

# String Search Approximation on Spatial Data through a Novel Approach

V.Phaneendra<sup>1</sup> \* Dr. D. Srujan Chandra Reddy<sup>2</sup>

<sup>1</sup>M.Tech Student, PBR VITS, Kavali, A.P., India.

<sup>2</sup>Professor, PBR VITS, Kavali, A.P., India.

## ABSTRACT

*In several applications finding objects closest to a specific location that contains a set of keywords. For example, online yellow pages allows user to specify an address and a set of keywords. In response, the user obtains a list of related information whose description contains these keywords and it's ordering according to their distance from the specified address. The complexities involved in nearest neighbor search on spatial data and keyword search on text data have been extensively studied individually. To the best of our knowledge there is no fully efficient method to answer spatial keyword queries exclusively, where the queries specify both the location and a set of keywords. Here the survey is done on current techniques to cope with the problem of string matching that allow errors. In many of the fast rising areas such as information retrieval and computational biology this is becoming a more and more important issue. In this study main focus is kept on spatial string searching and mostly on edit distance, its statistical behavior, its history and current developments, and the central ideas of the techniques and their complexities. The main objective of this survey is to present an overview of the new and efficient approach in approximating string search.*

**Keywords:** approximating the spatial search; nearest neighbor search in spatial data; spatial keyword queries, approximate string matching; edit distance; approximate string searching.

## 1. INTRODUCTION

Keyword search on large amounts of data is most crucial operation in a wide range of fields. Felipe et al. has extended its study to spatial databases [17], where keyword search becomes a fundamental building block for an increasing number of real-world applications, and proposed the IR2-Tree. A main limitation of the IR2-Tree is that it only supports exact keyword search. In reality, keyword search for retrieving approximate string matches is required. Because 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 purposes.

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 survey we focus on range queries and convert such queries as *Spatial Approximate String (SAS) queries*. SAS queries to Euclidean space, depicting a common scenario in location-based services: find all objects within a spatial range  $r$  (specified by a rectangular area) that have a description that is similar to "theatre". We denote SAS queries in Euclidean space as (ESAS) queries. Similarly, SAS queries to road networks (referred as RSAS queries). Given a query point  $q$  and a network distance  $r$  on a road network, we want to retrieve all objects within distance  $r$  to  $q$  and with the description similar to "theatre", where the distance between two points is the length of their shortest path.

Define the similarity between two strings is a key issue in SAS queries. The edit distance metric is often. Specifically, given strings  $\sigma_1$  and  $\sigma_2$ , the edit distance between  $\sigma_1$  and  $\sigma_2$ , denoted as  $\epsilon(\sigma_1, \sigma_2)$ , is defined as the minimum number of edit operations required to transform one string into the other. An insertion, deletion, or substitution of a single character is referred as the edit operations. If the different operations have different costs or the costs depend on the characters involved, we speak of general edit distance, Or else, if all the operations cost 1. We speak of simple edit distance or just edit distance (ed). In this last case we simply seek for the minimum number of edit operations to make both strings equal. For instance  $ed("survey","surgery") = 2$ . The edit distance has received a lot of attention because its generalized version is powerful enough for a wide range of applications. Despite the fact that most existing algorithms concentrate on the simple edit distance, most of them can be easily adapted to the generalized edit distance, and we pay attention to this issue throughout this work.

Here we have a clear example that,  $\varepsilon$  is symmetric, i.e.,  $\varepsilon(\sigma_1, \sigma_2) = \varepsilon(\sigma_2, \sigma_1)$ . For example, let  $\sigma_1 = \text{'theatre'}$  and  $\sigma_2 = \text{'theater'}$ , then  $\varepsilon(\sigma_1, \sigma_2) = 2$ , by substituting the first 'r' with 'e' and the second 'e' with 'r'. We do not consider the generalized operation of using exchange operation by swapping two characters in a string, while keeping others fixed. Here we have a standard method for computing  $\varepsilon(\sigma_1, \sigma_2)$  using dynamic programming for the strings of length  $n_1$  and  $n_2$  respectively, with the complexity of  $O(n_1 n_2)$ .

In this survey the key issue discussed is indexed search process, to build an apt data structure (for indexing) on the given text to speed up the search. As volatility of the text is the main reason to prevent keeping of indexes of text and extra space requirements. This survey mainly aims in explaining of the preliminary tools for string matching approximation with some of the extensions.

## II. RELATED WORKS

As many surveys are done on spatial queries for exact keyword matching. Chen et al. [1] proposed special indexing mechanism on the spatial and textual information. Zhou et al. [21] Suggested another indexing scheme that is a hybrid of spatial and inverted indexing mechanisms. For this R\*-tree is used and inverted indexes are built for the keywords as leaf nodes. As these are not the only techniques used for pruning. Some more studies [14], [5], [4] are proposed for pruning of spatial and textual data. In any way the spatial indexing is done on tree based data structure and textual data as the nodes.

For spatial databases the IR2-tree was proposed in [17] to perform exact keyword searching through kNN queries. As IR2-tree cannot support approximate string searches, as their estimation of selectivity was addressed therein, where the proposed R\* Tree cannot handle the Approximate string search in any way. Two more studies are there [7], [13] where ranking queries combine with both the spatial and text data according to the query object.

In another work related to string search LBAK tree was proposed for answering location-based keyword approximation queries. Here the key idea is to enlarge a tree-based spatial index (i.e. an R-tree) with a sub tree of q-grams of nodes for approximating edit-distance based string searching operations.

LBAK-tree was proposed after clear study on SAS queries on the Euclidean space [44], when it is compared with the MHR-tree [2], LBAK-tree achieves better querying time than the MHR-tree, but more space is required to execute. The LBAK-tree returns exact answers for the ESAS queries where the MHR-tree approximates the results. The final conclusion is

that LBAK-tree is opted for exact results and MHR-tree is opted for approximate solutions and for small space consumption. RSAS queries and selectivity estimation of SAS queries not been explored yet. So, individual string search approximation was extensively studied in this literature [3], [8], [9], [11], [19], [27], [30], [40], [42], [43].

The above works generally concentrates on similarity function to quantify the closeness between two strings.

## III. FEATURES OF THE MECHANISMS

### A. Approximate string search

Approximate string search is an important sub problem of approximate Keyword search, which is described as: To find a query similar to a given query string in the possible collection. Two main approaches are used to answer those queries. (a) *Trie based method*: where the collection of strings is stored on a trie (i.e. suffix tree). To answer an approximate string query, the trie and prune sub trees are traversed according to the query. Dynamic programming matrix with backtracking is the mostly used pruning method. (b) *Inverted-index method*. Here the key concept is to decompose each string into the collection of small overlapping substrings (termed as grams) and build an inverted index with these substrings.

A few more information on fast indexing and searching methods can be found in [12], [15], [16].

### B. Spatial approximate-keyword search

For Spatial Data Searching Approximation, MHR-tree was proposed to answer spatial data queries. MHR-tree is an enhanced version of R-tree [6] with min-wise signatures at every node for representing the collection of grams compactly, which is contained in objects of a sub tree. The set resemblance method is used between the signatures and the query strings for the pruning of sub trees in the tree. In this approach it is advantageous that does not require a lot of space for indexing because min-wise signatures are too small. Through this method we can miss some query answers due to their probabilistic nature.

For efficient working of this algorithm on some crucial web sites for achieving high query throughput for many concurrent users, the index structure upgraded to a tree-based spatial index with approximate string indexing such as a inverted index or a trie-based index. The main focus is kept on, how to efficiently combine these two types of indexes, and how the searching results the indexes for finding answers.

### C. Nearest Neighbor and Top-k Queries

$k$ -nearest neighbor queries ( $k$ NNs) for spatial databases is an old concept. Where most of the proposals use index structures take assistance of the  $k$ NN processing.  $k$ NN algorithm was introduced by Roussopoulos et al. [39]. In KNNs method R-tree[31] is used for indexing the points, finding potential nearest neighbors maintained according to priority queue, and the tree traversal is done according to a number of mathematical conditions. Branch-and-bound methods are used to modify the index structures to better suit problem [37], [41]. To perform this more efficiently, an incremental nearest neighbor algorithm on R\*-tree [22] was proposed.

### D. Location-Aware Text Retrieval Queries

Now a day's most familiar search engines (i.e. Google, Yahoo) are performing local search services which concentrates on retrieving the local content, (i.e. info related to stores and restaurants). Internally powerful algorithms are used. Here complete attention is kept on extracting geographic information from web. Search engines will use this extracted information. McCurley [38] explains the notion of geo-coding and geographic indicators found in web pages (i.e. zip codes, location names etc). In our study it is observed that the location-aware text retrieval work most closely related. Zhou et al. [47] handled the concept of retrieving web pages relevant to a keyword with in spatial data ranges. For that three approaches based on a loose combination of an inverted file and an R\*-tree is proposed. Building an R\*-tree for every distinct keyword on the web page containing the required keyword is the best approach. By this we can say that, to intersect the results from multiple queries, queries with multiple keywords need to access multiple R\*-trees. Substantial storage spaces are required to build individual R\*-tree for every keyword in query string.

Ian De Felipe et al. [17], proposed top-k spatial keyword search problem and also defined. IR2-Tree is proposed as best efficient indexing structure to store spatial and textual data for all set of objects. There are also efficient algorithms to insert and delete objects. An incremental algorithm is introduced to answer top-k spatial keyword queries using the IR2-Tree efficiently, based on the tight integration of data structures and algorithms applied on it for spatial database search and Information Retrieval operations.

Here we have described how to build an Information Retrieval R-Tree (IR2-Tree), which has R-Tree based structure. While execution of query an incremental algorithm is applied using the IR2-Tree for producing efficient top results of the query.

## IV. ALGORITHMS

### A. RSASSOL Algorithm

This algorithm is used for RSAS queries, occurs in five steps.

In first step for a given query, find all the sub graphs that lies within query range. Second step uses the filter-trees on sub graphs and retrieve points whose strings are nearly similar to the actual query string. Then the third step prunes away the candidate points by lower and upper bounds calculation from their distance point. In the fourth step further pruning of some more candidate points using the exact distance calculation between the actual query string and remaining candidate strings. String predicate is fully explored in this step. As a final step, the remaining candidate points and their exact distances to the query point and returns the final distances in query range.

### B. MPALT Algorithm

MPALT algorithm is determined as Multipoint Abbreviated List Table. In this algorithm multiple shortest paths, within the query range is calculated, between a single source point and multiple destination points in parallel. The distances computed and stored in the table as lower and upper distance bounds for every node from any destination.

In this algorithm source and destination shortest path distance is avoided by minimizing the accessing to the network, and also to the explored part of the network where it is very useful in the calculation of multiple shortest paths between multiple sources and destinations.

### C. Range Query and Edit distance

In range query data base operation all records are retrieved within the range of upper and lower boundaries. The Data structures for range query are Range tree is the data structure used for organizing range queries. Range query operation involves in preprocessing input data onto the data structure and to retrieve the efficient answers for the queries by taking any subset as an input.

Edit distance is a quantifying methodology of how two different strings are to one another and calculate minimum operations for transforming a string. Edit distances operation also find and do the the automatic spell checks and corrections and determine the candidate and required corrections if there is any misspelled words which has low distance to the word in the tree.

## V. FURTHER ENHANCEMENTS

After taking a vast study towards the search for data with exact grammar in queries. It is found that most of the cases aimed to find the data with incomplete knowledge about the queries they submit. Most of data can be found without proper spelling considerations

and its degree of uncertainty of error.

The system has developed to detect various kinds of data using brilliant string search algorithms. Like assumption to get a data misspelled or other error.

As search query is very important for searching data, through our algorithms we can refine the query. For more advancements in searching of data and refinement of queries more advance algorithms must be proposed for the future work.

## VI. CONCLUSION

By the end of this survey on String search approximation and Approximate string matching (defined as searching for substrings of a text that are within a predefined edit distance threshold from a given pattern). Throughout this survey we have presented and explained the main ideas behind the existing techniques, to classify them according to the type of approach proposed, and to show how they perform in practice in a subset of possible practical scenarios.

## VII. REFERENCES

- [1] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *SIGMOD*, 2006.
- [2] S. Alsubaiee, A. Behm, and C. Li. Supporting location-based approximate-keyword queries. In *GIS*, pages 61–70, 2010.
- [3] A. Arasu, S. Chaudhuri, K. Ganjam, and R. Kaushik. Incorporating string transformations in record matching. In *SIGMOD*, pages 1231–1234, 2008.
- [4] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1), 2009.
- [5] I. D. Felipe, V. Hristidis, and N. Risse. Keyword search on spatial databases. In *ICDE*, 2008.
- [6] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, 1984.
- [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 filter for approximate membership checking. In *SIGMOD*, pages 805–818, 2008.
- [9] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient 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 for similarity joins in data cleaning. In *ICDE*, pages 5–16, 2006.
- [12] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava. Fast indexes and algorithms for set similarity selection queries. In *ICDE*, 2008.
- [13] 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.
- [14] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In *SSDBM*, 2007.
- [15] C. Li, J. Lu, and Y. Lu. Efficient merging and filtering algorithms for approximate string searches. In *ICDE*, 2008.
- [16] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1), 2001.
- [17] I. D. Felipe, V. Hristidis, and N. Risse. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.
- [18] B. Yao, F. Li, M. Hadjieleftheriou, and K. Hou. Approximate string search in spatial databases. In *ICDE*, 2010.
- [19] 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.
- [20] D. Zhang, B. C. Ooi, and A. K. H. Tung. Locating mapped resources in web 2.0. In *ICDE*, 2010.
- [21] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma. Hybrid index structures for location-based web search. In *CIKM*, 2005.
- [22] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R\*-tree: an efficient and robust access method for points and rectangles. In *SIGMOD*, pp. 322–331, 1990.
- [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 of approximate string predicates in large databases. *The VLDB Journal*, 17(5):1213–1229, 2008.
- [26] M.-S. Kim, K.-Y. Whang, J.-G. Lee, and M.-J. Lee. n-gram/2l: a space and time efficient two-level n-gram inverted index structure. In *VLDB*, pages 325–336, 2005.
- [27] N. Koudas, A. Marathe, and D. Srivastava. Flexible string matching against large databases in practice. In *VLDB*, pages 1078–1086, 2004.
- [28] H. Lee, R. T. Ng, and K. Shim. Extending q-grams to estimate selectivity of string matching with low edit distance. In *VLDB*, pages 195–206, 2007.