The Role of Local Dimensionality Measures in Benchmarking Nearest Neighbor Search

Martin Aumüller

IT University of Copenhagen, Denmark
maau@itu.dk

Matteo Ceccarello

Free University of Bozen-Bolzano, Italy
matteo.ceccarello@unibz.it

Abstract

This paper reconsiders common benchmarking approaches to nearest neighbor search. It is shown that the concepts of local intrinsic dimensionality (LID), local relative contrast (RC), and query expansion allow to choose query sets of a wide range of difficulty for real-world datasets. Moreover, the effect of the distribution of these dimensionality measures on the running time performance of implementations is empirically studied. To this end, different visualization concepts are introduced that allow to get a more fine-grained overview of the inner workings of nearest neighbor search principles. Interactive visualizations are available on the companion website[1]. The paper closes with remarks about the diversity of datasets commonly used for nearest neighbor search benchmarking. It is shown that such real-world datasets are not diverse: results on a single dataset predict results on all other datasets well.

2012 ACM Subject Classification Theory of computation → Nearest neighbor algorithms; Computing methodologies → Simulation evaluation; Theory of computation → Computational geometry

Keywords and phrases Nearest neighbor search; Benchmarking

1 Introduction

Nearest neighbor (NN) search is a key primitive in many computer science applications, such as data mining, machine learning and image processing. For example, Spring and Shrivastava very recently showed in [30] how nearest neighbor search methods can yield large speed-ups when training neural network models. In this paper, we study the classical k-NN problem. Given a dataset $S \subseteq \mathbb{R}^d$, the task is to build an index on S to support the following type of query: For a query point $\mathbf{x} \in \mathbb{R}^d$, return the k closest points in S under some distance measure D.

In many practical settings, a dataset consists of points represented as high-dimensional vectors. For example, word representations generated by the `glove` algorithm [28] associate with each word in a corpus a d-dimensional real-valued vector. Common choices for d are between 50 and 300 dimensions. Finding the true nearest neighbors in such a high-dimensional space is difficult, a phenomenon often referred to as the "curse of dimensionality" [11]. In practice, it means that finding the true nearest neighbors, in general, cannot be solved much more efficiently than by a linear scan through the dataset (requiring time $O(n)$ for n data points) or in space that is exponential in the dimensionality d, which is impractical for large values of d.

While we cannot avoid these general hardness results [2], most datasets that are used in applications are not *truly* high-dimensional. This means that the dataset can be embedded onto a lower-dimensional space without too much distortion. Intuitively, the intrinsic

[1] `https://cecca.github.io/role-of-dimensionality-site/`

dimensionality (ID) of the dataset is the minimum number of dimensions that allows for such a representation [16]. There exist many explicit ways of finding good embeddings for a given dataset. For example, the Johnson-Lindenstrauss transformation [21] allows us to embed n data points in $\mathbb{R}^d$ into $\Theta((\log n)/\varepsilon^2)$ dimensions such that all pairwise distances are preserved up to a $(1+\varepsilon)$ factor with high probability. Another classical embedding often employed in practice is given by principal component analysis (PCA), see [22].

In this paper, we put our focus on *local measures of dimensionality.* In particular, we consider "local intrinsic dimensionality" (LID), a measure introduced by Houle in [16], an adapted version of "query expansion", a measure introduced by Ahle et al. in [1], and a local version of the "relative contrast" of the dataset introduced by He et al. in [15]. We defer a detailed discussion of these measures to Section 2. Intuitively, the LID of a data point $\mathbf{x}$ at a distance threshold $r > 0$ measures how difficult it is to distinguish between points at distance r and distance $(1+\varepsilon)r$ in a dataset. The Expansion of a data point $\mathbf{x}$ and a parameter $k > 0$ is the ratio of the distance of its $2k$-th nearest neighbor and its k-th nearest neighbor. The relative contrast (RC) of a data point $\mathbf{x}$ is the ratio between the mean distance of $\mathbf{x}$ to the points in the dataset and the distance to its nearest neighbor. The relative contrast of a dataset is then the average RC over all data points. Most importantly, all three measures are *local* measures that can be associated with a single query. It was stated in [17] that the LID might serve as a characterization of the difficulty of k-NN queries. One purpose of this paper is to shed light on this statement, as well as to compare it with the other measures.

A focus of this paper is an empirical study of how these local measures influence the performance of NN algorithms. To be precise, we will benchmark five different implementations [23] which employ different approaches to NN search. Four of them (`HNSW` [26], `IVF` [20], `Annoy` [8]), and `ONNG` [18] stood out as most performant in the empirical study conducted by Aumüller et al. in [5]. Finally, we included the very recent LSH-based approach (`PUFFINN`) from Aumüller et al. [7] that promises to give recall guarantees with an adaptive query algorithm.

Our experiments are based on the `ann-benchmarks` system from [5]. We describe their benchmarking approach and the changes we made to their system in Section 3. We analyze the distribution of local dimensionality measures of real-world datasets in Section 4. For all measures, we will see that there is a substantial difference between these distributions among datasets. We will then conduct two sets of experiments: First, we fix a dataset and choose as query set the set of points with smallest, medium, and largest estimated dimensionality measure, for each one of LID, RC, and query expansion. In addition, we choose a set of "diverse" query points w.r.t. their estimated dimensionality measure. As we will see, there is a clear tendency such that the larger the LID (resp. the smaller the RC and Expansion), the more difficult the query for all implementations. Among the three measures, the LID is the one for which this effect is most pronounced. Next, we will study how the different dimensionality distributions between datasets influence the running time distribution. In a nutshell, it cannot be concluded that any of the three dimensionality measures by itself is a good indicator for the relative performance of a fixed implementation over datasets.

In the first part of our evaluation, we work in the "classical evaluation setting of nearest neighbor search". This means that we relate a performance measure (such as the achieved throughput measured in queries per second) to a quality measure (such as the average fraction of true nearest neighbors found over all queries). While this is the most commonly employed evaluation method, we reason that this way of representing results in fact hides interesting details about the inner workings of an implementation. Using non-traditional visualization techniques provide new insights into their query behavior on real-world datasets. As one example, we see that reporting average recall on the graph-based approaches from [26, 18] hides an important detail: For a given query, they either find all true nearest neighbors or not a single one. This behavior is not shared by the three other approaches that we consider; all yield a continuous transition from "finding no nearest neighbors" to "finding all of them".

As a final point, we want, ideally, to benchmark on a collection of "interesting" datasets that show the strengths and weaknesses of individual approaches [29]. We will conclude that there is little diversity among the considered real-world datasets: While the individual performance observations change from dataset to dataset, the relative performance between implementations stays the same.

Our Contributions. The main contributions of this paper are

- a detailed evaluation of the distribution of local dimensionality measures of many real-world datasets used in benchmarking frameworks,
- a systematic way to create query workloads of a wide range of difficulty for nearest neighbor search,
- an evaluation of the influence of these different dimensionality measures on the performance of NN search implementations,
- considerations about the result diversity, and
- an exploration of different visualization techniques that shed light on individual properties of certain implementation principles.

We hope that our approach and the tools developed will find use in future benchmarking studies. In particular, the way to choose query workloads with varying difficulties results in interesting testbeds to benchmark implementations.

Related Work on Benchmarking Frameworks for NN. We use the benchmarking system described in [5] as the starting point for our study. Different approaches to benchmarking nearest neighbor search are described in [12, 13, 25]. We refer to [5] for a detailed comparison between the frameworks.

Related Work on the Meaningfulness of Nearest Neighbor Search. Beyer et al. [9] and Francois et al. [14] showed that under certain randomness assumptions and in the limit $d \to \infty$, nearest neighbor search queries become "meaningless", an effect usually referred to as the "concentration of distances". This means that the nearest and furthest neighbor of a data point become nearly indistinguishable. As mentioned in [15], these observations hold only asymptotically and usually do not occur in real-world datasets.

Relation to Conference Version. This paper is an extended version of the SISAP 2019 paper [6], which focused mainly on LID as a measure of local dimensionality. To have a better understanding of how much our observations generalized, this version includes two other measures (query expansion and relative contrast) and features a new NN implementation based on LSH (PUFFINN).

2 Local Dimensionality Measures

2.1 Local Intrinsic Dimensionality

We consider a distance-space $(\mathbb{R}^d, D)$ with a distance function $D\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. As described in [3], we consider the distribution of distances within this space with respect to a reference point $\mathbf{x}$. Such a distribution is induced by sampling n points from the space $\mathbb{R}^d$ under a certain probability distribution. We let $F\colon \mathbb{R} \to [0,1]$ be the cumulative distribution function of distances to the reference point $\mathbf{x}$.

▶ **Definition 1** ([16]). *The local continuous intrinsic dimension of F at distance r is given by*

$$ID_F(r) = \lim_{\varepsilon\to 0} \frac{\ln(F((1+\varepsilon)r)/F(r))}{\ln((1+\varepsilon)r/r)},$$

whenever this limit exists.

The measure relates the increase in distance to the increase in probability mass (the fraction of points that are within the ball of radius r and $(1+\varepsilon)r$ around the query point). Intuitively, the larger the LID, the more difficult it is to distinguish true nearest neighbors at distance r from the rest of the dataset. As described in [17], in the context of k-NN search we set r as the distance of the k-th nearest neighbor to the reference point $\mathbf{x}$.

Estimating LID We use the Maximum-Likelihood estimator (MLE) described in [24, 3] to estimate the LID of $\mathbf{x}$ at distance r. Let $r_1 \leq \ldots \leq r_k$ be the sequence of distances of the k-NN of $\mathbf{x}$. The MLE $\hat{\mathrm{ID}}_{\mathbf{x}}$ is then

$$\hat{\mathrm{ID}}_{\mathbf{x}} = -\left(\frac{1}{k}\sum_{i=1}^{k} \ln \frac{r_i}{r_k}\right)^{-1}. \tag{1}$$

Amsaleg et al. showed in [3] that MLE estimates the LID well. We remark that in very recent work, Amsaleg et al. proposed in [4] a new MLE-based estimator that works with smaller k values than (1).

2.2 Query Expansion

The concept of the Expansion around a query point at a distance threshold $r > 0$ was introduced by Ahle et al. in [1]. In their work, the query expansion $c^*_{\mathbf{x}}$ is the largest $c^*_{\mathbf{x}} > 0$ such that the number of points within distance $c^*_{\mathbf{x}} r$ is at most twice the number of points at distance r. They use this concept to show that an LSH approach can adapt to the query expansion. More precisely, the larger the query expansion, the less work is conducted by their adaptive query algorithm in expectation.

For our use case in k-NN search, we adapt the notion of query expansion as follows.

▶ **Definition 2.** *Given a data set S, an integer $k > 0$, and a data point $\mathbf{x}$, the Expansion of $\mathbf{x}$ at k is $dist(\mathbf{x}, \mathbf{x}_{2k})/dist(\mathbf{x}, \mathbf{x}_k)$, where $\mathbf{x}_i$ is the i-th nearest neighbor of $\mathbf{x}$ in S for $1 \leq i \leq |S|$.*

2.3 Relative Contrast

The concept of relative contrast (RC) was introduced by He et al. in [15]. Here, we concentrate on the following local variant.

▶ **Definition 3.** *Given a data set S, an integer $k > 0$, and a data point $\mathbf{x}$, let d_{mean} be the average distance of $\mathbf{x}$ to the points in S. The local relative contrast of $\mathbf{x}$ in S is then $d_{mean}/dist(\mathbf{x}, \mathbf{x}^*_k)$, where $\mathbf{x}^*_k$ is the k-th nearest neighbor of $\mathbf{x}$ in S.*

The relative contrast of the dataset S is the average local relative contrast over all points in a query set. It was shown in [15] that—if the relative contrast of the dataset is known—there is a way to choose LSH parameters to adapt to the RC. In the same way as query expansion, higher contrast means lower running time.

While both Expansion and RC relate the distance of a nearest neighbor to distances of other data points, RC has a much more global view on the dataset while Expansion considers distances between close points.

3 Overview over the Benchmarking Framework

We use the `ann-benchmarks` system described in [5] to conduct our experimental study. Ann-benchmarks is a framework for benchmarking NN search algorithms. It covers dataset creation, performing the actual experiment, and storing the results of these experiments in a transparent and easy-to-share way. Moreover, results can be explored through various plotting functionalities, e.g., by creating a website containing interactive plots for all experimental runs.

Ann-benchmarks interfaces with a NN search implementation by calling its preprocess (index building) and search (query) methods with certain parameter choices. Implementations are tested on a large set of parameters usually provided by the original authors of an implementation. The answers to queries are recorded as the indices of the points returned. Ann-benchmarks stores these parameters together with further statistics such as individual query times, index size, and auxiliary information provided by the implementation. See [5] for more details.

Compared to the system described in [5], we added tools to estimate the LID based on Equation (1), to estimate the query Expansion based on Definition 2, to estimate the RC based on Definition 3, pick "challenging query sets" according to the LID, query expansion, and RC of individual points, and added new datasets and implementations. Moreover, we implemented a mechanism that allows an implementation to provide further query statistics after answering a query. To showcase this feature, all implementations in this study report the number of distance computations performed to answer a query.[2]

4 Algorithms and Datasets

4.1 Algorithms

Nearest neighbor search algorithms for high dimensions are usually graph-, tree-, or hashing-based. We refer the reader to [5] for an overview over these principles and available implementations. In this study, we concentrate on the three implementations considered most performant in [5], namely `HNSW` [26], `Annoy` [8] and `FAISS-IVF` [20] (`IVF` from now on). We consider the very recent graph-based approach `ONNG` [18], and the recent LSH-based approach `PUFFINN` [7] in this study as well.

`HNSW` and `ONNG` are graph-based approaches. This means that they build a k-NN graph during the preprocessing step. In this graph, each vertex is a data point and a directed edge (u, v) means that the data point associated with v is "close" to the data point associated with u in the dataset. At query time, the graph is traversed to generate candidate points.

[2] We thank the authors of the implementations for their help and responsiveness in adding this feature to their library.

			LID		EXP		RC		
Dataset	**Data Points**	**Dim.**	**avg**	**median**	**avg**	**median**	**avg**	**median**	**Metric**
SIFT [19]	1 000 000	128	19.4	19.0	1.042	1.035	2.6	2.4	Euclidean
MNIST	65 000	784	13.9	13.1	1.057	1.053	2.2	2.0	Euclidean
Fashion-MNIST [31]	65 000	784	15.4	13.8	1.052	1.046	2.8	2.7	Euclidean
GLOVE [28]	1 183 514	100	17.9	17.6	1.054	1.041	2.3	2.1	Cosine
GLOVE-2M [28]	2 196 018	300	25.8	23.2	1.055	1.032	2.2	1.7	Cosine
GNEWS [27]	3 000 000	300	20.9	19.9	1.044	1.034	2.6	2.3	Cosine

Table 1 Datasets under consideration with their average local intrinsic dimensionality (LID), their query expansion (EXP), and their local relative contrast (RC). LID is computed by MLE [3] from the 100-NN of all the data points, EXP is computed by the fraction of distances to the 10-th NN and 20-th NN, and RC is computed relating the distance of the 10-th NN to the average distance computed from a random sample of 10 000 data points.

Algorithms differ in details of the graph construction, how they build a navigation structure on top of the graph, and how the graph is traversed.

`Annoy` is an implementation of a random projection forest, which is a collection of random projection trees. Each node in a tree is associated with a set of data points. It splits these points into two subsets according to a chosen hyperplane. If the dataset in a node is small enough, it is stored directly and the node is a leaf. `Annoy` employs a data-dependent splitting mechanism in which a splitting hyperplane is chosen as the one splitting two "average points" by repeatedly sampling dataset points. In the query phase, trees are traversed using a priority queue until a predefined number of points is found.

`IVF` builds an inverted file based on clustering the dataset around a predefined number of centroids. It splits the dataset based on these centroids by associating each point with its closest centroid. During query it finds the closest centroids and checks points in the dataset associated with those.

`PUFFINN` uses an adaptive trie-like multi-layer LSH data structure to guide the search. Using the probabilistic nature of LSH, it exploits adaptive termination criteria to give guaranteed recall [7] without the need of parameter tuning as in the other approaches. We note that PUFFINN does not support Euclidean distance and is thus missing in some plots.

We remark we used both `IVF` and `HNSW` implementations from `FAISS`[3].

4.2 Datasets

Table 1 presents an overview over the datasets that we consider in this study. We restrict our attention to datasets that are usually employed in connection with Euclidean distance and Angular/Cosine distance. For each dataset, we compute the LID distribution with respect to the 100-NN as discussed in Section 2, in order to get a stable estimate. Furthermore, we compute the Expansion using, for each point, the distance of its 10-th nearest neighbor as a threshold, as discussed in Section 2. The RC is estimated by computing the distance of each point to a sample of 3000 points. SIFT, MNIST, and GLOVE are among the most-widely used datasets for benchmarking nearest neighbor search algorithms. Fashion-MNIST is considered as a replacement for MNIST, which is usually considered too easy for machine learning tasks [31].

[3] `https://github.com/facebookresearch/faiss`

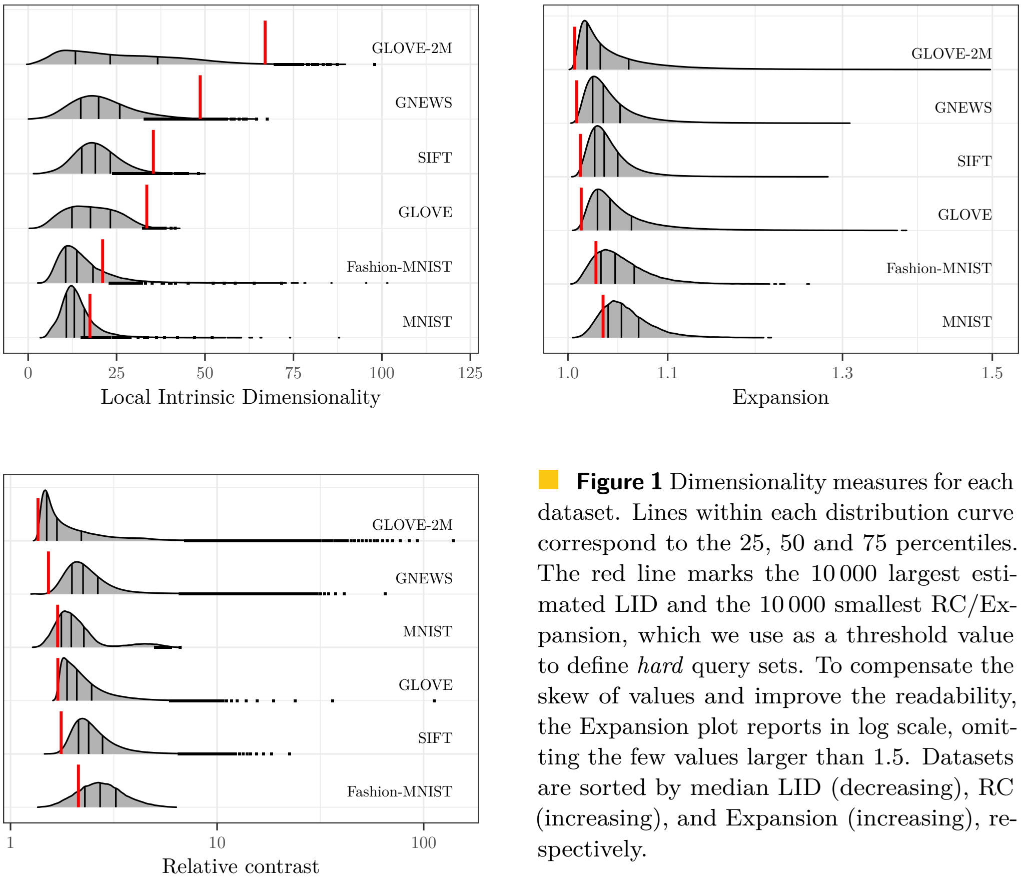


Figure 1 Dimensionality measures for each dataset. Lines within each distribution curve correspond to the 25, 50 and 75 percentiles. The red line marks the 10 000 largest estimated LID and the 10 000 smallest RC/Expansion, which we use as a threshold value to define *hard* query sets. To compensate the skew of values and improve the readability, the Expansion plot reports in log scale, omitting the few values larger than 1.5. Datasets are sorted by median LID (decreasing), RC (increasing), and Expansion (increasing), respectively.

Figure 1 provides a visual representation of the estimated distributions of LID, RC, and Expansion of each dataset, for $k = 100$. While the datasets differ widely in their original dimensionality, the median LID ranges from around 13 for MNIST to about 23 for GLOVE-2M. The distribution of LID values is asymmetric and shows a long tail behavior. MNIST, Fashion-MNIST, SIFT, and GNEWS are much more concentrated around the median compared to the two `GLOVE`-based datasets. Considering the RC measure, also its distribution is asymmetric and long tailed, with mean and median values pretty close to each other. Nonetheless, the distributions differ in their shape and the length of the tail. As for the Expansion, the values are very concentrated towards 1 (the minimum value), with extremely long tails which have been cut out of the figure for the sake of readability.

5 Evaluation

This section reports on the results of our experiments. Due to space constraints, we only present some selected results. More results can be explored via interactive plots at `https://cecca.github.io/role-of-dimensionality-site/`, which also contains a link to the source code repository. For a fixed implementation, the plots presented here consider the Pareto frontier over all parameter choices [5]. Tested parameter choices and the associated plots are available on the website.

Table 2 Pearson correlation of between the three different measures for each dataset. The correlations, although mild, are all statistically significant.

dataset	LID/Expansion	LID/RC	Expansion/RC
Fashion-MNIST	-0.621	-0.645	0.530
GLOVE	-0.571	-0.527	0.551
GLOVE-2M	-0.363	-0.253	0.325
GNEWS	-0.452	-0.321	0.312
MNIST	-0.569	-0.583	0.518
SIFT	-0.550	-0.527	0.400

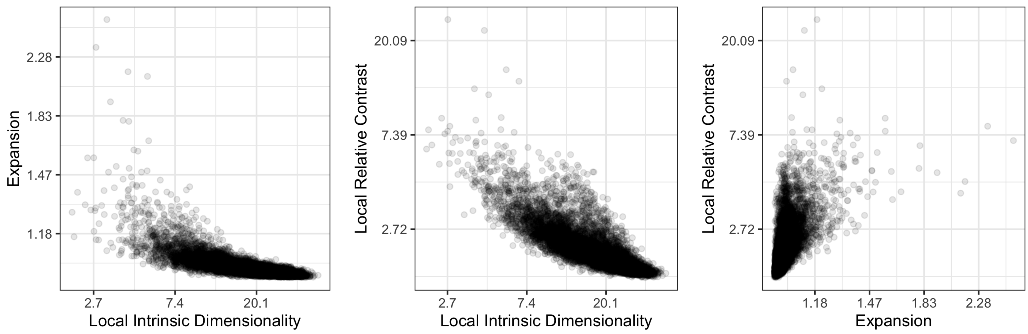


Figure 2 Scatterplots relating LID, Expansion, and RC for the GLOVE dataset for a sample of 10 000 data points. All the scales are logarithmic.

Experimental Setup Experiments were run on 2x 14-core Intel Xeon E5-2690v4 (2.60GHz) with 512GB RAM using Ubuntu 16.10 (kernel 4.4.0). Index building was multi-threaded, queries where answered in a single thread.

Quality and Performance Metrics As quality metric we measure the individual recall of each query, i.e., the fraction of points reported by the implementation that are among the true k-NN. As performance metric, we record individual query times and the total number of distance computations needed to answer all queries. We usually report on the throughput, i.e. the average number of queries that can be answered in one second, in the plots denoted as QPS for *queries per second*.

Objectives of the Experiments Our experiments are tailored to answer the following questions:

(Q1) How do LID, Expansion, and RC correlate with each other? (Section 5.1)
(Q2) How do the LID, Expansion, and RC of a query set influence performance of an implementation? (Section 5.2 and 5.3)
(Q3) How well does the number of distance computations reflect the relative running time performance of the tested implementations? (Section 5.4)
(Q4) How diverse are measurements obtained on datasets? Do relative differences between the performance of different implementations stay the same over multiple datasets? (Section 5.4)
(Q5) How concentrated are quality and performance measures around their mean for the tested implementations? (Section 5.5)

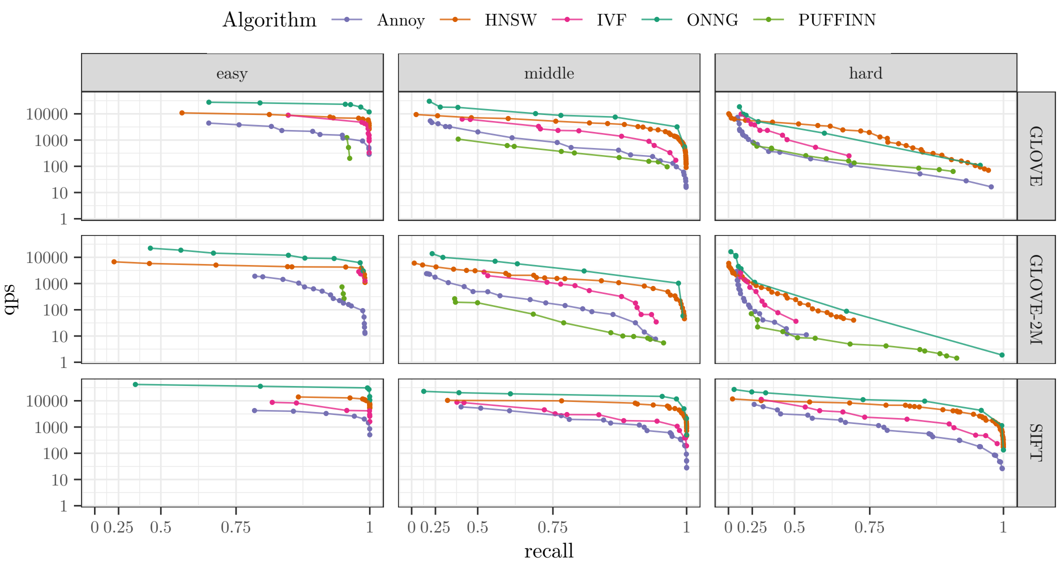


Figure 3 Recall-QPS (1/s) tradeoff – up and to the right is better – for queries selected according to LID, solved using different algorithms. Three datasets are considered here: GLOVE, GLOVE-2M and SIFT. The scale is logarithmic on the y axis and exponential on the x axis, to take into account the scale of the data.

Choosing Query Sets For each dataset, we select eight different query sets:

easy the 10 000 points with the lowest estimated LID (resp. the highest Expansion/RC)

medium the 10 000 points around the data point with median estimated LID (resp. Expansion/RC)

hard the 10 000 points with the highest estimated LID (resp. lowest Expansion/RC)

diverse 5 000 points chosen so to span the entire range of LID values (resp. Expansion/RC values). For the LID, we split all data points up into buckets, according to their rank by LID. For each query, we pick a non-empty bucket uniformly at random, and inside the bucket we pick a random point (with repetition). For Expansion and RC, we pick the 1 500 points with smallest and largest values, and add 2 000 points picked uniformly at random from the remaining points (with repetition).

Figure 1 marks with a red line the LID used as a threshold to build the *hard* queryset.

Main takeaways The following experimental evaluation presents a lot of results, giving the following main insights. First, we can use local dimensionality measures to build benchmark query sets of varying difficulty. Second, among these measures, the Local Intrinsic Dimensionality is the single most effective one at selecting queries of the desired accuracy. Then, the *diverse* query set is a good general benchmark, in that it includes queries of a wide range of difficulties. Finally, average performance measures are convenient but often hide interesting behaviour, which is best studied by looking at their distribution.

5.1 How Well do the Local Dimensionality Measures Correlate?

Figure 2 visualizes the correlation between the three different local dimensionality measures. As our working hypothesis, a higher LID score is associated with a higher difficulty for a

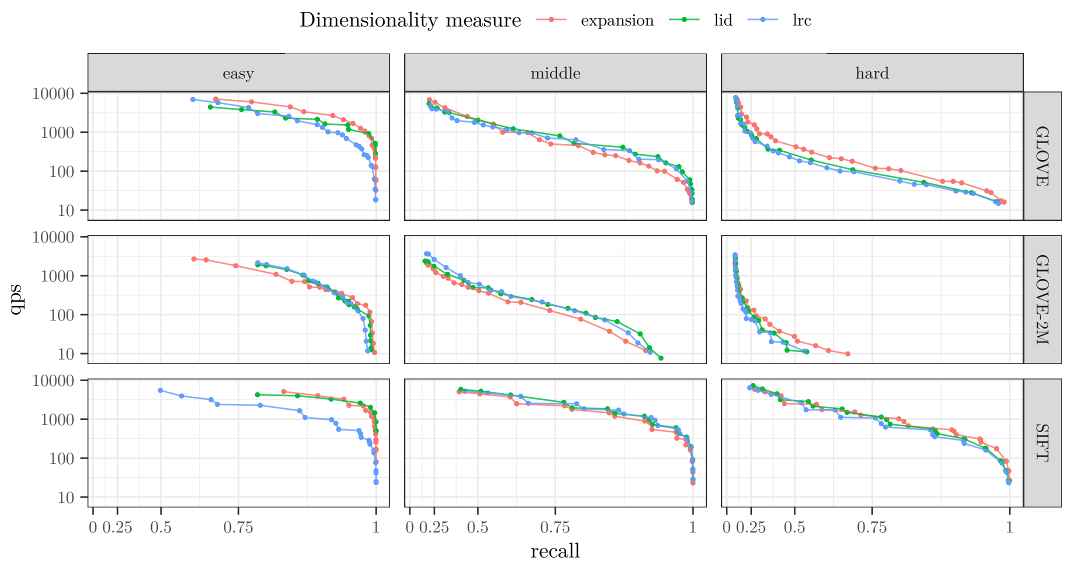


Figure 4 Recall-QPS (1/s) tradeoff – up and to the right is better – for algorithm Annoy solving queries selected according to LID, RC, and Expansion. Three datasets are considered here: GLOVE, GLOVE-2M and SIFT. The scale is logarithmic on the y axis and exponential on the x axis, to take into account the scale of the data.

query, as is a low Expansion or RC score. The plot shows some correlation between the scores: points with a high LID also have a low Expansion and RC, and vice versa. The effect is particularly marked for very easy points: points with a LID ≤ 5 also have a markedly high Expansion and RC, which are otherwise very concentrated around the mean. If we compute the Pearson correlation between the different measures (reported in Table 2) we can see that they are mildly correlated. We notice that for both Expansion and RC there exist some outliers: that is points with low LID, i.e., that are classified as easy queries, which have very small Expansion/RC, i.e., are categorized as a difficult query. Similarly, for the relation of Expansion and RC, we see a general correlation with a few unstructured outliers. It will be interesting to see how these correlations are reflected in the performance of an implementation.

5.2 Influence of Dimensionality Measures on Performance

Figure 3 reports the performance of different configurations of all the algorithms we consider on the GLOVE, GLOVE-2M, and SIFT datasets, drawing queries according to the LID. In these plots, the best performance is attained in the upper right corner: high recall and high throughput.

We observe a clear influence of the LID of the query set on the performance: the more difficult the query set, i.e., the larger the LID, the more down and to the left the graphs move, for all algorithms. This means that for higher LID it is more expensive, in terms of time, to answer queries with good recall.

For all datasets except GLOVE-2M (and GNEWS with the difficult query set), almost all implementations were still able to achieve close to perfect recall with the parameters set. This means that even for queries with large LID there are points in the dataset that can be

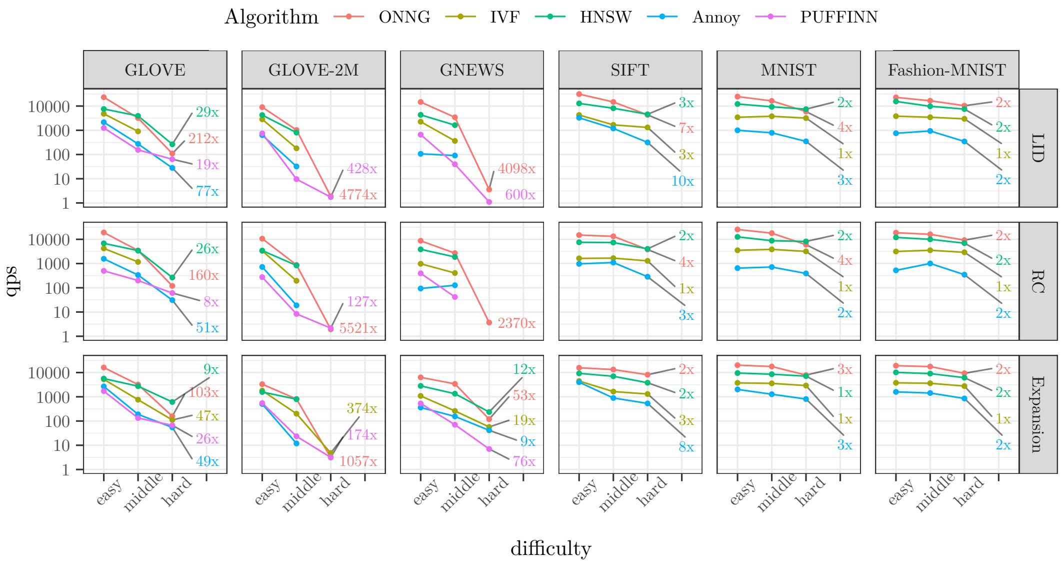


Figure 5 Change of performance of the fastest configuration achieving at least 0.9 recall as the difficulty of the dataset changes. The colored labels report the slowdown factor of the *hard* queryset compared with the *easy* one.

efficiently separated from the others.

We now turn our focus on the relationship of the three dimensionality measures with the performance of the algorithm. Figure 4 considers the same setup as before showing the results for `Annoy` for LID, Expansion, and RC. First of all, we observe that *easy*, *middle*, and *hard* query sets show the same behaviour we observed in Figure 3: selecting queries with lower Expansion/RC (or higher LID) makes them more difficult to solve for the algorithm. Furthermore, we note that, among the three dimensionality measures, the Expansion yields easier query sets with respect to the other two (i.e. the red line is always more up and to the right). At the same time, LID and RC yield query sets of comparable difficulty.

To better investigate the influence that dimensionality measures have for all datasets and implementations, consider Figure 5, which reports the change in performance of the fastest configuration attaining recall at least 0.9, for each algorithm. Clearly, all measures allow to select query sets which are progressively more difficulty to solve accurately for all algorithms. However, as shown by the labels in each plot, the LID allows to select *easy* and *hard* querysets that have a wider performance gap than the ones selected by Expansion or RC, also for the datasets in which all implementations achieve high recall on the hard query set.

5.3 Predictive Quality of Dimensionality Measures

In the previous two subsections, we found evidence that all dimensionality measures allow to pick query sets of various difficulties. Fixing the implementation and considering all datasets, how well does a dimensionality measure work between two different datasets? Figure 6 reports the queries per second of `Annoy` for a certain choice of datasets, with queries chosen from the middle, hard, and diverse query set.

Comparing results to the dimensionality measurements depicted in Figure 1, we first

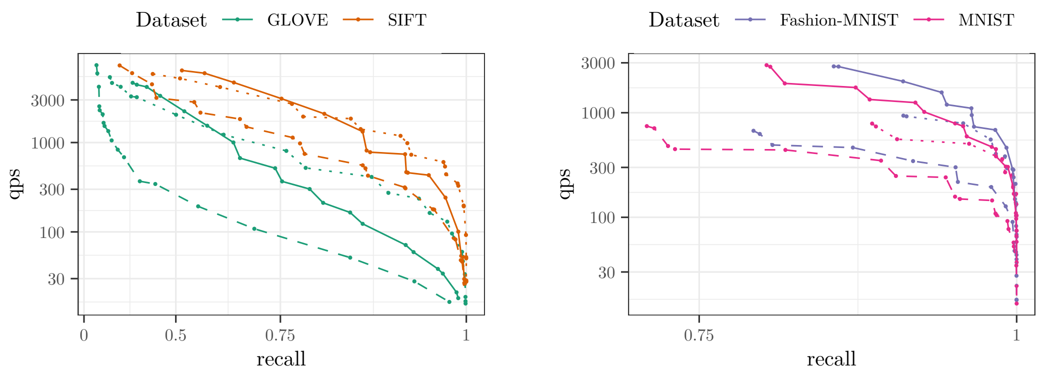


Figure 6 Recall-QPS (1/s) tradeoff – up and to the right is better – for Annoy on GLOVE, SIFT, Fashion-MNIST, and MNIST with queries selected according to LID. Dashed lines are *hard* query sets, solid lines are *diverse* query sets, dotted lines are *middle* query sets.

observe that the estimated median LID, RC and Expansion all give a good estimate on the relative performance of the algorithms on the data sets: recall that in Figure 1 the datasets are sorted by median score. (The plot is missing lines for GNEWS and GLOVE-2M, which are considerably more challenging according to Table 5.) As an exception, SIFT (middle) is much easier than predicted by its LID and Expansion distribution, but the RC measure predicts this, ranking SIFT lower than GLOVE. In particular, the hard SIFT instance (orange solid line) is as challenging as the medium GLOVE version (green dotted line). On the other hand, RC classifies MNIST as rather difficult to index, in particular compared to Fashion-MNIST. The plot on the right in Figure 6 clearly indicates that this is not true, and instead the two datasets are basically equivalent. From this, we cannot conclude that the considered local dimensionality measures as a single indicator explain performance differences of an implementation across different datasets.

It can also be seen from the plot that the diverse query set is more difficult than the medium query set. In particular, at high recall it generally becomes nearly as difficult as the difficult dataset. For many implementations, the reason for this behaviour is that they cannot adapt to the difficulty of a query. They only achieve high average recall when they can solve sufficiently many queries with high LID or low Expansion. The parameter settings that allow for such guarantees slow down answering the easy queries by a lot. This manifests in running times that are indistinguishable from those on the hard dataset, while only roughly 30% of the queries are characterized as difficult ones. As we shall see in Section 5.5, some algorithms are indeed able to adapt to the difficulty of the query. We believe that the "diverse" query sets thus allow for challenging benchmarking datasets for adaptive query algorithms.

As a side note, we remark that Fashion-MNIST is as difficult to solve as MNIST for all implementations, and is by far the easiest dataset for all implementations. Thus, while there is a big difference in the difficulty of solving the classification task [31], there is no measurable difference between these two datasets in the context of NN search.

5.4 Diversity of Results

Figure 7 gives an overview over how algorithms compare to each other among all "medium difficulty" querysets, selected according to the LID. Results for Expansion- and RC-based querysets are similar. We consider two metrics, namely the number of queries per second (top plot), and the number of distance computations (bottom plot). For two different average

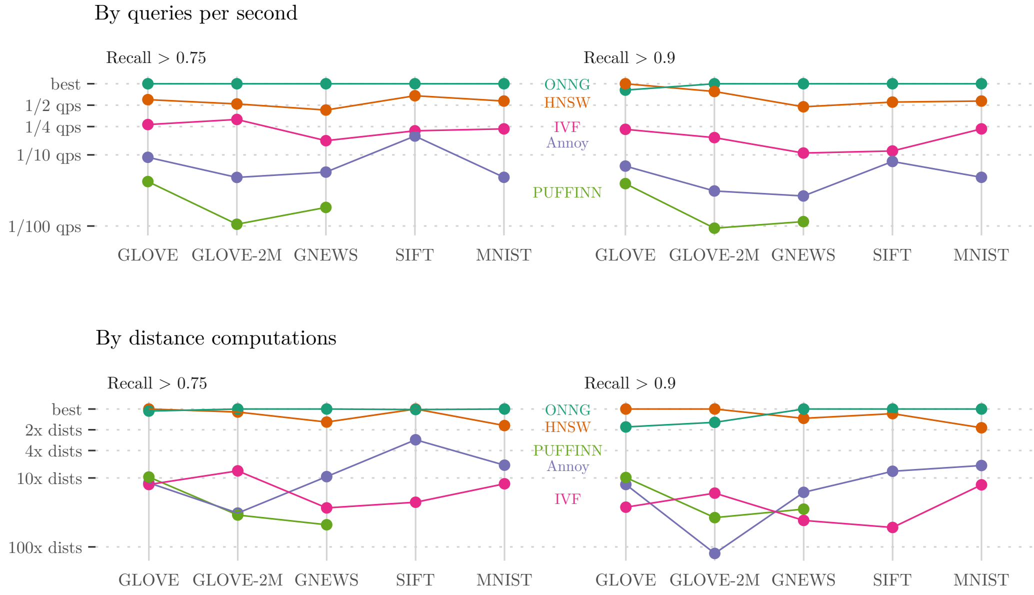
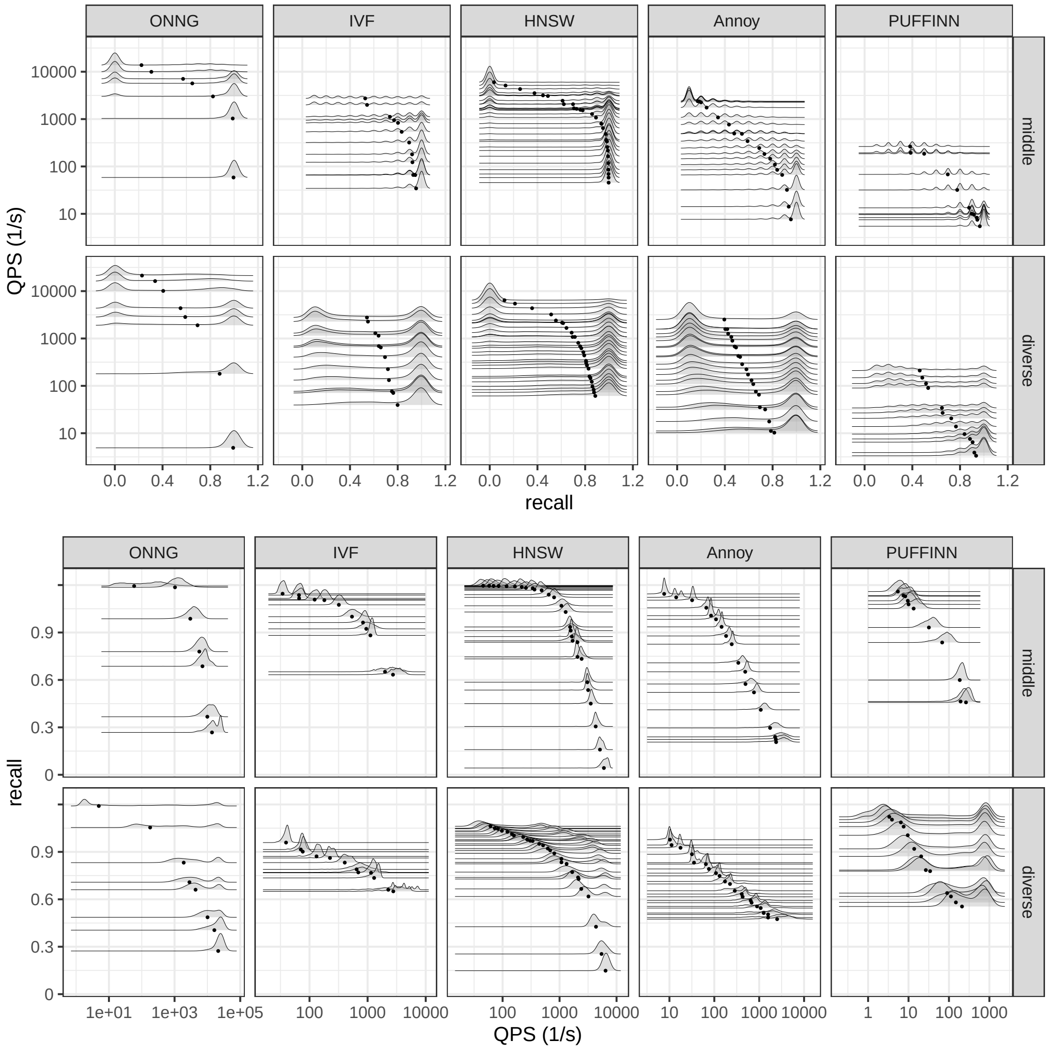


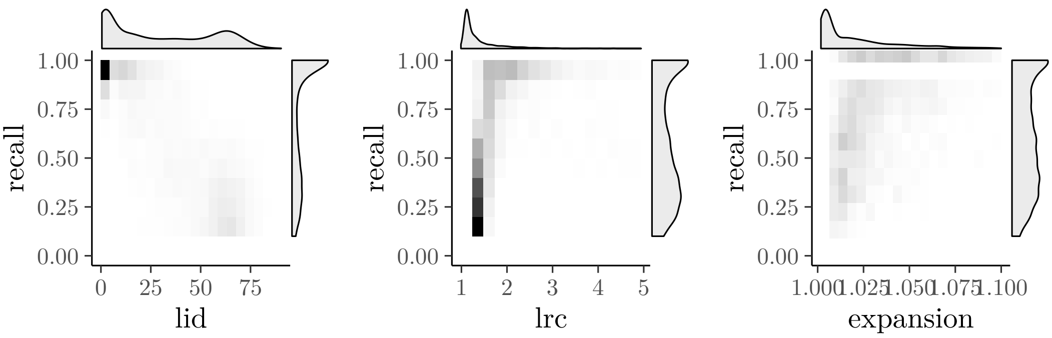
Figure 7 Ranking of algorithm on five different datasets, according to recall ≥ 0.75 and ≥ 0.9, and according to two different performance measures: number of queries per second (top) and number of distance computations (bottom). Both plots report the ratio with the best performing algorithm on each dataset, higher is better. Note that the scale is logarithmic.

recall thresholds (0.75 and 0.9) we then select, for each algorithm, the best performing parameter configuration that attains at least that recall. For each dataset, the plots report the ratio with the best performing algorithm on that dataset, therefore the best performer is reported with ratio 1. Considering different dataset, we see that there is little variation in the ranking of the algorithms. Only the two graph-based approaches trade ranks, all other rankings are stable. `Annoy` makes fewer distance computations (hence ranks higher in the figure) but is consistently outperformed by `IVF`.[4]

Comparing the number of distance computations to running time performance, we see that an increase in the number of distance computations is not reflected in a proportional decrease in the number of queries per second. This means that the candidate set generation is in general more expensive for graph-based approaches, but the resulting candidate set is of much higher quality and fewer distance computations have to be carried out. Generally, both graph-based algorithms are within a factor 2 from each other, whereas the other two need much larger candidate lists to achieve a certain recall. The relative difference usually ranges from 5x to 30x more distance computations for the non-graph based approaches, in particular at high recall. This translates well into the performance differences we see in this setting: consider for instance Figure 3, where the lines corresponding to `HNSW` and `ONNG` upper bound the lines relative to the other algorithms.



■ **Figure 8** Distribution of performance for queries on the GLOVE-2M (medium difficulty) dataset. Looking just at the average performance can hide interesting behaviour.



■ **Figure 9** Distribution of Recall vs. LID, RC, and Expansion plot on the GLOVE-2M dataset, using `Annoy`. Intensity reflects number of queries that achieve a combination of recall vs. LID (or RC or Expansion). The RC plot reports only queries with RC below 5. The Expansion plot reports only queries with Expansion below 1.1, which are the majority.

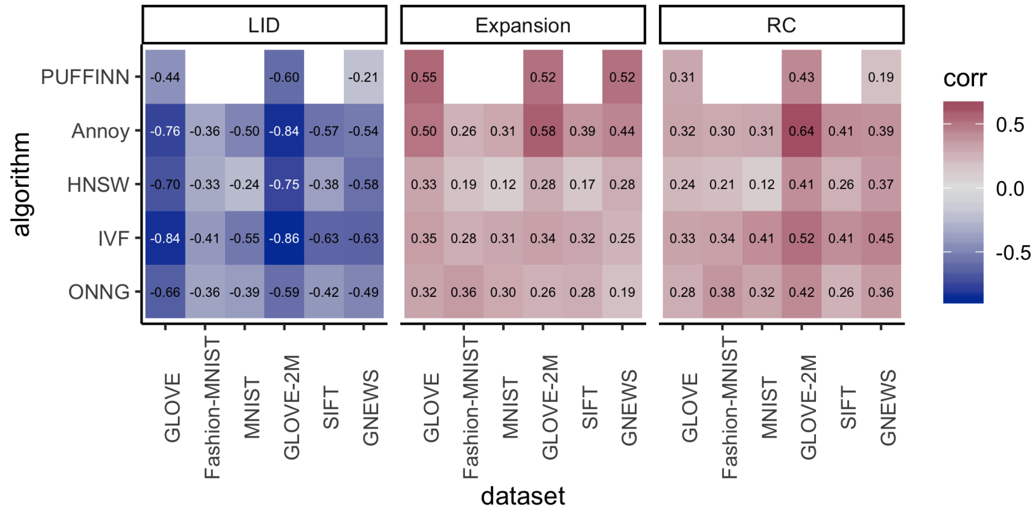


Figure 10 Average correlation between recall and dimensionality measure (across parameter configurations) for dataset/algorithm pairs, for each type of dimensionality measure. Note how LID correlates more strongly with recall. Furthermore, note that some algorithms (`Annoy`, `IVF`) are more sensitive than others to the dimensionality of the queries.

5.5 Reporting the Distribution of Performance

In the previous sections, we made extensive use of recall/queries per second plots, where each configuration of each algorithm results in a single point, namely the average recall and the inverse of the average query time. As we shall see in this section, concentrating on averages can hide interesting information in the context of k-NN queries. In fact, not all queries are equally difficult to answer. Consider the plots in Figure 8, which report the performance of the five algorithms on the GLOVE-2M dataset, with medium and diverse difficulty queries selected according to LID. The top 2x5 plots report the recall versus the number of queries per second for middle (top) and diverse (bottom) query sets, and black dots correspond to the averages. Additionally, for each configuration, we report the distribution of the recall scores: the baseline of each recall curve is positioned at the corresponding queries per second performance. Similarly, the bottom plots report on the inverse of the individual query times (the average of these is the QPS in the left plot) against the average recall. In both plots, the best performance is achieved towards the top-right corner.

Plotting the distributions, instead of just reporting the averages, uncovers some interesting behavior that might otherwise go unnoticed, in particular with respect to the recall. The average recall gradually shifts towards the right as the effect of more and more queries achieving good recalls. Perhaps surprisingly, for graph-based algorithms this shift is very sudden: most queries go from having recall 0 to having recall 1, taking no intermediate values, even for the query set that have very similar LID values. Taking the average recall as a performance metric is convenient in that it is a single number to compare algorithms with. However, the same average recall can be attained with very different distributions: looking at such distributions can provide more insight.

[4] We note that `IVF` counts the initial comparisons to find the closest centroids as distance computations, whereas `Annoy` did not count the inner product computations during tree traversal.

For the bottom plots and the middle query set, we observe that individual query times of all the algorithms are well concentrated around their mean. For the diverse dataset, algorithms might be able to adapt to the query difficulty. We observe that this is not true for `Annoy` and `IVF`. Both of them have a single peak in their query time, which means that they spend about the same time per query. On the other hand, `PUFFINN`, `HNSW`, and `ONNG` have two peaks in their performance distribution when they approach high recall. This means that they adapt to the presence of easy queries (where both `ONNG` and `PUFFINN` report with the same performance, and `HNSW` becomes slower for higher recall). It is surprising to see that all adaptive algorithms have two peaks, while the diverse query set is a mix of three different difficulties.

Figure 9 gives another distributional view on the achieved result quality. The plots show a run of `Annoy` on the GLOVE-2M dataset with diverse queries. On the top margin we see the distribution of estimated LID values (left plot), RC values (middle plot), and Expansion values (right plot) for the diverse query set, on the right margin we see the distribution of recall values achieved by the implementation. Each of the queries corresponds to a single data point in the recall/LID plot and data points are summarized through squares, where the color intensity of a square indicates the number of data points falling into this region. The plots show that the higher the LID of a query, there is a clear tendency for the query to achieve lower recall. Expansion and RC, instead, are less predictive in this setting: we can still observe that low Expansion (i.e. difficult) queries have low recall, but the relationship is less marked.

To further investigate the relationship between the dimensionality measures and the recall, we compute the correlation between each measure and the recall, reporting it in Figure 10. We observe that, as expected, the LID is is negatively correlated with the recall (i.e. the higher the LID, the harder it is to answer the query accurately), whereas the RC and Expansion are positively correlated. Looking at the magnitude of the correlation, we can clearly see that for any pair of dataset and algorithm, the recall correlates more strongly with the LID. Therefore, if we have to pick queries according to a single local dimensionality measure, the LID is the best predictor for the difficulty. Obviously, our observation that no single dimensionality measure is a perfect predictor for the difficulty of queries still holds. Interestingly, the choice of Expansion as a cost measure to which an LSH query algorithm may adapt to in [1] seems well-motivated: As the only out of five implementation, LSH-based `PUFFINN` shows the strongest correlation to Expansion and not to LID.

For space reasons, we do not report other parameter configurations and datasets, which nonetheless show similar behaviors. All of them can be accessed at the website.

6 Summary

In this paper we studied the influence of LID, RC, and Expansion to the performance of nearest neighbor search algorithms. We showed that all three measures allow to choose query sets of a wide range of difficulty from a given dataset. We also showed how different LID, RC, and Expansion distributions influence the running time performance of the algorithms. In this respect, we found that LID is a better predictor of performance than the other two. In any case, we could not conclude that the any of the three scores alone can predict running time differences well. In particular, SIFT is usually easier than GLOVE for the algorithms: while GLOVE's LID distribution would predict the opposite, the RC distribution correctly predicts this relationship between the datasets. However, the RC distribution does not predict differences correctly, either.

With regard to challenging query workloads, we described a way to choose diverse query sets. They have the property that for most implementations it is easy to perform well for most of the query points, but they contain many more easy and difficult queries than query workloads chosen randomly from the dataset. We believe this is a very interesting benchmarking workload for approaches that try to adapt to the difficulty of an individual query.

We introduced novel visualization techniques to show the uncertainty within the answer to a set of queries, which made it possible to show a clear difference between the graph-based algorithms and the other approaches. Furthermore, these visualizations allow to see whether a particular algorithm is able to adapt to the difficulty of the queries.

We hope that this study initiates the search for more diverse datasets, or for theoretical reasoning why certain algorithmic principles are generally better suited for nearest neighbor search. On a more practical side, Casanova et al. showed in [10] how dimensionality testing can be used to speed up reverse k-NN queries. We would be interested in seeing whether the LID can be used at other places in the design of NN algorithms to guide the search process or the parameter selection.

Acknowledgements

The authors would like to thank the anonymous reviewers of the conference version of this paper, who helped to improve the presentation of the paper. The research leading to these results has received funding from the European Research Council under the European Union's 7th Framework Programme (FP7/2007-2013) / ERC grant agreement no. 614331.

References

1. Ahle, T.D., Aumüller, M., Pagh, R.: Parameter-free locality sensitive hashing for spherical range reporting. In: SODA. pp. 239–256. SIAM (2017)
2. Alman, J., Williams, R.: Probabilistic polynomials and hamming nearest neighbors. In: FOCS'15. pp. 136–150
3. Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M.E., Kawarabayashi, K.I., Nett, M.: Estimating local intrinsic dimensionality. In: KDD'15. pp. 29–38. ACM (2015)
4. Amsaleg, L., Chelly, O., Houle, M.E., Kawarabayashi, K.i., Radovanović, M., Treeratanajaru, W.: Intrinsic dimensionality estimation within tight localities. In: Proceedings of the 2019 SIAM International Conference on Data Mining. pp. 181–189. SIAM (2019)
5. Aumüller, M., Bernhardsson, E., Faithfull, A.J.: Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Inf. Syst. **87** (2020), see https://arxiv.org/abs/1807.05614 for an open access version.
6. Aumüller, M., Ceccarello, M.: The role of local intrinsic dimensionality in benchmarking nearest neighbor search. In: SISAP. Lecture Notes in Computer Science, vol. 11807, pp. 113–127. Springer (2019)
7. Aumüller, M., Christiani, T., Pagh, R., Vesterli, M.: PUFFINN: parameterless and universally fast finding of nearest neighbors. In: ESA. LIPIcs, vol. 144, pp. 10:1–10:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
8. Bernhardsson, E.: Annoy, `https://github.com/spotify/annoy`
9. Beyer, K.S., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: ICDT. Lecture Notes in Computer Science, vol. 1540, pp. 217–235. Springer (1999)
10. Casanova, G., Englmeier, E., Houle, M.E., Kröger, P., Nett, M., Schubert, E., Zimek, A.: Dimensional testing for reverse k-nearest neighbor search. PVLDB **10**(7), 769–780 (2017)

11 Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. ACM Comput. Surv. **33**(3), 273–321 (Sep 2001). https://doi.org/10.1145/502807.502808

12 Curtin, R.R., Cline, J.R., Slagle, N.P., March, W.B., Ram, P., Mehta, N.A., Gray, A.G.: MLPACK: A scalable C++ machine learning library. J. of Machine Learning Research **14**, 801–805 (2013)

13 Edel, M., Soni, A., Curtin, R.R.: An automatic benchmarking system. In: NIPS 2014 Workshop on Software Engineering for Machine Learning (2014)

14 François, D., Wertz, V., Verleysen, M.: The concentration of fractional distances. IEEE Trans. Knowl. Data Eng. **19**(7), 873–886 (2007)

15 He, J., Kumar, S., Chang, S.: On the difficulty of nearest neighbor search. In: ICML. icml.cc / Omnipress (2012)

16 Houle, M.E.: Dimensionality, discriminability, density and distance distributions. In: Data Mining Workshops (ICDMW). pp. 468–473. IEEE (2013)

17 Houle, M.E., Schubert, E., Zimek, A.: On the correlation between local intrinsic dimensionality and outlierness. In: SISAP'18. pp. 177–191 (2018). https://doi.org/10.1007/978-3-030-02224-2_14

18 Iwasaki, M., Miyazaki, D.: Optimization of Indexing Based on k-Nearest Neighbor Graph for Proximity Search in High-dimensional Data. ArXiv e-prints (Oct 2018)

19 Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Trans. Pattern Anal. Mach. Intell. **33**(1), 117–128 (2011). https://doi.org/10.1109/TPAMI.2010.57

20 Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. CoRR **abs/1702.08734** (2017)

21 Johnson, W.B., Lindenstrauss, J., Schechtman, G.: Extensions of Lipschitz maps into Banach spaces. Israel Journal of Mathematics **54**(2), 129–138 (1986)

22 Jolliffe, I.: Principal component analysis. Springer (2011)

23 Kriegel, H., Schubert, E., Zimek, A.: The (black) art of runtime evaluation: Are we comparing algorithms or implementations? Knowl. Inf. Syst. **52**(2), 341–378 (2017)

24 Levina, E., Bickel, P.J.: Maximum likelihood estimation of intrinsic dimension. In: NIPS'15. pp. 777–784 (2005)

25 Li, W., Zhang, Y., Sun, Y., Wang, W., Zhang, W., Lin, X.: Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement (v1.0). CoRR **abs/1610.02455** (2016)

26 Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. ArXiv e-prints (Mar 2016)

27 Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013)

28 Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)

29 Smith-Miles, K., Baatar, D., Wreford, B., Lewis, R.: Towards objective measures of algorithm performance across instance space. Computers & Operations Research **45**, 12 – 24 (2014)

30 Spring, R., Shrivastava, A.: Scalable and sustainable deep learning via randomized hashing. In: KDD'17. pp. 445–454 (2017). https://doi.org/10.1145/3097983.3098035

31 Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. CoRR **abs/1708.07747** (2017)

A Additional experiments

A.1 How Well is Running Time Reflected in Distance Computations

Figures 11 and 12 present the same setup as in subsection 5.2, but this time relating recall to the number of distance computations required to achieve that recall. This cost measure

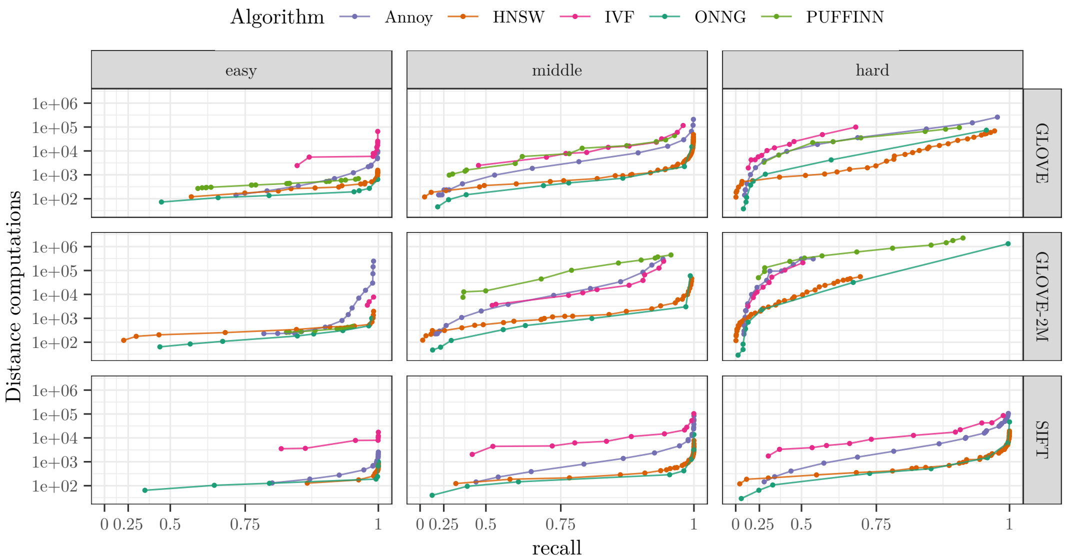


Figure 11 Recall-Distance computations tradeoff – down and to the right is better – for queries selected according to their LID. Three datasets are considered here: GLOVE, GLOVE-2M and SIFT. The scale is logarithmic on the y axis and exponential on the x axis, to take into account the scale of the data.

is more robust to implementation details and gives a more general view on how well an approach is able to efficiently index the data set.

Let us consider Figure 11. For the recall vs. distance computations trade-off, we aim for all curves to be down and to the right, which reflects high recall with a small number of distance computations. In general, the trend observed in the running time study continues for distance computations: the easy, middle, and hard query sets are progressively more difficult to answer. Graph-based approaches compute considerably fewer distances, and there is little difference in these two approaches. With regard to the other approaches, `Annoy` computes fewest distances, but turns out to be the slowest implementation on most of the data and query sets combinations.

Figure 12 shows the influence of the three different dimensionality measures for `Annoy`. First, we notice that there is remarkably little difference between the three different dimensionality measures in terms of distance computations, in particular for SIFT. For the difficult query set, we see that Expansion provides the easier-to-index queries, whereas RC provides considerably more difficult queries than the two others considering the easy queryset. LID provides the best of both worlds.

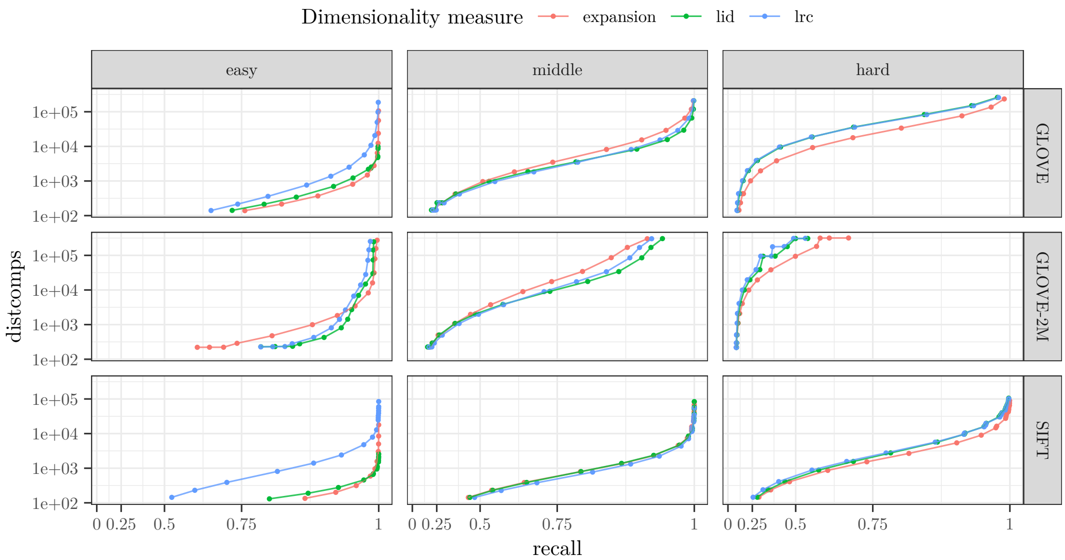


Figure 12 Recall-Distance computations tradeoff – down and to the right is better – for queries solved with Annoy. Three datasets are considered here: GLOVE, GLOVE-2M and SIFT. The scale is logarithmic on the y axis and exponential on the x axis, to take into account the scale of the data.