree in that takes into account the probable pruning capability provided by the string match predicate and the spatial predicate simultaneously. The MHR-tree is based on the R∗-tree. In each node, it stores the signature of the string information of its children nodes. Before exploring how to build the MHR-tree and answer ESAS queries with MHR-tree, we will initiate the min-wise signature, which is used to summarize the string information.

SAS queries exist on road networks as well. In Figure 2, given a query point and a network distance on a road network, we want to retrieve all objects within that range and with the string similar to “theatre”. Here, the network distance between two points is defined as the length of the shortest path. We define those queries as Spatial Approximate String queries in Road Networks (RSAS), respectively.

Formally, We have a road network t=(V, E), Where V(E) denotes the set of nodes(edges). Each point pi ∈ P resides on an edge (ni, nj) ∈ E, where ni < nj. We locate pi by (ni, nj) and the offset to ni. Therefore, the coordinates of points in O are not necessarily needed in the RSAS queries. For the RSAS queries, we have the similar definitions ESAS query that the distance in measured by the network distance instead of Euclidean distance. Another 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.

III. THE ESAS QUERIES

A. *Edit distance pruning*

Computing edit distance accurately 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 created by sliding a window of length q over the characters of σ. To deal with the special case at the initiation and the end of σ, that have fewer than q characters, one may introduce special characters, such as “#” and “$”, which are not in ∑. This helps abstractly enlarge σ by prefixing it with q − 1 occurrences of “#” and suffixing it with q − 1 occurrences of “$”. Hence, each q-gram for the string σ has exactly q characters.

Example 1: The q-grams of length 2 for the string theatre are {#t, th, he, ea, at, tr, re, e$}. The q-grams of length 2 for the string theater are {#t, th, he, ea, at, te, er, r$}.

Let, For strings σ1 and σ2 of length |σ1| and |σ2|, if ε(σ1, σ2) = τ, then

|gσ1∩gσ2 | ≥ max(|σ1|, |σ2|)−1 − (τ − 1) * q.

Example: q=2, τ=2

σ1=theatre, σ2=theater

gσ1{#t,th,he,ea,at,tr,re,e$}

gσ2{#t,th,he,ea,at,te,er,r$}

B. *Min-wise signature*

The MinHash is that min-wise independent permutations are a technique for quickly estimating how similar two sets are. One need the hash function h to define a random permutation on n elements, where n is the total number of distinct elements in the union of all of the sets to be compared. But because there are n! Different permutations, it would require Ω(n log n) bits just to specify a truly random permutation, an infeasible large number for even moderate values of n.

The min-wise independent families of permutations were first introduced in many research papers. A family of min-wise independent permutations F must satisfy the following equations. Let the universe of elements be U, for any set X that is defined by fundamentals from U, i.e., X ⊆ U, for any x ∈ X, when π is chosen at random in F we have:

$$\mathrm{pr}(\min\{\pi(X)\}) = \pi(x)) = \frac{1}{|X|}$$

In the above formula [3.1], π(X) produces a permutation of X and π(x) is the location value of x in the resulted permutation and min{π(A)} = min{π(x)|x ∈ A}. In other words, all fundaments of any fixed set X have an equal probability to be the minimum value for set X under permutation π from a min-wise independent family of permutations. The min-wise independent family of permutations is useful for estimating set resemblance. The set resemblance of two sets A and B is defined as:

$$\rho(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

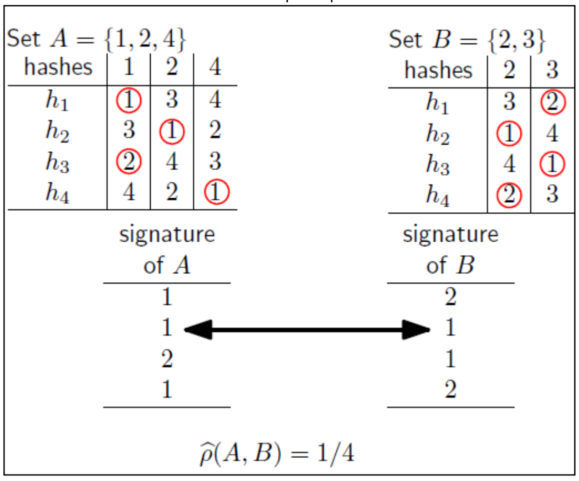


Fig. 3: Estimating set resemblance

IV. THE R-TREE

R-trees are tree data structures used for spatial access methods, i.e., for indexing multi-dimensional information such as geographical coordinates, rectangles or polygons. The R-tree was proposed by Antonin Guttman in 1984 and has found significant use in both theoretical and applied contexts. A common real-world usage for an R-tree might be to store spatial objects such as restaurant locations or the polygons that typical maps are made of: streets, buildings, outlines of lakes, coastlines, etc. and then find answers quickly to queries such as "Find all museums within 2 km of my current location", "retrieve all road segments within 2 km of my location" to display them in a navigation system or "find the nearest gas station" although not taking roads into account. The R-tree can also accelerate nearest neighbor search .

A. *Properties of R-tree:*

Let M be the maximum number of entries that will fit in one node. Let m ≤ M/2 be a parameter specifying the minimum number of entries in one node.
Then an R-Tree must satisfy the following properties:

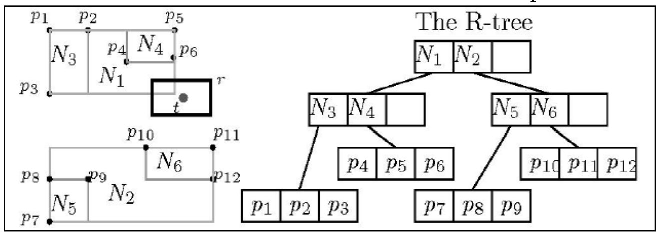
1) Every leaf node contains between m and M index records, unless it is the root.
2) For each index-record Entry (I, tuple-identifier) in a leaf node, I is the MBR that spatially contains the n-dimensional data object represented by the tuple-identifier.
3) Every non-leaf node has between m and M children,
4) unless it is the root.
5) For each Entry (I, child-pointer) in a non-leaf node, I is the MBR that spatially contains the regions in the child node.
6) The root has two children unless it is a leaf.
7) All leaves appear on the same level.

V. THE MHR-TREE

Suppose the disk block size is B. The R-tree and its variants (R∗-tree in particular) share a related principle. They first group ≤ B points that are in spatial immediacy with each other into a minimum bounding rectangle (MBR). These points will be stored in a leaf node. The process is repeat d until all points in P are assigned into MBRs and the leaf level of the tree is completed. The resulting leaf node MBRs are then further grouped together recursively till there is only one MBR left. Each node u in the R-tree is related with the MBR enclosing all the points stored in its sub-tree, denoted by MBR(u). Each internal node also stores the MBRs of all its children.
The query algorithm for the MHR tree for finding the similarity and construct the tree consists of the following steps:

– Step 1: Let database be a FIFO queue initialized to ∅ (L=∅), and dataset also initialized to ∅ (A = ∅);
– Step 2: Let minimum bounding rectangle (MBR) be the root node of R(tree) and Insert Minimum bounding Rectangle into database.
– Step 3: When database is not null then, Let MBR is the head element of the database then pop out MBR.
– Step 4: If MBR is a leaf node then every point (p ∈ up) All points are assigned to MBR
– Step 5: If P is contained in R then (Points are associated with root tree)
– Step 6: Find the signature of node by using this formula
– if | gp ∩gσ| ≥ max(|σp|, |σ|)−1−(τ −1) ∗ q
– Step 7: if length of string signature less than threshold value (ε(σp , σ) < τ) then inserts point into dataset else every child entry into MBR
– Step 8: If root and MBR intersect then
– Calculate s (g = gwi ∪ gσ) based on s (gwi), s(gσ) ;
– Step 9: Then read child node and insert into database

This procedure is recursively applied in a bottom-up fashion until the root node of the R-tree has been processed.


p_1 p_2 p_5 p_3 p_4 p_6 N_3 N_4 N_1 r t
p_{10} p_{11} p_8 p_9 p_{12} p_7 N_5 N_6 N_2
The R-tree
N_1 N_2
N_3 N_4 N_5 N_6
p_4 p_5 p_6 p_{10} p_{11} p_{12}
p_1 p_2 p_3 p_7 p_8 p_9
Fig. 4: MHR-tree: Basic idea

VI. THE RSAS QUERIES

A. *The RSASsol method*

We partition a road network G = {V, E} into m edge-disjoint sub graphs G1, G2, . . . , Gm, where m is a user parameter, and build one string index Filter Tree for strings in each Sub graph. We also select a small subset VR of nodes from V as reference nodes.

1) *Reference node:*

They are used to prune candidate points or nodes whose distances to the query point q are out of the query range r.

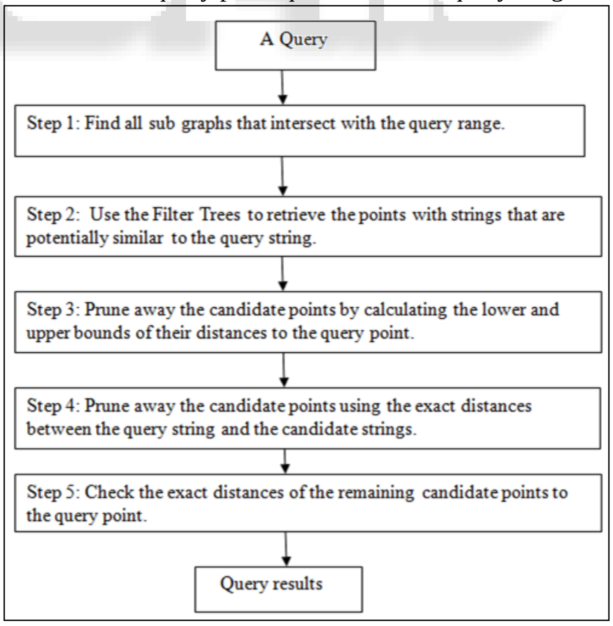

A Query
Step 1: Find all sub graphs that intersect with the query range.
Step 2: Use the Filter Trees to retrieve the points with strings that are potentially similar to the query string.
Step 3: Prune away the candidate points by calculating the lower and upper bounds of their distances to the query point.
Step 4: Prune away the candidate points using the exact distances between the query string and the candidate strings.
Step 5: Check the exact distances of the remaining candidate points to the query point.
Query results

Fig. 5: Overview of the RSASSOL algorithm

Conceptually, our RSAS query framework consists of five steps.

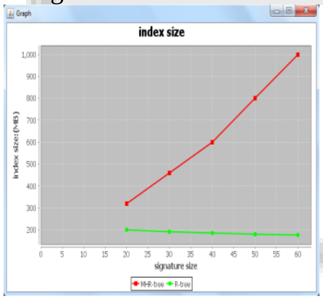
– Given a query, we first find all sub graphs that intersect with the query range.

– Next, we use the Filter Trees of these sub graphs to retrieve the points whose strings are potentially similar to the query string.
– In the third step, we prune away some of these candidate points by calculating the lower and upper bounds of their distances to the query point, using reference node(VR).
– The fourth step is to further prune away some candidate points using the exact edit distance between the query string and strings of remaining candidates. After this step, the string predicate has been fully explored.
– In the final step, for the remaining candidate points, we check their exact distances to the query point and return those with distances within r.

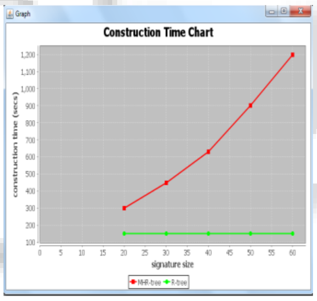
We name this algorithm RSASSol and details of this algorithm. We use d(o1, o2) to denote the network distance of two objects o1, o2 (where an object can be a network vertex, or a point on the network).

VII. Experimental Results

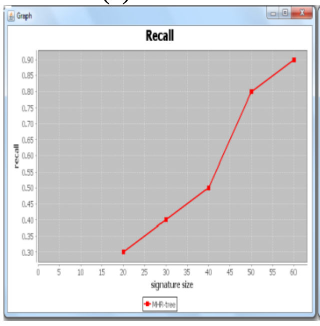
For ESAS and RSAS queries, the real datasets were obtained from the open street map. Each dataset contains the streets for a state in the USA. Each point has its longitude and latitude coordinates and several string attributes. We combine the state, country, town names for a point as its associated string. For our experiments, we have used the Texas (TX) and California (CA) datasets, since they are the largest few in size among different states.



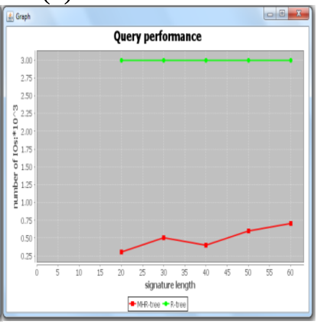
(a)Index size



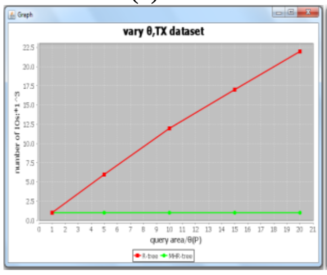
(b) Construction time



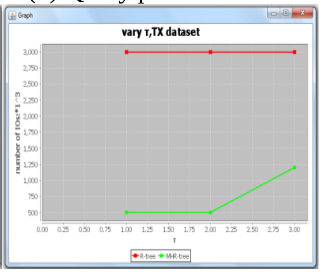
(c) Recall



(d) Query performance



(e) Vary Ø, TX dataset



(f) Vary T, TX dataset

Fig. 6: Results with TX dataset

VIII. Contribution Work

I developed keyword search in large spatial database by voice reorganization.

IX. Conclusions

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. Future work includes examining spatial approximate sub-string queries, designing methods that are more update-friendly.

References

[1] Feifei Li Member IEEE, Bin Yao, Mingwang Tang, Marios Hadjieleftheriou, "Spatial Approximate String Search", IEEE Transactions on Data Mining,vol.25, no.6,2013,15.
[2] R. Prasanthini1, Dr. K. Kavitha2,"Avoiding Traffic Congestion Based on MHR tree in Carpool Services", International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169, Volume: 4,627 – 630.
[3] B. Yao, F. Li, M. Hadjieleftheriou, and K. Hou, "Approximate string search in spatial databases", In ICDE, 2010.
[4] Sellis, T., Roussopoulos, N. Faloutsos, C., "The R+-tree: a Dynamic Index for Multi- Dimensional Objects", VLDB, 1987.
[5] S.Alsubaiee, A.Behm, and C. Li. "Supporting location based approximate keyword queries" In GIS,pages 61-70,2010.