RESEARCH ARTICLE OPEN ACCESS

Assessment of Efficacy among String Quest

K.Manju[1], R. Brindha[2], V. Lathika[3], Mr. A. M. Ravishankkar[4], Mr. T. Yoganandh[5]

[1,2,3]*UG Student-Department of CSE Jay Shriram group of institutions, Avinashipalayam, India*
[4,5]*Assistant Professor- Department of CSE Jay Shriram group of institutions, Avinashipalayam, India*

ABSTRACT
In a large spatial database the work mainly deals with the approximate string search. Particularly we examine the query range append with the string similarity search in both Euclidean space and Road network. We dub this query the spatial approximate string (SAS) query. We propose an approximate solution in Euclidean space, the D-tree which embeds min-wise signatures into an R-tree. The min-wise signature for an index node u keeps a clear representation of the union of q-grams from the sub tree of u. We find the pruning functionality of such signatures based on the set resemblance between the query string and the q-grams from the sub tree of index nodes. We discuss about the estimation of selectivity of a SAS query in Euclidean space, for which we present a novel adaptive algorithm to find the balanced partitions using both the spatial and string information stored in the tree. In Road networks, we propose a novel exact method, RSASSOL, which significantly performs the baseline algorithm. The RSASSOL method partitions the road network, adaptively searches relevant sub graphs, and prunes candidate points using both the string matching index and the spatial reference nodes. The efficiency and effectiveness of our approaches is by the extensive experiments on large real data set**s.**

Keywords- Approximate string search, range query, road network and spatial databases

I. INTRODUCTION

Data mining also known as Knowledge Discovery or Knowledge Discovery in Database (KDD), is the process of extracting or mining the data from large amount of databases. Text mining is one of the applications of data mining which mainly involves the process of extracting interesting information and knowledge from unstructured text documents. Keyword search over a large amount of data is an important operation in a wide range of domains. In spatial databases, wherekeyword search becomes a fundamental building block for an increasing number of real-world applications, and proposed the IR^2-Tree. A main limitationof the IR^2-Tree is that it onlysupports exact keyword search. Approximate string search is necessarywhen users have a fuzzysearchcondition, 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 search could be combined with any type of spatial queries. In this work, we focus on range queries and dub such queries as Spatial Approximate String (SAS) queries. We denote SAS queries in Euclidean space as (ESAS) queries. Similarly, it extends SAS queries to road networks (referred as RSAS queries).In ESAS, this motivates the need for string matching. A critical component of record matching involves determining whether two strings are similar or not: Two strings are considered matches if their corresponding (string) attributes are similar. String similarity is typically measured via a similarity function that, given a pair of strings returns a number between 0 and 1 a higher value indicating a greater degree of similarity with the value 1 corresponding to equality. This function is used to perform a similarity join between two input relations that returns pairs of strings whose similarity is above an input threshold.

II. RELATED WORK

1. INCORPORATING FORMAL STRING INDEX TRANSFORMATION IN RECORD MATCHING

In this paper, the author considered the problem of record matching for user-defined string transformation as input. The similarity between two strings is defined by transformations coupled with an underlying function of similarity. To lookup an input record against a table of records, where we make this approach with effectiveness by a fuzzy match operation. Here we have an additional table of transformation as input. The cognizant of transformations is nothing but the improvement in the quality of record matching and the efficient retrieval based on our index structure. The characteristic of this scenario is the most input sub-strings do not match with any member of the dictionary, sowe developed a compact filter which efficiently filters out a large number of sub-strings that cannot match with any member of dictionary. For membership, the

sub-strings which pass out filter are then verified by checking it. We demonstrate real datasets that approach significantly outperforms both current best exact methods as well as probabilistic methods, that may not identify a small percentage of matching sub-strings.

2. RETRIEVING TOP-K PRESTIGE-BASED RELEVANT SPATIAL WEB OBJECTS

In this paper, the author handles the problem of retrieving web documents which is relevant to query of a keyword within a pre-specified spatial region. There are two stages in query processing. In the First stage, indexing is for the filtration of web document. In the second stage, employed the another index. (e.g.., R-tree). To integrate the R-tree with signature files here a hybrid index structure is proposed. For the purpose of pruning the search space at a query time, both spatial information and text information is utilized by enables the hybrid index structure. However, this proposal is limited by its use of signature files (e.g.., the number of false matches is linear in the collection size and there is no sensible way of using signature files for handling ranking queries). To process a new type of query the combination of R^*-tree and bitmap indexing is developed by hybrid index structure is called l-closest keyword query. To enable the efficient processing of the location-aware top-k ranking query, it utilizes both location and text information to prune the search space which integrates the R-tree and inverted files for the IR-tree in hybrid index structure.

3. SELECTIVITY ESTIMATION IN SPATIAL DATABASES

In this paper, the author proposed a several new techniques for spatial selectivity estimation. These techniques are based on the spatial indices, binary space partitioning, and the novel notion of spatial skew. In database the critical component of query processing is selectivity estimation. For the spatial selectivity estimation there will be a very little work in providing accurate and efficient techniques, despite the increasing popularity of spatial databases. In this domain the relational techniques do not perform well because the spatial data defers from the relational data. From the previously known techniques,the author can able to show that : (a) Sampling and parametric techniques which work well in the relational one-dimensional world do not work well for spatial data. (b) A BSP based partitioning that we call Min-skew outperforms the other techniques over a broad range of query workloads and datasets.

III. EXISTING SYSTEM

Approximate string search is necessary when the users have a fuzzy condition search or spelling error when submitting the query. By completely ignoring the spatial component of a query, we evaluate only the string predicate by matching the index which is built as a string in both ESAS and RSAS query to produce a direct solution. The string solution which contains a point is not satisfied by the spatial predicate has been pruned in post processing step after all similar strings has been retrieved.

1) The string solution suffers the same scalability and performance issues as the spatial solution.
2) In existing spatial databases additionally we answer for SAS query to enable the efficient processing of standard spatial queries which is a spatial-oriented solution.

IV. PROPOSED SYSTEM

In our proposed system, we divide a roads network G={V,E} into t edge-disjoint sub graphs $G_1,G_2,\ldots\ldots,G_t$, where t is a user parameter, and for each sub graph build one string index. From V as reference nodes, we also select a small subset V_R of nodes: they are used to prune candidate points/nodes whose distance to query point q are out of query range r. In our RSAS query framework it consists of five steps.

1) We find all the sub graphs which intersect with the query range.
2) To retrieve the points we use the filtration tree of the sub graphs whose string are potentially similar to the query string.
3) We performing the calculation of lower and upper bounds of their distance to the query point, using V_R to prune away some of the candidate points.
4) Between the query string and the strings of candidates the exact edit distance is performing to prune away some further candidate points. After this step, the string predicate has been fully explored.
5) We do the checking process of their exact edit distance to the query point for the remaining candidate points to return those with distance within r.

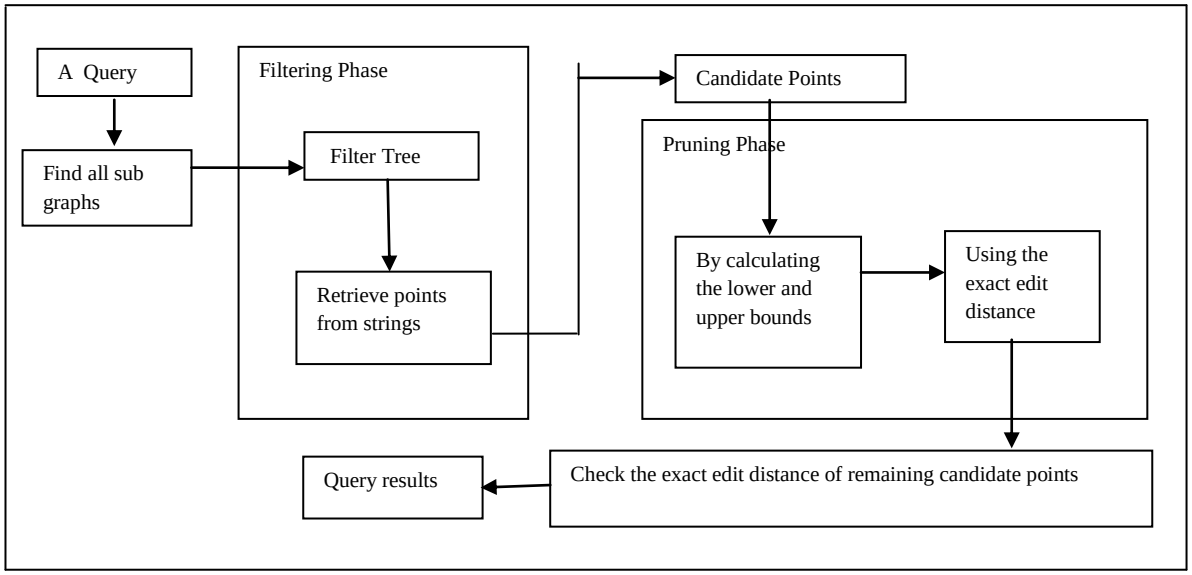

Figure 1.SYSTEM ARCHITECTURE

In RSASSOL algorithm, we find all the sub graphs that intersect with the query range. Here we employee the Dijkstra's algorithm, that is starting from the query point q to traverse nodes in G. we analyze that sub graphs for further explorations, whenever this traversal needs the first node of new sub graph. When we reach the boundary of query range automatically the algorithm terminates.To find the points from G_i that may share similar strings to the query strings, examine theeach sub graph G_i, we use the approximate string search over G_i's filter tree as the next pruning step. Then using the spatial predicate, we prune the candidate points by computing lower and upper bounds on their distance to q using V_R, in a similar way to the ALT algorithm. Given a candidate point p on an age $w=(m_i,m_j)$, the shortest path from p to a reference node m_r must pass through either m_i or m_j.

Network distance

$d(p,m_r)=\min(d(p,m_i)+d(m_i,m_r),d(p,m_j)+d(m_j,m_r))$
where

$d(m_i,m_r),d(m_j,m_r)$ are available from $RDIST_i$ and $RDIST_j$ respectively,
$d(p,m_i)$ is the distance offset of p to m_i which is available in the adjacency list and the points file of m_i,

$d(p,m_j)=NDIST(m_i,m_j)-d(p,m_i)$
where
$NDIST(m_i,m_j)$ is available in the adjacency list of m_i.

We compute $d(p,m_r)$ on the fly rather than explicitly storing the distance between a point and a reference node since the number of points much larger than the number nodes in G. given $d(p,m_r)$ and $d(q,m_r)$ for every $m_r \epsilon V_R$, we then obtain the distance lower and upper bounds between p and q using the triangle inequality. Besides the batch verification, we support one-at-a-time-verification, which implement as follows. Verification model consists of two phases: first, the building phase, second, querying phase. In building phase, we create λ(r) tuples { id, r, hash_sig, wt} for each dictionary string r and each signature generated by r. Hash code of signature is denoted as hash_sig and wt is the weight of the string r.

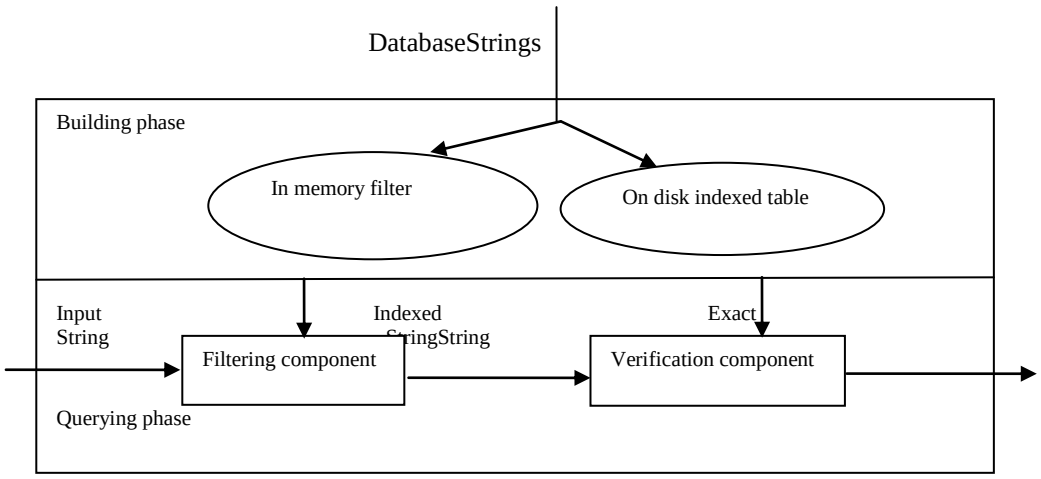

Figure 2. OVERVIEW OF THE FRAMEWORK

RSASSOL ALGORITHM

1. Find the set X of ids from all the sub graphs
2. Set B = ß ,B_c= ß
3. For each sub graph id j ϵ X do
4. Find all points ids in Ki whose associated strings α ' may satisfy ϵ(α ' ,α) ≤ r using filter tree
5. for every point P_i ϵ B_c do
6. Calculate $b^+(p_i , q)$ and $b^-(p_i ,q)$ as discussed
7 .if $b^+(p_i ,q) \leq r$ then
8. if ϵ $(\alpha_i ,\alpha) \leq r$ then move p_i from B_c to B
9. else delete p_i from B_c else if $b^-(p_i ,q) > r$ then delete p_i from B_c
10. for every point p_i ϵ B_c do
11. if $\epsilon(\alpha_i ,\alpha) > r$ then
12. delete p_i from B_c
13. Use the MPALT algorithm to find all points p 's in B_c
14. Return B

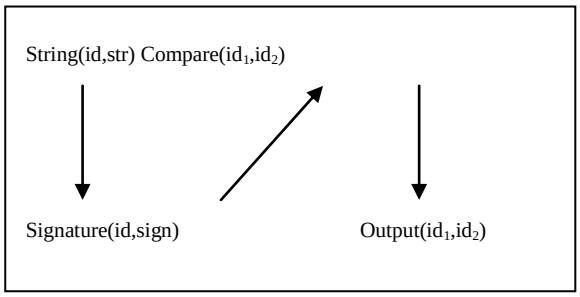

Figure 3.SS-JOIN IMPLEMENTATION

In this section, we perform the String Similarity joins using edit distance, which is one of the most common distance functions for string. SS-Joins are closely related to set-containment joins, which has been the main part of several previous works. Generally, similarity joins are closely related to proximity search. The goal is to retrieve thelookup closest object.

APPROXIMATE MEMBERSHIP CHECKING

Input: Z,□, S =<t_1 ,t_2.....>
1. Build the filter f (Z,□)
2. Index Z for verification
3. for (Start= 1 to lZ l – L + 1)
4. for (length =1 to L)
5. m ← < t_{start},$t_{start+1}$,$t_{start+length-1}$>
6. if(f. prune(m)==true) continue
7. if(Ɔ r ε Z, S. t Similarity(r, m)≥ delta)
8. Output m

V. EXPERIMENTAL SETUP

The system is developed using Java and is used in the system development. MYSQL is used as a back end for this system development. Input to this project is the user's string which is related to the keyword which already in database.

1) The process of entering the string to collect the exact information about the string.
2) The particular string makes a compare with related strings which is already stored in the database by using the ids.
3) Then the id shortlisted the string which is related to the user string.

4) At last by applying the Dijkstra's algorithm the user can get the exact information for the particular string.

The attention is finally paid to extract the exact information from the database based on the comparison between the strings.

VI. CONCLUSION

In this paper, we use the edit distance for the string predicate as the similarity measurement. And also address the problem of query selectivity estimation for the queries in Euclidean space. In the selectivityestimation for the query range on road networks where proposed. Even though, they can only able to estimate the number of loads and edges in the range. Future work includes examines the queries of spatial approximate sub-string, designing methods that are more user friendly and solving the selective estimation problem for RSAS queries.

REFERENCES

[1] S. Acharya, V. Poosala, and S. Ramaswamy, "Selectivity Estimation in Spatial Databases", Proc. ACM SIGMOD Int'l Conf Management of Data, 1999.

[2] S. Alsubaiee, A. Behm, and C. Li, "Supporting Location-Based Approximate-Keyword Queries", Proc. SIGSPATIAL 18th Int'l Conf. Advances in Geographic Information Systems (GIS), 2010.

[3] A. Arasu, S. Chaudhuri, K. Ganjam, and R. Kaushik, "Incorporating String Transformations in Record Matching", Proc. ACM SIGMOD Int'l Conf. Management of Data, 2008.

[4] A. Arasu, V. Ganti, and R. Kaushik, "Efficient Exact Set-Similarity Joins", Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB), 2006.

[5] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: an Efficient and Robust Access Method for points and Rectangles", Proc. ACM SIGMOD Int'l Conf. Management of Data, 1990.

[6] X. Cao, G. Cong, and C.S. Jensen, "Retrieving Top-k Prestige-Based Relevant Spatial Web Objects", Proc. VLDB Endowment, vol. 3, 2010.