Towards overcoming the re-ranking bottleneck

Piotr Indyk (MIT)

Piyush Anand, Piotr Indyk, Ravishankar Krishnaswamy, Sepideh Mahabadi, Vikas C Raykar, Kirankumar Shiragur, Haike Xu, Graph-Based Algorithms for Diverse Similarity Search, ICML 2025.

Haike Xu, Sandeep Silwal, Piotr Indyk, A Bi-metric Framework for Fast Similarity Search, The 1st Workshop on Vector Databases, ICML 2025.

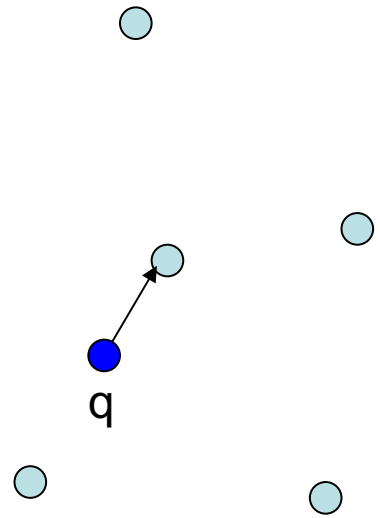
Piotr Indyk, Haike Xu, Worst-case performance of popular approximate nearest neighbor search implementations: Guarantees and limitations, NeurIPS 2023.

Plan

- Nearest neighbor search
- Filtering/re-ranking
- Improving over re-ranking
  - Digression: graph-based algorithms
  - Case 1: Searching with cheap proxy metrics
  - Case 2: Diverse nearest neighbor

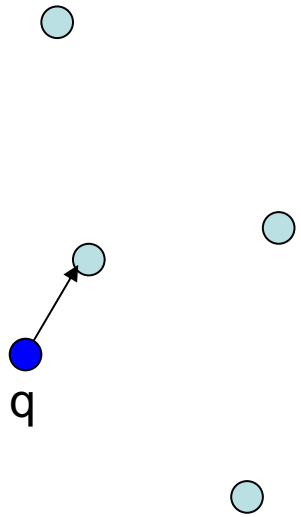
What: Nearest Neighbor Search

- **Given:** a set P of n points in a d-dimensional space R^d
- **Goal:** build a data structure which, given any query $q \in R^d$ returns a point $p \in P$ minimizing the distance $D(p,q)$
- Want:
  - Fast running time
  - Small data structure size



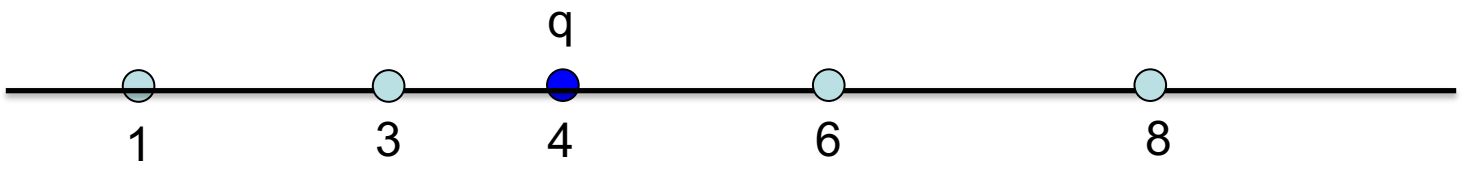
The nearest neighbour search problem arises in numerous fields of application, including:

- Pattern recognition – in particular for optical character recognition
- Statistical classification – see k-nearest neighbor algorithm
- Computer vision
- Computational geometry – see Closest pair of points problem
- Databases – e.g. content-based image retrieval
- Coding theory – see maximum likelihood decoding
- Data compression – see MPEG-2 standard
- Robotic sensing[2]
- Recommendation systems, e.g. see Collaborative filtering
- Internet marketing – see contextual advertising and behavioral targeting
- DNA sequencing
- Spell checking – suggesting correct spelling
- Plagiarism detection
- Similarity scores for predicting career paths of professional athletes.
- Cluster analysis – assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense, usually based on Euclidean distance
- Chemical similarity
- Sampling-based motion planning



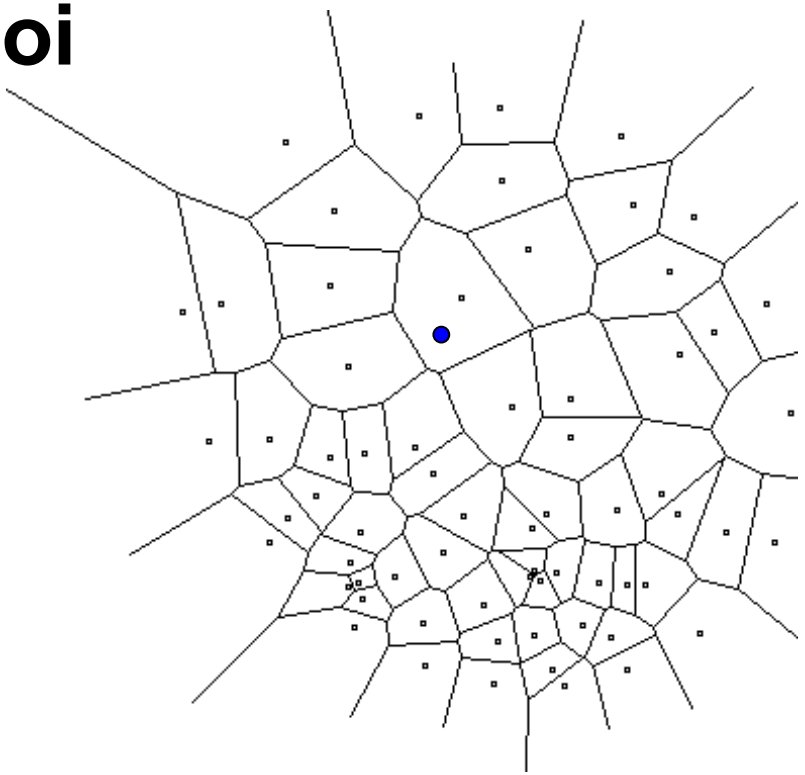
How: Nearest neighbor for d=1

- Pointset P: $x_1 < x_2 \ldots < x_n$, $x_i \in R$
- Query: $q \in R$
- Sufficient to find the interval that q belongs to
- Algorithm:
  - During the preprocessing, sort and store the input: O(n) space
  - To answer query, perform binary search: O(log n) time



Nearest neighbor for d=2

- Space partitioning: **Voronoi diagram**
- Given q, find the cell q belongs to
- Performance:
  - Query time: O(log n)
  - Space: O(n)

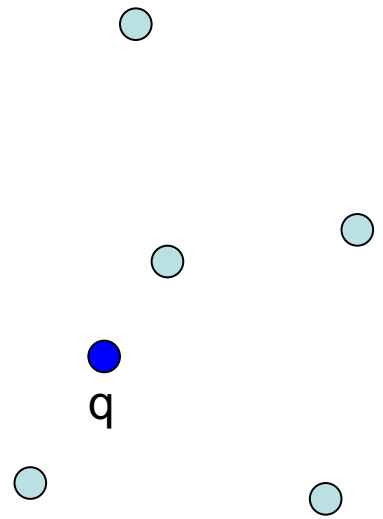


The case of d>2

- Problem: Voronoi diagram has size

$$n^{\lceil d/2 \rceil}$$

  - This is a lot of space!
    (again, think $n \gg 10^6$, $d>50$)

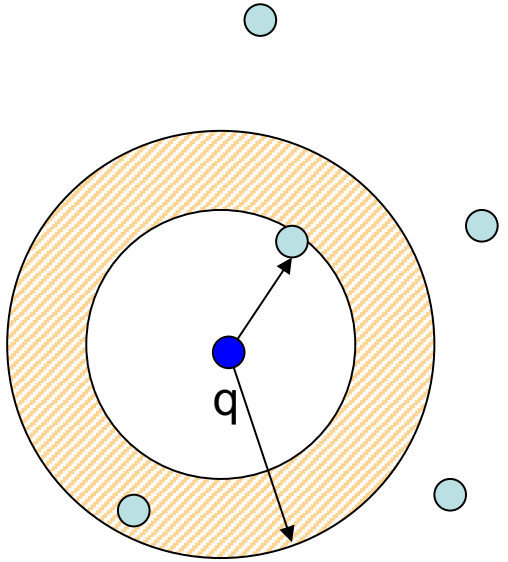


- We can also perform a linear scan: O(dn) space, O(dn) time
- These are pretty much the only known general solutions
- “The curse of dimensionality”

Relaxation 1: Theory

- **Given:** a set P of n points in some space X under some metric d, parameter $\varepsilon>0$
- **Goal:** data structure which, given any query q returns $p' \in P$, where

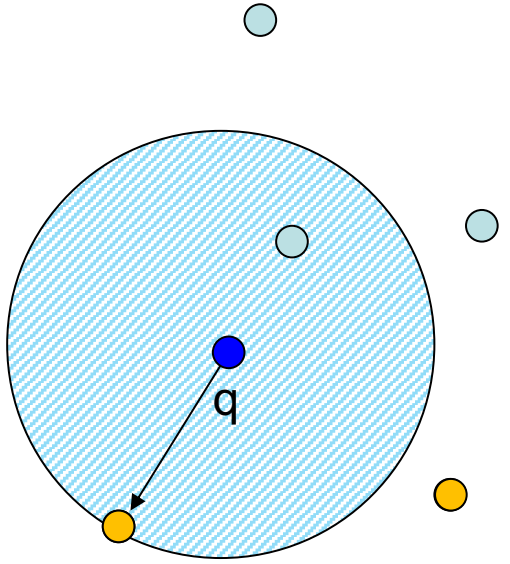
$$d(p',q) \le (1+\varepsilon) \min_{p \in P} d(p,q)$$



"(1+ε)-approximate nearest neighbor"

Relaxation 2: Practice

- **Given:** a set P of n points in some space X under some metric d, parameter k
- **Goal:** data structure which returns as many top k nearest neighbors as possible
  - Recall@k: the fraction of top k nearest neighbors returns
- These two relaxations are correlated, but distinct
- We will use either, depending on the context



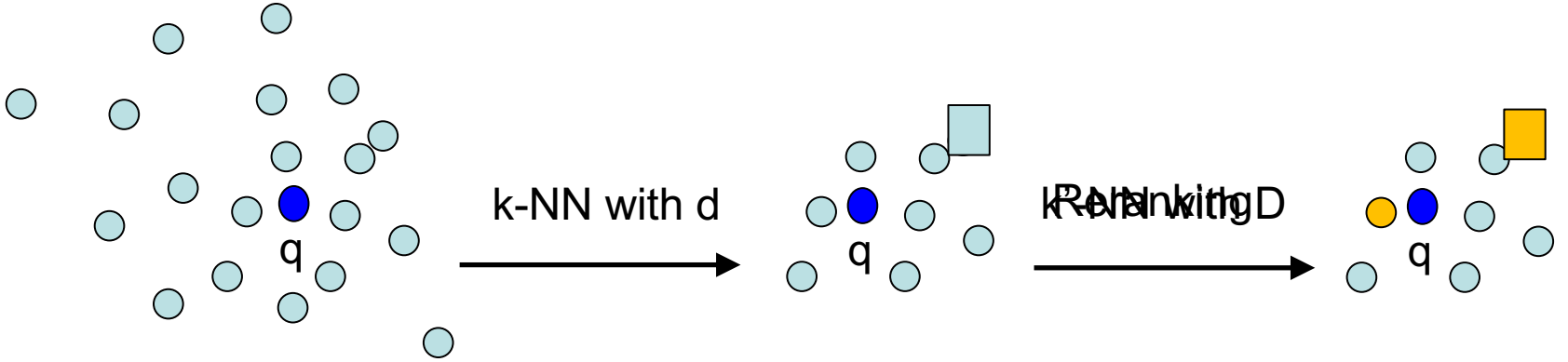
k=2

Recall@k=0.50

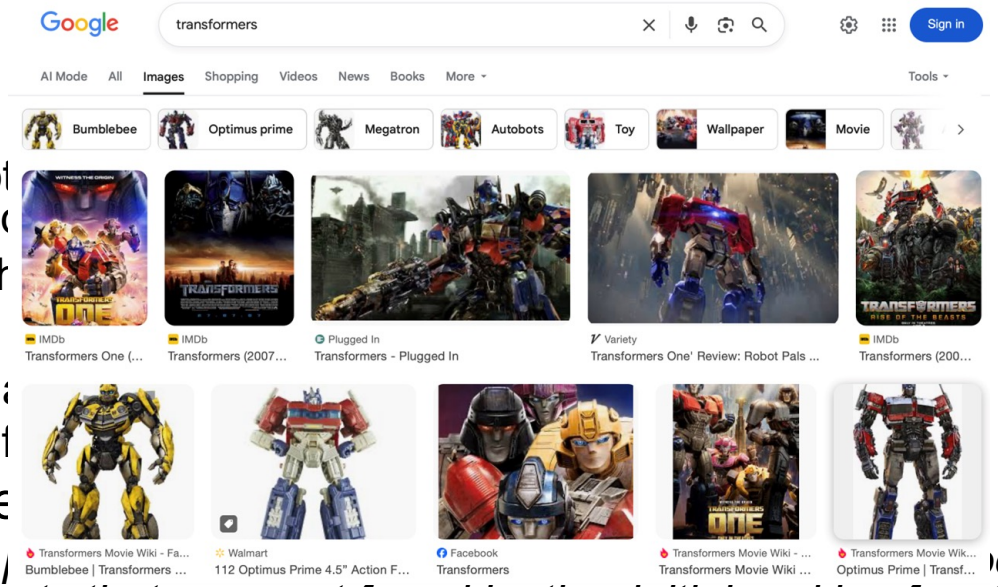
Plan

- Nearest neighbor search: defs
- **Filtering/re-ranking**
- Improving over re-ranking
  - Digression: graph-based algorithm
  - Case 1: Searching with cheap proxy metrics
  - Case 2: Diverse nearest neighbor

Using nearest neighbor search: filtering/re-ranking



- Examples:
  - Diversity constraints
  - Refining results using anot
    induced by the Euclidean
  - Thousands of papers on th
- Benefits:
  - Query time much faster tha
  - Preprocessing also much f
- Drawback: Accuracy limite

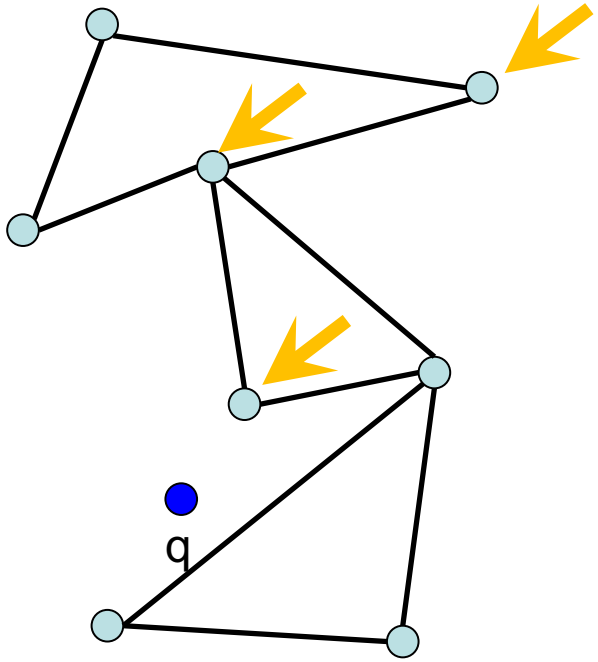


"The performance of a re-ranking ol, however. This is because documents that were not found by the initial ranking function have no chance of being re-ranked." (MacAvaney, Tonellotto, Macdonald. 2022)

Plan

- Nearest neighbor search: defs
- Filtering/re-ranking
- **Improving over re-ranking**
  - Case 1: Searching with cheap proxy metrics
  - Case 2: Diverse nearest neighbor

Graph-based algorithms



Since 2017: HNSW, NGT, NSG, DiskANN, SSG, Kgraph, DPG, NSW, SPTAG-KDT…

Plan

- Nearest neighbor search: defs
- Filtering/re-ranking
- Improving over re-ranking
  - Digression: graph-based algorithm
  - **Case 1: Searching with cheap proxy metrics**
  - Case 2: Diverse nearest neighbor

Improving over reranking: theory

Informal Theorem [Xu-Silwal-Indyk'25]: given an accurate approximate graph-based algorithm, if we

- build a data structure **using a proxy metric d**, but then
- switch and answer queries **using the ground truth metric D,** and run it slightly longer,

then we get an accurate approximate answers with respect to the ground truth **D.**

(even if **d** and **D** are not that close).

This gives us a template for constructing data structures that use cheap proxy metrics **d** to speed-up search using expensive ground truth metrics **D**.

Experiments on text

- d: bge-micro-v2 (open source):
  - Embeds text into Euclidean vectors
  - 0.00043 secs per embedding
- D: SFR-Embedding-Mistral (open source):
  - Also embeds text into Euclidean vectors
  - 0.13 seconds per embedding
- D: Gemini-2.0-Flash (proprietary):
  - Compares d(q,p) vs. d(q,p')
    (could use it to compute d(q,p), but does not work that well)
  - 0.01 cents per comparison

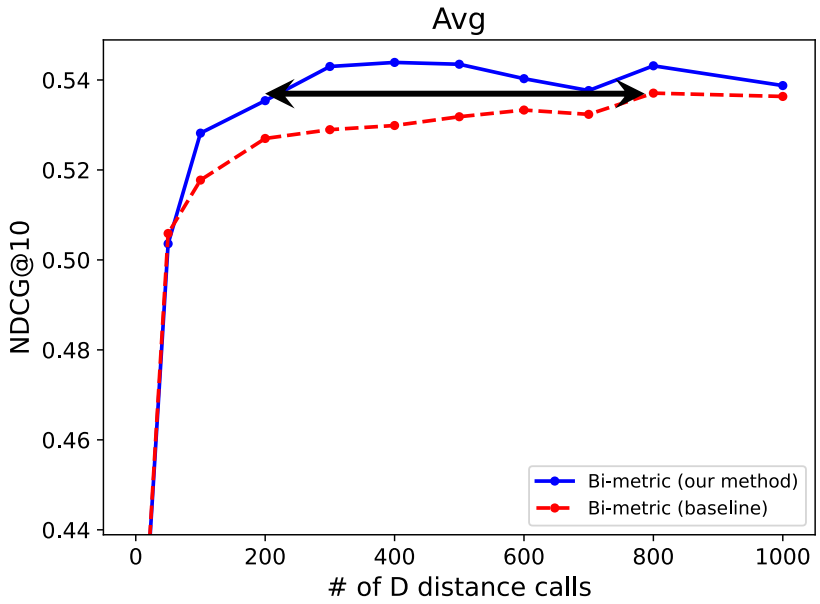
Text retrieval setup

- MTEB benchmark data sets, models from Hugging Face leaderboard (below)

Model	Model Size (Million Parameters)	Memory Usage (GB, fp32)	Average	ArguAna	ClimateFEVER	CQADupstackRetrieval	DBPedia	FEVER	FiQA2018	HotpotQA
Linq-Embed-Mistral	7111	26.49	60.19	69.65	39.11	47.27	51.32	92.42	61.2	76.24
NV-Embed-v1	7851	29.25	59.36	68.2	34.72	50.51	48.29	87.77	63.1	79.92
SFR-Embedding-Mistral	7111	26.49	59	67.17	36.41	46.49	49.06	89.35	60.4	77.02
voyage-large-2-instruct			58.28	64.06	32.65	46.6	46.03	91.47	59.76	70.86
gte-large-en-v1.5	434	1.62	57.91	72.11	48.36	42.16	46.3	93.81	63.23	68.18
GritLM-7B	7242	26.98	57.41	63.24	30.91	49.42	46.6	82.74	59.95	79.4
e5-mistral-7b-instruct	7111	26.49	56.89	61.88	38.35	42.97	48.89	87.84	56.59	75.72
LLM2Vec-Meta-Llama-3-supervis	7505	27.96	56.63	62.78	34.27	48.25	48.34	90.2	55.33	71.76
voyage-lite-02-instruct	1220	4.54	56.6	70.28	31.95	46.2	39.79	91.35	52.51	75.51
gte-Qwen1.5-7B-instruct	7099	26.45	56.24	62.65	44	40.64	48.04	93.35	55.31	72.25
LLM2Vec-Mistral-supervised	7111	26.49	55.99	57.48	35.19	48.84	49.58	89.4	53.11	74.07
...										
bge-micro-v2	17	0.06	42.56	55.31	25.35	35.07	32.25	74.99	25.59	53.91

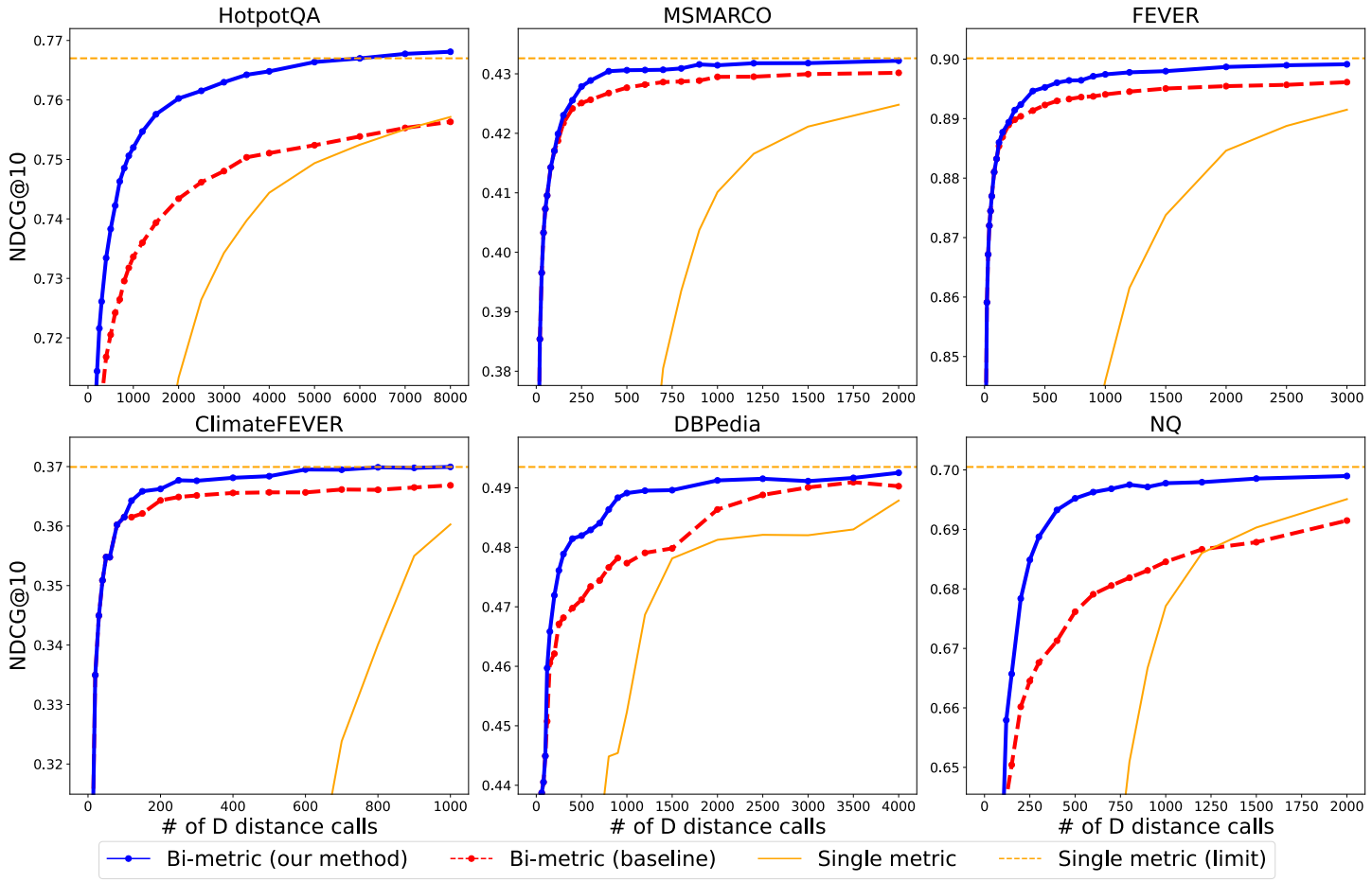
Results 1

- Setup:
  - Algorithm: modified DiskANN
  - d=bge-micro, D=Gemini
  - Data: large (>1 MB) MTEB benchmark data sets
    - Use only 500 queries, so that we can afford it
- Results (averaged over all data sets)



Results 2

d=bg-micro, D=SFR-Embedding-Mistral



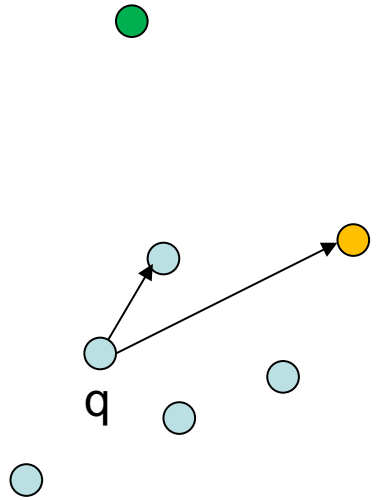
Plan

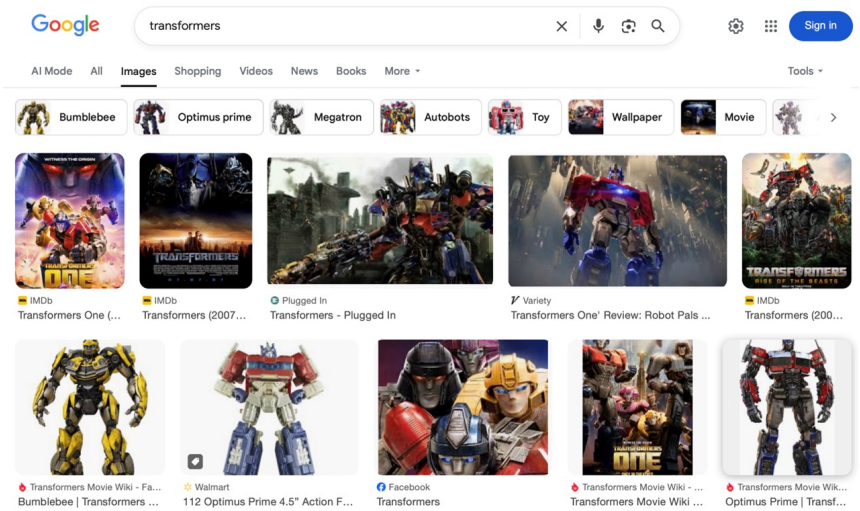
- Nearest neighbor search: defs
- Filtering/re-ranking
- Improving over re-ranking
  - Digression: graph-based algorithm
  - Case 1: Searching with cheap proxy metrics
  - **Case 2: Diverse nearest neighbor**

    (with another digression to graph-based algos)

Diverse Nearest Neighbor Search

- **Given:** a set P of n points in a d-dimensional space R^d , with colors and a parameter k
- **Goal:** build a data structure which, given any query $q \in R^d$ returns k points of different color minimizing the distance to q





Our results

Authors	Comment	Space	Query Time
Anand, Indyk, Krishnaswamy, Mahabadi, Raykar, Shiragur, Xu'25	**diverse**	**$k\ 2^{O(\dim)}\ n \log \Delta$**	**$k^2\ 2^{O(\dim)} \log^2 \Delta$**
Indyk, Xu'23: DiskANN (non-diverse)	non-diverse	$2^{O(\dim)}\ n \log \Delta$	$2^{O(\dim)} \log^2 \Delta$

We were able to modify existing non-diverse DiskANN graph-based algorithm to solve the diverse problem.

- Space: multiplied by k
- Time: multiplied by k^2

Diverse DiskANN

Building the Graph:

- Pruning
  - If $d(v,w) \leq \frac{1}{\alpha} \cdot d(u,w)$
    - Prune the edge (u,w) only if either
      1. $col[v] = col[w]$
      2. Or we have connected u to at least k different colors in the ball of radius $\frac{1}{\alpha} \cdot d(u,w)$ around w

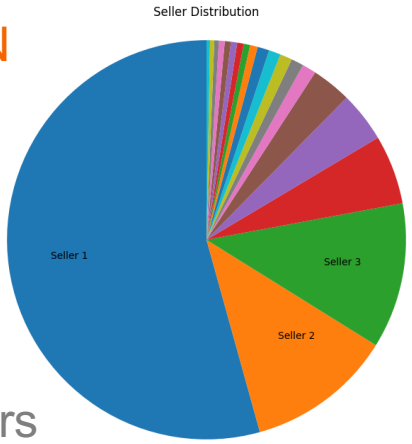
Diverse DiskANN

- **Query answering algorithm:**
  - Start from k points that all have different colors $a_1, \ldots, a_k$
  - In each iteration, **swap one point** a_i with a neighbor point a' that
    - is closer to the query
    - has a different color from the rest of the points

Experiments

Algorithms:

- Standard DiskANN Build & Search + Postprocessing to ensure diversity
- Our algo 1: Standard DiskANN Build + Diverse DiskANN Search
- Our algo 2: Diverse DiskANN Build + Diverse DiskANN Search



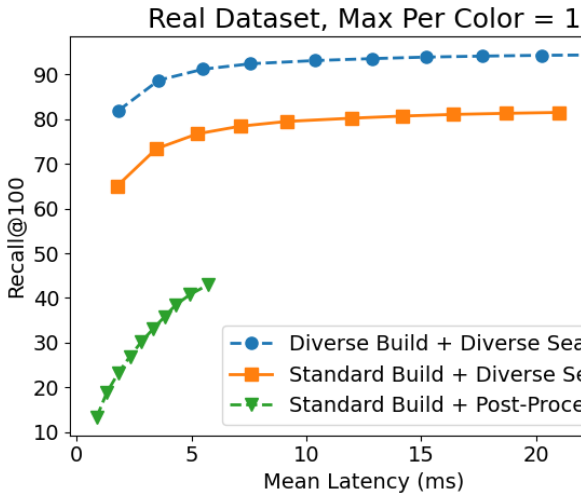
Datasets:

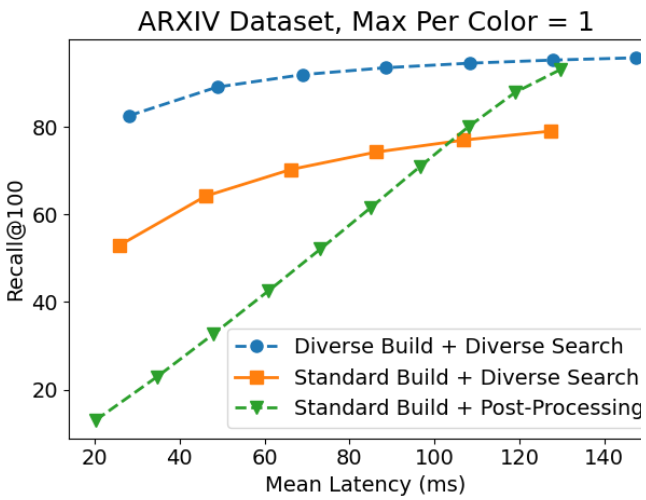
- **Ads dataset**: 20 Million vectors, 5000 queries, 64 dimensions
- **Semi-Synthetic Arxiv**: 2 Million, 1536 dimensions, 1000 colors (uniform on {1,2,3} w.p. 0.9 and uniform on the rest w.p. 0.1)
- **Semi-Synthetic SIFT**: 1 Million, 128 dimensions, 1000 colors (one color w.p. 0.8 and uniform on the rest w.p. 0.2)

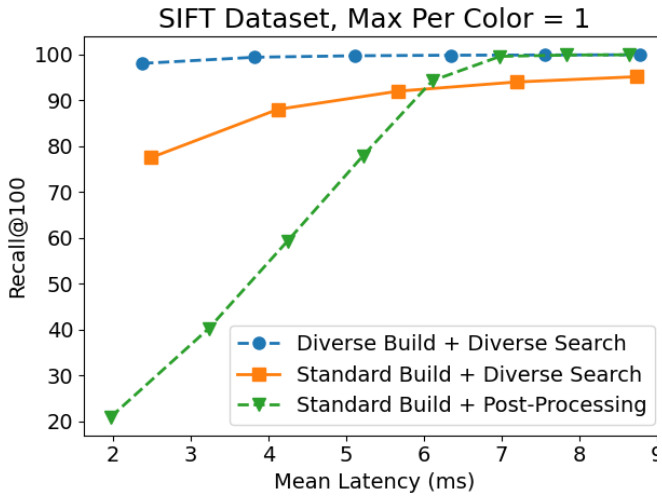
Parameters:

- $k = 100$

Recall vs Latency







Baseline: Standard DiskANN + Postprocessing to ensure diversity

Our algo 1: Standard DiskANN Build + Diverse DiskANN Search

Our algo 2: Diverse DiskANN Build + Diverse DiskANN Search

Note: the experiments are run on “standard preprocessing” DiskANN, while the analysis is for “slow preprocessing” DiskANN

Conclusions

- Can improve over re-ranking
  - Diversity
  - Proxy vs accurate metrics
- Other cases where such an improvements are possible?
- Other problems:
  - Better algorithms for diverse NN?
    - Reduce the dependence on k
  - Tri-metric setting?

THANKS!