Automated Data Discovery in Similarity Score Queries

Fatih Altiparmak[1], Ali Saman Tosun[1,2],
Hakan Ferhatosmanoglu[1], and Ahmet Sacan[1,3]

[1] The Ohio State University, Dept. of Computer Sci. & Eng., Columbus, OH
{altiparm,sacan,hakan}@cse.ohio-state.edu
[2] The University of Texas at San Antonio, Dept of Computer Science
tosun@cs.utsa.edu
[3] Middle East Technical University, Dept. of Computer Eng., Ankara, Turkey

Abstract. A vast amount of information is being stored in scientific databases on the web. The dynamic nature of the scientific data, the cost of providing an up-to-date snapshot of the whole database, and proprietary considerations compel the database owners to hide the original data behind search interfaces. The information is often provided to researchers through similarity-search query interfaces, which limits a proper and focused analysis of the data. In this study, we present systematic methods of data discovery through similarity-score queries in such "uncooperative" databases. The methods are generalized to multi-dimensional data, and to L-p norm distance functions. The accuracy and performance of our methods are demonstrated on synthetic and real-life datasets. The methods developed in this study enable the scientists to obtain the data within the range of their research interests, overcoming the limitations of the similarity-search interface. The results of this study also present implications in data privacy and security areas, where the discovery of the original data is not desired.

1 Introduction

An ever growing amount of information is being served on the Web. Some of this information is in the form of inter-linked HTML pages, which are crawled, indexed, and made accessible by the search engines such as Google (google.com), Yahoo (yahoo.com), and MSN (msn.com). The portion of the Web that is accessible via search engines is termed the *surface Web*. A far greater amount of information is believed to be hidden behind databases whose content is not accessible through static URL links. It was estimated that this *deep Web* contained 7,500 terabytes of data – 500 times larger than the surface Web [5].

The information in the hidden Web is available only as a response to dynamically issued queries to the search interface of the databases. Recent efforts have focused on categorizing these databases at the absence of content summaries [13], or on building meta-search engines that provide a unified search interface to these databases [20]. However, most of these efforts were limited to *text databases* and ignored the *numeric databases* prevalent in scientific repositories.

J.R. Haritsa, R. Kotagiri, and V. Pudi (Eds.): DASFAA 2008, LNCS 4947, pp. 440–451, 2008.

In scientific databases, the data is usually represented as multi-dimensional feature vectors. Due to the nature of the data and the large quantity of information, similarity search has emerged to be the de facto form of query in scientific applications such as high-energy physics [24], geographic information systems (GIS) [6], financial time series databases [14], medical imaging [19, 17], and bioinformatics [15].

The degree of similarity between objects in a database is often quantified by a distance measure, e.g., Euclidean distance, operating on the multi-dimensional data objects or the feature vectors extracted from the data objects. For example, a user may pose a query over a medical database asking for X-rays that are similar to a given X-ray in terms of Euclidean distance of multi-dimensional texture feature vectors [18, 16]. 3D Shape histograms of proteins are used to identify their similarities [1]. Similarity query is usually implemented by finding the closest feature vector(s) to the feature vector of the query data. This type of query is known as nearest neighbor (NN) query [21] and it has been extensively studied in the past [11, 12, 23, 2, 3, 9, 4, 7]. A closely related query is the ϵ-range query where all feature vectors that are within ϵ neighborhood of the query point q are retrieved.

Bandwidth and resource constraints and the continuous nature of the data acquisition itself, whether it be from manual contributions from researchers or automated sensor input, make the maintenance of an up-to-date, downloadable snapshot of the whole database unfeasible. Therefore the database providers limit the data access to the similarity query interface they provide. Still other providers practice this limitation due to privacy or proprietary concerns.

Even though similarity query over the database is one of the first steps of gaining valuable information about the entity under consideration, it is insufficient for further scientific investigation. The scientists often seek to acquire a portion of the database relevant to their research question. In this study, we overcome the limitation imposed by the similarity query interface, and show that the data of interest can be automatically retrieved while minimizing the burden on the resource constraints of the database owners.

The results of this study also have critical implications in database security, where the discovery of the data is not desired by the providers. Specifically, we show that the whole database can be discovered through similarity queries. We give recommendations for preventive measures where privacy or proprietary concerns lead the query interface limitation.

We have previously identified two main models of similarity search queries [22]:

- *Reply Model.* Client queries vector x and database responds with the closest k vectors y_i $(i = 1 \ldots k)$.
- *Score Model.* Client queries vector x and database responds with similarity score $\| x - y \|$, where y is the closest vector in the database to the x.

In this paper, we focus on the Score Model, where a rigorous analysis was missing. The contributions of this paper can be summarized as follows:

- Data discovery through similarity score queries is proven under a general probing strategy.
- A strategy using query histories is developed to improve the efficiency of the data discovery
- The methods are generalized to multi-dimensional case, and to $L_p - norm$ distance measures

2 Methods

In the score model, upon receiving a similarity query x from user, the database responds with the similarity score $\parallel x - y \parallel$ [8], where y is the closest point in the database to x. Assume that l_1, l_2, and u_1, u_2 are the coordinates of the two corners on the same diagonal of the minimum rectangle bounding the region of interest in 2-dimensional space. Further assume that the closest distance between any two points in the database is given as c.

A basic data discovery approach where the database consists of a *single* n-dimensional vector y was given in [22]. We reiterate this basic approach and its proof here in Algorithm 1 for completeness. The function $createVector(d, i, v)$ creates a d dimensional vector which has 0 in all dimensions but v in the i^{th} dimension. The n-dimensional vector y can be discovered using $n + 1$ queries. So, for 2 dimensions, 3 queries would be needed.

Algorithm 1. Algorithm to discover y

1: n = dimensionality of vectors x and y
2: $q_1 = sim\_search([0, 0, ...0])$
3: **for** i= 1 to n **do**
4: $\quad q_2 = sim\_search(createVector(n, i, 1))$
5: $\quad y_i = \frac{q_2^2 - q_1^2 - 1}{-2}$
6: **end for**

Lemma 1. *Algorithm 1 discovers y.*

Proof. Consider i^{th} iteration of the loop. We have $q_1 = (\sum_{k=1}^{n}(y_i)^2)^{1/2}$ and $q_2 = ((y_i - 1)^2 + \sum_{k=1, k\neq i}^{n}(y_i)^2)^{1/2}$. By simple algebra we get $q_2^2 - q_1^2 = -2y_i + 1$. We can solve this equation to find y_i.

Now let us consider a database with large number of tuples. In this case, the basic approach in Algorithm 1 can not be used since queries q_1 and q_2 can return scores based on different vectors in database. The example shown in Figure 1 returns a score of 1 for similarity search $(0, 0)$ and returns a score of $1/2$ for similarity search $(0, 1)$. These are not comparable since their distances are to different nodes.

In the following sections, we first explain our proposed discovery strategy for 2 dimensions using l_2 (Euclidian) distance and then generalize it to n dimensions using l_p norm as the distance metric. For notational clarity, we first consider the

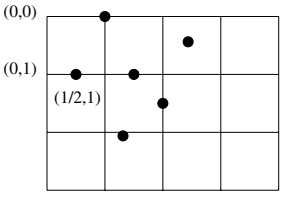

Fig. 1. Discovery of multiple points. Algorithm 1 is insufficient in correctly discovering the data.

problem of discovering the whole database; i.e., all the data points lie within the region of interest. The extension to discovery of a region of the database is trivial and discussed later.

2.1 Discovery of Multiple Data Points in 2 Dimensions

In order to discover every point in the database, closely located queries need to be sent to ensure that all the queries return a score to the same data point. Assume that the whole space is divided into equally spaced squares and that the length of an edge of the square, which is also the distance between two consecutive probes, is $c\prime$. In the best case, $n + 1$=3 probes having the coordinates x,y,x,y+$c\prime$,x+$c\prime$,y are required to return scores to the same data point to guarantee discovering it.

Recall that we have the probes and the associated scores, but do not have any information identifying the data point for the returned score. Therefore, to discover data points when the databases use the score model, we must take advantage of all the information at hand: the returned scores and the distance between probes, $c\prime$. The task becomes finding a probe distance $c\prime$ such that putting a condition on the returned scores for the nearby probe points guarantees that the score returned by each of them is to the same data point. In the following lemma we put a condition, maximum score, on the returned distance for a probe to guarantee that each of its $c\prime$ distanced neighbors returns a score to the same data point.

Lemma 2. *Let the score for a probe $p = (x, y)$ be δ where $\delta \leq \frac{c}{4}$ and the closest point to probe p be $q = (s, t)$. Then the closest point to probes $p_2 = (x + \frac{c}{4}, y)$, $p_3 = (x - \frac{c}{4}, y)$, $p_4 = (x, y + \frac{c}{4})$, and $p_5 = (x, y - \frac{c}{4})$ is $q = (s, t)$.*

Proof. Consider the distance $d(p_2, q)$. By triangle inequality we have $d(p_2, q) \leq d(p_2, p) + d(p, q) \leq \delta + \frac{c}{4}$. Therefore, $d(p_2, q) \leq \frac{c}{4} + \frac{c}{4} \leq \frac{c}{2}$. Since c is the smallest distance between pairs, closest point to probe $p_2 = (x + \frac{c}{4}, y)$ is $q = (s, t)$. Proofs for p_3, p_4, p_5 are similar.

The required distance, $c/4$, and $c\prime$, $\frac{c}{4}$, are selected such that sum of them is $\leq c/2$ and at least one of the corners of the square a data point lies in returns a score $\leq$ the required distance, $c/4$ to the point. Algorithm shown in Figure 2 utilizes lemma 2 to discover all points in a two dimensional space.

Algorithm 2. The General Probe Algorithm

```
1: α1 = ⌈(u1 − l1)/(c/4)⌉ + 1
2: α2 = ⌈(u2 − l2)/(c/4)⌉ + 1
3: for i= 0 to α1 − 1 do
4:    for j= 0 to α2 − 1 do
5:       probe[1] = l1+i c/4
6:       probe[2] = l2+j c/4
7:       dist_{i,j} = dist_search(probe)
8:    end for
9: end for
10: for i= 0 to α1 − 2 do
11:    for j= 0 to α2 − 2 do
12:       if dist_{i,j} ≤ c/4 then
13:          y1 = ((dist^2_{i+1,j} − dist^2_{i,j})/(c/4) − 2(l1 + i c/4) − c/4)/(−2)
14:          y2 = ((dist^2_{i,j+1} − dist^2_{i,j})/(c/4) − 2(l2 + j c/4) − c/4)/(−2)
15:          if y not in database then
16:             save y
17:          end if
18:       end if
19:    end for
20: end for
```

Theorem 1. *The General Probe Algorithm in Algorithm 2 discovers the whole database.*

Proof. To make the proof complete we need to show for each data point that

(1) at least one probe exists such that distance between the probe and the data point is $\leq \frac{c}{4}$ and
(2) algorithm guarantees to find the data point.

The proof for each item will be made separately.

(1) Since voronoi regions cover the whole region, each data point lies in a square having edge of $\frac{c}{4}$. Hence, the longest distance between a data point and the nearest corner of the square it lies into can be $c\sqrt{2}/8$ which is less than $c/4$. Thus, at least one of these probes will return a score $\leq \frac{c}{4}$ to the point.

(2) As stated by Lemma 2, each of the probes with indices $\{i, j\}, \{i+1, j\}$ and $\{i, j+1\}$ returns the distance to the the same data point if the returned distance for $\{i, j\} \leq \frac{c}{4}$. Let us call this data point y. We have $dist_{i,j} = ((l_1 + i\frac{c}{4} - y_1)^2 + (l_2 + j\frac{c}{4} - y_2)^2))^{1/2}$ and $dist_{i+1,j} = ((l_1 + (i+1)\frac{c}{4} - y_1)^2 + (l_2 + j\frac{c}{4} - y_2)^2))^{1/2}$. By simple algebra we get $dist^2_{i+1,j} - dist^2_{i,j} = \frac{c}{4}(-2y_1 + 2(l_1 + i\frac{c}{4}) + \frac{c}{4})$. We can solve this equation to find y_1. We can find y_2 similarly by considering probe with indices $\{i, j+1\}$ instead of $\{i+1, j\}$. Since we have shown that such a probe exist for each data point, the algorithm will discover the whole database.

The General Probe Algorithm divides the space into a grid and queries the database with the corners of this grid. The distance between two consecutive

probes is $c/4$ to guarantee that for each data point, at least one of the probes belonging to the corners of the square that the data point lies in returns a score less than or equal to the required distance, $c/4$. Therefore, there is at least one probe within $c/4$ distance to each point and by using Lemma 2 the point can be discovered.

2.2 A Progressive Probing Algorithm

A very small value for c, the minimum distance between data points, can cause the General Probe Algorithm to generate an infeasible number of probes. As suggested in [22], a progressive probing scheme can be utilized to hierarchically sample the database, and in turn, to discover the data in finer detail as the number of probes increases.

The progressive scheme is expected to be better at discovering more of the database with earlier queries since it spreads the probe points. The total of the indices is used and the level of a probe is calculated by the modula operation. For example, for two dimensions modula of the sum of row number and column number is used for calculating the level of a probe. As an example, *levelno* for a two dimensional space divided into 5 columns and 5 rows is 4 ($\lceil$max row index + 1 + max column index + 1$\rceil$). The probe pattern of progressive discovery for this space is given in Figure 2.

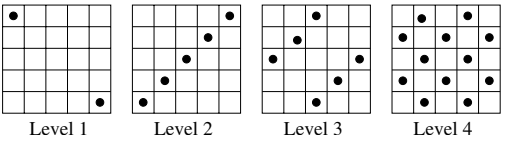

Fig. 2. Levels of Progressive Probing in 2 dimensions

2.3 Exploiting the Query History

The General Probe Algorithm is based on querying the database at each of the interval points. However, some of these probes may be redundant, and it is possible to eliminate such probes based on the information obtained from previous queries. To discover every point in the database, the space can be divided in a way that there exist (n+1) probes for each data point that return score$\leq$ $c/2$ to the point. The returned scores for most of the queries will be larger than c/2, which would guarantee that the nearby probes return scores to the same data point. It is possible to eliminate unnecessary probes based on the information obtained from previous queries.

In Figure 3, the database returns distance value s for a query probe p whose nearest neighbor is the data point y. The $c\prime$ is taken as $c/4$ as found above. Lemma 2 states all probes having distance $c\prime$ to a probe p will return a score to the same data point if the returned score for p is $\leq c/4$. In the proof of this Lemma we showed that all these probes will have at most a score of $c/2$. Since

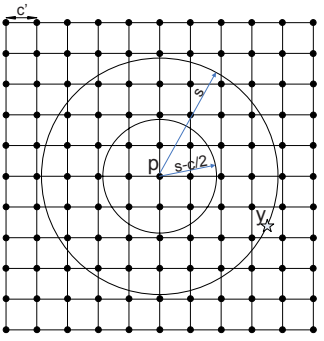

Fig. 3. Elimination of Redundant Probes

there is no other data element within the outer circle, any probe in the inner circle cannot have a score $\leq c/2$. Hence, the probes in the inner circle except p are redundant and can be eliminated.

2.4 Extension to Higher Dimensions and to l_p Norm

We will extend the solution presented for 2 dimensions to n dimensions in three steps. As a first step, we will show that there is a solution for the simple case where the database has only one point. Then the function to calculate the total number of probes in n dimensions is provided. This function depends on the distance metric (l_p norm) and n. The last step is to generalize Lemma 2, on which solution was built, to the case where we have n dimensions and use the l_p norm as the metric.

Definition 1. *l_p norm between n dimensional vectors x and y is defined as* $l_p(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}$.

The solution shown in Algorithm 3 is an extended version of the solution in Algorithm 1. There are two differences between these solutions. The first difference is that we send a value of -1 instead of a value of 1 in the appropriate dimension when we are gathering information about a dimension. The other difference is that while for 2 dimensions using l_2 norm we can find an exact solution for y_i, we can only show that a unique real solution exists for n dimensions using l_p norm.

Algorithm 3. Algorithm to discover y for n dimensions

```
1: q1 = sim_search([0, 0, ...0])
2: for i = 1 to n do
3:     q2 = sim_search(createVector(n, i, −1))
4:     Compute yi
5: end for
```

Theorem 2. *Algorithm in Algorithm 3 discovers y for every l_p.*

Proof. Consider i^{th} iteration of the loop. Using the definition of l_p norm, we have

$$q_1 = (\sum_{j=1}^{n} |y_j|^p)^{1/p}$$

and

$$q_2 = (|y_i + 1|^p + \sum_{j=1, j\neq i}^{n} |y_j|^p)^{1/p}$$

By using above equation we have

$$q_2^p - q_1^p = |y_i + 1|^p - |y_i|^p$$

Left hand side is known since we know q_1, q_2 and p. As q_1 and q_2 are returned for the same data point, we know that this equation has at least one real solution, y_i. Therefore, we need to prove the uniqueness of the solution to discover y_i. We need to show that this is an increasing function in each of the possible 3 intervals, ≥ 0, $[-1, 0)$, and < -1, to prove the uniqueness of the solution.

- $y_i \geq 0$: The expression becomes $(y_i + 1)^p - (y_i)^p$. If we take the derivative with respect to y_i we get

  $$p(y_i + 1)^{p-1} - p(y_i)^{p-1}$$

  Since $p \geq 1$ above expression is always positive. Therefore, in this interval the function is increasing.
- $-1 \leq y_i < 0$: The expression becomes $(y_i + 1)^p - (-y_i)^p$. If we take the derivative with respect to y_i we get

  $$p(y_i + 1)^{p-1} + p(-y_i)^{p-1}$$

  Above expression is always positive therefore, in this interval the function is increasing.
- $y_i < -1$: The expression becomes $(-(y_i + 1))^p - (-y_i)^p$. If we take the derivative with respect to y_i we get

  $$-p(-(y_i + 1))^{p-1} + p(-y_i)^{p-1}$$

  If we reorganize the expression we get

  $$p((-y_i)^{p-1} - (-(y_i + 1))^{p-1})$$

  Since the first term in the parenthesis is always greater than the second one, the derivative is always positive in this interval. Hence, in this interval the function is increasing.

The function is increasing in all of the intervals, thus it is an increasing function. As a result, independent from p we have a unique real solution for y_i and it can be computed by Newton's method.

The total number of probes for n dimensions is found by multiplying the number of probes needed for each dimension separately. The following theorem considers this fact and the gives the method to find the common distance between closest probes, $c\prime$.

Theorem 3. *For the score model, general scheme in n dimensions utilizing* l_p *norm as the distance metric requires* $\prod_{i=1}^{n} \lceil \frac{u_i - l_i}{c\prime} \rceil + 1$ *probes to discover all elements of an n dimensional database where* $c\prime$ *is equal to the minimum of* $c/4$ *and* $\frac{c}{2(n)^{1/p}}$.

Proof. The general scheme divides the whole space into hypercubes having an edge of $c\prime$ which is selected to guarantee that the returned score for at least one of the probes belonging to the corners of the region that a data point lies in is $\leq$ the required distance, $c/4$. To make this guarantee, $c/4$ should be the longest distance between a point and its closest probe. Hence, the diagonal of the hypercube should be $c/2$ and an edge of it, $c\prime$ should be $\frac{c}{2(n)^{1/p}}$. However, we should also consider the requirement that $c\prime$ should be $\leq$ c/4. So, $c\prime$ is the minimum of c/4 and $\frac{c}{2(n)^{1/p}}$ for n dimensions using l_p norm.

Lemma 2 is at the heart of the solution for 2 dimensions using l_2 norm shown in Figure 2. The extended version of this Lemma should be used for the solution for n dimensions using l_p norm. We will not make the proof, instead we will verify that the sum of the required distance, $c/4$ and the $c\prime$ is $\leq c/2$. The value of $c\prime$ for n dimensions using l_p norm is given in Theorem 3 as the minimum of $\frac{c}{2(n)^{1/p}}$ and $c/4$. So, the sum will be less than or equal to $c/2$. As a result, if the distance returned for a probe is less than or equal to $c/4$, all probes which are in the $c\prime$ distance to this probe will return a score to the same data point. In total, there are $2n$ such probes.

The proof outline for proving the existence of a unique real solution when the database contains a large number of data points, and the probe that has a score $\leq c/4$ is considered with its $c\prime$ neighbors is to follow the same path as Theorem 2.

3 Experiments and Results

The methods developed above are applied on four datasets. Three of the datasets are 2-dimensional each of which represents a different type of distribution. The first dataset is latitude and longitude of road crossings in Maryland. This data set is a good example for uniformly distributed data. The second dataset has points mostly clustered in one region. The third skewed dataset is a correlated data typically seen in time series data such as stock price movements [10]. For each of these three data sets, if the data set contained more than 1000 points, we used a subsampled version with 1000 points. The fourth data set is a clinical data obtained from a pharmaceutical company[1]. The data contains measurements of

[1] We wish to thank Pfizer, Inc. for kindly providing the patient dataset.

4 blood ingredients for 244 patients. Half of these patients were suffering from arthritis, while the other half were from a healthy control group.

The General and Progressive methods were applied with query history to eliminate unnecessary probes. Since we do not utilize any distance $> c/2$ while discovering a data point, we used already sent queries to eliminate probes which have scores more than $> c/2$. Results are summarized in Figure 4.

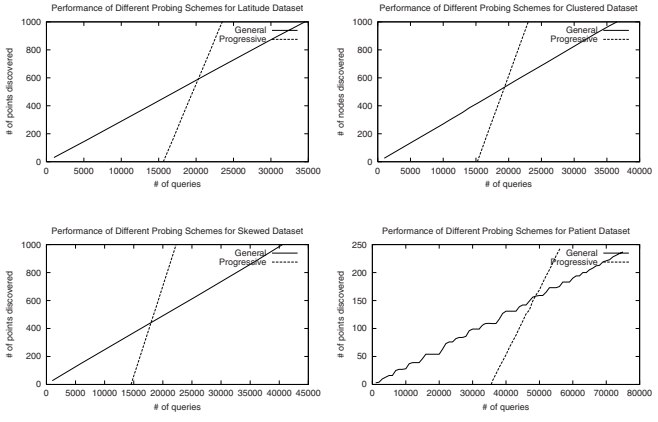

Fig. 4. Performance of General and Progressive probing using history

Because of the leveling strategy used in the progressive model, queries that are next to each other are not sent until the last level. Therefore, this strategy does not find any points until it starts to send the probes on this last level. On the other hand, the distribution of probes resulting from the progressive scheme eliminates more probes than the general scheme. This phenomenon is shown in the graphs in Figure 4. The progressive scheme does not begin to discover points until the general scheme has already discovered about half the database, but it completely discovers the database before the general scheme does.

4 Discussion

Web-based search engines and many biomedical and clinical databases utilize similarity search as their major type of query. In this paper we showed how the data in numeric databases can be discovered through similarity score queries. The methods we have developed were extended to multi-dimensional data, and to l_p-norm distance measures. Using a progressive scheme and exploiting the results of previous queries, we were able to improve the performance over the general probing strategy significantly.

Using the methods we have developed, it is now possible to discover data within a range, or the whole database using results of similarity score queries. This effectively removes the limitation imposed by the query interfaces and lets

the researchers extract the data of their interest through the query interface channel they have been provided with. Instead of downloading an outdated snapshot of the whole database, the researchers can obtain the up-to-date information for the portion of the database that they are interested in.

We believe that the data discovery methods provided here would also relieve the database providers of the burden of providing customized data to each request coming from the researchers. The database providers will not have to spend effort to extend the query interface, or to compile data for individual research interests.

The results of this study also present critical implications for data security [22], where the original data is hidden intentionally, and their discovery is not desired. If this is the case, we have shown that providing a similarity score query interface can not hide the original data. There are certain measures the database owners can practice for detection and prevention of data-discovery attacks, following the results of this study.

A simple data protection mechanism can rely on investigating the number of queries to the same nearest neighbor data point. We have shown that at least $n+1$ probes are required for the discovery of a data point in n-dimensional space. By refusing to reply to requests that cause the number of such probe queries to exceed n, the database system can effectively prevent malicious discovery of the data.

References

[1] Ankerst, G., Kastenmüller, M., Kriegel, H., Seidl, T.: Nearest neighbor classification in 3d protein databases. In: Proc. 7th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB 1999) (1999)

[2] Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching. In: 5th Ann. ACM-SIAM Symposium on Discrete Algorithms, pp. 573–582 (1994)

[3] Berchtold, S., Bohm, C., Keim, D., Kriegel, H.: A cost model for nearest neighbor search in high-dimensional data space. In: Proc. ACM Symp. on Principles of Database Systems, Tuscon, Arizona, June 1997, pp. 78–86 (1997)

[4] Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is nearest neighbor meaningful. In: Int. Conf. on Database Theory, Jerusalem, Israel, January 1999, pp. 217–225 (1999)

[5] BrightPlanet.com. The deep web: Surfacing hidden value (2000) Accessible at, `http://brightplanet.com`

[6] Cheng, X., Dolin, R., Neary, M., Prabhakar, S., Kanth, K.V.R., Wu, D., Agrawal, D., Abbadi, A.E., Freeston, M., Singh, A.K., Smith, T.R., Su, J.: Scalable access within the context of digital libraries. In: Advances in Digital Libraries, pp. 70–81 (1997)

[7] Ciaccia, P., Patella, M.: PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In: Proc. Int. Conf. Data Engineering, San Diego, California, March 2000, pp. 244–255 (2000)

[8] Du, W., Atallah, M.: Protocols for secure remote database access with approximate matching. In: 7th ACM Conference of Computer and Communications Security (ACMCSS 2000), The First Workshop on Security and Privacy in E-commerce (2000)

[9] Ferhatosmanoglu, H., Stanoi, I., Agrawal, D., Abbadi, A.E.: Constrained nearest neighbor queries. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, Springer, Heidelberg (2001)
[10] Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., El Abbadi, A.: Vector approximation based indexing for non-uniform high dimensional data sets. In: Proceedings of the 9th ACM Int. Conf. on Information and Knowledge Management, McLean, Virginia, November 2000, pp. 202–209 (2000)
[11] Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., El Abbadi, A.: Approximate nearest neighbor searching in multimedia databases. In: Proc of 17th IEEE Int. Conf. on Data Engineering (ICDE), Heidelberg, Germany, April 2001, pp. 503–511 (2001)
[12] Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: 30th ACM Symposium on Theory of Computing, Dallas, Texas, May 1998, pp. 604–613 (1998)
[13] Ipeirotis, P.G., Gravano, L., Sahami, M.: Probe, count, and classify: Categorizing hidden web databases. In: SIGMOD Conference (2001)
[14] Jacob, K.J., Shasha, D.: Fintime – a financial time series benchmark (March 2000), `http://cs.nyu.edu/cs/faculty/shasha/fintime.html`
[15] Kahveci, T., Singh, A.K.: Efficient index structures for string databases. The VLDB Journal, 351–360 (2001)
[16] Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., Protopapas, Z.: Fast and efficient retrieval of medical tumor shapes. IEEE Transactions on Data Engineering (TKDE 1998) (1998)
[17] Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., Protopapas, Z.: Fast nearest neighbor search in medical image databases. The VLDB Journal, 215–226 (1996)
[18] Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., Protopapas, Z.: Fast nearest neighbor search in medical image databases. In: Proceedings of the Int. Conf. on Very Large Data Bases, Mumbai, India, pp. 215–226 (1996)
[19] Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., Protopapas, Z.: Fast and effective retrieval of medical tumor shapes. IEEE Trans. Knowl. Data Eng. 10(6), 889–904 (1998)
[20] Meng, W., Yu, C.T., Liu, K.-L.: Building efficient and effective metasearch engines. ACM Computing Surveys 34(1), 48–89 (2002)
[21] Roussopoulos, N., Kelly, S., Vincent, F.: Nearest neighbor queries. In: Proc. ACM SIGMOD Int. Conf. on Management of Data, San Jose, California, May 1995, pp. 71–79 (1995)
[22] Tosun, A.S., Ferhatosmanoglu, H.: Vulnerabilities in similarity search based systems. In: CIKM, pp. 110–117. ACM, New York (2002)
[23] Weber, R., Bohm, K.: Trading quality for time with nearest-neighbor search. In: Proc. Int. Conf. on Extending Database Technology, Konstanz, Germany, March 2000, pp. 21–35 (2000)
[24] Whalley, M.R.: The Durham-RAL high energy physics database - HEPDATA. Computer Physics Communications 57(1-3), 536–537 (1990)