ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Web Personalisation Using ANN for Query System Optimization

[I]Richa Soni, [II]Gurpreet Kaur
[I]Student, Dept. of CSE, Chandigarh University, Gharuan, Mohali, India
[II]Asstt. Prof., Dept. of CSE Chandigarh University, Gharuan, Mohali,India,

Abstract
Data mining is the powerful new technology with great potential to analyze important information in the data base .The application area of web usage mining is web personalisation. Web personalization is provide the user the relevant information from a large data set. Here we focus on the query system to find the most appropriate result. We suggest a simple modification to the Kd-tree search algorithm for nearest neighbour search resulting in an improved performance of spatial clustering algorithm .To prove that our approach is better the proposed algorithm is compared with the traditional algorithm (Kd Tree) and also investigate the brute force algo .Approximate Nearest neighbour search (ANN), also known as closest point search which is an optimization problem for finding closest points in metric spaces. Brute force is a trivial but very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement. Brute force algo requires no preprocessing phase .A Brute-force algorithm for string matching problem has two inputs to be considered: pattern and text. A k-d tree is a data structure known for organizing some number of points in a space with k dimensions. K-d trees are very useful for range and nearest neighbour searches. In this paper, we studied k-d tree algorithm and implement to our proposed algo to show the better results and than compared to traditional algorithm. An approximate nearest neighbour is to evaluate and compare the efficiency of the data structure when applied on a particular number of data points, and focus on execution time. The work performed is to enhance the performance of kd tree and compared to traditional algo to obtain the optimal results. The aim of the algorithm is to make faster, more accurate and efficient data structure primarily depends on a particular data set. We have implemented a new modified k-d tree for better performance which can be applied for the enhancement of query system in the web personalisation.

Keywords
Data Mining, Data Structure, Range Searching, Nearest neighbour, Approximate K-NN, K-d tree, Brute-force, Spatial clustering, Web Personalisation.

I. Introduction

A focus of this research is to improve performance of the kd Tree approach and to demonstrate its performance in a real-world problem. Another objective is to investigate the different algo for the range query system.There are a large number of methods, techniques and algorithms that organize, manage, and maintain objects in a structured manner. The purpose of approximate k-nearest neighbour(ANN) is to calculate the nearest neighbour on the basis of value of k, that specifies how many nearest neighbours are to be considered to define the class of a sample data point[1]. The k-nearest neighbour join combines each point of one point set with its k nearest neighbours [2]. The general model of a ANN query is that the user gives many query types such a point query in multidimensional space and a distance metric for measuring distances between points in this space. The system is then tried to find the K closest or nearest answers in the database from the submitted query (i.e. query point) to give the most nearest result. Generally distance metrics may include: Euclidean distance [3]. Given a set of n points in a d-dimensional space, the k-d tree is constructed recursively as follows. First, one finds a median of the values of the ith coordinates of the points (initially, i = 1). That is, a value M is computed, so that at least 50% of the points have their ith coordinate greater or equal to M, while at least 50% of the points have their coordinate smaller than or equal to M. The value of x is stored, and the set V is partitioned into VL and VR , where VL contains only the points with their ith coordinate smaller than or equal to M, and |VR| = |VL |±1. The process is then repeated recursively on both VL and VR, with i replaced by i + 1 (or 1, if i = d). When the set of points at a node has size 1, the recursion stops. The ANN implementation can be done using k-d tree algorithm increase the speed of basic ANN algorithm [4], [5].The simplest version of the ANN algorithm is the 'Brute Force' implementation and consists of three stages. The first stage is to calculate all of the `distances' from each query point to every reference point in the training set. The second stage is to sort these distances and select the k objects that are the closest from which the third and final stage of classification can be performed. More formally,ANN finds the K closest (or most similar) points to a query point among N points in a d- dimensional attribute (or feature) space. K is the number of neighbours that are considered from a training data set [1]

A. Data Mining

Data mining is the process of discovering meaningful, new correlation patterns and trends by shifting through large amount of data stored in repositories, using pattern recognition techniques as well as statistical and mathematical techniques. Data mining is that it scans through a large volume of data to discover patterns and correlations between attributes. Thus, though there are techniques like clustering, decision trees, etc., existing in different disciplines, these are not readily applicable to data mining as they are not designed to handle large amounts of data (Virmani A.1998). Data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

Following can be done using Data mining process [7].

i. Data collection

In this stage, data is collected from a various sources. The usage data is collected from a range of possible sources after which, their contents and structure is recognized.

ii. Data Pre-Processing

The data collected during first stage is usually diverse and voluminous. Therefore it is necessary to preprocess it by filtering unnecessary and irrelevant data, predicting and filling in missing values, removing noise, transforming it into more useful format, and resolving the inconsistencies.

iii. Knowledge Discovery

This stage employs machine learning and statistical methods on pre-processed web data in order to extract the patterns of website usage. There are four major machine learning approaches that are often found in literature: clustering, classification, association discovery and sequential pattern discovery [10].

iv. Knowledge Post-processing

To achieve personalization, it is essential to apply post-processing to the knowledge obtained after the passage of raw data through previous four stages. The outcome of post-processing is used by human experts who act accordingly to accomplish the personalization task.

B. Range Searching

At first sight it seems that database has little to do with geometry. The queries about data in database can be interpreted geometrically. In this case the records in the database are transformed into points in multidimensional space and the queries about records are transformed into the queries over this set of points. Every point in the space will have some information of person associated with it. Consider the example of the database of personal administration where the general information of each employee is stored. Consider an example of query where we want to report all employees born between 2014 and 1989, who earns between Rs.30000 and Rs.40000 per month. The query will report all the points that whose first co-ordinate lies between 2014 and 1989, and second co-ordinate lies between 30000 and 40000. In general if we are interested in answering queries on d -fields of the records in our database, we transforms the records to points in d-dimensional space. Such a query is called rectangular range query, or an orthogonal range query.

C. Data Structure

Data structures are ways to organize data (information). Data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. Data structures are a method of representing of logical relationships between individual data elements related to the solution of a given problem. Data structures are the most convenient way to handle data of different types including abstract data type for a known problem. The components of data can be organized and records can be maintained. Further, the record formation leads to the development of abstract data type and database systems. In data structures, we also have to decide on the storage, retrieval and operation that should be carried out between logically related items. For example, the data must be stored in memory in computer understandable format, i.e. 0 and 1 and the data stored must be retrieved in human-understandable format, i.e. ASCII. In order to transform data various operations have to be performed [8].

D. ANN

Approximate K-nearest neighbour uses the training set directly to classify an input when an input is given. When using a k nearest neighbour algorithm on an input with d attributes the input is classified by taking a majority vote of the k (where k is some user specified constant) closest training records across all d attributes. "Closest" as used means the distance an attribute is away from the same attribute of the training set, using some specified similarity metric. Approximate K nearest neighbour (ANN) in which nearest neighbour is calculated on the basis of value of k, that specifies how many nearest neighbours are to be considered to define class of a sample data point. The training points are assigned weights according to their distances from sample data point. But still, the computational complexity and
memory requirements remain the main concern always. The general model of a ANN query is that the user gives many query types such a point query in multidimensional space and a distance metric for measuring distances between points in this space [13]. The system is then tried to find the K closest or nearest answers in the database from the submitted query (i.e. query point). Generally distance metrics may include: Euclidean distance, Manhattan distance, etc. Approximate K-nearest neighbour algorithm (ANN) is part of supervised learning that has been used in many applications in the field of data mining, statistical pattern recognition and many others. It is usual to use the Euclidean distance. The algorithm on how to compute the approximate K-nearest neighbors is as follows:

1. Determine the parameter K = number of nearest neighbors beforehand.
2. Calculate the distance between the query-instance and all the training samples. You can use any distance algorithm.
3. Sort the distances for all the training samples and determine the nearest neighbors based on the K-th minimum distance.
4. Since this is supervised learning, get all the Categories of your training data for the sorted value which fall under K.
5. Use the majority of nearest neighbours as the prediction value of the query instance [14].
6. Approximate Nearest Neighbour (ANN)

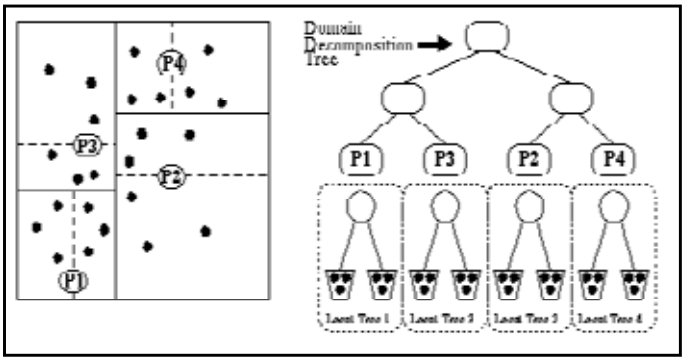
In some applications it may be acceptable to retrieve a "good guess" of the nearest neighbor. In those cases, we can use an algorithm which doesn't guarantee to return the actual nearest neighbour in every case, in return for improved speed or memory savings. Often such an algorithm will find the nearest neighbour in a majority of cases, but this depends strongly on the dataset being queried [15].

E. KD Tree

A k-d tree, or k-dimensional tree, is a data structure used for organizing some number of points in a space with k dimensions. It is a binary search tree with other constraints imposed on it. k-d trees are very useful for range and nearest neighbour searches.

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

The root-cell of this tree represents the entire simulation volume. The other cells represent rectangular sub-volumes that contain the mass, center-of-mass, and quadrupole moment of their enclosed regions. It was one of the early structures used for indexing in multiple dimensions. Each level of k-d tree partitions the space into two partitions, the partitioning is done along one dimension of the node at the top level of the tree, along another dimension in nodes at the next level, and so on, iterating through the dimensions. The partitioning proceeds in such a way that, at each node, approximately one half of the points stored in the sub tree fall on one side, and one half fall on the other. Partitioning stops when a node has less than a given maximum number of points [16].



Their purpose is always to hierarchically decompose space into a relatively small number of cells such that no cell contains too many input objects. This provides a fast way to access any input object by position. We traverse down the hierarchy until we find the cell containing the object.

Typical algorithms construct k-d trees by partitioning point sets recursively along with different dimensions.

Each node in the tree is defined by a plane through one of the dimensions that partitions the set of points into left/right (or up/down) sets, each with half the points of the parent node. These children are again partitioned into equal halves, using planes through a different dimension. Partitioning stops after log n levels, with each point in its own leaf cell [17]. k-d Tree Algorithm The k-d tree is a binary tree in which every node is a k dimensional point. Every non-leaf node can be thought of as implicitly generating a splitting hyper-plane that divides the space into two parts, known as half-spaces. Points to the left of this hyper-plane represent the left sub tree of that node and points right of the hyper-plane are represented by the right sub tree. The hyper-plane direction is chosen in the following way: every node in the tree is associated with one of the k dimensions, with the hyper-plane perpendicular to that dimension's axis. So, for example, if for a particular split the "x" axis is chosen, all points in the sub tree with a smaller "x" value than the node will appear in the left sub tree and all points with larger "x" value will be in the right sub tree. In such a case, the hyper-plane would be set by the x-value of the point, and its normal would be the unit x-axis [18]. The nearest neighbour search (NN) algorithm aims to find the point in the tree that is nearest to a given input point. This search can be done efficiently by using the tree properties to quickly eliminate large portions of the search space. Searching for a nearest neighbour in a k-d tree proceeds as follows:

1. Starting with the root node, the algorithm moves down the tree recursively, in the same way that it would if the search point were being inserted (i.e. it goes left or right depending on whether the point is less than or greater than the current node in the split dimension).
2. Once the algorithm reaches a leaf node, it saves that node point as the "current best".
3. The algorithm unwinds the recursion of the tree, performing the following steps at each node:

1. If the current node is closer than the current best, then it becomes the current best.
2. The algorithm checks whether there could be any points on the other side of the splitting plane that are closer to the search point than the current best. In concept, this is done by intersecting the splitting hyper-plane with a hyper-sphere around the search point that has a radius equal to the current nearest distance. Since the hyperplanes are all axis-aligned this is implemented as a simple comparison to see whether the difference between the splitting coordinate of the search point and current node is less than the distance (overall coordinates) from the search point to the current best.

1. If the hyper-sphere crosses the plane, there could be nearer points on the other side of the plane, so the algorithm must move down the other branch of the tree from the current node looking for closer points, following the same recursive process as the entire search.
2. If the hyper-sphere doesn't intersect the splitting plane, then the algorithm continues walking up the tree, and the entire branch on the other side of that node is eliminated. When the algorithm finishes this process for the root node, then the search is complete.

G. Brute Force

Approximate k-nearest neighbour (ANN) search using a brute force approach as well as with the help of the k-d tree will be used to reach of the main objective of this research (i.e. to speed-up K-nearest neighbour searches). Brute-force search or exhaustive search is a very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement [19].

1. Compute all the distances between the query point and reference points.
2. Sort the computed distances.
3. Select the k reference points with the smallest distances.
4. Classification vote by k nearest objects.
5. Repeat steps (1 to 4) for all query points.

II. Experimental Results and Outputs

In this paper, presented experimental results was conducted for computing execution time for nearest neighbours, with number of data set points assuming are organized in 2 dimensions. In first experiment,as shown in fig 3 the implementation of brute force is done so as to investigate the algorithm of query system. In second experiment, as shown in figure 4. Analyzes the traditional kd Tree algorithm that computes execution time between randomly generated number of data set points. Number of data set points is 10. Enter range one number after the another: (min, max) numbers are 1 to 10. Randomly data set of 10 points is generated and the distance in nearest neighbour & execution time (in microseconds) is computed by traditional kd Tree algorithm. In third experiment, as shown in figure 4. We presented an optimised k-d tree algorithm that computes the execution time between number of data set points. Number of data set points in file is 10. We entered two co-ordinates of the point and execution time (in microseconds) is computed by the k-d tree nearest neighbour algorithm.

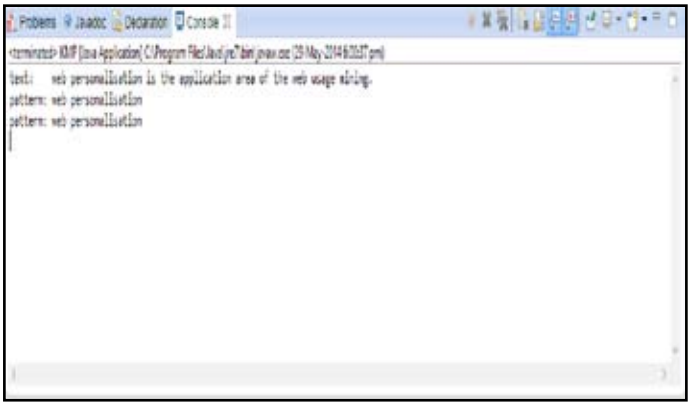


Fig. 3: Implementation of Brute Force algorithm

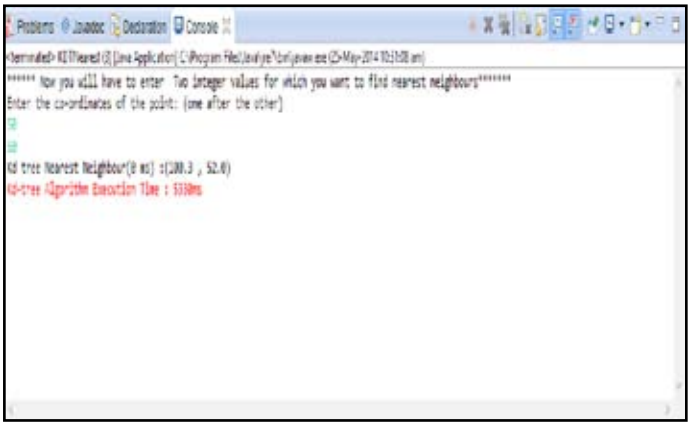


Fig. 4 : execution time of traditional KD Tree

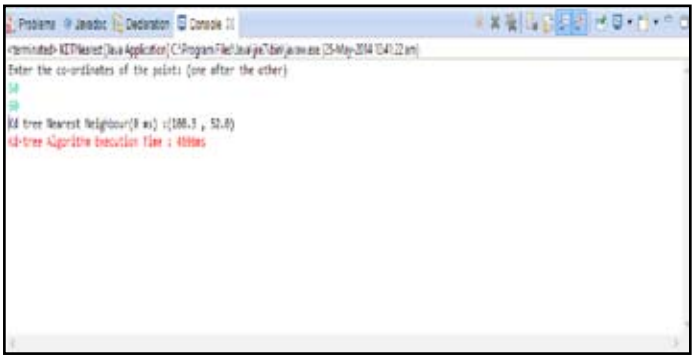


Fig. 5: Execution time of Optimised KD Tree

A. Table

Given tables show the execution time of both the algorithms when number of data set points are computed. The execution time required by both approaches was different. Distance found by both approaches with the same points. The optimised k-d Tree algorithm needs less number of seconds in comparing with traditional kd Tree approach.

Table 1: Results of optimised kd Tree

Coordinate values		Results	Execution Time (T) ms
0	10	(4.7,11.1)	9581
10	20	(7.8,12.3)	8034
34	54	(20.5,7.9)	5080
50	60	(100.3,52.0)	4596
2000	20	(105.5,30.9)	5720
4	8	(4.9,5.8)	6907
10	1	(6.7,3.3)	6767
7	1	(5.1,1.2)	4002
0.80	3	(3.3,1.5)	11112
14	12	(7.8,12.3)	3564

Table 2: Results of Traditional kd Tree

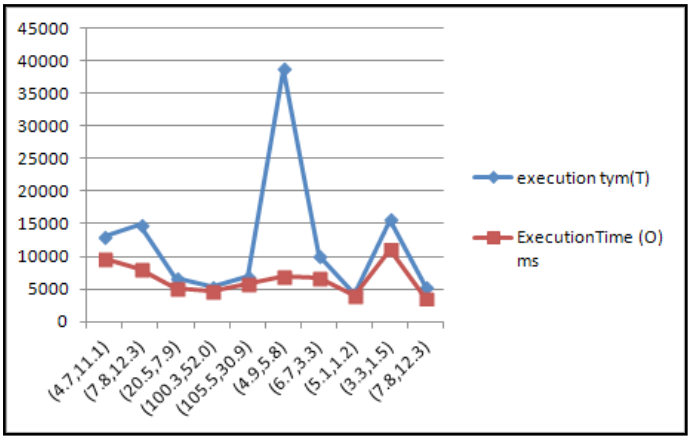
Coordinate values		Results	Execution Time (T) ms
0	10	(4.7,11.1)	12994
10	20	(7.8,12.3)	14864
34	54	(20.5,7.9)	6628
50	60	(100.3,52.0)	5330
2000	20	(105.5,30.9)	6923
4	8	(4.9,5.8)	38775
10	1	(6.7,3.3)	10034
7	1	(5.1,1.2)	4329
0.80	3	(3.3,1.5)	15645
14	12	(7.8,12.3)	5252

Table 3: Comparision of Traditional(T) and Optimised kd tree(O)

Results	Execution Time (T) ms	Execution Time (O) ms
(4.7,11.1)	12994	9581
(7.8,12.3)	14864	8034
(20.5,7.9)	6628	5080
(100.3,52.0)	5330	4596
(105.5,30.9)	6923	5720
(4.9,5.8)	38775	6907
(6.7,3.3)	10034	6767
(5.1,1.2)	4329	4002
(3.3,1.5)	15645	11112
(7.8,12.3)	5252	3564

The experiments have been performed on a 2.60 GHz PC, with Intel (R) core (TM) i5 -3230 and 2 GB memory. The experiment was conducted for computing distance and execution time for nearest neighbours, with number of data set points assuming are organized in 2 dimensions. The experiment has been performed to compare the performance of both, traditional k-d tree and optimised kd Tree , for data set points in a 2-dimentional space. The execution time required by both approaches was different. Distance found by both approaches with the same points. Better performance was achieved when the optimised k-d tree was used for larger number of data sets for nearest neighbours. It shows the execution time and distance required by traditional k-d tree and Optimised kd Tree when number of data set points are computed. However, number of data set points for nearest neighbours, better performance was observed using the optimised kd tree.

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)



Graph1: Traditional Kd Tree vs Optimised K-d Tree

The optimised k-d Tree algorithm needs less number of seconds in comparing with traditional kd Tree especially when using a large volumes of data in a 2-dimension space.

III. Conclusion

The use of the approximate k-nearest neighbour with K-d Tree data structure and comparing its performance to the traditional approach. The work is done between two techniques and select the best one. The results of the work performed in this paper revealed better performance using the optimised k-d Tree, compared to the traditional K-d Tree approach. The aim of the algorithm is to make faster, more accurate and efficient data structure primarily depends on a particular data set. It can be further expanded as by changing the k-d tree traversal technique. We have proposed a new modified k-d tree. From the implementation of traditional k-d tree and optimised K-d Tree algorithms we now can conclude that methods in optimised k-d tree are comparatively faster than Traditional K-d Tree.

IV. Acknowledgement

I would like to thank my guide Er.Gurpreet Kaur,Assistant Professor in Department of Computer Science and Er Iqbaldeep Kaur, Associate Professor C.U(Panjab) and Engineering at Chandigarh University, Punjab, India for motivating me to do research work on the topic- Web Personalisation using ANN algorithm. I would also like to thank my mother, my father and my sister for their continuous support, cooperation, and guidance throughout my M.Tech work. Moreover I would also like to thank my Professors who were always there at
the need of the hour and provided with all the help and facilities, which I required, for my thesis work.

References

References

[1] *A.N.Pathak," A Study on Selective Data Mining Algorithms", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011.*

[2] *Apostolos Papadopoulos, Department of Informatics Aristotle UniversityThessaloniki, Greece" Performance of Nearest Neighbor Queries in R-trees",1996.*

[3] *D. A. White and R. Jain. Similarity indexing with the SS-tree. In Proceedings of the 12th IEEE International Conference on Data Engineering, pages 516–523, New Orleans, LA, February 1996.*

[4] *David M. Mount , "ANN Programming Manual", University of Maryland, 2010.*

[5] *E.Jan , "Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration", Journal of Software Engineering for Robotics, , 1-12, February 2012.*

[6] *H.Kaiming ," Computing Nearest-Neighbor Fields via Propagation-Assisted KD-Trees",2011.*

[7] *J.You , Key Laboratory of Machine Perception, Peking University," Optimizing kd-trees for scalable visual descriptor indexing",2009.*

[8] *K.Suhas," Introspection of various K-Nearest Neighbor Techniques", Proc. of the Intl. Conf. on Advances in Computing and Communication – ICACC 2013.*

[9] *K.Tapas," An Efficient k-Means Clustering Algorithm: Analysis and Implementation", IEEE Transactions On Pattern Analysis And Machine Intelligence, VOL. 24, NO. 7, JULY 2002.*

[10] *L. Bentley," Multidimensional Binary Search Trees in DatabaseApplications", IEEE Transactions On Software Engineering, VOL. SE-5, NO. 4, JULY 1979.*

[11] *M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In VISSAPP(1), pages 331–340, 2009.*

[12] *M.Marius , David G. Lowe," Fast Approximate Nearest Neighbors With Automatic Algorithm Configuration",2008.*

[13] *Mohammed Otair,"Approximate K-Nearest Neighbour Based Spatial Clustering Using K-D Tree", International Journal of Database Management Systems (IJDMS) Vol.5, No.1, February 2013.*

[14] *Mount D. and Arya S., "ANN: A Library for Approximate Nearest Neighbor Searching", 2005.*

[15] *Nitin Bhatia, Vandana, "Survey of Nearest Neighbor Techniques", International Journal of Computer Science and Information Security, Vol. 8, No. 2., 2010.*

[16] *P. Zezula, Amato G, V. Dohnal, and M. Batko. Similarity Search: The Metric Space Approach. Springer-Verlag, Berlin, Germany, 2006.*

[17] *P.John Authenticated Multistep Nearest Neighbour Search in Data mining International Journal of Communications and Engineering Volume 01– No., Issue: 02 March2012.*

[18] *R. F Sproull, "Refinements to Nearest Neighbor Searching", Technical Report, International,1991*

[19] *R.Jean-Baptiste," A Kd-Tree Algorithm To Discover The Boundary Of A Black box Hypervolume", hal-00816704, version 1 - 22 Apr 2013.*

[20] *Rina Panigrahy, "Nearest Neighbor Search using Kd-trees", citeseerx.ist.psu.edu., 2006.*

[21] *S. Agarwal and P. Niyogi. Stability and generalization of bipartite ranking algorithms. In Proceedings of COLT , pages 32–47, 2005.*

[22] *S. Dhanabal, S. Chandramathi,"A Review of various k-Nearest Neighbor Query Processing", 2011.*

[23] *S. M. Beitzel, E. C. Jensen, A. Chowdhury, and O. Frieder. Varying approaches to topical web query classification. In SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pages 783–784, New York, NY, USA, 2007.*

[24] *S.Jagan," A fast all nearest neighbor algorithm For applications involving large point-clouds", Computers & Graphics 31 ,2007.*

[25] Stoimen"s web log: Computer Algorithms: Brute Force String Matching. Systems, Man Cybernatics, Vol.8, pp 311-313, March 2012.

[26] T. Bailey and A. K. Jain, "A note on Distance weighted k-nearest neighbor rules", IEEE Trans., 1978.

[27] T. L.Wang and D. Shasha. Query processing for distance metrics. In D.McLeod, R. Sacks-Davis, and H. Schek, editors, Proceedings of the 16th International Conference on Very Large Databases, pages 602–613, Brisbane, Australia, August 1990.

[28] V.Deepika ," Comparison of Brute-Force and K-D Tree Algorithm", International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 1, January 2014.

[29] William R. Mark, Gordon Stoll, "Fast kd-tree Construction with an Adaptive Error-Bounded ",2006.

[30] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. J. Mach. Learn. Res., 4:933–969, 2003.