Evaluating and Generating Query Workloads for High Dimensional Vector Similarity Search

https://github.com/Cecca/hephaestus

Matteo Ceccarello

Alexandra Levchenko

Ioana Ileana

Themis Palpanas

Check out the paper!

The setting

Nearest neighbor search is one of the most basic operations in data analysis:

- pattern recognition
- information retrieval
- clustering
- classification
- anomaly detection
- machine learning in general….

The *k*-Nearest Neighbors (*k*-NN) problem

Given a dataset *S* in a metric space with distance function *d,* for a query point *q* find the set of *k* points most similar to it

$$\text{find } X \subseteq S \text{ s.t. } |X| = k \text{ and } d(q, X) \leq d(q, S \setminus X)$$

Brute forcing the solution is impractical, index data structures are used to reduce the number of distance computations:

- Exact indices (e.g. MESSI, DSTree…)
- Approximate indices (e.g. HNSW, IVF, LSH ….)

Indexing for *k*-NN

Low dimensional case

- kd-trees
- R-trees
- ball-trees
- M-trees
- cover-trees
- vp-trees
- ……

Due to the *curse of dimensionality* the above indices are not effective in the high dimensional case

High dimensional case

Exact approaches

- MESSI
- DSTree
- ….

Approximate approaches

- Graph based (e.g. HNSW)
- Clustering based (e.g. IVF)
- Hashing based (e.g. LSH)

Reliable and informative benchmarks are fundamental for algorithm designers and practitioners alike

- Several implementations of competing approaches
- Many parameters that influence the performance/accuracy tradeoff
- The tradeoffs depend on the dataset and queries

We address the problem of generating queries and benchmarks of user-defined hardness

Our contributions:

(a) An overview of different query hardness measures
(b) Methods to generate synthetic queries of given target hardness
(c) Experiments showing the effectiveness of our methods

Easy and hard queries

The cost can be measured by the number distance computations required to answer it with a given recall (say 0.9)

Easy

queries require few distance computations, hence are fast to answer

Hard

queries require computing many distances, hence are slower

The overall performance depends on the distribution of difficulties of queries, that should not be left to random chance

The *empirical hardness*

Of course, a given query might be easier for an index than for another one.

The *empirical hardness* is the number of distance computations normalized by the number of points in the dataset.

< 1	$= 1$	> 1
Better than brute force	Just like brute force	Worse than brute force

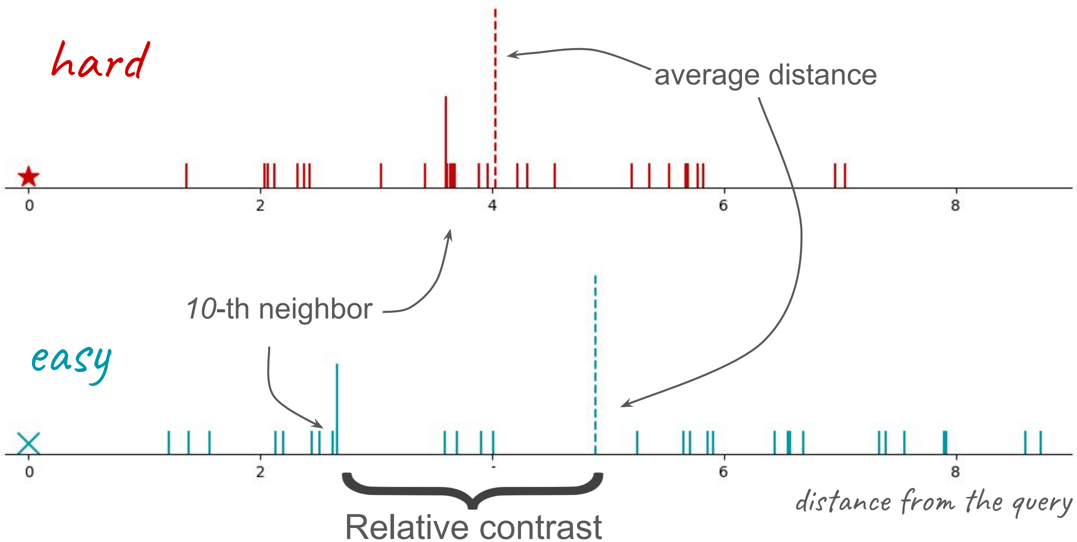
What makes a query intrinsically difficult?

- A good candidate is the position of the query relative to the dataset
- We can look at the distribution of distances, condensing it to a single number

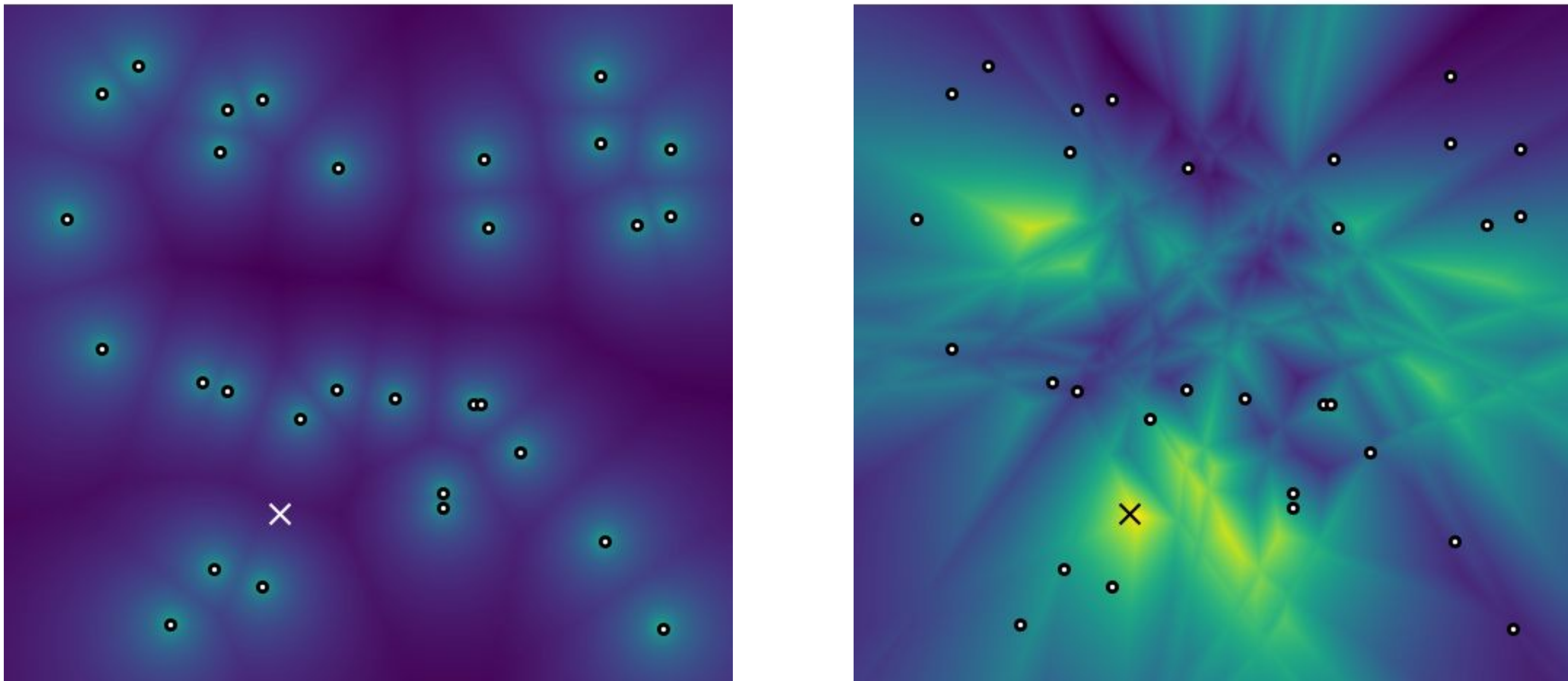
$$\mathrm{LID}_k(\vec{q}) = -\left(\frac{1}{k}\sum_{i=1}^{k}\log\frac{r_i}{r_k}\right)^{-1}$$

$$\mathrm{RC}_k(\vec{q}) = \frac{\frac{1}{n}\sum_{i=1}^{n} r_i}{r_k}$$

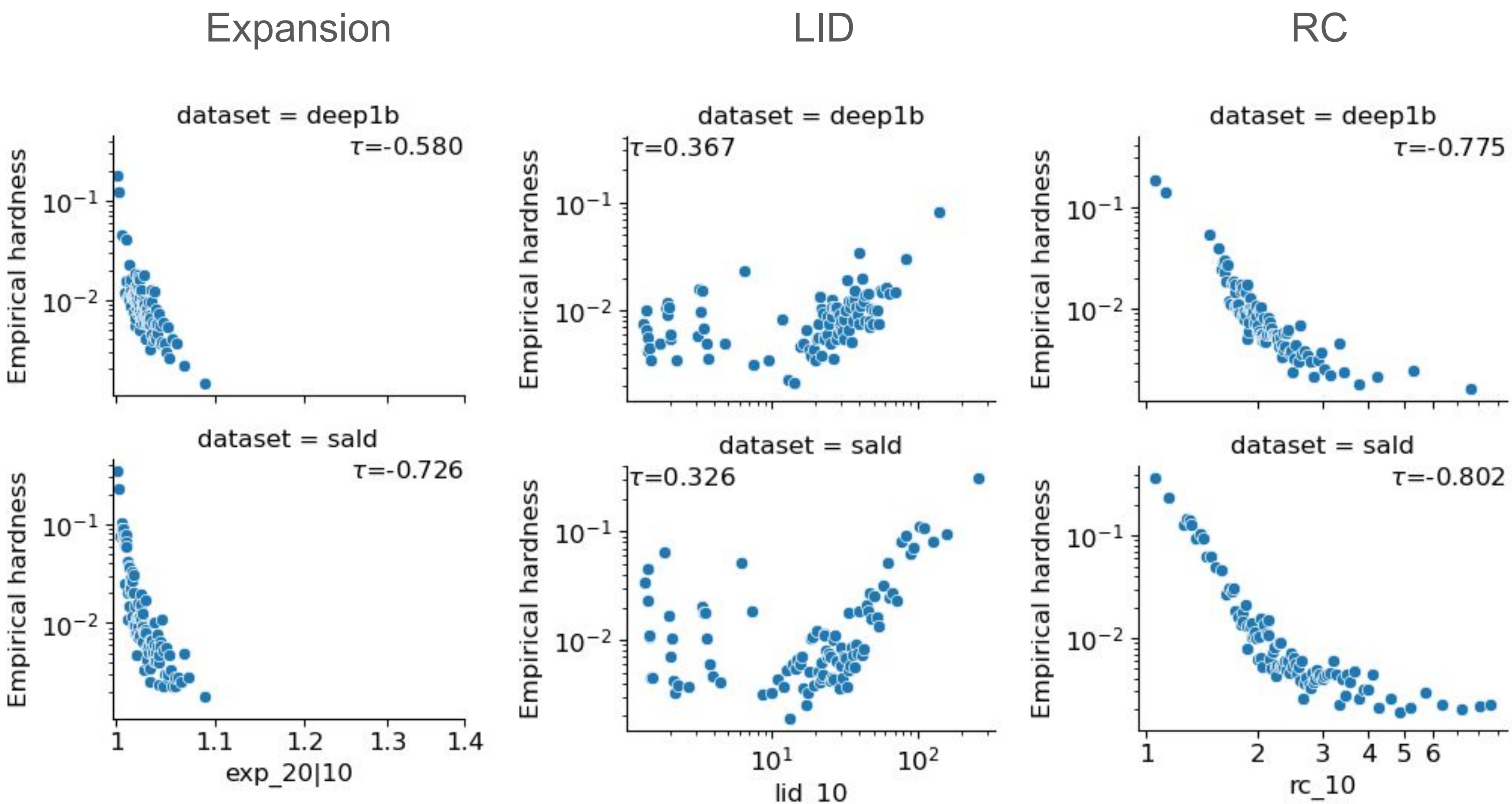
$$\mathrm{Expansion}_k(\vec{q}) = \frac{r_{2k}}{r_k}$$



The distribution of relative contrast in $\mathbb{R}^2$ relative to a set of 30 points



$k = 1$ $\quad$ $k = 10$

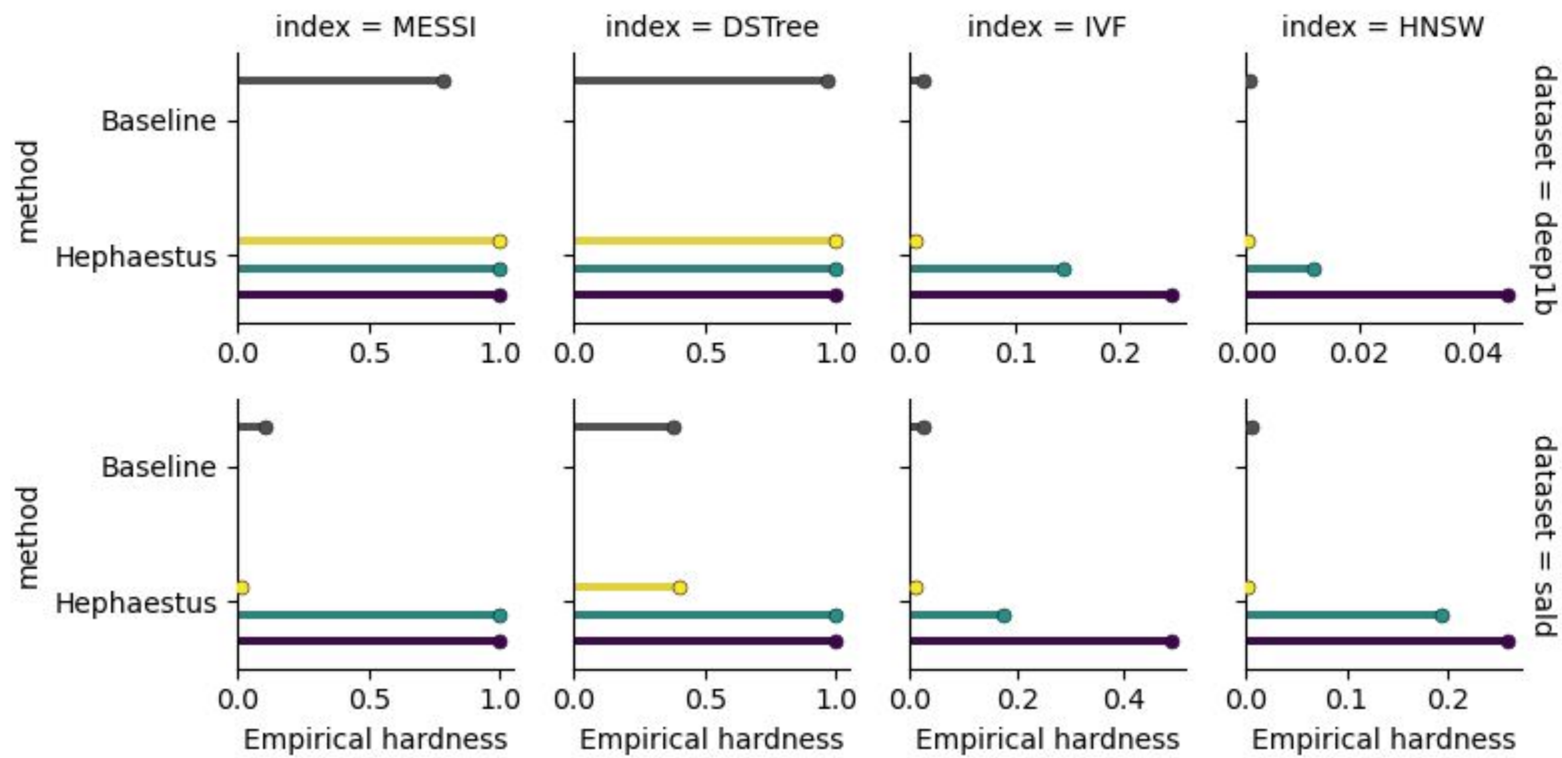

Expansion
LID
RC
dataset = deep1b
dataset = sald
τ=-0.580
τ=-0.726
τ=0.367
τ=0.326
τ=-0.775
τ=-0.802
Empirical hardness
exp_20|10
lid_10
rc_10

Can we synthesize queries of a desired difficulty?

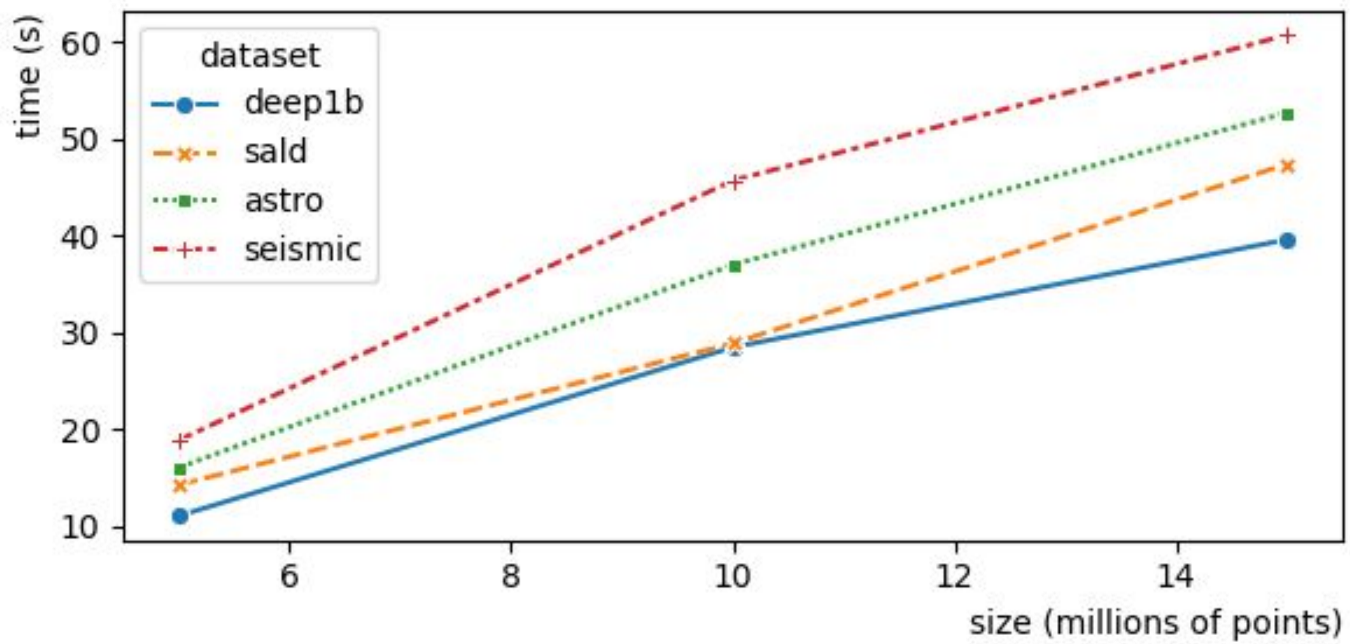
For benchmarking purposes it is of interest to be able to create query workloads with a controlled distribution of difficulties.

Yes! With our method Hephaestus!

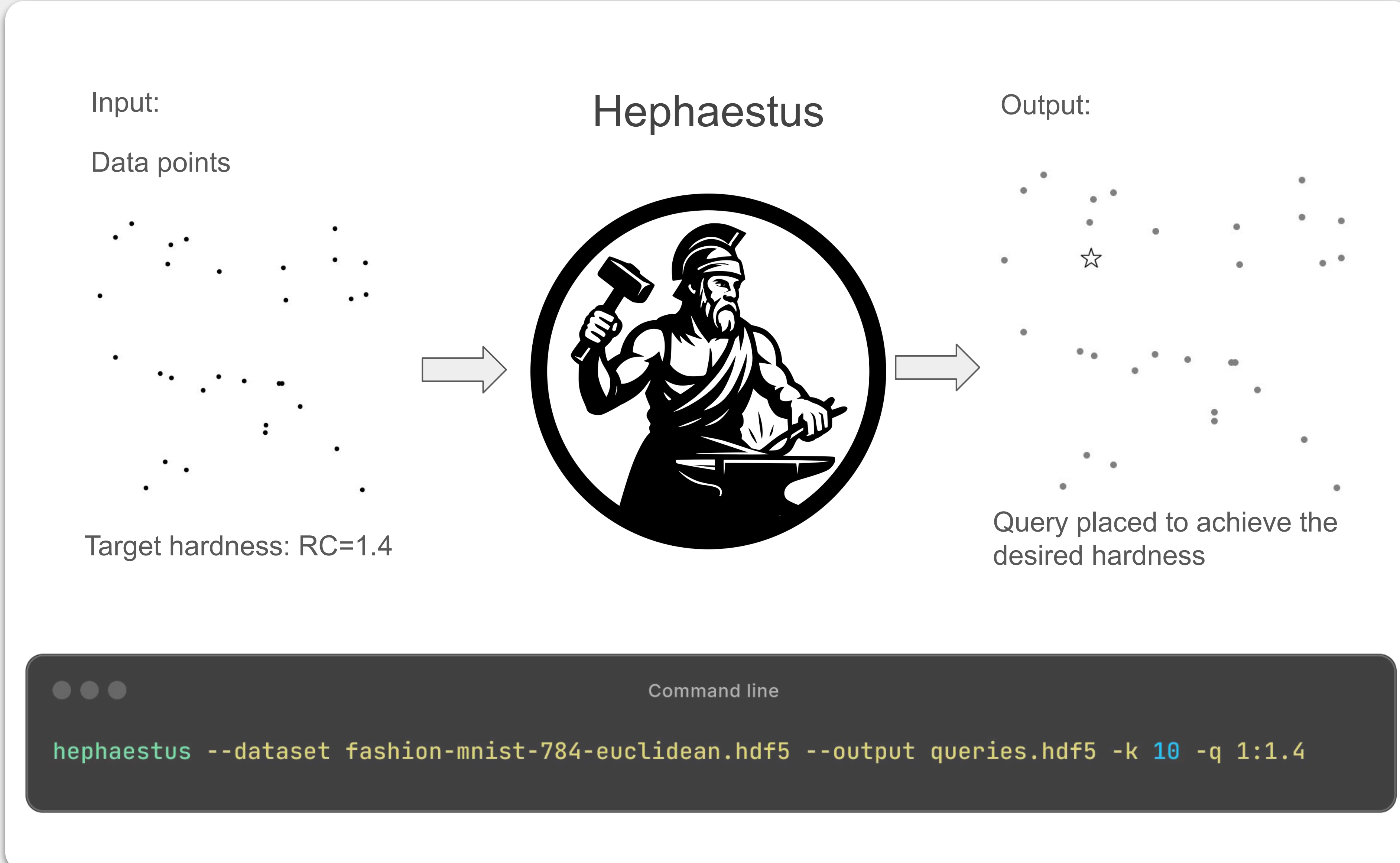
Effectiveness



Empirical hardness of easy, medium and hard queries



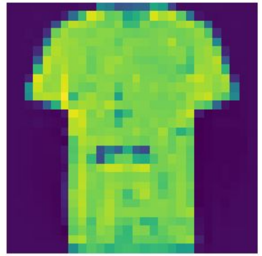
Time to generate synthetic workloads at different dataset scales


Input:
Data points
Target hardness: RC=1.4
Hephaestus
Output:
Query placed to achieve the desired hardness
Command line
hephaestus --dataset fashion-mnist-784-euclidean.hdf5 --output queries.hdf5 -k 10 -q 1:1.4

Easy query

Relative contrast 4

IVF index inspects 4.6% of the dataset to answer it

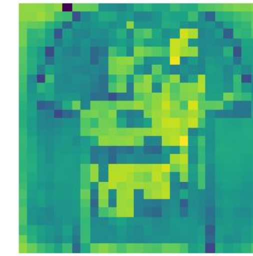


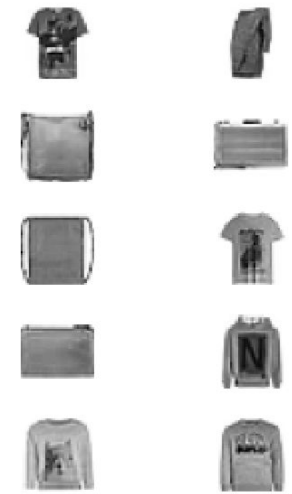
All answers are shirt items, very similar to the query

Hard query

Relative contrast 1.09

IVF index inspects 45.1% of the dataset to answer it!





Some answers are shirt items, while others are bags

Conclusions

- We give an overview of different measures to assess the hardness of nearest neighbor queries
- We assess the correlation between these measures and the effort invested by different data structures
- We propose Hephaestus, a scalable method to generate synthetic query workloads with the desired hardness level
- We provide an open source Python implementation of Hephaestus

Evaluating and Generating Query Workloads for High Dimensional Vector Similarity Search

KDD 2025

https://github.com/Cecca/hephaestus

Matteo Ceccarello

Alexandra Levchenko

Ioana Ileana

Themis Palpanas

Check out the paper!