Unique indexing model in Geospatial database Paradigm

Bhagwant Singh*,
[1]Department of Informatics, School of Computer Science
University of Petroleum & Energy Studies (UPES)
E-mail: Bhagwant.singh@ddn.upes.ac.in

Dr. Kingshuk Srivastava*
Department of Informatics, School of Computer Science
University of Petroleum & Energy Studies (UPES)
E-mail: ksrivatava@ddn.upes.ac.in

Dr. D.K. Gupta*
Department of Petroleum Engineering and Earth Sciences, School of Engineering
University of Petroleum & Energy Studies (UPES)
E-mail: dkgupta@ddn.upes.ac.in

Abstract:

Smart technologies have been presenting data under multiple dynamic perspective. Analyses of such data to generate knowledge base, helps analysist in formulating solution for any agile problems. The Data available for processing have intrinsic spatial and temporal signature, hence increasing the complexity of the data model. Today's location enabled devices are decisive of various on-the-run problems with a solution incorporating every customization. Therefore, extracting the relevant data of the precise location at the given time, keeping the global relationship in harmony is the need of the hour. This paper presents a framework for capturing location aware unconventional data and present co-relation of perspective, agile problem, spatial signature and data model in term of spatial search.

Keywords: Spatial Indies, Top-K search, Quad-tree, h_s code

1. Introduction

Location pairing is an important and inherent obligation of every service provider worldwide. Taking in incalculability of the engendered spatial data at an inimitable rate, probing information turn out to be a decisive job. Diverse web/mobile podiums such as online directories (yellow pages), social media sites (twitter, Facebook, flicker), web crawlers (Google, Yahoo, Bing), and GPS enables devices (airline, taxi, rail) provide semantic location aware data of the user. Perception of analogous data with intrinsic location beheld under different context deliver unlikely fallouts. Hence encouraging the use of advanced and improved spatial temporal data processing tools are highly suggested. Location aware services tends to record trajectory of the object to understand cohesive nature of the object in consideration. For that reason, location get associated with the objects, its different interaction with environment and review about the activities in the neighbourhood. Location and characteristic keywords can be paired together and research ensuring effective search of information has been undertaken during the last decade [16][8][9]. Combining spatial search and keyword search has presented hybrid search mechanism to work with current spatial data known as "Spatial keyword Search". Top- K spatial keyword query is one of the most remarkable algorithms used to manipulate existing spatial data. Over a time, advancements in the Top-K search algorithm to cater different problems, but no research optimize the latitude and longitude with associated keyword. The current paper presents a framework to search spatial data with unique location code and hybrid indexing techniques.

2. Literature Review

Spatial search required to search attribute and location separately [Costes et. al (2019)] [Georgiou et. al (2019)] [Zheng et. al (2016)]. Presenting the perspective search of the information, indexes are generated both keyword index and spatial index. Based on the core design techniques, the spatial indexes can be presented as R-tree based

* Energy Acres, Bidholi. Dehradun- 248007, Uttarakhand, India

[Zhang et. al (2016)] [Zheng et. al (2016)], Grid Base [Griffith et. al (2016)] [Giannotti et. al (2007)], and spatial filling curve based [Hong et. al (2017)] [Griffith et. al (2016)] [Zhang et. al (2016)]. Among these three, the R-tree based index presents much more efficient and moldable results, hence current paper presents the literature supporting R-tree and its applications to cater spatial search. Conventionally to engender keyword search index, inverted index or hash table are used to index keywords of large dataset [Zheng et. al (2016)] . For spatial indices include inverted R-tree [Zhou et. al (2005)] , SFC-QUAD [Christoforaki et. al (2011)] , S2I [Zhang et. al (2013)] , IR2-tree [Felipe et. al (2008)] , KR∗-tree [Zhou et. al (2005)] , IR-tree [Wu et. al (2011)] [Cong et. al (2009)] , WIBR-tree [Wu et. al (2011)] , and SKI [Cary et. al.(2010)] . [Zhou et. al (2005)] paper, generate keyword-object list and create R-tree for each keyword in search to form IR-tree. With the arrival of the query only the R-tree supporting all the keywords are addressed for which incremented nearest neighbourhood technique is used [Griffith et. al (2016)] . [Christoforaki et. al (2011)] paper, fused mathematical tool such as space filling curve with a keyword file generated using the inverted indexes. To reduce processing time, keywords were blocked together and the parent index of the block is searched in the S2I [Rocha et. al (2011)]. Spatial similarly to S2I tree, IR2-tree incorporated unique reference file with each leaf node of R-tree [Felipe et. al (2008)] . Li et al. proposed BR-tree [Griffith et. al (2016)], where R-tree location of the objects and B-Tree organize keywords. The algorithm followed two approaches that is keyword search first or location search first. [Griffith et. al (2016)] [Zheng et. al (2016)] ,paper combine both IR-tree and IR2-tree to formulate IL- Quadtree. In IL- Quadtree, for each keyword linear quad trees is generated using Morton code and different bits are used to present the status of quadrant [Griffith et. al (2016)].

For searching for confined area the KR∗-tree was purposed by [Hariharan et. al (2007)] , IR2-tree is the other variant of IR-tree proposed by Cong et al. [Li et. al (2012)] [Li et. al (2011)] [Cong et. al (2009)]. A Similar IRLi-tree store Integrated Inverted file at the nodes. Other remarkable research combining R-tree with inverted index are DIR-tree, CDIR-tree, introduced by Cong et al. and SKI proposed by [Cong et. al (2009)]. For supporting multi query search, WIBR-tree was proposed by [Wu et. al (2011)] . With every tree structure proposed the specified query was solved but effect the MBR and result in change in processing time.

Top K spatial keyword search query which follow scoring function or kNN (k nearest neighbourhood) to find the relevant results formulate the base for various model to find stationary and non-stationery (in-motion) objects. Various spatial search queries are designed for stationary object [Zheng et. al (2016)] [Christoforaki et. al (2011)] [Felipe et. al (2008)], where one find the probability of having an object at particular location depending upon the list of keywords given as stated above. For mobile objects, a location-aware top-k text retrieval (LkT) [Cong et. al (2009)] query was proposed by Cong et al. paper; along with query engine for bi directional motion in asMkSK query and RSTkNN query [Lu et. al (2011)] . These query engine use language models and a probabilistic ranking function to find the best fit for the location aware query. Another study carried out by Chen et al. presented Boolean range continuous query and proposed IQ-tree to support logically related keywords. [Zheng et. al (2016)] proposed I3 Integrated Inverted Index to support logical join among the search query. Different researcher presented Top K with different input criteria's such as sliding window [Griffith et. al (2016)] , distributed database [Zhang et. al (2016)] and adjusting weights of the scoring function dynamically [Hong et. al (2017)].

[Cao et. al (2010)] paper propose a model that find out result based on the priority of the keyword used in location-aware top-k prestige-based text retrieval (LkPT) query. Literature presented by Cho. X et al. supports extracting block of output object, m closest keywords (mCK) query and collective spatial keyword querying (CSKQ) [Cao et al (2011)] return all object holding minimum distance from the object suggested by the keywords and minimum distance between the output objects.

Different research carried out by [Alvares et. al (2007)] [Yan et. al (2011)] presented work to capture semantic search pattern of the path travelled. All of these research also used the top k spatial keyword search query with sematic engine. Literature suggest that Top K spatial keyword search query can be used for various other perspectives such as navigation [Rocha et. al (2011)], location based type search [Wu et. al (2011)] , and rigorously ranking of an event, based on location and region of occurrence [Zheng et. al (2016)] . Conclusive from the related work, Top K spatial keyword search query is most optimal technique to deal with rapidly generated spatial data over various platform; prominently social media.

3. Framework

Spatial data are nominal attribute data, accompanied by logical layered view over it. This logical view is responsible for the latitude/longitude of the data along with their intrinsic relational behavior. All the object exist in the reality have strong sense of dependency among each other, which define the object existence. Design of the logical view of the spatial data requires through understanding of the data along with well-defined structure, which present with the guard condition of the logical expression. Coordinate system such as Universal Transverse Mercator (UTM) is on such logical expression to design the logical view of any spatial data available for analysis.

Design of the over-layer view helps in the retrieving the spatial information and perform various operations of search, insertion and updating. The complexity of the design of the logical expression increase when the analyst processes schema-less data. For such data, there are many ways to form logical expression, this paper present the "Column BASE- Data Fetch and Index" (CBDFI) model for performing a fast spatial data search. The model also used a unique location code to traverse globally to and from a certain point in reality. This presented hexadecimal code formulate the base for searching spatial data in CBDFI model.

The black-box view of the architecture of CBDFI model include data encoder, followed by spatial pre-fetch and Column database. Data encoder; extract latitude, longitude from presented structural, semi-structural & un-structural data and store then in controlled fashion for further processing. Following which sexagesimal location is rehabilitated into hybrid ASCII code called as Hybrid Spatial ASCII (h_s code) of 12 bits. The code is converted for 0 - 90^o latitude; 0 - 60 minutes; 0 - 60 seconds 0- 180^o longitude. The h_s code comprises of alphabet A-Z and decimal number 0-9. These codes present the similar characteristic of hexadecimal code hence h_s code is also called as hexadecimal spatial code. The generated h_s code will be stored in spatial pre fetch. The column database will provide storage schema for the whole process. The purposed framework is designed to create unique indexing key which will incorporate the h_s code and attribute indexer to formulate hybrid indexer for CBDFI model.

Hybrid spatial indexer (HhI-S indexer) ; identify the unique keyword for the each generated h_s code and index these hexadecimal code with the help up of scaled up indexing algorithm HhI-S indexer combining IR-tree and H_s I-tree (hybrid h_s code index tree) which combine linear quad tree with inverted index and hashing [Costes et. al (2019)] . At the final stage of the HhI-S indexer, the keyword and the associated index will populate the leaf node of the tree data structure. During the search call the most appropriate node and the corresponding child nodes will be picked for understanding the correlation of the keyword with the corresponding h_s code.

The h_s code can associate with any keyword based on the similarities of the h_s code. Ideally, there can be four different cases to define the relationship between the generated h_s code and keywords. These cases are as follows:

Case1: Each generated h_s code generate unique keyword. Therefor number of keyword and generate h_s codes are the same,

Case 2: Multiple generate h_s codes are associated with same keyword. Therefor better and fast indexing,

Case 3: Multiple keywords are associated with same generated h_s code. Therefore creating duplicate keywords and increasing the complexity and

Case 4: Multiple keywords are associated with multiple generate h_s code.

In the CBDFI model the multiple generated h_s code are associated with single keyword, hence reducing the conversion time of the indexes and enhance spatial search. The HhI-S indexer will be following the standard topological vector model such that following relationship is maintain (Table 1).

Table1: Topological structure to store h_s code of Hy-S indexer

H_s code	**Latitude**	**Longitude**
GCGCICO33	0"	
ICOCOCO35		18"
So on ….		

a: h_s table

Index	**Keyword**	**Lat/Long**
Index1	Keyword1	0"/ 12"
	Keyword2	8"/18"
So on…..		

b: Index table

Index	**Keyword**	**H_s code Start**	**H_s code End**
Index 1	Keyword 1	GCGCICO33	ICICICO33
	Keyword 2		

c: Hy-S Indexer table

The CBDFI model will convert these columns, with each column stating the latitude, or longitude or altitude of an index and will return the generated h_s code. The columnar reference will generate the H_s I- tree and will

facilitate the spatial search. The framework of the CBDFI model (Figure 1) will be divided into four major areas; the data encoder, hybrid spatial query engine followed by query optimizer and finally the columnar database Monet DB. Formation of the hybrid indexer will be deriving the whole search algorithm. The created h_s code will formulate a linear quad tree and will support the Top k spatial keyword search query. After formation of the indexes, probalistic model [Singh et. al (2019)] will be used to ensure the conversion of the query. For each generated index all the related keywords, theirs associated objects/entities and correlated surrounding objects will be stored. Considering the storage schema various different NoSQL databases can be deployed.

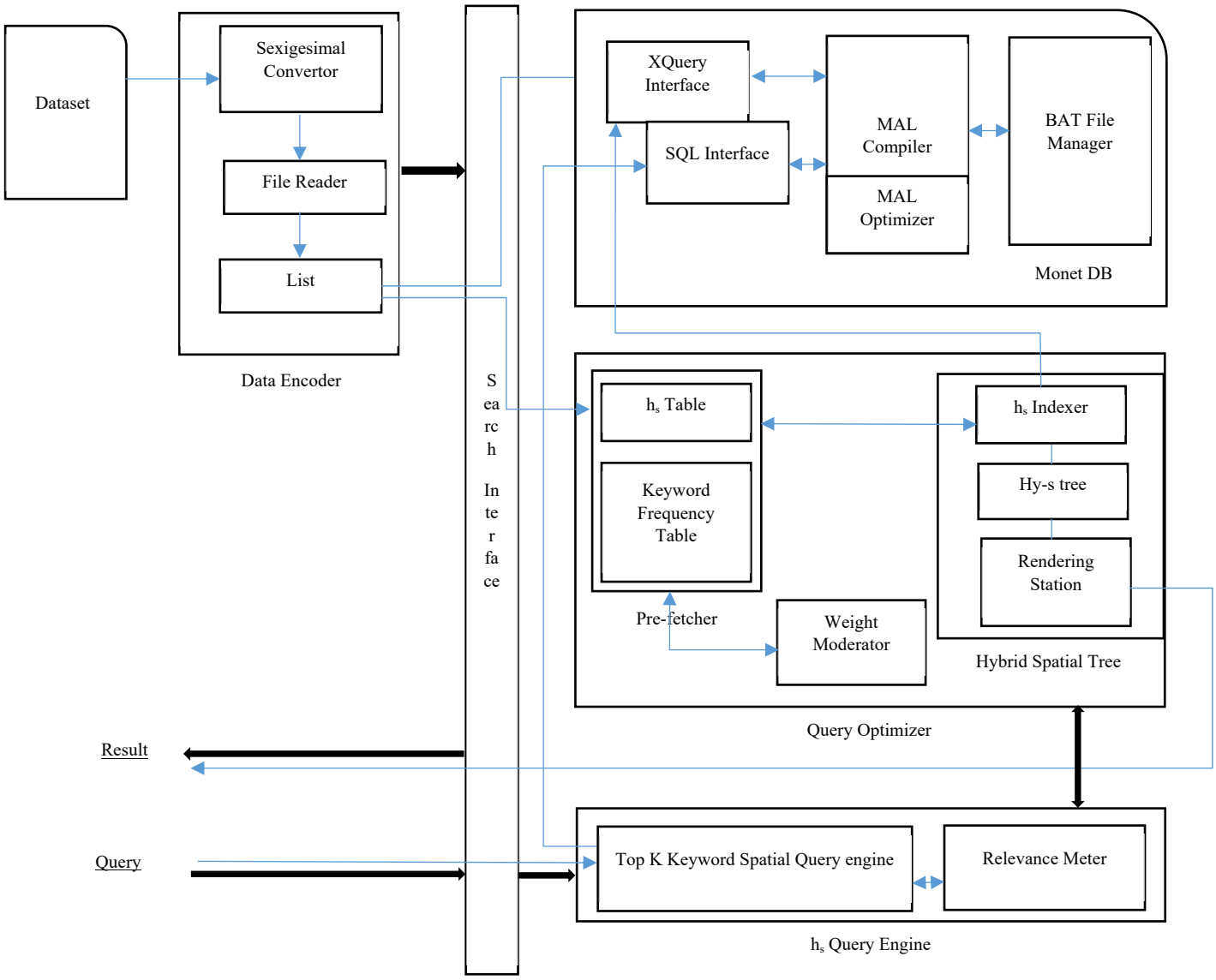


Figure 1: CBDFI Block Architecture

With due understanding of the dynamics of this search mechanism, column database serve the purposed [Costes et. al (2019)] . [Singh et. al (2019)] paper present a clear comparison of the different columnar databases for storing spatial data in No SQL schema. [Srivastava et. al (2012)] paper state various problems faced while integrated such data in No SQL schema. Therefore, the results will be stored in the column database and on every query fired by the user, engine will check the node of the tree for the suitable match and return the result of the query.

4. Conversion (hs code)

The CBDFI model work on h_s code converted from the latitude and longitude sexagesimal value. The converted h_s code showcase a unique pair of character values, arranged in a manner to generate unique codes for each spatial

location values. Transition of each code form a pair to another formulate the rules for searching the location values during query call. The frequency of repetition of the each bit result in forming the guard condition for search.

Prime number and prime factors plays a very important role in enhance the processing time of different mathematical calculation [Hadraba]. The h_s convertor also rely on the prime factor of the total keyword identified in the dataset. Keyword (h_s code) generator start with conversion of the extracted latitude and longitude into weighted ASCII code. As the latitude and longitude follow the standard decimal numbers system, therefor the weighted ASCII code follow the same convention. Weighted ASCII code of number are as follow

0 -99 = ASCII code of tenth place * 10 + ASCII code of unit place **(1)**

100- 180 = ASCII code of hundred place * 100 + ASCII code of tenth place * 10

+ ASCII code of unit place **(2)**

For any number = ASCII code * Weight of Hundred + ASCII code * Weight of tenth

+ ASCII code * Weight of unit. **(3)**

After converting the weighted ASCII code, formation of alphabet set (Palpha) take place ensuring only alphabet whose ASCII code is prime number and these prime number is prime factor of the total keywords. The whole process of generating h_s code depend upon the division of weighted ASCII code from each element of Palpha set until and unless one finds the first prime dividend. On identification of the first prime dividend, ASCII code of the matched element p € Palpha{} is subtracted from the weighted ASCII code to find revised weighted ASCII code. The process keep on repeating until the remainder is less than 65. For example latitude 2° (see Table 2):

Weighted ASCII code = 48*10 +50 = 530, (from 1,2,3)

Palpha {C,G,I,O,S,Y}.

Weighted ASCII code divided by Palpha : 530 / p € Palpha{} **(4)**

The convertor will generate the h_s code for each location in degree, minute and second format. The h_s code generated will be 9 bit long for values 0-59,63 and 10 bits for 60-62, 64-90. These 9/10 bit code will be enveloped in 12 bit format by suffixing the 3/2 bits at the MSB. The 9 bit code from 0-47 will be prefixed with Y and followed 48 -59,63 will be prefixed with I at the 9th bit.

Table2: Step by step conversion of Location 2° N latitude into h_s code

Weighted ASCII	First Prime Remainder	Remainder ASCII	Code
530	71	459	G
459	67	392	C
392	73	319	I
319	67	252	C
252	73	179	I
179	67	112	C
112	79	33	O

The 10th and 11th bit will comprises of 2 bit code to understand the position of the number pair in the location. The normalized code will be 12 bit which will uniquely identify the value. These 10 bit code are further converted into 8 bit code for more simplified calculation. The algorithm demonstrate the working of h_s code generator

Algorithm1: Generating h_s code

```
1: for i in range(n,n+1):
2:         u ← i%10
3:         u ← dict1[u]
4:          t ← int(i/10)
5:          t ← dict1[t]*10
6:          s ← u+t
7:         num.append(s)
8:         print(num)
9: for i in range(len(num)):
10:        while j<7:
11:                while(num[i]>dict2[j]):
12:                  temp ← (num[i] / dict2[j])
13:                  temp ← round(temp)
14:             if(temp - int(temp) <= 0.54):
15:                  temp ← int(temp)
16:             else:
17:                temp ← round(temp)'''
18:                if(sympy.isprime(temp) == True or temp==1 or temp==0):
19:                z ← dict2[j]
20:                k ← chr(z)
21:                      num[i]- ← dict2[j]
22:                      key  ← key+k
23:                j← 0
24:                else:
25:                          j ← j+1
26:                break
27:         if(num[i]<dict2[j]):
28:                           break
29:                if(num[i]<67):
30:                          key +  ←  str(num[i])
31:  result.append(key)
32:  key           ""
33:  print(result)
```

The following process flow converting the location value into h_s code is presented in Figure2

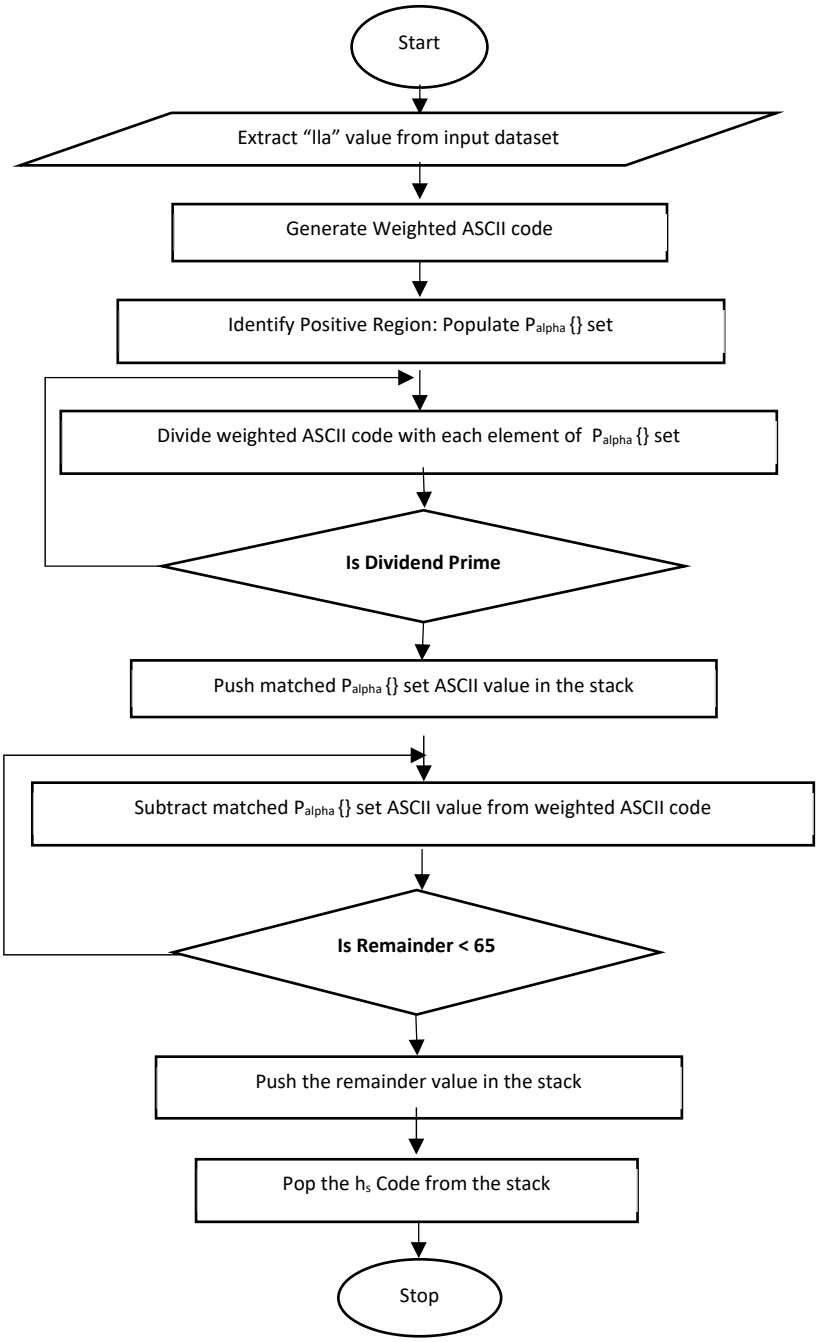


Figure 2: Process flow diagram h_s code generation for bit(0 -9)

For backtracking ASCII value of the each alphabet in h_s code is used. The first three bits from the MSB are separately read and summation of ASCII values of the remaining bits (9th bit to LSB) will give the weighted ASCII code which will be converted back to the decimal equivalent. The 10th bit is checked for the ASCII code. If the 10th bit ASCII code is 79 or 83, only then the value is added to summation of 9 bits (Figure 3). Frequency f(p) of each p € Palpha{} presented in the h_s code help in searching the exact code at improved rate. One of the most important realization is the frequency of G, if the f(G) where G € Palpha{}is 1 the h_s code from 44-53 is encountered. Algorithm for both h_s code backtracking the location is as follows:

Algorithm2: Retrieving location from h_s code

```
1: code ← input()
2: j ← len(code)
3: for k in range(j):
4:         if k < 1 :
5:                 sum + ← dict(code[k])
6:         elsif k =1:
7:                 num1← int(code[1])
8:         else:
9:                 num ← int(code[0])
10:                num ← num1*10 + num
11:                sum ← sum+num
```

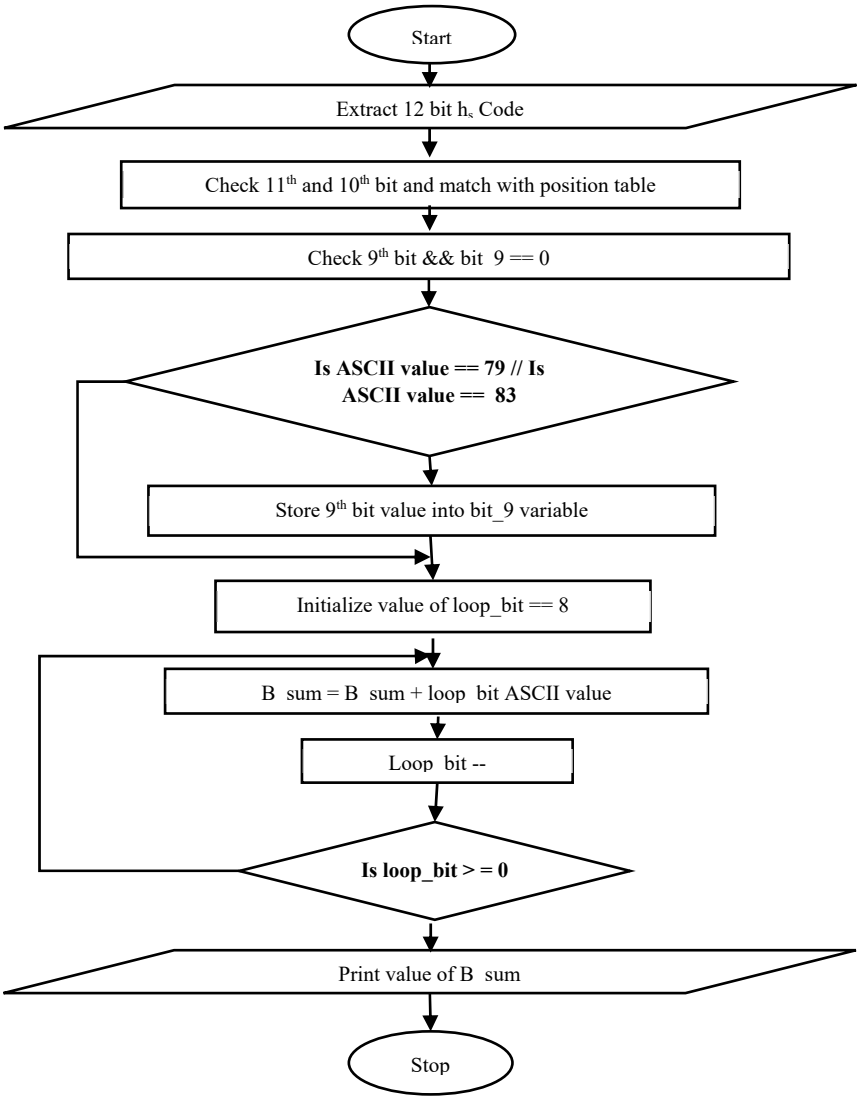


Figure 3: Process flow diagram for Back tracking from hs code

5. Hs I tree

In order to store the h_s code in the most efficient manner a hybrid spatial tree data structure is generated. This tree is a linear quad tree and upsclaed version of IL-tree. Similar to the utility of the IL-tree and its variants to process spatial indexing, the purposed tree create a data structure to store the 12 bit location code in four quarants and associate keyword with each location. The generated h_s code will be represented in the quad tree fashion For each object encountered by the system, the generated h_s code will be stored in the defined tree structure with two layers storing the position and the pair of h_s code and the third layer will be linked when the HhI-S indexer calculate the index and the corresponding keywords. The third layer present the correlation of the keyword with itself and similar keywords identified in the given set of location values with respect to the each quadrant of the h_s code. These set will help in identifying the next location that should be traversed in order to find the similar keyword.

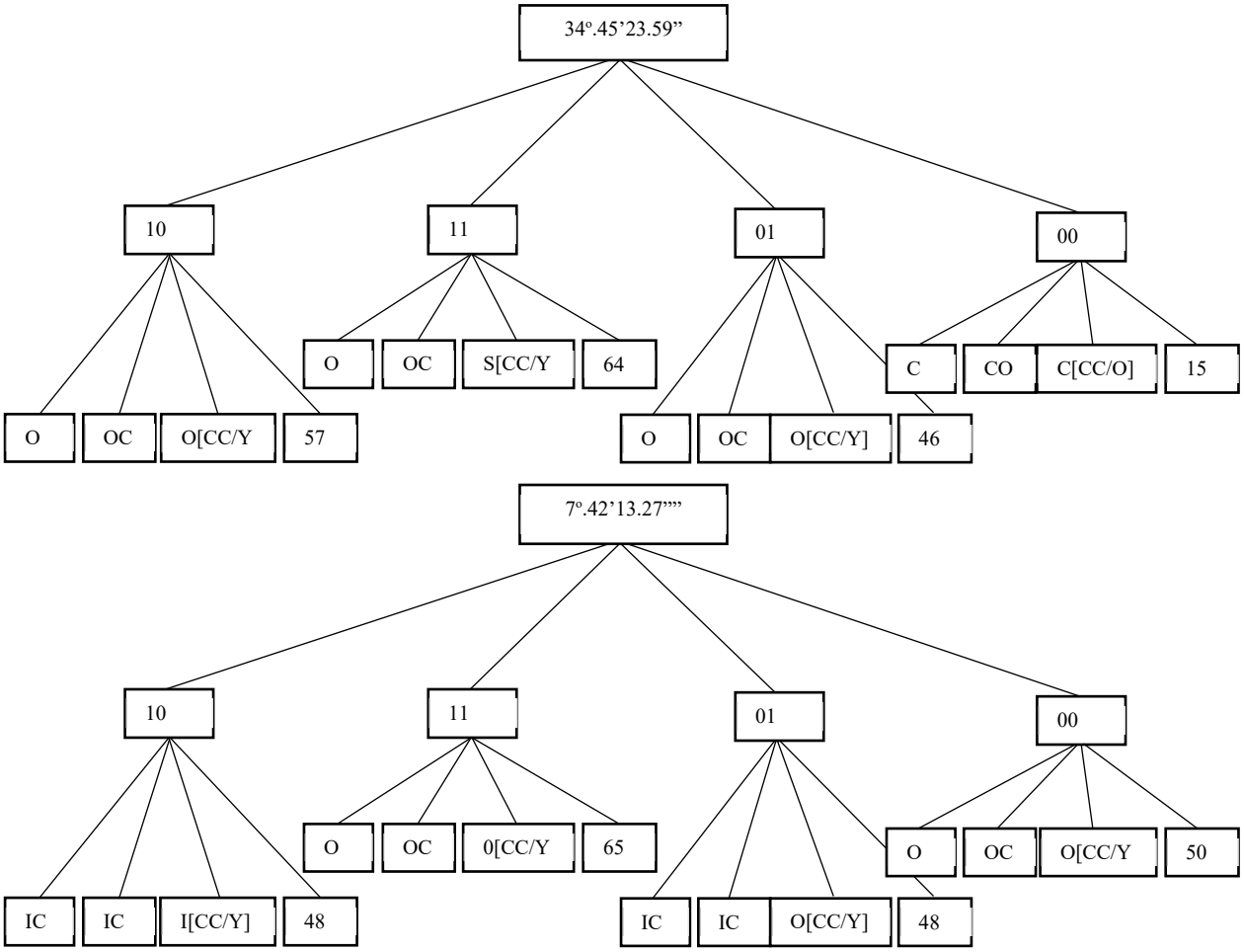


Figure4: Hybrid Hs I -tree for h_s code

These correlation values also will help in identify the hidden relationship of the object in search and will help in faster retrieval of the query fetched from the purposed framework. For the given location (34°.45'23.59" N and 7°.42'13.27" E) of the object O tagged within post of user handler; the Hs I tree node will be presented in quad tree format (Figure 4). Layer 3 will be added when multiple keywords will be encountered in the search query. The insertion, updating and deletion will be carried out in the same way as the linear quad tree.

6. Hidden Markov Model for h_s Code

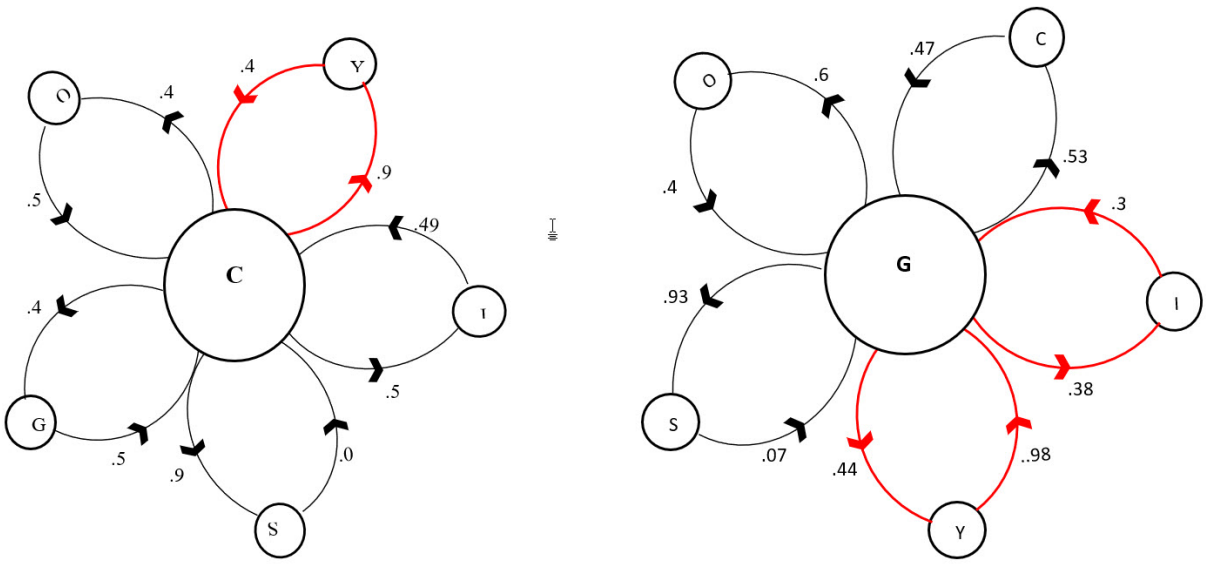
The generated h_s codes are group of four 12 bits number for each latitude and longitude location. These codes can be further simplified to produce single 7 bit cumulative code of the given location. During the process of encoding the h_s code into 7 bit hexadecimal code various different observation and hidden states are produces which thrives the probabilistic model of the generated location code. The algorithm for encoding 12 bits h_s code in 6 bits hexadecimal code is as follows:

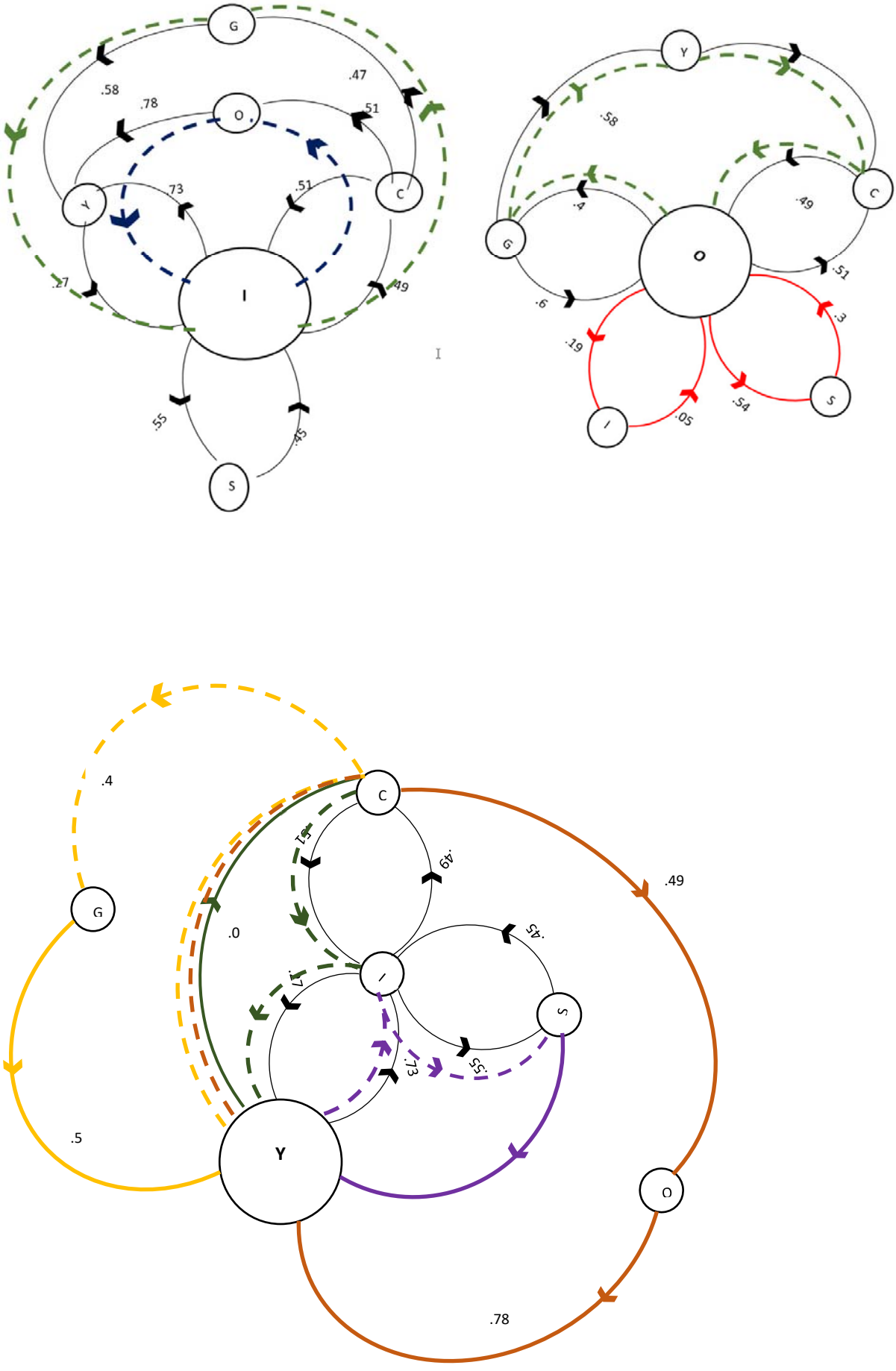
Algorithm3: Encoding 12 bit h_s code into 6 bit hexadecimal code

```
1: h ⟵ input("enter hs code",)
2: t ⟵ h[-9:-8]
3: m ⟵ h[-8:-4]
4: d ⟵ h[-1:-0]
5: nh ⟵ m+t+ char(d)
6: p1 ⟵ nh[7:6]
7: p2 ⟵ nh[5:4]
8: p3 ⟵ nh[3:2]
9: pd ⟵ nh[1:0]
10: x ⟵ nh[7]
11: y ⟵ nh[2]
12: hd1⟵ x+y
13: hx1⟵ dist1.get("hd1")
14: hd2⟵ nh[6:3]
15: hx2⟵ dist2.get("hd2")
16: hxhs⟵ hx1+hx2+pd
```

The converted hexadecimal code for each segment of the sexigeminal code will be converted followed by hexadecimal athematic to produce cumulative location hexadecimal code. List of p1, p2 and p3 will produce various state of the h_s code. Combination of these state and their transaction is one such evaluation problem that can be solved using Hidden Markova model (hmm).

A though deduction of the above created h_s codes, outline the following outcomes such as: firstly the number of observations are 90; secondly there are 17 state pair whose combination will produce an observation and thirdly the produced hidden markov model will work with 19 hidden state. The sequence of the state will produce the unique h_s code and its maximum probability to have a particular observation will be calculated by hmm. Alike any markov model, markov assumption is also applicable in this presented model i.e. for the prediction of future state only the present state matter, there is no correlation of past state with future state. Figure 5 present state transition probability of 6 orginal states forming the h_s code.




G
.58
.78
O
.47
.51
Y
.73
.51
C
.27
I
.49
.55
.45
S
Y
.58
G
.4
O
.49
C
.6
.51
.19
.3
.05
.54
I
S
.4
C
.49
.0
.51
.45
G
.17
I
S
.73
.55
.5
Y
O
.78

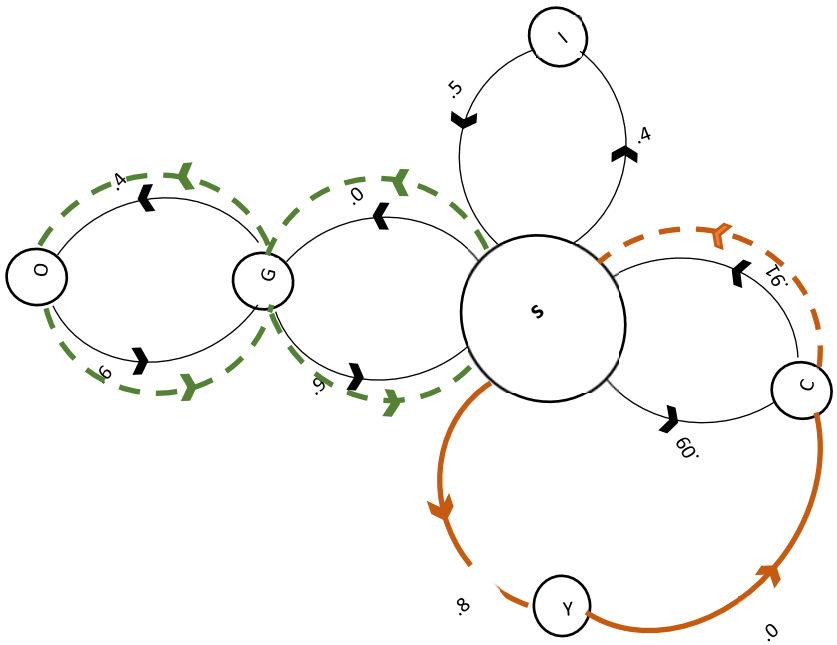


Figure4: State transition chat of different state in 12 bits h_s code

Pair of these states will formulate the 17 unique state of markov model h_s codes which can be defined as follows :

$S = s_1,s_2,s_3,\ldots\ldots s_N$ {where S Є U (Universal Set) where N = 17}

$T = t_{11}, t_{12}, t_{13}, \ldots\ldots\ldots\ldots t_{N1},\ldots\ldots\ldots t_{NM}$ {where T Є transition probability matrix, where each t_{ij} represent the probability of state i to transit to state j.

$\Pi = \pi_1, \pi_2, \ldots\ldots\ldots\ldots \pi_N$ Π Є to an initial probability distribution over the state, π_i, N = 17

This markov chain will further designed HMM λ= (T, Є,π) for hs codes and will be defined as following:

$T = t_{11}, t_{12}, t_{13}, \ldots\ldots\ldots\ldots t_{N1},\ldots\ldots\ldots t_{NM}$ A transition probability matrix T, where each t_{ij} represent the probability of state i to transit to state j. Summation of each row and column of the transition matrix should be 1; N=M = 17

$Є = Є_i(\phi_t)$ Emission probability of observation being generated at state i

$\Phi = \phi_1, \phi_2 \ldots\ldots\ldots \phi_T$ Set of T observation from the set of possible observation at a state

$S = s_1,s_2,s_3,\ldots\ldots s_N$ a set on N state; here N = 17

$\Pi = \pi_1, \pi_2, \ldots\ldots\ldots\ldots \pi_N$ an initial probability distribution over the state, π_i is the probability that the markov chain will start at state i and if $\pi_j = 0$ then state j can be never be initial state; here N = 17

This defined HMM λ= (T, Є,π) will be formulating the mathematical base for the derivation of the hs code and finally the HhI –S indexer.

7. Conclusion.

Various different research has been undertaken to improve the search efficiency of Top-K spatial keyword based query models. Following the two fashion that is keyword index first or spatial index first present the analysis with better result depending upon the query inputted. In contrast to the presented techniques this paper present a new data structure Hy-S tree which present the latitude/ longitude value in quad tree structure. With each node as a unique pair of character value for location. Transitions from character pair to pair assist in deriving hidden relationship and also traversal to the next neighboring node. The h_s code has converted 90 code into unique code with 61 numerical values associated with code hence compressing the size of table by 30 percent. In comparison with the commercial geo-hash technique which uses 9 bit code, the purposed h_s code can be further reduce into 8 bit and follow the similar computation and reducing the total throughput of the system.

Acknowledgements

This work is presented under my PhD research work at the University of petroleum and Energy studies. I thank my Guide Dr. Kingshuk Srivatava and Co- guide Dr, D.K. Gupta for their guidance and motivation to conduct this research. We thank School of computer science and School of Engineering for providing us with relevant resource to conduct this study

References

[1] A. Cary, O. Wolfson, and N. Rishe, “Efficient and scalable method for processing top-k spatial boolean queries,” in SSDBM, 2010, pp. 87–95.
[2] Alvares, L., Bogorny, V., Kuijpers, B., de Macedo, J., Moelans, B., Vaisman, A.: A Model for Enriching Trajectories with Semantic Geographical Information. In: GIS, pp. 1–8 (2007)
[3] B. Costes and J. Perret, “A hidden Markov model for matching spatial networks,” Journal of Spatial Information Science, no. 18, Jun. 2019, doi: 10.5311/josis.2019.18.489.
[4] Cao, X., Cong, G., Jensen, C., Ooi, B.: Collective Spatial Keyword Querying. In: SIGMOD (2011)
[5] Cao, X., Cong, G., Jensen, C.S.: Retrieving top-k prestige-based relevant spatial Web objects. Proc. VLDB Endowment 3(1-2), 373–384 (2010)
[6] Christoforaki, M., He, J., Dimopoulos, C., Markowetz, A., Suel, T.: Text Vs. Space: Efficient Geo-Search Query Processing. In: Proceedings of the 20Th ACM International Conference on Informationand Knowledge Management, pp. 423–432 (2011)
[7] Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial Web objects. Proc.VLDB Endowment 2(1), 337–348 (2009)
[8] C. Zhang, Y. Zhang, W. Zhang, and X. Lin, “Inverted Linear Quadtree: Efficient Top K Spatial Keyword Search,” IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 7, pp. 1706–1721, Jul. 2016.
[9] D. A. Griffith, M. M. Fischer, and J. LeSage, “The spatial autocorrelation problem in spatial interaction modelling: a comparison of two common solutions,” Letters in Spatial and Resource Sciences, vol. 10, no. 1, pp. 75–86, Jun. 2016, doi: 10.1007/s12076-016-0172-8.
[10] De Felipe, I., Hristidis, V., Rishe, N.: Keyword Search on Spatial Databases. In: ICDE, pp. 656–665(2008)
[11] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen, “Joint top k spatial keyword query processing,” TKDE, 2011.
[12] D. Zhang, K.-L. Tan, and A. K. H. Tung, “Scalable top-k spatial keyword search,” in EDBT, 2013, pp. 359–370
[13] G. Cong, C. S. Jensen, and D. Wu, “Efficient retrieval of the top-k most relevant spatial web objects,” PVLDB, vol. 2, no. 1, 2009.
[14] Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory Pattern Mining. In: SIGKDD, pp. 330–339 (2007)
[15] G. Li, J. Feng, and J. Xu, “Desks: Direction-aware spatial keyword search,” in ICDE, 2012.
[16] H.-J. Hong, G.-M. Chiu, and W.-Y. Tsai, “A single quadtree-based algorithm for top-k spatial keyword query,” Pervasive and Mobile Computing, vol. 42, pp. 93–107, Dec. 2017
[17] H. Georgiou, N. Pelekis, S. Sideridis, D. Scarlatti, and Y. Theodoridis, “Semantic-aware aircraft trajectory prediction using flight plans,” International Journal of Data Science and Analytics, Mar. 2019.
[18] I. D. Felipe, V. Hristidis, and N. Rishe, “Keyword search on spatial databases,” in ICDE, 2008.
[19] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørv°ag, “Efficient processing of top-k spatial keyword queries,” in SSTD, 2011.
[20] K. Zheng, B. Zheng, J. Xu, G. Liu, A. Liu, and Z. Li, “Popularity-aware spatial keyword search on activity trajectories,” World Wide Web, vol. 20, no. 4, pp. 749–773, Sep. 2016, doi: 10.1007/s11280-016-0414-0
[21] Lu, J., Lu, Y., Cong, G.: Reverse Spatial and Textual K Nearest Neighbor Search. In: SIGMOD (2011)
[22] M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, and T. Suel, “Text vs. space: efficient geo-search query processing,” in CIKM, 2011.
[23] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, “Processing spatial- keyword (sk) queries in geographic information retrieval (gir) systems,” in SSDBM, 2007.
[24] Singh, B. & Srivastava, K. & Gupta, D.. Future prospects and challenges in geospatial database for handling of big data concept: A review. International Journal of Recent Technology and Engineering. 7. 140-144. April 2019
[25] Srivastava, K., Sridhar, P.S.V.S. and Dehwal, A. 'Data integration challenges and solutions: a study', International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, No. 7, July 2012
[26] Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., Aberer, K.: Semitri: a Framework for Semantic Annotation of Heterogeneous Trajectories. In: EDBT, pp. 259–270 (2011)
[27] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma, “Hybrid index structures for location-based web search,” in CIKM, 2005.
[28] Wu, D., Yiu, M., Jensen, C., Cong, G.: Efficient Continuously Moving Top-K Spatial Keyword Query Processing. In: ICDE (2011)
[29] Zhou, Y., Xie, X.,Wang, C., Gong, Y., Ma,W.: Hybrid Index Structures for Location-Based Web Search. In: CIKM, pp. 155–162 (2005)
[30] Z. Li, Y. Li, and M. L. Yiu, “A Spatial Insight for UGC Apps: Fast Similarity Search on Keyword-Induced Point Groups,” in 2019 20th IEEE International Conference on Mobile Data Management (MDM), 2019
[31] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. L. Lee, and X. Wang, “Ir- tree: An efficient index for geographic document search,” IEEE Trans. Knowl. Data Eng., vol. 23, no. 4, pp. 585–599, 2011
[32] Adam Hadraba; DIPLOMA THESIS; Inverted index implementation
[33] Chuleerat Jaruskulchai and Canasai Kruengkrai; Building Inverted Files ThroughEfficient Dynamic Hashing