simMachines

SIMILARITY SEARCH & PATTERN RECOGNITION

✉ info@simmachines.com

Whitepaper:

Performance of the simMachines R-01 Similarity Search Engine

▶ Arnoldo Müller-Molina, PhD
[*Founder, Principal Data Scientist*]
☎ +1 (260) NEAREST / (260) 632-7378
✉ arnoldoMuller@simmachines.com

PART 1

INTRODUCTION

The future belongs to the
companies and people that turn
data into products.
Mike Loukides, O'Reilly Radar [1]

Similarity search is a key technology for data scientists. It can be defined as the retrieval of the "closest" objects to a query in a database. An object can be a vector of d dimensions but it can also be a string of characters or even a graph. There are no restrictions on the type of data you can process. New similarity search applications are constantly being developed, ranging from language translation systems [2] to intellectual property protection [3]. Similarity search is regarded as an important tool to address the problem of information overload [4]. With this type of engine you can perform data classification. You can also accelerate existing machine learning algorithms. The information challenges of this age require fast similarity search to support cost-effective data analysis.

▶ Similarity search is essential to overcome the problem of information overload.

The simMachines R-01 engine provides outstanding performance and ease of use. Our engine is a natural choice for those who require to handle large data-sets, the goldmines of our times. In what follows, we give a brief introduction to similarity search. In part 2 we experimentally compare our engine against the popular method *LSH* (Locality Sensitive Hashing) [5] and other established methods from the SISAP Metric Spaces Library [6]. We also test our engine on a large data-set of 120 million DNA sequences.

▶ Our R-01 engine provides the highest performance and it is very easy to use!

✉ info@simmachines.com

:: 1.1 Background

▶ Skip to the benchmarks (part 2) if you already know about similarity search.

Similarity search is known also as the *nearest neighbor* problem or the *post-office* problem. A large number of similarity search algorithms [4, 7, 8, 9] exist. A key concept in most of these indexes is the idea of *metric space*. For the space, a distance function is defined to compare the similarity of two objects. The index does not "know" the structure of the objects it is handling, it only has access to a function that calculates the similarity between two objects.

:: 1.2 Metric Space

▶ Our engine also accepts functions that are not metrics.

Let $\mathcal{M} = (\mathcal{D}, d)$ be a metric space for a domain of objects $\mathcal{D}$ and $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$, *a total distance function* that satisfies the following properties:

$$\forall x, y \in \mathcal{D}, d(x, y) \geq 0$$
$$\forall x, y \in \mathcal{D}, d(x, y) = d(y, x)$$
$$\forall x, y \in \mathcal{D}, d(x, y) = 0 \iff x = y$$
$$\forall x, y, p \in \mathcal{D}, d(x, y) \leq d(x, p) + d(p, y)$$

To use the simMachines R-01 engine, the user must define a distance function d. This is the only piece of information required. Our product does not assume any details about the structure of the objects. We allow distance functions that do not comply with any of the previous properties. As long as the distance consistently measures similarity, R-01 can handle it.

:: 1.3 Similarity Queries

Given a collection $X \subseteq \mathcal{D}$, we define three type of similarity queries:

- *k-nearest neighbor query* (k-NN) returns the k closest elements to the query object q. Formally, the set $R \subseteq X$ such that $|R| = k$, for any $x \in R$ and any $y \in X - R : d(q, x) \leq d(q, y)$.
- *Range query*: Given $q \in \mathcal{D}$ and r it retrieves all the objects within distance r, that is the set $\{x \in X | d(x, q) \leq r\}$.
- *Range k-NN query*: the intersection of the previous two types of queries.

:: 1.4 The simMachines R-01 Similarity Search Engine

At simMachines, we have developed a new technique that outperforms by a large margin other similarity search indexes. The engine is the result of 6+ years of research. This section explains features present in the engine.

To use the simMachines R-01 engine, the user must provide a distance function. This distance function may follow the properties explained in the previous section, but it does not have to be a metric. Our engine provides k-NN and range k-NN queries. Despite decades of research, standard similarity search indexes degrade considerably when the space to search has a high intrinsic dimension [10]. This problem is called the *curse of dimensionality*.

▶ Approximate similarity search is often as good the exact version of the problem.

Since providing exact solutions to the similarity search problem is so difficult, some researchers [11, 12] have focused on providing approximate solutions. In this context, the similarity search index is allowed to return a result whose distance to the query is at most c times from the closest result. The simMachines R-01 engine can be configured to return exact solutions ($c = 1$) or approximate solutions ($c > 1$). Both modes of operation can be accomplished at very high speeds. A recall based configuration is also possible. R-01 can be set to return a recall value x in average. Even when exact solutions are required, R-01 provides superior performance as we shall see in part 2.

Many similarity search indexes [4] require to set complex parameters. They also force the user to know in detail the characteristics of the index. With R-01 , the user focuses on instantiating the index in one line. The rest is carried by insert, delete and search methods.

PART 2

BENCHMARKS

Data is the next Intel Inside

Tim O'Reilly

In this section, we evaluate the performance of the simMachines R-01 engine in multiple scenarios.

:: 2.1 Evaluation

We evaluate the results using a cost measure. Let $d\#$ be the number of distance computations evaluated for a query, and $|DB|$ the number of elements in the database. The cost is defined as $\frac{d\#}{|DB|}$. Let R_A be the approximate result set returned by LSH or R-01 and R the exact result set. Recall is defined as $\frac{|R_A \cap R|}{|R|}$. Let v_a be the approximate distance value returned and v the closest nearest neighbor to a given query. The approximate error c is defined as $\frac{v_a}{v}$. We output the average results of 100 queries that were randomly selected and removed from the original data-sets.

:: 2.2 Audio

We downloaded the audio data-set provided by the LSHKit project [13]. This data-set contains 54000 vectors of 192 dimensions. We employed the Multi-Probe LSH [14] implementation provided by LSHKit, and tuned it using the

▶ The audio data-set is considered to be of high dimensionality [13].

✉ info@simmachines.com

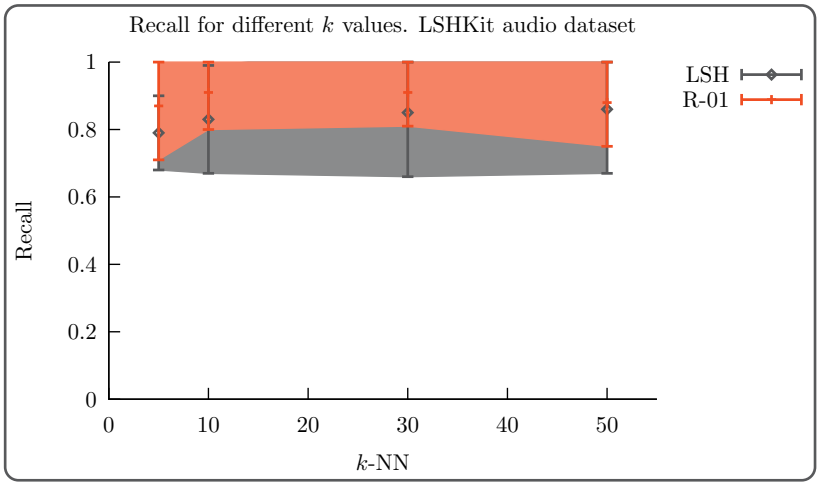

Figure 2.1: Performance of R-01 in the audio data-set. Target recall is set to 80% in both methods. Cost is the ratio of distance computations measured over the total number of elements in the data-set. k is the number of elements to be retrieved. [More recall is better]

tutorial provided by the author. LSHKit and R-01 were set to retrieve objects with a recall of 80%.

Figures 2.1 and 2.2 summarize respectively the recall obtained for both methods and the cost involved. We can observe that R-01 provides the same recall level or better than LSH while using an order of magnitude less distance computations.

It is worthwhile to mention that LSH employs about 8 hash tables with different transformations to minimize the error. On the other hand, the simMachines R-01 data structure is using only one index to achieve an order of magnitude improvement.

:: 2.3 Vector Data-sets

▶ The SISAP Metric Spaces Library [6] contains well established indexes, it is used in similarity search publications.

In this section, we test the performance of the R-01 engine when no error is allowed. We compare our engine against indexes from the metric spaces library [6]. The following indexes are used:

- Fixed Height Queries Tree (**fqh**). A variant of the Fixed-Queries Tree [15]

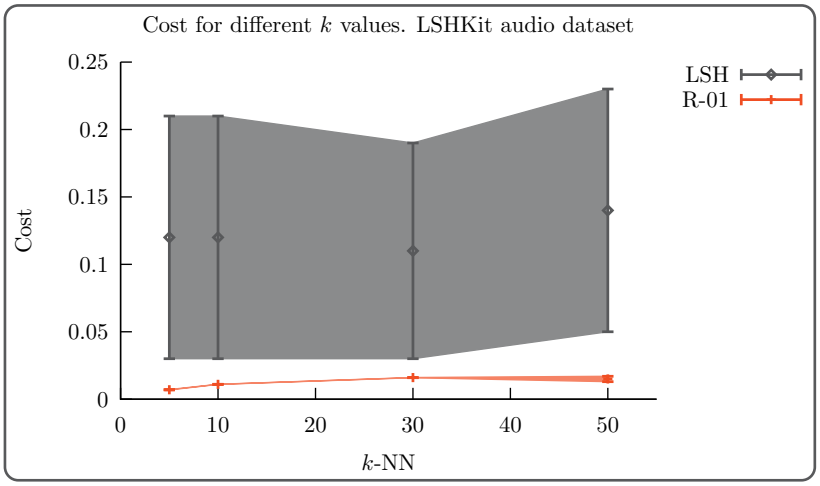

Figure 2.2: Performance of R-01 in the audio data-set. Target recall is set to 80%. k is the number of elements to be retrieved. Cost is the ratio of distance computations measured over the total number of elements in the data-set. [Less is better]

which is in turn a variant of the Burkhard-Keller. The height of the tree is restricted to h levels.

- Generalized Hyperplane Tree [16] and Bisector Tree [17] (**ght**). The implementation is a mixture of the best features of both data structures.
- List of Clusters [18] (**lcluster**). A list of balls holding b elements on each ball.
- Vantage Point Tree [19] (**mvp**)
- Spatial Approximation Tree [20] (**sat**). A self-adjusting Voronoi-like partitioning of the space.
- Linear AESA [21] (**pivots**). Employs i pivots to prune the search space. This index is used as a baseline as it requires full scans on the projection and therefore it cannot scale.

We ran our tests on the audio data-set (section 2.2), and the following data-sets from the metric spaces library:

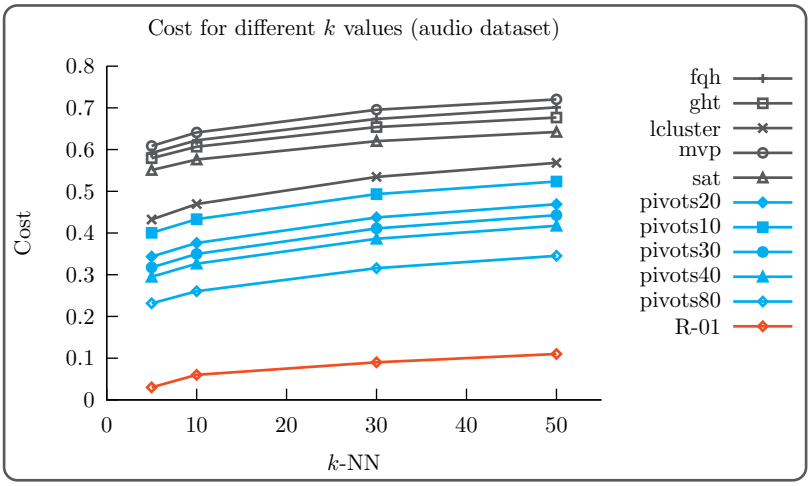

Figure 2.3: Performance of R-01 in the audio data-set. k is the number of elements to be retrieved. Cost is the ratio of distance computations measured over the total number of elements in the data-set. [Less is better]

- **nasa**: 40150 vectors of 20 dimensions generated from NASA images [22]
- **colors**: 112544 color histograms of 112 dimensions [23]

In figures 2.3, 2.4 and 2.5 we show respectively the results of the experiments for data sets audio, colors and nasa. The plots clearly show how R-01 consistently outperforms all the other data structures. R-01 is 5× to 10× faster than the other methods.

:: 2.4 Large DNA Data-set

▶ The genome is an important and complex data-set. Its large size makes it an excellent test environment.

We extracted 120 million DNA fragments of 20 base pairs each from the human genome data-set [24]. We indexed the fragments using the R-01 engine and the permutation index (Perm) of Amato *et al.* [25]. We performed $k = 5$ nearest neighbor queries setting the engine to return values within $c = 1.4$. We measured different performance metrics as the database grows from 500000 fragments to 120 million fragments. We ran our tests on an Intel Xeon processor with 24GB of RAM running Ubuntu Linux 10.04 64 bits and Java 1.6.0_24.

Figure 2.6 shows the quality result for the whole experiment. Our engine fulfills the promise of answering results in average of $c = 1.4$. Perm returns

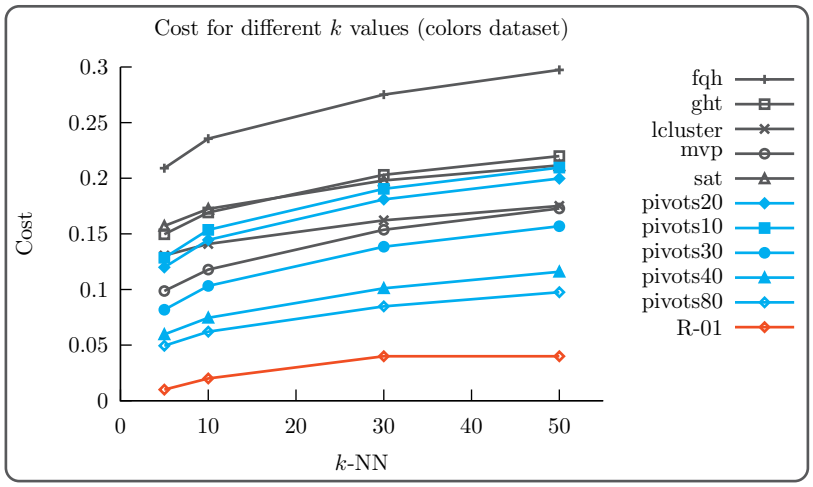

Figure 2.4: Performance of R-01 in the colors data-set. k is the number of elements to be retrieved. Cost is the ratio of distance computations measured over the total number of elements in the data-set. [Less is better]

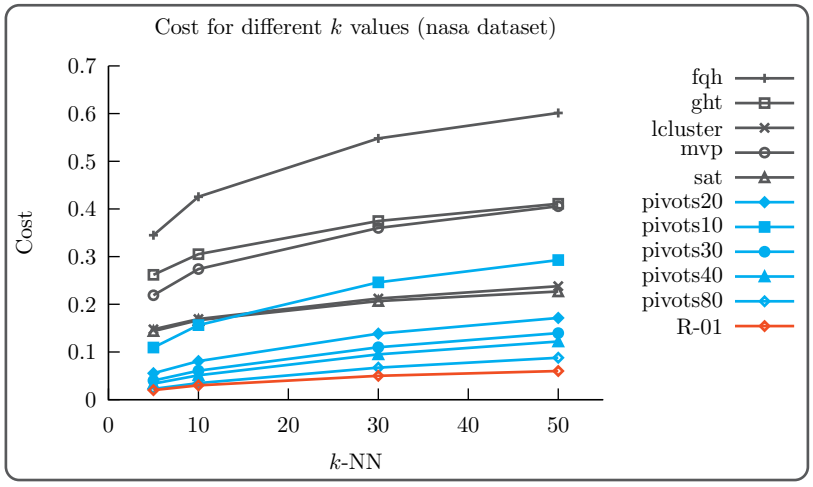

Figure 2.5: Performance of R-01 in the nasa data-set. k is the number of elements to be retrieved. The distance computation count is displayed in the y axis. [Less is better]

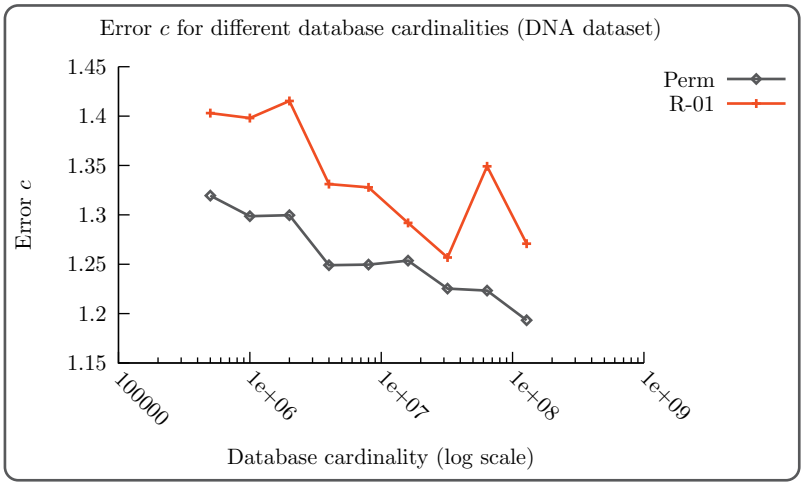

Figure 2.6: Result quality of R-01 and Perm in the DNA data-set for different database cardinalities. The indexing methods were configured to return in average $c = 1.4$. The average error c returned was recorded for each database cardinality. [Values around c=1.4 are the expected result]

a lower c because it calculates more distance computations. Figure 2.7 shows the average time in milliseconds for each query for different database cardinalities. When the database reaches 120 million objects, R-01 answers queries in 23 milliseconds. On the other hand, Perm rapidly becomes slower as the database grows.

▶ R-01 provides reliable result quality, with robust scalability on very large datasets.

It is our hope that these experiments convince you about the scalability of the R-01 engine. Only in 20 milliseconds the engine is capable of answering queries in a database of 120 million objects.

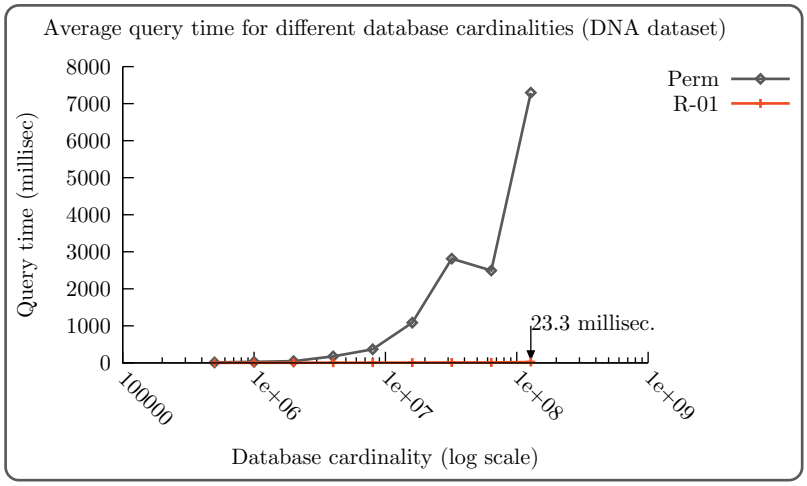

Figure 2.7: Performance of R-01 and Perm in the DNA data-set for different database cardinalities. The number of milliseconds elapsed for each query was recorded and averaged between all the queries. [Less is better]

PART 3

CONCLUSIONS

> The ability to take data to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it that's going to be a hugely important skill in the next decades
>
> *Hal Varian, The McKinsey Quarterly [26]*

In this white-paper we have explained general concepts related to similarity search. We introduced the simMachines R-01 engine and we compared it against popular similarity search schemes.

▶ R-01 provides the highest performance and ease of use.

Our tests demonstrated that our engine is 10× faster and uses 10× less space than LSH. Additionally, the engine achieves 5× - 10× improvements over well established methods. We could verify that our engine answers queries in 20 milliseconds even in complex data-sets of 120 million objects.

Data analysis tasks required in a wide array of industry fields can now be served by the simMachines R-01 high performance similarity search engine. R-01 provides very high quality results on very complex search spaces at very high speeds.

✉ info@simmachines.com

:: About the Author

Dr. Arnoldo Müller-Molina has successfully applied similarity search techniques to solve complex biological questions at the Max Planck Institute for Molecular Biomedicine in Germany. Additionally, he has published peer-reviewed articles related to similarity search [3, 27, 28, 29, 30]. Dr. Muller-Molina wrote a popular open source similarity search engine during Google Summer of Code 2007 [31] and other open source projects [32, 33]. At Intel, he developed high volume database applications employed in the analysis of factory production data and corporate financial information. Arnoldo holds a PhD in computer science from Kyushu Institute of Technology (Japan) where he specialized in software similarity analysis for intellectual property protection and similarity search data structures where he held a Monbusho scholarship from the Japanese Ministry of Education, Culture, Sports, Science & Technology. He obtained a computer science degree from Instituto Tecnológico de Costa Rica in 2001.

BIBLIOGRAPHY

[1] "http://radar.oreilly.com/2010/06/what-is-data-science.html."

[2] M. M. Gonzalez and T. Endo, "Tense and mood decision with similarity search in japanese to spanish machine translation," in *IMECS*, pp. 86–91, 2009.

[3] A. J. Müller-Molina and T. Shinohara, "Fast approximate matching of programs for protecting libre/open source software by using spatial indexes," in *SCAM '07*, (Washington, DC, USA), pp. 111–122, IEEE Computer Society, 2007.

[4] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Similarity Search: The Metric Space Approach*. Secaucus, NJ, USA: Springer-Verlag, 2005.

[5] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 518–529, Morgan Kaufmann Publishers Inc., 1999.

[6] http://www.sisap.org/Metric_Space_Library.html.

[7] H. Samet, *Foundations of Multidimensional and Metric Data Structures*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[8] G. R. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces (survey article)," *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 517–580, 2003.

[9] E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin, "Searching in metric spaces," *ACM Comput. Surv.*, vol. 33, no. 3, pp. 273–321, 2001.

[10] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 194–205, Morgan Kaufmann Publishers Inc., 1998.

[11] J. M. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions," in *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, (New York, NY, USA), pp. 599–608, ACM, 1997.

[12] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *J. ACM*, vol. 45, no. 6, pp. 891–923, 1998.

[13] http://lshkit.sourceforge.net/.

[14] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: efficient indexing for high-dimensional similarity search," in *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pp. 950–961, VLDB Endowment, 2007.

[15] R. A. Baeza-Yates, W. Cunto, U. Manber, and S. Wu, "Proximity matching using fixed-queries trees," in *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, CPM '94, (London, UK), pp. 198–212, Springer-Verlag, 1994.

[16] J. K. Uhlmann, "Satisfying general proximity / similarity queries with metric trees," *Information Processing Letters*, vol. 40, no. 4, pp. 175 – 179, 1991.

[17] I. Kalantari and G. McDonald, "A data structure and an algorithm for the nearest point problem," *IEEE Trans. Softw. Eng.*, vol. 9, pp. 631–634, September 1983.

[18] E. Chávez and G. Navarro, "A compact space decomposition for effective metric indexing," *Pattern Recogn. Lett.*, vol. 26, pp. 1363–1376, July 2005.

[19] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, (Philadelphia, PA, USA), pp. 311–321, Society for Industrial and Applied Mathematics, 1993.

[20] G. Navarro, "Searching in metric spaces by spatial approximation," *The VLDB Journal*, vol. 11, pp. 28–46, 2002. 10.1007/s007780200060.

[21] L. Micó, J. Oncina, and E. Vidal, "An algorithm for finding nearest neighbours in constant averagetime with a linear space complexity," *Recognition Methodology and Systems*, pp. 557–560, 1992.

[22] http://www.dimacs.rutgers.edu/Challenges/Sixth/software.html.

[23] http://www.dbs.informatik.uni-muenchen.de/~seidl/DATA/histo112.112682.gz.

[24] http://www.ncbi.nlm.nih.gov/.

[25] G. Amato and P. Savino, "Approximate similarity search in metric spaces using inverted files," in *Proceedings of the 3rd international conference on Scalable information systems*, InfoScale '08, (ICST, Brussels, Belgium, Belgium), pp. 28:1–28:10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[26] "http://www.mckinseyquarterly.com/Hal_Varian_on_how_the_Web_challenges_managers_2286."

[27] A. J. Müller-Molina, K. Hirata, and T. Shinohara, "A tree distance function based on multi-sets," in *ALSIP'08, PAKDD Workshops*, pp. 90–100, 2008.

[28] A. Müller and T. Shinohara, "Efficient similarity search by reducing i/o with compressed sketches," in *SISAP*, IEEE/ACM, 2009.

[29] A. Müller-Molina and T. Shinohara, "On the configuration of the similarity search data structure d-index for high dimensional objects," in *Computational Science and Its Applications - ICCSA 2010*, vol. 6018 of *Lecture Notes in Computer Science*, pp. 443–457, Springer Berlin / Heidelberg, 2010.

[30] A. Müller, "Obsearch: A high performance similarity search engine for java," in *SISAP*, IEEE/ACM, 2009.

[31] "http://code.google.com/p/obsearch/."

[32] "http://furia-chan.org."

[33] "http://treelib.berlios.de."