An Efficient Semantic-Based Organization and Similarity Search Method for Internet Data Resources

Peige Ren, Xiaofeng Wang, Hao Sun, Baokang Zhao, Chunqing Wu

An Efficient Semantic-based Organization and Similarity Search Method for Internet Data Resources

Peige Ren, Xiaofeng Wang, Hao Sun, Baokang Zhao, Chunqing Wu

School of Computer Science
National University of Defense Technology
Changsha, Hunan, CHINA

renpeige@163.com, xf_wang@nudt.edu.cn, sunhao4257@gmail.com, {bkzhao, chunqingwu}@nudt.edu.cn

Abstract. A large number of data resources with different types are appearing in the internet with the development of information technology, and some negative ones have done harm to our society and citizens. In order to insure the harmony of the society, it is important to discovery the bad resources from the heterogeneous massive data resources in the cyberspace, the internet resource discovery has attracted increasing attention. In this paper, we present the iHash method, a semantic-based organization and similarity search method for internet data resources. First, the iHash normalizes the internet data objects into a high-dimensional feature space, solving the "feature explosion" problem of the feature space; second, we partition the high-dimensional data in the feature space according to clustering method, transform the data clusters into regular shapes, and use the Pyramid-similar method to organize the high-dimensional data; finally, we realize the range and kNN queries based on our method. At last we discuss the performance evaluation of the iHash method and find it performs efficiently for similarity search.

Keywords. feature space; high-dimensional index; similarity search

1. Introduction

With the rapid expansion of information technology and the popularization of the Internet throughout the world, the data type and amount in the internet is growing in amazing speed, there are more and more data resources with different type appearing in the cyberspace, such as videos, digital images, text documents, etc. But some negative resources are harming our society and citizens, such as violent videos, pornographic pictures, reactionary remarks or bad public sentiments. To insure the harmony and development of the society, monitoring and discovering negative data resources from the heterogeneous massive data resources have become more and more important. To organize and mine internet data resources effectively, semantic-based similarity search has been advocated; while for the massiveness and heterogeneity of the internet data resources, it's difficult to discovery the negative data resources quickly and accurately.

The data resources in the internet may contain thousands of features, for discovering all resources that are semantic-similar to a given query, traditional methods represent the data objects as points in a high-dimensional feature space and use a distance function to define the similarity of two objects. But the

data presented by high-dimensional feature space led to the "curse of dimensionality" problems. These problems are mainly in the following aspects [1-3]: 1. The data in the high dimensional feature space is very sparse, so it is difficult to organize and manage the data during the similarity search in the high dimensional feature space; 2. In the high dimensional feature space there is a tendency for data objects to be nearly equidistant to a given query object, so it's not easy to find the data objects which are have similar semantic to a given query data; 3. In high dimensional space, the overlap degree of classic high-dimensional index methods proposed before usually became large with the increase of the dimension, which increased the access path and the query cost during the similarity search.

So far, there is a long stream of research on realizing the semantic-based similarity search in high dimensional feature space, and people have present a number of high-dimensional index techniques, such as R-tree [4], KD-tree [5], VP-tree [6], M-tree [7], Pyramid-Technique [8], iDistance [9], etc., but most of them cannot meet people's needs of realizing semantic-based similarity search in the internet environment very well. First of all, the features of internet data objects are various and massive, showing the phenomenon of "explosion of the features", so it is difficult to manage the data objects in a uniform feature space. Secondly, exiting techniques are usually proposed for special purpose, such as the pyramid technique is efficient for window queries over uniform data, but inefficient for clustered data or kNN queries; while the iDistance method is usually efficient for kNN queries, but inefficient for window or range queries; moreover, multi-dimensional indexes such as R-trees and M-trees are inefficient for range queries in high-dimensional feature space, and are not adaptive to data distributions. Besides, traditional kNN query is realized by processing a sequence of range queries R(q, r) with a growing radius r, which is costly and cannot meet the needs of users.

Motivated by the above disadvantages of exiting indexing technologies for high-dimensional feature space, we developed the iHash method, a semantic-based data management and similarity search method. The key contributions of our work are: 1.By normalizing the internet data objects to a feature space of fixed dimension, our method can solve the "explosion of the features" problem of the feature space; 2.By clustering the data objects according to their distribution, and transforming the data clusters into regular shapes, we can use Pyramid-similar method to map high-dimensional data into 1-dimensional values; 3.We realize the range query and kNN query for semantic-based similarity search based on the iHash method. Besides, to improve the performance of kNN queries, we employ a low-cost heuristic to find an estimated distance, and transform the kNN query to a range query with the estimated distance, which can reduce the query cost significantly.

The rest of the paper is organized as follows. Section 2 provides an overview of related work. Section 3 introduces the preliminaries of this paper. Section 4 give an overview of our method-iHash. In Section 5, we realize the range query and kNN query using the iHash. The experimental results are presented in Section 6, and finally we conclude in Section 7.

2. Related work

Semantic-based similarity search has attracted a lot of attention recently. To speed up similarity searches, a number of high-dimensional indexes were developed in previous studies, such as R-tree, KD-tree, VP-tree, M-tree, X-tree [12] and their variations. High-dimensional indexes aims to accelerate the querying efficiency on multidimensional data, the basic idea of this indexes is to organize the underlying data from a global view, and all the dimensionality should be considered synthetically, the nature function of them is pruning-cutting away a lot of useless searching paths. However, most of the

existing indexs suffer from the curses of dimensionality, that is, their similarity search performance degrades dramatically with the increasing of the dimensionality, which cannot meet our needs.

Stefan Berchtold et al. proposed the Pyramid-Technique for similarity search in high-dimensional data spaces. The basic idea of Pyramid-Technique is to partition the hypercube-shaped high-dimensional data space into subspaces, the subspaces are pyramid shaped, the center point of the high-dimensional space is the common top of the pyramids, and the surfaces of data space are bottoms of the pyramids. According to the space partition, the data objects in high-dimensional space can be mapped to 1-dimensiona values and the B+-tree [14] can be used to manage the transformed data. The Pyramid-Technique is effective for uniformly distributed data set. But in actual life, the data objects are usually irregular distributed, and the Pyramid-Technique cannot process irregular distributed data effectively.

Jagadish et al. presented the iDistance method for k-nearest neighbor (kNN) query in a high-dimensional metric space. The basic idea of this method is to cluster data objects firstly and find a reference point (cluster center) for each cluster; each data object is assigned a one-dimensional value according to the distance to its cluster's reference object, and the one-dimensional values are indexed in a B+-tree. The iDistance method is effective for uniformly distributed and irregular distributed data sets, but it cannot support the window query. Besides, according to the analysis in [10, 13], the most efficient method for similarity search in very high dimensional uniformly distributed data space may be the sequential scan method.

Additionally, with respect to the k-nearest neighbor (kNN) query, traditional methods [9] usually execute a series of range queries iteratively until finding k answers, which is costly. The X-tree [15] and SS-tree [16] use different node size to improve the query efficiency, and GHT* [17] method use a hyperplane to divide the data space ad use a heuristic to improve the efficiency of similarity query.

3. Preliminaries

In this section we present a brief introduction of similarity searching in the metric spaces and describe the K-means clustering method, further, we present the Pyramid-Technique, since our method will take advantage of these technologies.

1.1 3.1. Similarity search in metric space

More formally, a metric space can be described as a pair M= (D, dist), where D is the domain of feature values and dist is a distance function with the following properties for all objects x, y, z∈D:

dist(x, y)≥0 (non negativity);

dist(x, y)=dist(y, x)(symmetry);

dist(x, y)≤dist(x, z)+dist(z, y)(triangle inequality);

d(x, y)=0 if x=y (identity).

Here we refer to the data objects in the feature space and two main types of similarity queries:

1) Range query Range (q, r) retrieve all elements that are within distance r to q, that is, {x∈I: dist (q, x)≤r};

2) k-nearest neighbor query kNN(q, k) retrieve the k closest elements to q, that is, retrieve a set $A \subseteq D$ such that $|A| = k$ and $\forall x \in A,\ \forall y \in D - A$, dist (q, x)≤dist(q, y).

1.2 3.2 The Pyramid-Technique

The basic idea of the Pyramid-Technique is to map the high-dimensional data objects in high-dimensional data space into 1-dimensional values, and then use the traditional index structure B+-tree to manage the 1-dimensional values, which is aimed to eliminate the effects of "curse of dimensionality" during similarity search in high-dimensional data space. The Pyramid-Technique first divide hypercube shaped data space into a series of pyramids and ensure a pyramid number for each pyramid. The pyramids share the center point of the high-dimensional data space as their common top, and the bottoms of pyramids are the surfaces of the data space. Then any one high-dimensional data in a pyramid can be mapped to a 1-dimensional value pv_v according to the pyramid number i and height from the high-dimensional data to the pyramid top h_v, that is, $pv_v=i+h_v$ (as shown in Figure 1), and all the 1-dimensional values transformed from the high-dimensional data are indexed by the B+-tree.

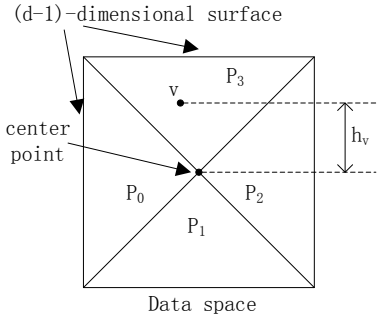

Fig. 1. The Pyramid technique

4. The Overview of the iHash

1.3 4.1. The normalization of the high-dimensional data

To solve the problem of the variousness of the data objects and the inconsistent of the feature's number of different objects, we use a hash function to normalize the data objects, mapping the data objects of different features to a fixed feature space.

First, we express the high-dimensional data object with a feature array, such as a data object A= {A0, A1, …, An-1}, Ai (0≤i≤n-1) denotes a feature of the data object A, Ai.attr denotes the name of the feature Ai, Ai.val denotes the value of Ai. Since the internet data object has its unique feature types and feature values, every data object has exclusive feature array and is mapped to different feature space from others, so it is hard to maintain and manage the mass of data objects in the internet. To map different data objects to the same feature space, here we define a hash function iHash1(Ai.attr), mapping the data object A to a normalized feature array B of M elements according to names of the data object features, and returning the specific locations of the A's features in feature array B. The main purpose of the function iHash1 is to map different kinds of internet data objects into a same feature array structure of a fixed size M, so the internet data objects can be mapped into a same M-dimensional feature space. Specifically, the iHash1 function inputs the name strings of the features, and gets values of range [0, M-1] after several operations, further, we can select an appropriate size of M to avoid the collision during the hash. Besides we define the other hash function iHash2(Ai.val) to map the feature values in A into a set of N-bit values, and store them to corresponding locations in the feature array B. The hash process can be formalized as $B = \text{iHash1}(A_i.\text{attr}) \oplus \text{iHash2}(A_i.\text{val})$, where B is an array of M elements; each element size is N bits; the element values represent the features in A. In this paper, we choose SHA-1algorithm as the iHash2

function, it inputs the values of every feature, gets 160-bit summaries after encoding, then maps the summary into N-bit values and stores them to the corresponding locations in B. The iHash function can map different internet data objects into a same feature space, which is convenient for the operations on the data objects. Figure 2 shows the process of the data normalization.

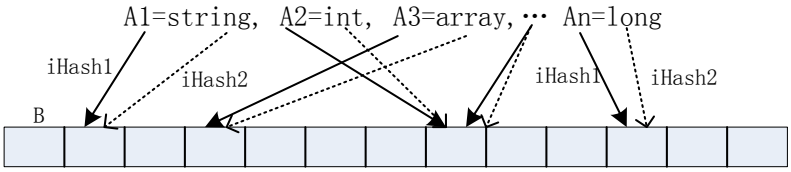

Fig. 2. Data normalization

1.4 4.2. Data clustering and transformation

In real life the data objects in feature space distribute irregularly, they are often clustered or correlated in the space, so space-based partitioning makes the high-dimensional index inefficient (such as the Pyramid Technique). To address the deficiency, a more appropriate data-based partitioning strategy would be used to identify clusters from the space. In our method, we adopted the K-means clustering algorithm to divide the data objects in the high-dimensional data space, ensuring the data objects of similar semantic to be in a same cluster as far as possible. The number of the data clusters K is defined as a tuning parameter, can vary according to specific data distribution and applications.

The data clusters obtained by K-means methods are hypersphere shaped, but the Pyramid-Technique are based on the data spaces are hypercube shaped. To use the Pyramid-similar method to manage the high-dimensional data, we need to transform the data clusters into hypercube shaped. Figure 3 shows the process of data clustering and transformation. After the transformation, the data clusters are a series of hypercube-shaped data subspaces, the value of each side length is 1. And the cluster center is transformed into the center of hypercube, that is, its coordinate is [0.5, 0.5… 0.5]. The transformation is a one-to-one mapping, and as proved in [13], we can get the right answers by operating in the transformed data clusters.

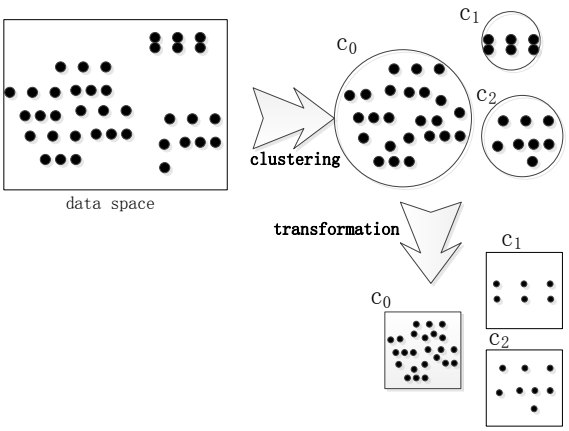

Fig. 3. Data clustering and transformation

1.5 4.3. Space partitioning and data mapping

In section 4.2, the data clusters have been transformed into regular hypercube shaped, so we can use a method that like Pyramid-Technique [8, 18] to partition each data cluster and map the high-dimensional data objects into 1-dimensional values.

For each n-dimensional data cluster, we first take the cluster center as a common top and the surfaces of the data cluster as the bottoms to partition the data cluster into 2n hyper-pyramids, and define pyramid-index i for each pyramid p_i. Next, in each hyper-pyramid, we map the high-dimensional data objects to 1-dimensional values based on the deviations from data objects to the top of the pyramid and the pyramid-index i. The 1-dimensional value of a high-dimensional data object v can be expressed as $p_v=i+h_v$, where h_v is the deviation of v to the pyramid top.

We can observe that the pyramid-index i is an integer and the deviation h_v is a real number in the range [0, 0.5], so all high-dimensional data objects in pyramid p_i will be mapped into the interval [i, i+0.5] and different intervals are disjunct. Besides, we can note that there may be more than one high-dimensional data objects are mapped to a same 1-dimensional value, that is because different data objects with a pyramid may have a same deviation from the pyramid top, so the map is not a one-to-one transformation(As shown in Figure 4).

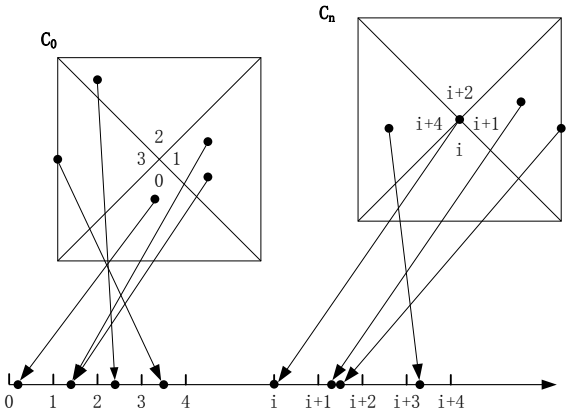

Fig. 4. Space partitioning and data mapping

1.6 4.4. Index Creation

To facilitate speedy semantic-based similarity query, we use the B+-tree to manage the transformed 1-dimensional values. Since having got the pyramid value p_v of any high-dimensional data object v, we can easily manage all the high-dimensional data objects with a B+-tree using their pyramid values as keys. The high-dimensional data are stored in the leaf nodes of the B+-tree, and leaf nodes of the B+-tree are linked to the left and right siblings, so when the search region is changed during a similarity search, it is easy to search the neighboring nodes. Besides, the B+-tree data structure can efficiently realize the insert, delete and update operations of data objects during the similarity search.

5. Semantic-based Similarity Searching

1.7 5.1. Range Query

The Range (q, r) query algorithm is usually the base algorithm for more sophisticated similarity queries. In this paper, the query can be processed in several logical phase. Let R (q, r) be a range query, in the first phase, we determine the data clusters which are overlapped with the query range. Firstly, we transform the query R to T(R), and map the query range to a one-dimensional interval $[q_{low}, q_{high}]$ using the pyramid technique, so we can easily get the clusters that are overlapped with the query range by comparing the query interval $[q_{low}, q_{high}]$ and the intervals of each cluster in the B+-tree, and filter out clusters that do not intersect with the query. In the second phase, we determine the query subspaces. Specifically, we can get the subspaces by computing the intersected regions of the query interval and the pyramid interval in the B+-tree, and mapping the intersected regions to the transformed feature space. In the third phase, we determine the final query answers. The data objects in the query subspaces are the set of candidate data objects, thus we scan the data set and determine the final answers for the query T(R). Since the relationship of the transformed data objects and the original data objects are bijection, so we can easily get the final answers.

1.8 5.2. kNN Query

The traditional kNN query $NN_k(q)$ is realized by processing a sequence of range queries R(q, r) with a growing radius r, but the execution of multiple range query iterations is costly. To reduce the response time and query cost, the kNN query can be realized by the following steps:

(1) estimate an approximate distance of the k-th nearest data object from the query point q;

(2) transform the kNN query to a range query $R(q, r_k)$ where r_k is the estimated distance;

(3) perform the range query $R(q, r_k)$ to retrieve the k closest data objects to query point q.

To obtain the estimated distance r_k, we employ a low-cost heuristic to find k data objects that are near the query point q. The heuristic algorithm can be described as followed.

Firstly, compute the 1-dimensional pyramid value p_q of the query point q and determine the location of p_q in the B+-tree, traverse through the B+-tree leaves alternately, add the first found k data objects to the answer set randomly (for a pyramid value may map to more than one data objects), and compute the r_k value; secondly, examine the data objects x while its pyramid value $p_x \in I$, $I=[p_q-r_k, p_q+r_k]$, if the distance $d(q, x)<r_k$, remove the k-th data object from answer set, add x into answer set, and update r_k and I; thirdly, iterate the second step until the whole interval I has been searched and we will get the final estimated distance r_k. So we can obtain the k nearest neighbors of the query point by processing the range query $R(q, r_k)$.

6. Performance Evaluation

In this section, we analyze the performance of our method by studying the experimental results. All the experiments were performed on a computer with Intel(R) Core (TM) 2 Quad CPU Q8300 2.5GHz and 4GB RAM, each index page is 4kB. The operating system is CentOS 5. In the experiment, we generated 8, 16, 32, 64, 128-dimensional synthetic clustered datasets following a Gaussian distribution with variance of 0.05. Besides, a real data collections was used, we abstracted the color image features

from 68040 pictures to form a feature space using our method. During each experiment, for every experiment result we run 20 times and compute the average value as the final result.

1.9 6.1. Range query

In the range query experiment, we observe the effect of the dimension of the dataset on the response time. We choose synthetic clustered datasets as input data, the input datasets are respectively 8, 16, 24, and 32-dimensional data objects, and the dataset size is 1000000. Besides, we choose the sequential scan and Pyramid-Technique as the references of our method. As shown in Figure 5, we can observe that the response time of three methods increases with the dimensionality, meanwhile Pyramid-Technique and the iHash can overcome the adverse effects of the "curse of dimensionality" in some extent, since they have less response time than the sequential scan method; moreover, the iHash is more efficient than the Pyramid Technique with the increase of the dimensionality, that is because with the increase of the dimensionality of the high-dimensional data space, the data objects in the high-dimensional space will get sparser, and the Pyramid technique will access more useless space. The result show that the iHash method scales well with the change of dimensionality of high-dimensional data space.

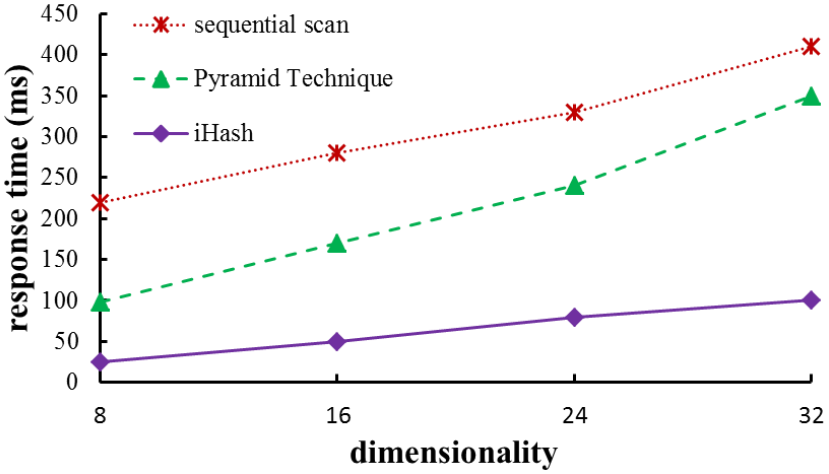

Fig. 5. Effects of dimensionality

1.10 6.2. kNN query

For k-Nearest Neighbor (kNN) query, we use a real data collection that consist of 68040 pictures, we first abstracted the color image features from these pictures to form a 32-dimensional feature space using the hash function presented in section 3.1, and we took the iDistance technique and sequential scan method as the references of our method. In the kNN query experiment, we mainly study the influence of the result set size k on the query response time. As shown in Figure 6, we can observe that the query response time of the three techniques increase with the increasing of the result set size k, and the iDistance and the iHash methods have much better performance compared to the sequential scan method, that is because that during the kNN queries using the two methods, many irrelevant data points to the query point can be excluded from the search range. Besides, the iHash method can do better than the iDistance method in some extend. We can conclude that by efficient space division and estimating the distance of the k-th nearest data point, the iHash can achieve good performance during kNN query.

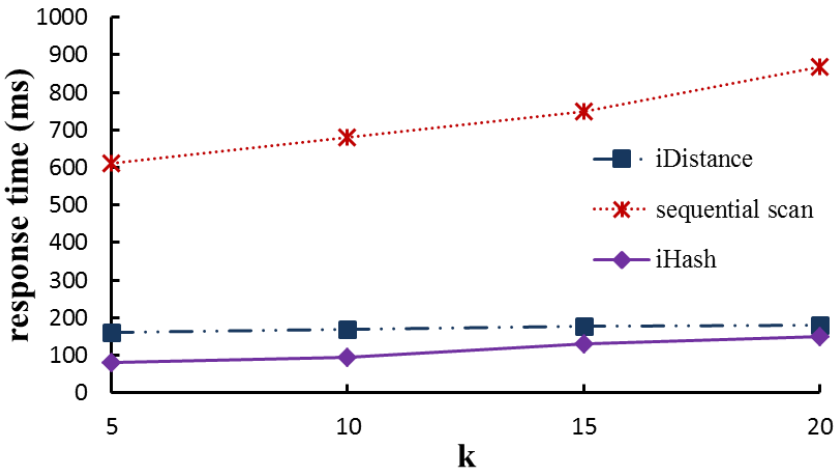

Fig. 6. Effects of result set size

7. Conclusion

With the development of information technology and society, semantic-based similarity search in the internet is of growing importance; the basic idea of similarity search is to efficiently obtain the data objects near to a given query in the high-dimensional data space. In this paper we proposed the iHash method, an efficient semantic-based organization and similarity search method for internet data resources. Our method normalizes the internet data objects into a high dimension feature space; besides, we partition and transform the feature space into hypercube-shaped subspaces so that the Pyramid-similar technique can be applied to index the high-dimensional data objects; finally, we realize the range and kNN queries based on our method. Our experimental evaluation employs synthetic and real data collections and the experimental result shows that our approach performs efficiently both in range queries and kNN queries.

2 Acknowledgment

The work described in this paper is partially supported by the grants of the National Basic Research Program of China (973 project) under Grant No.2009CB320503, 2012CB315906; the project of National Science Foundation of China under grant No. 61070199, 61103189, 61103194, 61103182, 61202488, 61272482;the National High Technology Research and Development Program of China(863 Program) No. 2011AA01A103, 2012AA01A506, 2013AA013505, the Re-search Fund for the Doctoral Program of Higher Education of China under Grant No. 20114307110006, 20124307120032, the program for Changjiang Scholars and Innovative Research Team in University (No.IRT1012), Science and Technology Innovative Research Team in Higher Educational Institutions of Hunan Province("network technology"); and Hunan Province Natural Science Foundation of China (11JJ7003).

3 References

1. CHRISTIAN BÖHM et al., “Searching in High-Dimensional Spaces—Index Structures for Improving the Performance of Multimedia Databases.” ACM Computing Surveys, 2001, 33(3):322-373.
2. EDGAR CHÀVEZ et al., “Searching in Metric Spaces.” ACM Computing Surveys, 2001, 33(3): 273-321.

3. P. Zezula, G. Amato, V. Dohnal, and M. Batko, “Similarity Search: The Metric Space Approach.” Advances in Database Systems, 2006
4. Antonin Guttman. “R-trees: a dynamic index structure for spatial searching.” SIGMOD , 1984, pp. 47-57.
5. Bentley JL. “Multidimensional Binary Search Trees Used for Associative Searching.” Communications of the ACM, 1975, 18(9): 509-517.
6. Fu AW, Chan PM, Cheung YL, Moon YS. “Dynamic vp-Tree Indexing for n-Nearest Neighbor Search Given PairWise Distances.” VLDB, 2000
7. Ciaccia T, Patella M, Zezula P. “M-tree: An Efficient Access Method for Similarity Search in Metric Spaces.” VLDB, 1997, pp. 426-435.
8. Berchtold S, Böhm C, Kriegel HP. “The Pyramid-Technique: Towards Breaking the Curse of Dimensionality.” SIGMOD , 1998, pp. 142-153.
9. H.V. Jagadish, Beng Chin Ooi, et al., “iDistance: An Adaptive B^+-tree Based Indexing Method for Nearest Neighbor Search.” ACM Transactions on Database systems, 2003, pp. 1-34.
10. Rui Zhang, Beng Chin Ooi, Kian-Lee Tan, “Making the Pyramid Technique Robust to Query Types and Workloads.” Proceeding of the 20th International Conference on Data Engineering (ICDE’ 04), 2004
11. A.K. JAIN, M.N. MURTY, P.J. FLYNN, “Data Clustering: A Review.” ACM Computing Surveys, 1999, 31(3): 264-323
12. H.-P. K. S. Berchtold, D. Keim. The x-tree: An index structure for high-dimensional data. In VLDB, 1996
13. Zhang, Rui, Beng Chin Ooi, and K-L. Tan. "Making the pyramid technique robust to query types and workloads." Data Engineering, 2004. Proceedings. 20th International Conference on. IEEE, 2004.
14. Comer D., “The Ubiquitous B-tree.” ACM Computing Surveys, 1979, 11(2): 121-138.
15. Berchtold, Stefan, Daniel A. Keim, and Hans-Peter Kriegel. "The X-tree: An index structure for high-dimensional data." Readings in multimedia computing and networking (2001): 451.
16. White, David A., and Ramesh Jain. "Similarity indexing with the SS-tree." Data Engineering, 1996. Proceedings of the Twelfth International Conference on. IEEE, 1996.
17. Batko, Michal, Claudio Gennaro, and Pavel Zezula. "Similarity grid for searching in metric spaces." Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures. Springer Berlin Heidelberg, 2005. 25-44.
18. Berchtold, Stefan, Christian Boehm, and Hans-Peter Kriegel. "High-dimensional index structure." U.S. Patent No. 6,154,746. 28 Nov. 2000.