An Empirical Evaluation of Intrinsic Dimension Estimators

Cristian Bustos[1], Gonzalo Navarro[2], Nora Reyes[1], and Rodrigo Paredes[3]

[1] Dpto. de Informática, Universidad Nacional de San Luis
[2] Computer Science Department, University of Chile
[3] Departamento de Ciencias de la Computación, Universidad de Talca, Chile
{cjbustos,nreyes}@unsl.edu.ar, gnavarro@dcc.uchile.cl, raparede@utalca.cl

Abstract. In this work, we study the behavior of different algorithms that attempt to estimate the intrinsic dimension (ID) in metric spaces. Some of these algorithms were developed specifically for evaluating the complexity of the search on metric spaces, based on different theories related to the distribution of distances between objects on such spaces. Others were designed originally only for vector spaces and they have been adapted so that they can be applied to metric spaces.
To determine the goodness of the ID estimation obtained with each algorithm —or at least determine which one fits the best to the actual difficulty of the search process on the tested metric spaces— we make comparisons using two indices, one based on pivots and the other on compact partitions. This allows us to verify if the considered ID estimators reflect the actual hardness of searching over the considered spaces.

1 Introduction

Nowadays, similarity search in metric spaces has received much attention in the algorithmic community due to the numerous applications it has. It is an important problem in many fields, ranging from multimedia information retrieval to machine learning, classification, and searching the Web. Therefore, it has been proposed a wealth of practical algorithms to solve it. These algorithms are effective on certain metric spaces, but their performance is poor in others.

The similarity between a set of objects $\mathbb{U}$ is modeled using a *distance function* (or *metric*) $d : \mathbb{U} \times \mathbb{U} \mapsto \mathbb{R}^+ \cup \{0\}$ that satisfies the properties: triangle inequality, strict positivity, reflexivity, and symmetry. In this case, the pair $(\mathbb{U}, d)$ is called a *metric space* [7, 23, 21].

In some applications, the metric spaces are of a particular kind called "vector spaces", where the elements consist of D coordinates of real numbers. There are many works that benefit of geometric properties of vector spaces, but normally they cannot be extended to general metric spaces, where the only available information is distance between objects. In the general case, the distance is very expensive to compute, then the main objective is to reduce the number of distance evaluations. In contrast, vector space operations tend to be simpler and thus, the primary objective is reduce the number of I/O operations carried out.

Similarity queries are usually of two types. For a given database $S \subseteq \mathbb{U}$ with size $|S| = n$, $q \in \mathbb{U}$ and $r \in \mathbb{R}^+$, $(q,r)_d = \{x \in S \mid d(q,x) \leq r\}$ denotes a *range query*. The other type of query is the k *nearest neighbor* one, denoted by $kNN_d(q)$, which retrieves the k closest elements to q in S. Formally, it retrieves a set $R \subseteq S$ such that $|R| = k$ and $\forall\ u \in R, v \in S \setminus R$, $d(q,u) \leq d(q,v)$.

A naïve way to answer both query types is by performing an exhaustive search in the database. That is, all the database elements are compared with the query q, and then those elements that are "close enough" to it are returned (*brute force algorithm*). Obviously, this approach is too expensive for real applications. Thus, researchers have achieved different ways to efficiently search in general metric spaces. There has been some significant progress in this area, mostly around the idea of building an index; i.e. a data structure that allows to reduce the number of evaluations of the distance function during the resolution of a query.

In vector spaces, the *curse of dimensionality* describes a phenomenon where performance of all existing algorithms decays exponentially as dimension grows. In general, the complexity of solutions in metric space are measured as the number of distance evaluations, because the absence of coordinates prevents a complexity analysis in terms of the representation size.

In this work, we study the behavior of some algorithms trying to estimate the *intrinsic dimension* (ID) in metric spaces. Some of them were developed specifically for evaluating the complexity of the search on metric spaces, based on different theories related to the distribution of distances between objects on such spaces. Others were designed originally for vector spaces and they have been adapted so that they can be applied to metric spaces.

To determine the goodness of the estimation of ID obtained with each algorithm —or at least determine which one fits the best to the actual difficulty of the search process on the tested metric spaces— we have made comparisons using two indices, one based on pivots and the other on compact partitions. This allows us to verify whether the considered algorithms to estimate the ID reflect the actual hardness of searching over the considered metric spaces, and also, which one offers the best estimation.

2 Intrinsic Dimension Estimators for Vectorial Spaces

There are several interesting applications where the data is represented as D-dimensional vectors in $\mathbb{R}^D$. For instance, in pattern recognition applications, objects are usually represented as vectors [15]. So, data is embedded in $\mathbb{R}^D$, even though this does not imply that its *real* dimension is D.

There are many definitions of ID. For instance, the ID of a given dataset is the minimum number of free variables needed to represent the data without lost of information [2]. In general terms, a dataset $\mathbb{X} \subseteq \mathbb{R}^D$ has ID $M \leq D$, if its elements fall completely within a M-dimensional subspace of $\mathbb{R}^D$ [13]. Another intuitive notion is the lesser effort in the searching, the lower space ID.

Even in vector spaces, there are many reasons to estimate the ID of a dataset. Using more dimensions (more coordinates in the vectors) than necessary can

bring several problems. Firstly, the space needed to store the data is an issue. For instance, if the ID of a dataset $\mathbb{X} \subseteq \mathbb{R}^D$ is $M \leq D$ and $|\mathbb{X}| = n$, so $n \times D$ real coordinates are needed to store the original data. Instead, if we know that the ID is M, we will just store $n \times M$ real coordinates ($n \times M \leq n \times D$), and the space savings is greater depending on how lower is M with respect to D. Also, as the amount of available information increases, compressing the data storage becomes even more important. Secondly, as the asymptotic complexity of the algorithms is monotonically increasing with respect to the dataset dimensionality, a dimensionality reduction can produce and important CPU time reduction.

There are two approximations to estimate the ID of a vector space [15, 2], namely, the *local* and *global* methods. The local ones make the estimation by using the information contained in sample neighborhoods, avoiding the data projection over spaces of lower dimensionality. The global ones deploy the dataset over a M-dimensional space using all the dataset information.

In particular, this work is focused on global ID estimators. That is, in every case we consider all the dataset information in order to obtain a estimation as accurate as possible. Global methods can be split in three families, namely, projection techniques, multidimensional scaling methods and fractal based methods. Among them, the last two are more suitable to metric spaces, so we have selected and adapted some representatives of these groups.

3 Intrinsic Dimension Estimators for Metric Spaces

In general metric spaces, due to the curse of dimensionality severely affect the performance of the search process, knowing the intrinsic dimension can help to chose a metric index appropriate to the space dimension. For instance, in low ID spaces, where the searching is easier, pivot based indices usually are more suitable. However, they can perform very bad in high ID spaces, or hard spaces, where compact partition based indices, such as the LC are more efficient.

Hence, a proper estimation of the operating dataset ID is very important, as can help to improve the time and memory costs of the selected solution.

Nevertheless, there are very few dimension estimators we can apply directly in general metric spaces; and the ID estimators that are proper to metric spaces can only consider the dataset objects and its distance. Thus, in the following we analyze some methods to estimate the ID of vector spaces and others to general metric spaces, and later we show how all of them have been adapted in this work. Furthermore, as multidimensional spaces are a particular case of metric spaces, these estimators can also be applied to obtain the ID of D-dimensional spaces.

3.1 Fractal Based Methods

Unlike other families, fractal based methods can estimate non-integer ID values. The most popular techniques of this family are *Box Counting* [17], which is a simplified version of the *Haussdorff dimension* [10, 18], and *Correlation* [3].

The dimension estimation by Box Counting D_B of a set $\Omega \subseteq \mathbb{R}^D$ is defined as follow: if $v(r)$ is the number of boxes of size r needed to cover Ω, then D_B is

$$D_B = \lim_{r \to 0} \frac{\ln(v(r))}{\ln\left(\frac{1}{r}\right)} \tag{1}$$

In this method, the boxes are multidimensional regions of side r on each dimension (that is, they are hipercubes of side r). Regrettably, even though efficient algorithms have been proposed, the Box Counting dimension can be computed only for low dimension datasets. This is because its algorithmic complexity grows exponentially with the dimension.

Estimating the dimension by Correlation is an alternative to Box Counting. It is defined as follow. Let $\Omega = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^D$ and the correlation integral $C_m(r) = \lim_{n \to \infty} \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} I(||x_j - x_i|| < r)$, where $I(\cdot)$ is an indicator function. Intuitively, $C_m(r)$ is the fraction of object pairs whose distance is lower than r. So, the dimension estimation by Correlation D_C is

$$D_C = \lim_{r \to 0} \frac{\ln(C_m(r))}{\ln r} \tag{2}$$

The most popular method to estimate the dimension by Correlation and Box Counting is the log-log plots. It consists in plotting $\ln(C_m(r))$ versus $\ln(r)$. The dimension by Correlation is the slope of the lineal section of the curve. To estimate the dimension by Box Counting we replace $\ln(C_m(r))$ by $\ln(v(r))$.

However, as we are interested in the general case of metric spaces, and here we not always have coordinates, we consider *balls* of radius r. That is, the set of objects within a distance r from a reference object o. In this case, we randomly pick the reference objects from the database elements, and we count the number $B(r)$ of balls of radius r we need to cover the dataset. To do so, we use the *List of Clusters* (LC) [6], whose code is available from SISAP [12], with the variant of fixed radius and centers chosen at random, and obtain as result the LC length.

Thus, to estimate the dimension by Box Counting, which in this case is Ball Counting, we replace $\ln(v(r))$ by $\ln(B(r))$, plot $\ln(B(r))$ versus $\ln\left(\frac{1}{r}\right)$ in log-log and obtain the slope of the linear section of the curve by using linear regression with least squares over the experimental data $\left(\ln(B(r)), \ln\left(\frac{1}{r}\right)\right)$.

3.2 Distance Exponent

In [22], the authors discuss the problem of the selectivity estimation for range queries in real world metric spaces, including spatial or multidimensional datasets as special cases. The main contribution is, surprisingly, several datasets follow the so-called *Power Law*. They call *Distance Exponent* the exponent of the power law. It plays an important role when analyzing real metric spaces. In [22], they show how to use it to derivate formulae for estimating the selectivity of range query; for instance, the number of objects relevant to the query, the number of I/O access to answer the query when the data is stored in secondary memory, the amount of time needed to answer the query, and so on.

For the sake of finding a formula that estimates the number of neighbors of objects within a distance r in a n-objects dataset, they introduce the following notions: (i) the *Distance Graphic* of a metric set is the number of object pairs at distance at most r versus the distance r, and both axis are drawn in logarithmic scale; and (ii) the *Distance Exponent* is the slope of the line that better fits the distance graphic in case it is linear for a range of scales. Using these two notions, they define the *Distance Law*.

Distance Law - *Given a dataset of n objects from a metric space with distance function $d(x, y)$, the average number of distances lower than a radius r follows a power law; that is, the average number of neighbors $\overline{nb}(r)$ within a distance r is proportional to $r^{\mathcal{D}}$:*

$$N \cdot \Phi(r) = \overline{nb}(r) \propto r^{\mathcal{D}} \tag{3}$$

If a dataset has a metric to evaluate the distance between every object pair, then this graphic can always be drawn, even if it has not an spatial property. Moreover, they show that the distance graphic shows an almost linear behavior for many databases, both real and synthetic. Clearly, the distance graphic requires $O(n^{2)}$ distance computations. To avoid this cost, $\overline{nb}(r)$ is estimated using an index [22]. That is, a way to estimate the distance exponent $\mathcal{D}$ of a dataset stored in a metric index is by means of the very same index.

The index used in [22] to estimate $\overline{nb}(r)$ is the *M-tree* [8]. But, as in this work we are only interested in comparing the different ID measures and indexing the space is not strictly necessary (as we do not pretend to solve queries) we compute directly $\overline{nb}(r)$, considering a reference object chosen at random from the dataset. Next, we only determine the number of elements at distance r from that object.

3.3 Fastmap

This method arises from the proposal in [14] of a fast algorithm to map objects as points in a k-dimensional space (k being defined by the user), so that the dissimilarities are preserved. So, this algorithm could allow to map any metric (or vector) space into a k dimensional vector space. The objective is speeding up the searching process in traditional or multimedia databases.

To do so, objects are mapped into a k-dimensional space using k feature extraction functions, provided by domain experts [14]. The main drawback is defining such feature extraction functions. For example, in the metric case of strings with the edit distance [16], it is not clear which features can be considered.

As it is relatively easier for the domain expert to provide a distance function to compare objects (than provide several feature extraction functions), Faloutsos and Lin proposed [11] a generalization of the method of [14], trying to map the objects in k dimensional points using only the domain experts' distance function.

Therefore, Fastmap considers the problem of, given a n object dataset from a metric space $(\mathbb{U}, d)$, find n image points in a k-dimensional target space, such that the distances between the objects in the original space are preserved as much as possible, between the image points in the target k-dimensional space.

For the sake of evaluating the dissimilarity preservation in the target space, an *stress* function is defined as follows:

$$stress^2 = \frac{\left(\sum_{i,j}(\hat{d}_{ij} - d_{ij})^2\right)}{\left(\sum_{i,j} d_{ij}^2\right)} \tag{4}$$

where, d_{ij} is the dissimilarity measure (the distance of the original space) between object objects o_i and o_j, and $\hat{d}_{ij}$ is the Euclidean distance between their respective images p_i and p_j. The stress function gives the relative error that the distances in the target space suffer in average after the transformation. To reach the objective, Fastmap begins with an estimation which is iteratively improved, until no additional improvement is possible.

In the metric case, we can consider that we have a distance function between the objects or a matrix of $n \times n$ distances between all the object pairs. So, the objective of *Fastmap* [11] is to find n points in the k-dimensional space, whose Euclidean distances correspond to the given matrix of $n \times n$ distances in the original space. The crux is assuming that objects are points in an unknown m-dimensional space (with unknown m), and trying to project these points over k mutually orthogonal directions. The challenge is computing all these projections using only the distance matrix. Fastmap projects the objects over lines carefully selected. To do so, two objects o_a and o_b are chosen, and it considers the "line" passing through them in the original space. The projections of the objects over this line are obtained using the *cosine law*:

Theorem 1 (*Cosine Law*) Any triangle $o_a \overset{\triangle}{o_i} o_b$ satisfies that:

$$d(o_b, o_i)^2 = d(o_a, o_i)^2 + d(o_a, o_b)^2 - 2x_i' d(o_a, o_b) \tag{5}$$

Eq. 5 can be solved for x_i' to compute the projection of object o_i:

$$x_i' = \frac{d(o_a, o_i)^2 + d(o_a, o_b)^2 - d(o_b, o_i)^2}{2d(o_a, o_b)} \tag{6}$$

Thus, the input of *Fastmap* is a set S of size n and and, in each iteration, computes the coordinates of all the n objects over the new axis. So, after k iterations, it produces a k-dimensional target space S' where each object $o_i \in S$ is mapped to a k-coordinates vector $p_i = (x'_{i,1}, x'_{i,2}, \ldots, x'_{i,k}) \in S'$, where $x'_{i,j}$ is the j-th projection of the image p_i of the object o_i.

In our case, we want to estimate the number of projections needed so that the target space reaches a mapping with a small enough *stress*. This is, to determine the minimum number of projected dimensions so that the metric space objects mapped onto the vector space preserve their distances. Thus, we modify the *Fastmap* algorithm. Instead of considering a prefixed value k of projected dimensions, each time it projects a new dimension, we compute the *stress* of the candidate target space. So, if the difference between the current *stress* and the previous one is significative, we compute another projection. This increases the number of coordinates of the target space. Otherwise, the target space current dimension is reported as the estimation of the ID of the original metric space.

3.4 Intrinsic Search Difficulty

In [5], a measure of the intrinsic complexity of searching a general metric space is introduced. It reflects the expected behavior of the searching algorithms on the metric space, is easy to estimate and is independent of the searching algorithm.

Several authors [1, 4, 9] have proposed to use the *distance histogram* to characterize the hardness of searching an arbitrary metric space. In this trend, the main objective of [5] was to define a quantitative measure of the intrinsic search hardness based on the histogram, which is not necessary related to the concept of dimension. But, it allows to lower bound it.

The intuition of [5] is that as the metric space is harder to search, the mean μ of its distance histogram grows and/or its variance σ^2 reduces. Obviously, we do not have the histogram of the whole space, but we can approximate it using the histogram of the dataset $S \subset \mathbb{U}$.

Definition: Let μ be the mean and σ^2 be the variance of the histogram of distances of a metric space. Then, its *intrinsic search difficulty* is:

$$\rho = \frac{\mu^2}{2\sigma^2} \tag{7}$$

As it can be seen, the intrinsic search difficulty grows with the histogram mean and with the inverse of its variance. Also, given an arbitrary and unknown metric space, the intrinsic difficulty can be measured by means of simple statistics, with a reasonable number of distance evaluations between objects obtained at random from the dataset. It is easier and cheaper than other techniques, in particular with respect to those based over a definition of dimension.

In [19], it is presented an axiomatic system, consisting of three axioms, that an intrinsic dimension function must satisfy. The author demonstrates by a theorem that the intrinsic dimension measure ρ satisfies these axioms, although two of them in a weaker version. Besides, in [20], some goals that should have an intrinsic dimension function are posed and ρ achieves nearly four of them.

As the measure ρ has been specifically designed for metric spaces, we do not adapt it. In this case, we consider the dataset S and we compute all the distances $d(x, y), \forall\, x, y \in S$ and then we compute the average $\mu = \frac{1}{n^2} \sum_{x,y \in S} d(x, y)$ and the variance $\sigma^2 = \frac{1}{n^2} \sum_{x,y \in S} (d(x, y) - \mu)^2$. Finally, we obtain the value of $\rho = \frac{\mu^2}{2\sigma^2}$ and report it as the value that estimates the ID of the metric space.

4 Experimental Results

In the following, we show the experimental evaluation of the four different ID estimators for general metric spaces. We want to explore their behavior and determine whether any of them represents appropriately the relation between the dimension and the hardness of the similarity searching in the particular metric space, also known as *the curse of dimensionality.*

We consider two kinds of metric spaces, depending on the data source:

Real world: these are metric spaces obtained from real world applications. For instance, a feature vectors space of images obtained from a NASA image set.

Synthetic: these are spaces generated artificially so that they present some interesting characteristic to be evaluated. For instance, uniformly distributed vectors in $\mathbb{R}^D$ with known dimension.

4.1 Synthetic Metric Spaces

This class of metric space contains the vector spaces. They are treated as metric spaces as we do not consider the coordinates information. We want to test two conditions. In the first, we generate the vectors with uniform distribution, so that the representational dimension matches the ID. In the second, we generate vectors with Gaussian distribution, so that the representational dimension is greater than the ID.

Uniformly Distributed Vectors with Euclidean Distance We generate four datasets of 100,000 vectors uniformly distributed in the unitary cube $[0, 1)^D$, with D = 5, 10, 15 and 20. As the coordinates of the vectors have uniform distribution in $[0, 1)$, we can control exactly the ID of the each vector space. For shortness, the four spaces are called C5, C10, C15 y C20, that stand for vectors of 5, 10, 15 o 20 coordinates.

Fig. 1(a) depicts the estimations for these four metric spaces. As it can be seen, Fractal estimator is not be able to reflect correctly the dimension increasing. On the other hand, the other three estimators increase the values of their measures as the dimension grows.

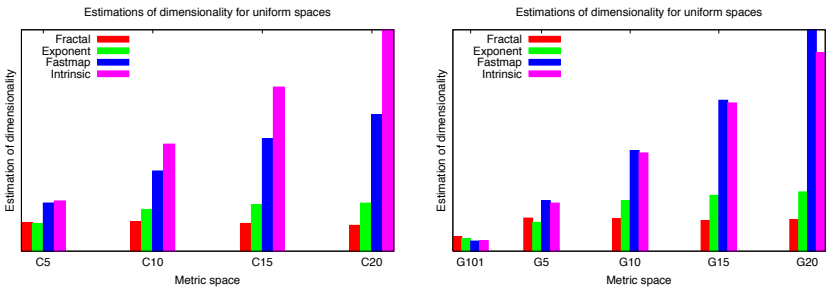
Gaussian Distributed Vectors with Euclidean Distance We generate 100,000 vectors in $\mathbb{R}^D$ with mean $\mu = 1$ and variance $\sigma^2 = 0.1$, with D = 5, 10, 15 y 20. In these spaces, there is no, a priori, clusters of elements. For shortness, these spaces are called G5, G10, G15 y G20.

We also generate 100,000 vectors in $\mathbb{R}^{101}$ with mean $\mu = 1$ and variance $\sigma^2 = 0.1$ with 200 clusters (the cluster centers are uniformly distributed in the space). For shortness, this space is called G101.

Figure 1(b) shows the estimations obtained with Fractal, Distance Exponent, Fastmap, and Intrinsic Search Difficulty, for these metric spaces. As it can be noticed, again Fractal estimation fails in these spaces because it is not capable of showing the increase of the dimension. As occurred before, the other three estimators reflect seamlessly the grow of the dimension, although Fastmap and Intrinsic Search Difficulty seem to highlight them more.

4.2 Real Metric Spaces

We pick four spaces from [12] (available at `http://www.sisap.org/library/dbs/`) in order to estimate their IDs with the four dimension estimators. The selected spaces are a representative sample for this kind of spaces. They are the following:



(a) Uniform spaces. (b) Gaussian spaces.

Fig. 1. Comparison of dimensionality estimations for synthetic metric spaces.

Dictionary: it is a dictionary of 69,069 English words. In this space, we use a discrete function, the *Edit Distance* or *Levenshtein Distance* [16]. Fig. 4(a) (in the Appendix A) shows the distance histogram of Dictionary, which is very concentrated.

NASA: this is a set of 40,700 images from NASA, represented as feature vectors of 20 real coordinates per vector, under the Euclidian distance. They were generated from images downloaded from the NASA site [4]. Fig. 4(b) (in the Appendix A) shows the distance histogram of NASA.

Histograms: this is a dataset of 112,682 histograms of medical images, each one composed by 8-D color histograms of 112 real components [5]. As any quadratic form function can be used as distance in this case, we also have chosen the Euclidean distance, due to it is the simplest alternative. Fig. 4(c) (in the Appendix A) illustrates the distance histogram of this space.

Documents: this space has 1,265 documents, represented as vectors according to the vectorial model of documents used in the Information Retrieval field. To compare documents we use the *cosine distance*. Each vector has a coordinate for each vocabulary term in the colection. So, documents can be seen as points in a vector space of high representational dimension. The documents are files obtaines form the TREC-3 colection [6]. Fig. 4(d) (in the Appendix A) shows that the distance histogram of this space is very irregular.

To measure the intrinsic hardness of the searching, we consider two search algorithms, one belonging to compact partition based indices and the other to pivot based indices.

For these experiments we perform 10 executions of the algorithms, building the index with the 90% of the database elements, and reserving the resting 10% (chosen at random) to make the queries. So, objects considered as queries do not belong to the index. Thus, we show results averaged over the 10 executions. In each execution, the objects in the metric space are permutated. Therefore,

[4] Available at `http://www.dimacs.rutgers.edu/Challenges/Sixth/software.html`.
[5] Available at `http://www.dbs.informatik.uni-muenchen.de/~seidl/DATA/histo112.112682.gz`.
[6] Available at `http://trec.nist.gov`.

each of the 10 indices considers a different dataset S, and, of course, the query objects are also differents. Finally, we only analyze the cost of range queries.

For these metric spaces, the query radii used in the experiments are:

Dictionary: As the metric is discrete, we use radii 1, 2, 3, y 4, retrieving in average approximately the 0.003%, 0.037%, 0.326% and 1.757% of the database.

NASA: This is a continuous metric, so we use radii 0.012, 0.285 and 0.53, retrieving in average approximately the 0.01%, 0.1% and 1% of the dataset.

Histograms: This metric is also continuous. Then, to retrieve in average approximately the 0.01%, 0.1% and 1% of the dataset, we use query radii 0.051768, 0.082514 and 0.131163.

Documents: Again, as the distance is continuous, we use query radii 0.14, 0.15 y 0.195, that retrieve in average the 0.01%, 0.1% and 1% of the database.

From the pivot based algorithm family, we use the generic pivot algorithm. To do so, we chose at random a set of pivots $\mathcal{P} = \{P_1, P_2, \ldots, P_k\} \subset S$ of size $|\mathcal{P}| = k$. So, in this case, we store kn distances to the pivots. For each space, we experimentally determine the number of pivots that obtains the best searching performance. Therefore, the results shown for each case correspond to the best possible ones for this method, in the corresponding metric space.

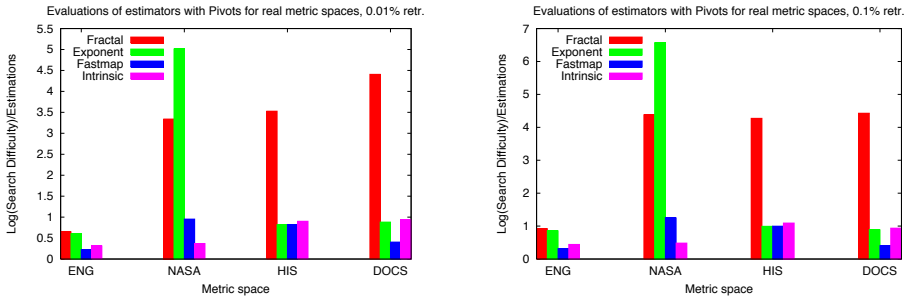
For the case of compact partition based algorithms, we use the *List of Clusters* (LC) [6], which is very representative of this family of algorithms. We use the LC variant that has a maximum size for each cluster. For each real-world metric space considered, we experimentally determine the cluster size whose perfomance is the best, and this is the result shown in the plots.

Figures 2 and 3 illustrate the correlation between the intrinsic difficulty of searching with the Pivots index and the List of Clusters, respectively; and the estimation reported for each considered ID estimator. For lack of space, we only show the results of the search that retrieve the 0.01% and 0.1% of the database approximately. As it is aforementioned, Fastmap and the Intrinsic Search Difficulty appear as the better ones because the ratio between the logarithm of search costs and their estimations are low. That is, the estimations obtained with these estimators are proportional to search costs for all real metric spaces considered and also for both types of indices.

5 Conclusions

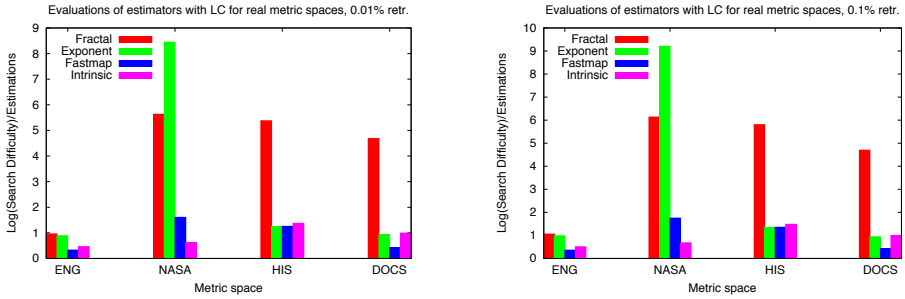
One of the main obstacles to the design of efficient search techniques in metric spaces is the existence of high-dimensional spaces in real applications. Therefore, it is necessary to have estimations of the intrinsic dimension of a metric space to determine the most appropriate and efficient method according to the particular dimension of the given metric space.

Therefore the aim of this study is to recognize the state of the art in this area and compare experimentally some of the important measures found, in order to provide empirical evidence to decide which measure better reflects the real difficulty of the similarity searches in metric spaces.



(a) Retrieving 0.01%. (b) Retrieving 0.1%.

Fig. 2. Comparison of estimators for each real metric space considered, using Pivots.



(a) Retrieving 0.01%. (b) Retrieving 0.1%.

Fig. 3. Comparison of estimators for each real metric space considered, using List of Clusters.

Although our results are inconclusive, we have shown, for example, that the Fractal dimension estimator has not been useful, while the other three measures appear to be suitable. Hence, this work appears as a contribution to the development and understanding of the search problem and its intrinsic difficulty of metric spaces, and to validate proper intrinsic dimension estimators of metric spaces with other adapted from vector spaces, allowing to achieve knowledge still unevaluated in this area.

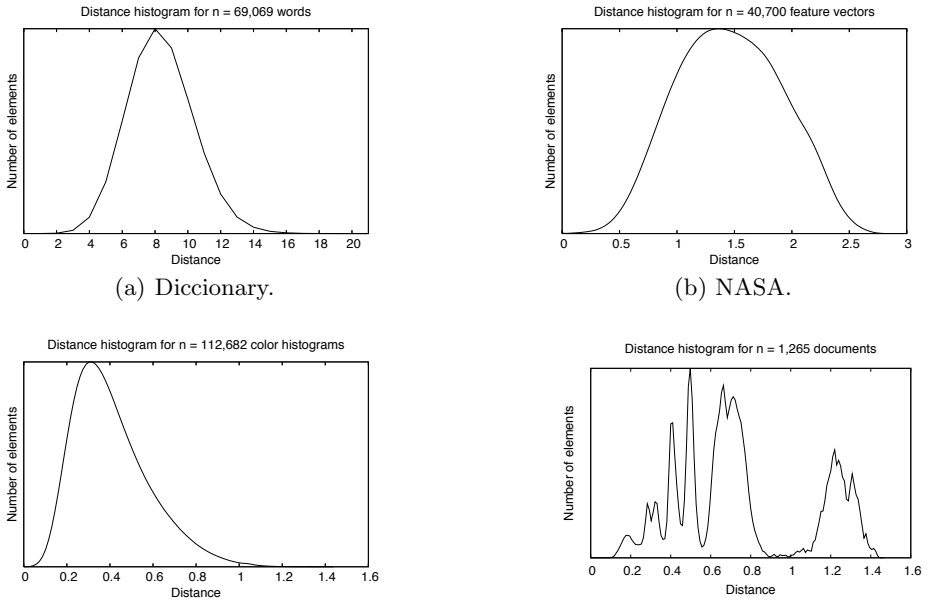
As future work we plan to analyze other estimators, as the concentration dimension [19], and evaluate them exhaustively on even more metric spaces.

References

1. S. Brin. Near neighbor search in large metric spaces. In *Proc. 21st Conf. on Very Large Databases (VLDB'95)*, pages 574–584, 1995.

2. F. Camastra. Data dimensionality estimation methods: a survey. *Pattern Recognition*, 36(12):2945–2954, 2003.
3. F. Camastra and A. Vinciarelli. Estimating the intrinsic dimension of data with a fractal-based method. *IEEE TPAMI*, 24(10):1404–1407, 2002.
4. E. Chávez and J. Marroquín. Proximity queries in metric spaces. In *Proc. 4th South American Workshop on String Processing (WSP'97)*, pages 21–36. Carleton University Press, 1997.
5. E. Chávez and G. Navarro. Towards measuring the searching complexity of metric spaces. In *Proc. Intl. Mexican Conf. in Computer Science (ENC'01)*, volume II, pages 969–978. Sociedad Mexicana de Ciencias de la Computación, 2001.
6. E. Chávez and G. Navarro. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, 26(9):1363–1376, 2005.
7. E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001.
8. P. Ciaccia, M. Patella, and P. Zezula. M-tree: an efficient access method for similarity search in metric spaces. In *Proc. 23rd VLDB*, pages 426–435, 1997.
9. Paolo Ciaccia, Marco Patella, and Pavel Zezula. A cost model for similarity queries in metric spaces. In *PODS*, pages 59–68, 1998.
10. J. P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Rev. Mod. Phys.*, 57:617, 1985.
11. C. Faloutsos and K.-I. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. 1995 ACM SIGMOD Intl. Conf. on Management of Data*, pages 163–174. ACM Press, 1995.
12. K. Figueroa, G. Navarro, and E. Chávez. Metric spaces library, 2007. Available at `http://www.sisap.org/Metric_Space_Library.html`.
13. K. Fukunaga. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.
14. H. V. Jagadish. A retrieval technique for similar shapes. In *SIGMOD Conference*, pages 208–217. ACM Press, 1991.
15. A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
16. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
17. B. Mandelbrot. *Fractals: Form, Chance and Dimension*. W. H. Freeman, San Francisco, 1977.
18. E. Ott. *Chaos in dynamical systems*. Cambridge University Press, Cambridge, New York, 1993.
19. V. Pestov. Intrinsic dimension of a dataset: what properties does one expect? In *2007 Intl. Joint Conf. on Neural Networks (IJCNN)*, pages 2959–2964, Aug 2007.
20. V. Pestov. An axiomatic approach to intrinsic dimension of a dataset. *Neural Networks*, 21(23):204 – 213, 2008. Advances in Neural Networks Research: 2007 International Joint Conference on Neural Networks (IJCNN).
21. H. Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
22. C. Traina Jr., A. J. M. Traina, and C. Faloutsos. Distance exponent: a new concept for selectivity estimation in metric trees. Research Paper 99-110, School of Computer Science, Carnegie Mellon University, 03/1999 1999.
23. P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search: The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer, 2006.

A Histograms of the real word metric spaces



(a) Diccionary. (b) NASA.

(c) Histograms. (d) Documents.

Fig. 4. Distance histograms for each real metric space considered.