State-of-the-art **Filtered Vector Search** Research Opportunities

Yannis Chronis

Helena Caminal

Yannis Papakonstantinou

Fatma Özcan

Anastasia Ailamaki

Acknowledgements to: Manos Chatzakis

ETH zürich

Google Cloud

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

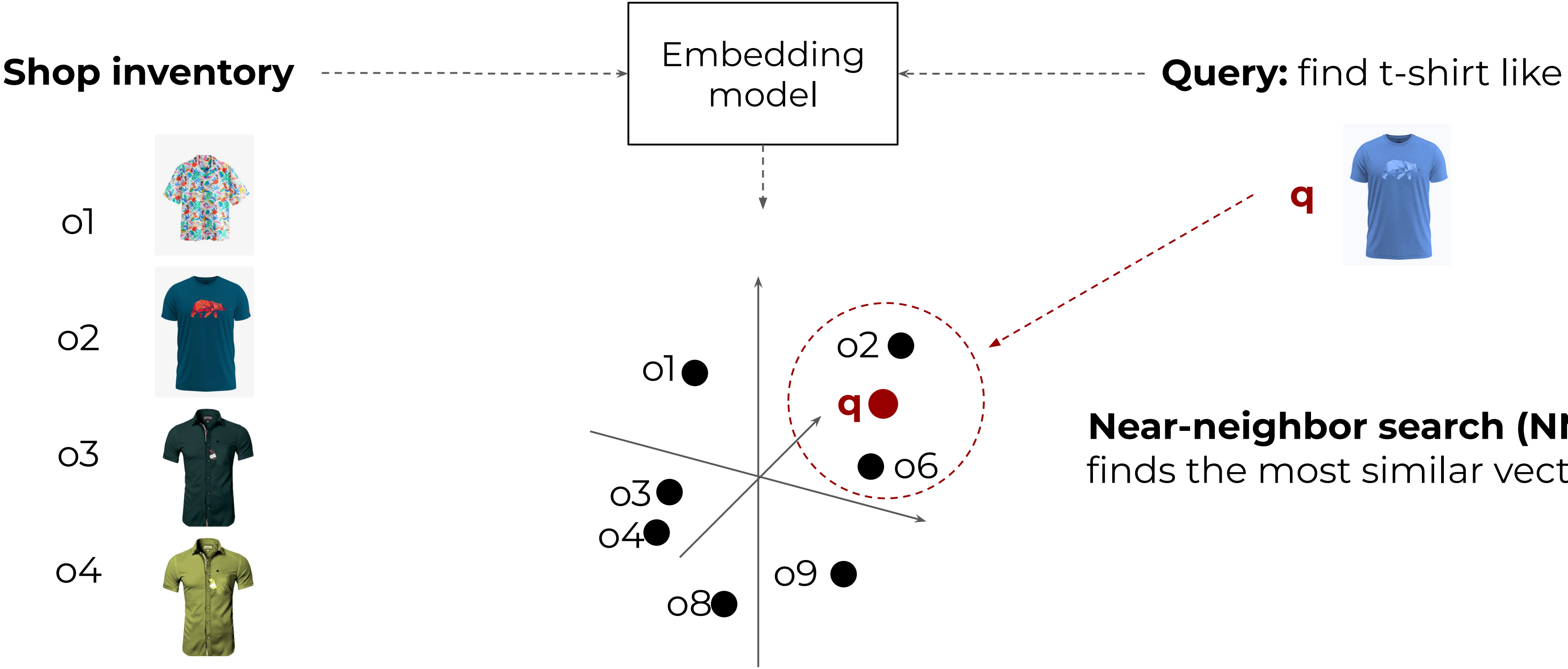
Outline

1) Background
2) Databases and Vector Search & Quality Performance Tradeoff
3) Basic Execution Methods & Challenges
4) Specialized Filtered Vector Search Indices
5) Future Research Directions

Background

Why and how do we search vectors?

Vector Search: Searching multi-modal data

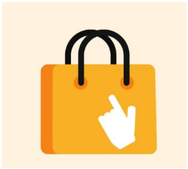


Near-neighbor search (NNS)
finds the most similar vectors

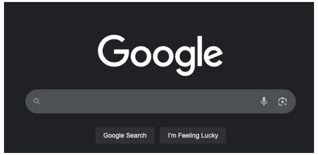
vectors
Vector Search radius

Vector search is already a core operator

Recommendation system
Find the 10 most similar products to my purchase



Semantic search
Find 5 modern and minimal apartments



Information retrieval
Google search

RAG
Find relevant data and augment an LLM prompt

Gemini

Embeddings

- Models embed objects in a multidimensional space
- Modality-specific → toward general models
- Dimension sizes: [100s - 1000s dimensions]* (e.g.: [0.2, 0.1, 0.42, 1.2, ...])
- **Distance captures similarity**

SELECT ...
FROM *tableX*
ORDER BY distance(***q***, *vector_col)*
LIMIT BY **K**

sort on the **distance from a query vector**

top-K nearest neighbors

Embeddings make data "structured"

[https://openai.com/index/clip/]

*Openai large3 models is 3072 dimensions

Near-neighbor search (NNS) is not scalable when it's accurate

Query: find top-3 vectors close to vector **q**

NNS

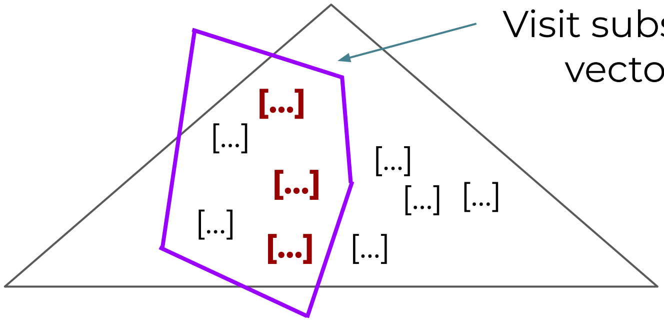
[0.2, 0.1, ...]
[0.5, 0.1, ...]
[0.3, 0.6, ...]
[0.5, 0.1, ...]
...
[0.5, 0.1, ...]

Calculate distance from **q** and **sort**

	distance
[0.2, 0.1, ...]	0.5
[0.5, 0.1, ...]	1.1
[0.3, 0.6, ...]	2
[0.5, 0.1, ...]	5
...	
[0.5, 0.1, ...]	50

Expensive: O(#vectors)

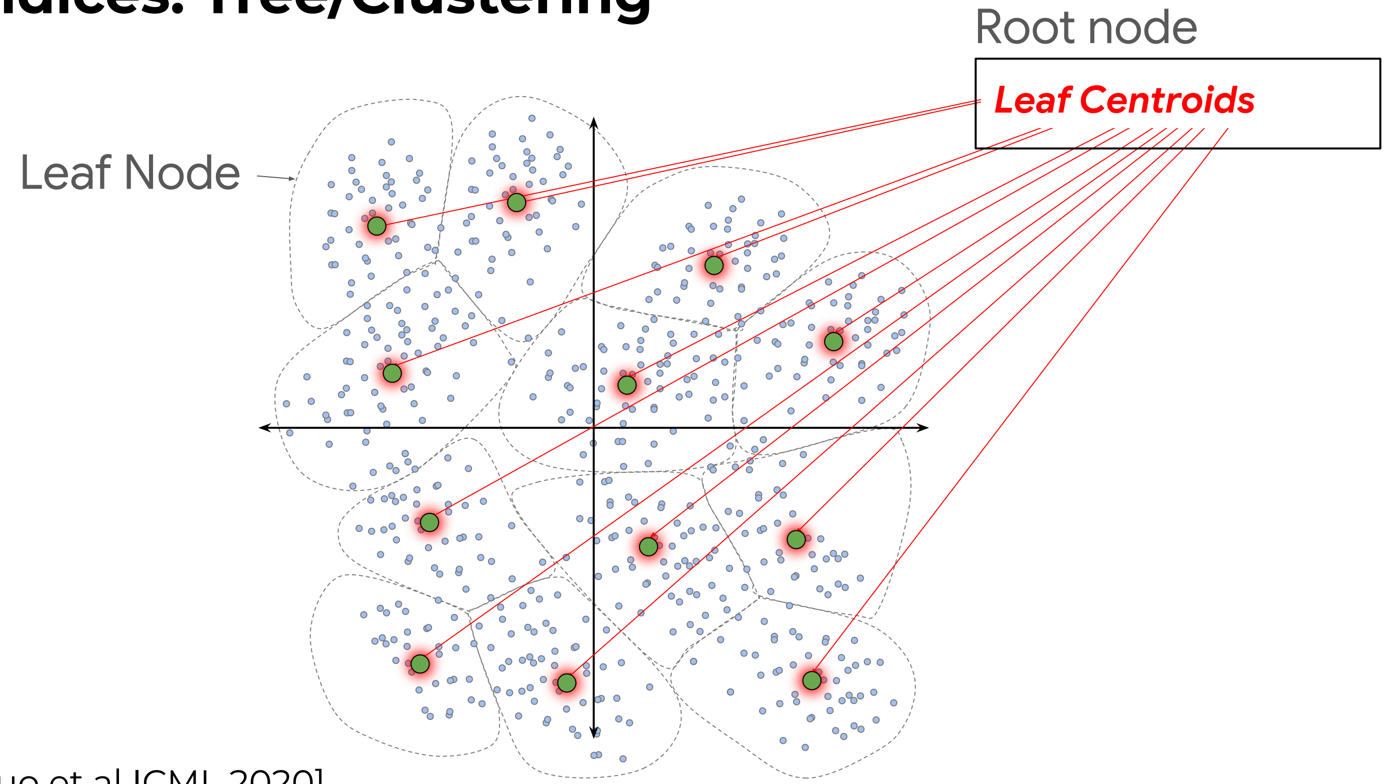
Approximate-NNS via Vector Indices



Visit subset of vectors

Trade-off accuracy for performance

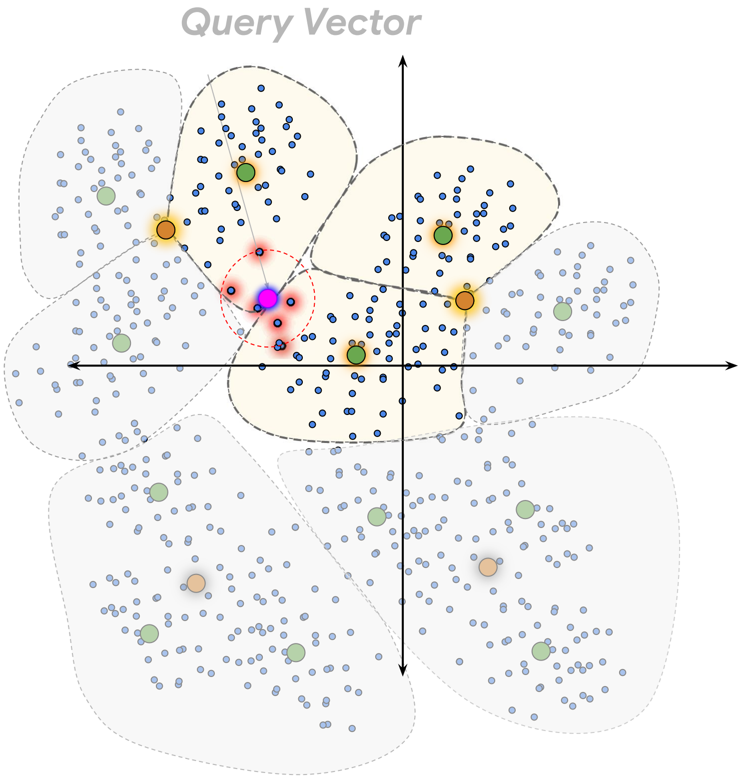
Vector Indices: Tree/Clustering



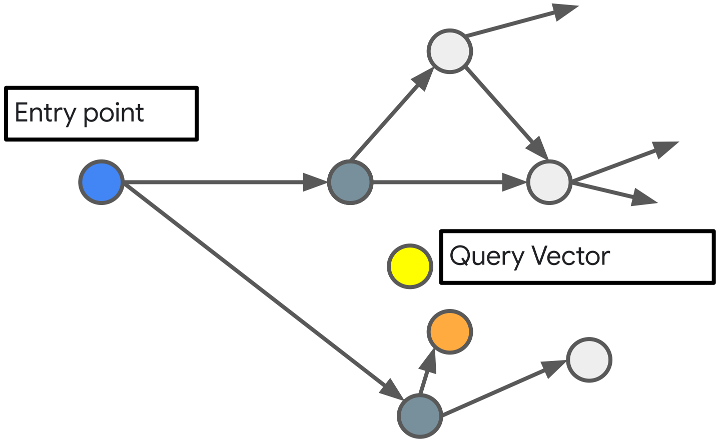
SCANN [Ruiqi Guo et al ICML 2020]
IVF [Dmitry Baranchuk et al ECCV 2018]

Vector Indices: Tree/Clustering

Given a query vector,
find the closest centroids/leaves,
compute the distances to their vectors



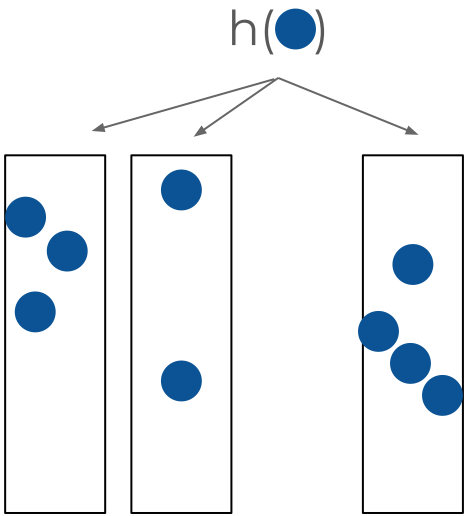
Vector Indices: Graph



num_neighbors = 2
(typically ~20 in practice)

DiskANN [Suhas Jayaram Subramanya et al NeurIPS 2025]
HSNW [Yury Malkov et al ITPM 2020]

Vector Indices: Hash



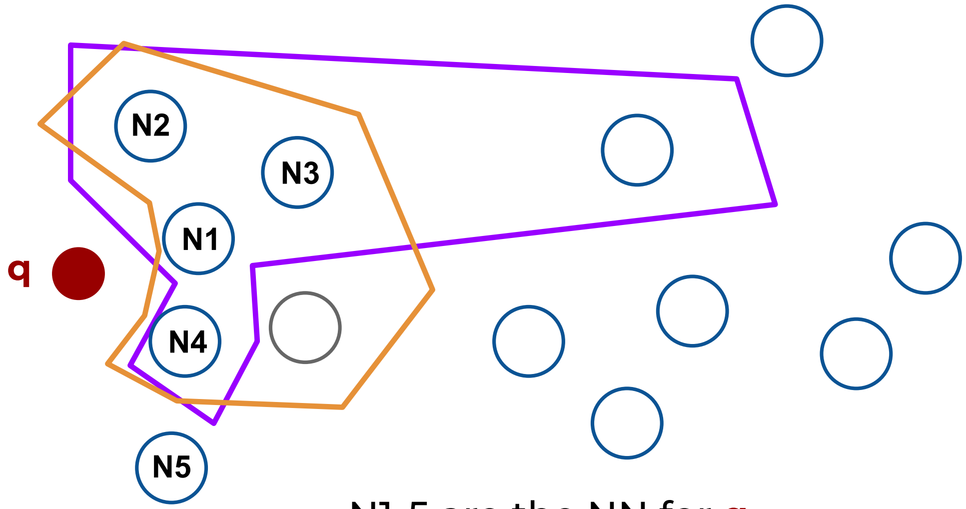
LSH based, ...

LSH [Alexandr Andoni et al NeurIPS 2015]

Measuring the quality of ANN

Typically, we use **recall@k**:

$Recall@k = |AN\_k \cap N\_k| / k$



N1-5 are the NN for **q**

Not always a good metric!!

Algorithm 1: Recall@5 = 4

Algorithm 2: Recall@5 = 4

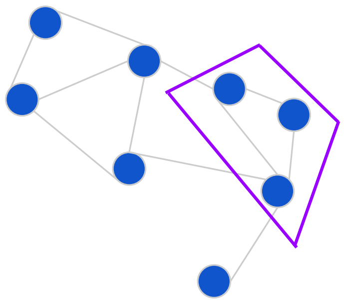
Alternative methods
- RDE@k
- TDK@k

[Marco Patella et al SISAP 2008]

Databases and Vector Search

Controlling the Quality vs Performance Trade-off

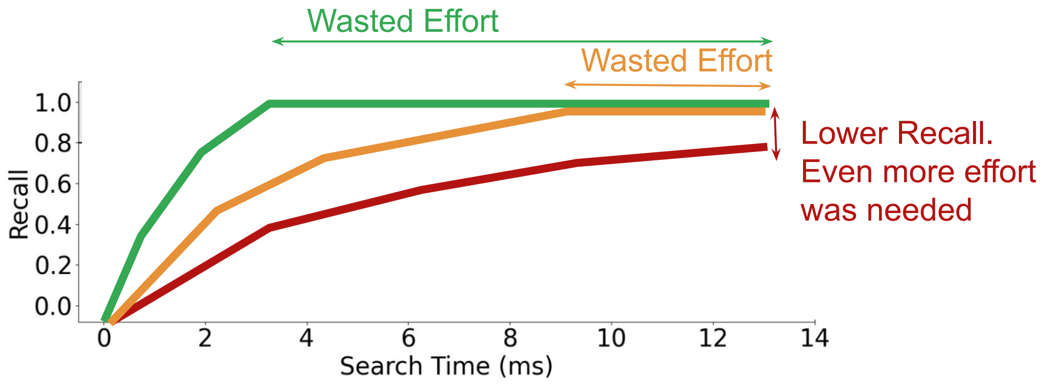
Performance vs Recall Trade-off in Approximate Nearest Neighbor Search



ANNS_search(**q**, K, *search_effort_params*)

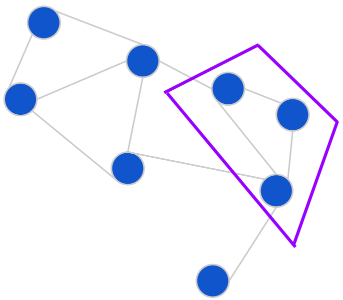
How **many vectors** to visit to achieve a user specified **target recall** ?

Challenge
Queries have different hardness, different search effort is needed



[Manos Chatzakis et al SIGMOD 2026]

Challenge: tuning the search effort parameters



ANNS_search(**q**, K, *search_effort_params*)

Hard for users and experts to tune

Uniform autotuning for all queries
Learned offline models (eg Google's CloudSQL VectorAssist)

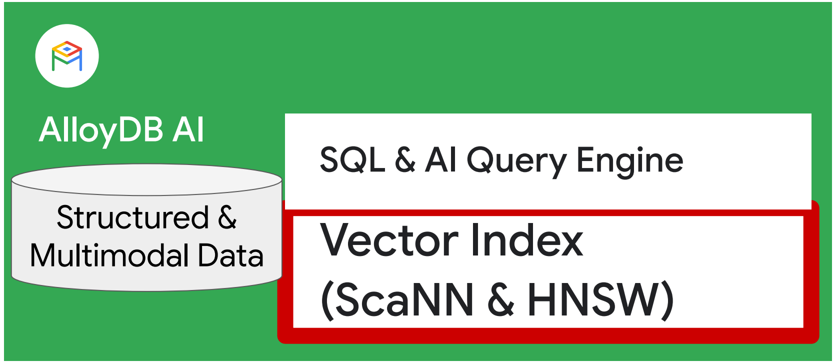
Different for each query
A model predicts the search effort parameters for each query/index

Adaptive
Decide to continue/stop (early stop) based on the current search state

[Manos Chatzakis et al SIGMOD 2026, Tiannuo Yang et al ICDE 2024, Jason Ansel et al PACT 2014]

Vector Search in SQL

Increases search quality by making use of structured + unstructured



Deep integration in SQL
=> always up-to-date results

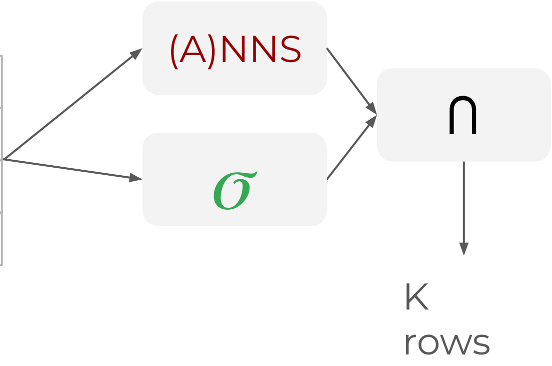
Combines & optimizes SQL + vector queries
=> ease-of-use, higher relevance and optimized performance
-> filtered vector search increasingly hot in R&D

> At Target, we used AlloyDB to improve our online search experience. We used the ability to combine our structured and unstructured data to enhance the accuracy of natural language search queries by 20%!"
>
> Visagan Subburayalu, VP of Infrastructure & Cybersecurity, Target

Filtered Vector Search (FVS): query structured and unstructured data

SELECT ...
FROM *shop_invectory*
WHERE ***col1 = True and col2 > 5***
ORDER BY distance(***q***, *vector_col)*
LIMIT BY **K**

vector_col	**col1**	**col2**
[0.2, 0.1, ...]	True	3
[0.5, 0.1, ...]	False	4
...		
[0.5, 0.1, ...]	True	5



Challenging but a “subcase”

FVS = SQL + Vectors :

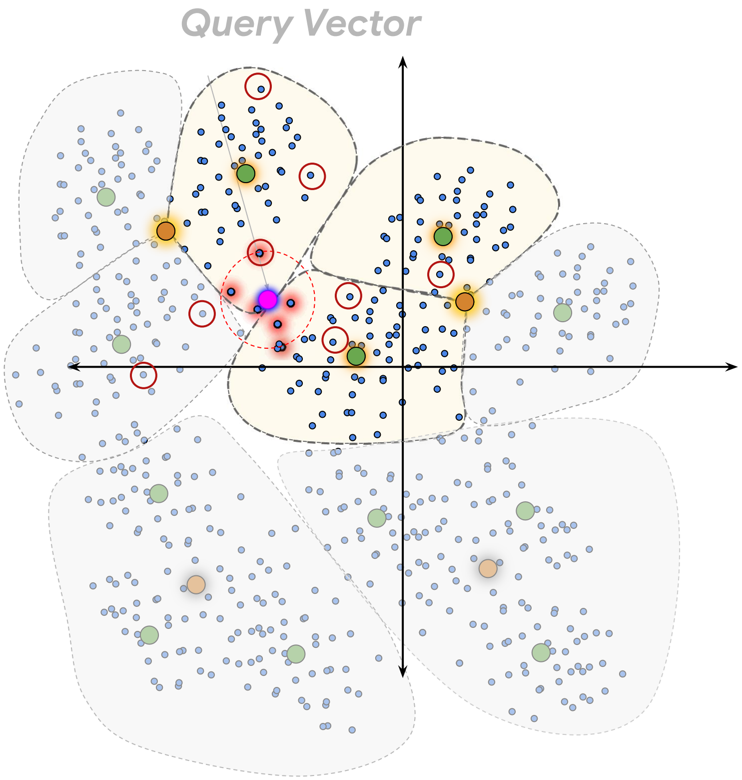
- \+ Joins
- \+ Subquery expressions
- \+ Multiple vectors search (on both sides of a join)
- \+ Dataset is not vector + tag stored in main memory (common setup for most information retrieval scenarios)

Filtered Vector Search (FVS)

For a **query vector q**,
find the closest centroids/leaves,
compute the distances to their
vectors that satisfy the conditions

For a LIMIT k query
-> there may not even be k
rows/vectors that satisfy the condition
-> there may be k but the ones
furthest away are inferior solutions

Inspect more centroids/leaves but the wasted effort Vs recall tradeoff becomes harder



Quality & Ease-of-Use North Star(s) of Filtered Vector Search

Deliver performance & quality in a user-friendly way

Out-of-the-box high recall

Should also work for filtered vector searches of many selectivities

Stable recall

Developer tunes parameters for ~ target recall of pure vector search.
System more-or-less delivers target recall for filtered vector searches.

Declarative Recall

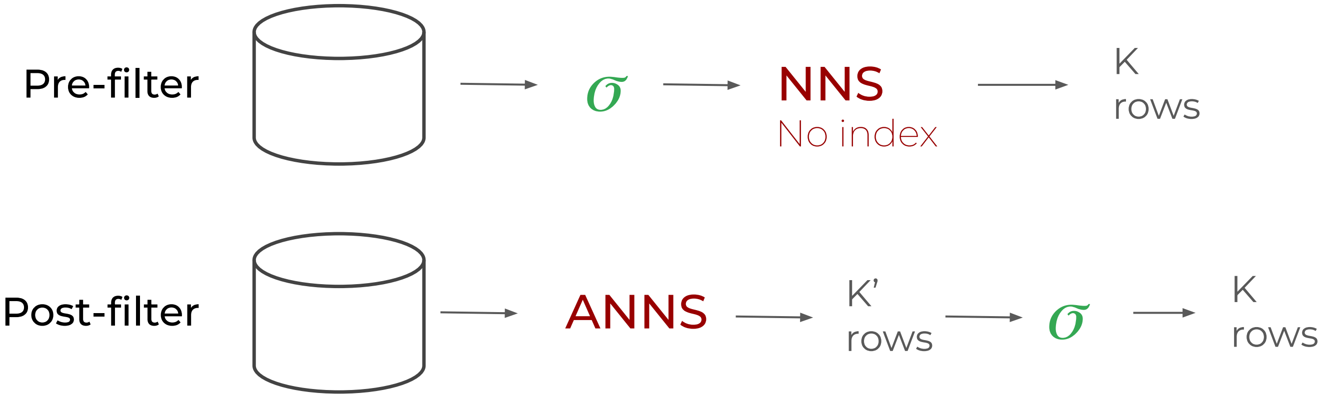
Developer declares the target recall of the query. The database configures all the parameters to achieve the dev specified target recall and works for filtered vector search also.

* with high performance

Filtered Vector Search

Basic Execution Methods + Challenges

Basic execution methods



Expensive
if filter is not selective

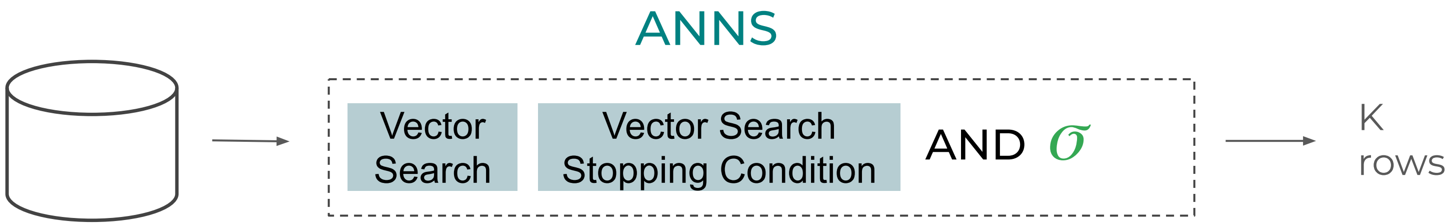
Recall & Performance Challenge

K' >> K
Wasted effort

K' > K // σ(K') < K
Low recall

σ: filter

Inline-Filter



Blur the line between ANNS and filtering to improve accuracy and performance

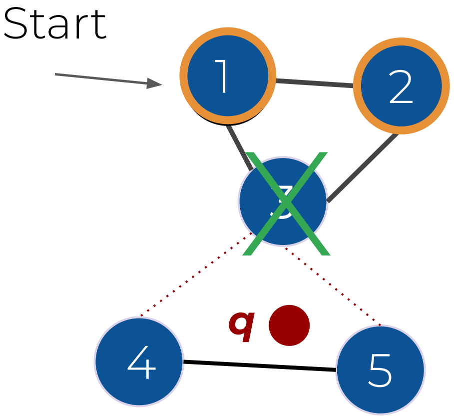
* multiple implementations of inline filtering

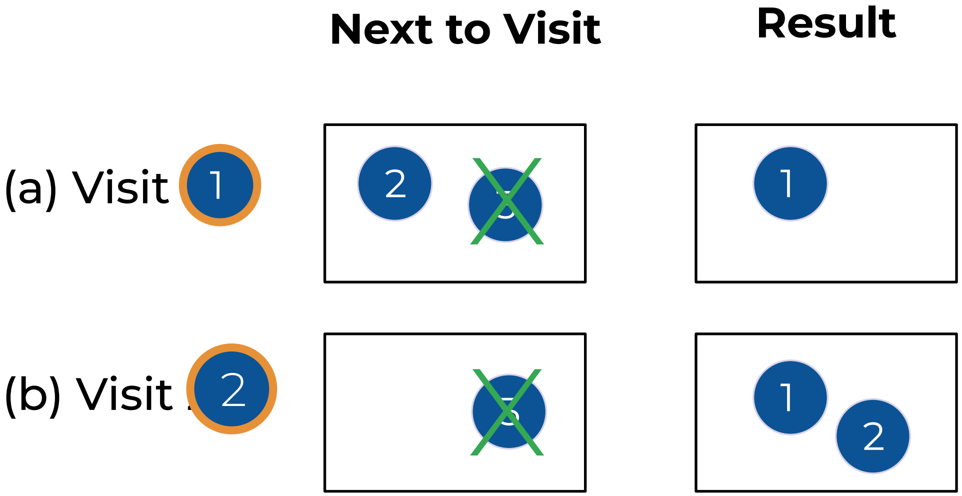
σ: filter

Predicate Subgraph Traversal

Graph Inline-Filtering #1

Filtered-out nodes
DO NOT
participate in navigation





Search Stops -> Low Recall

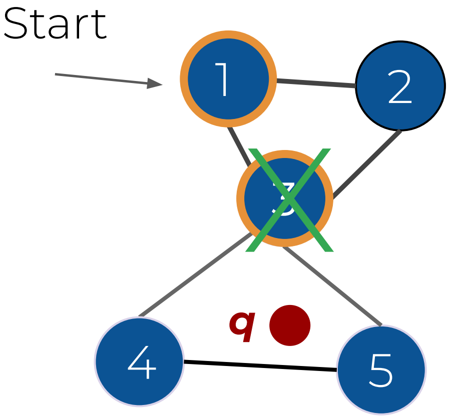
Connectivity Breaks

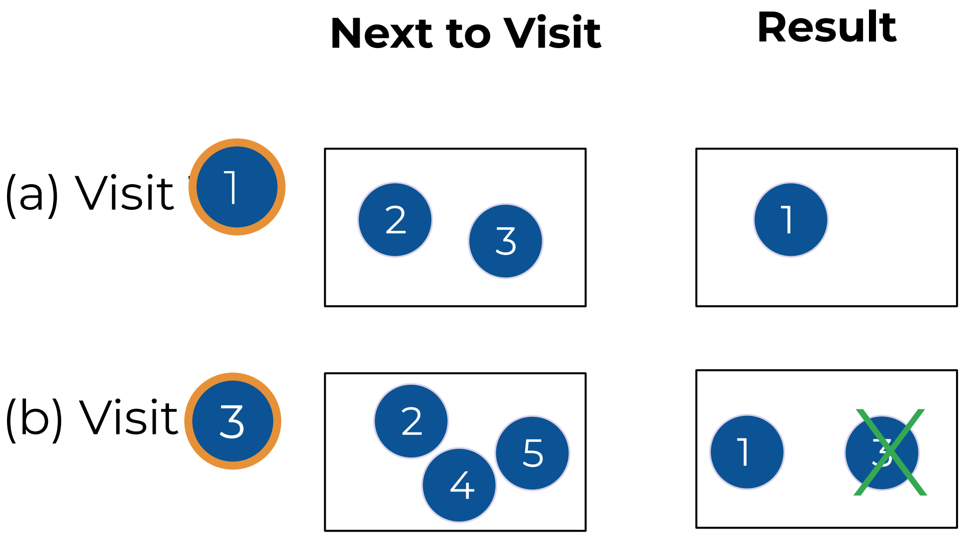
[https://weaviate.io/blog/speed-up-filtered-vector-search]

Sweeping

Graph Inline-Filtering #2

Filtered-out nodes
DO
participate in navigation

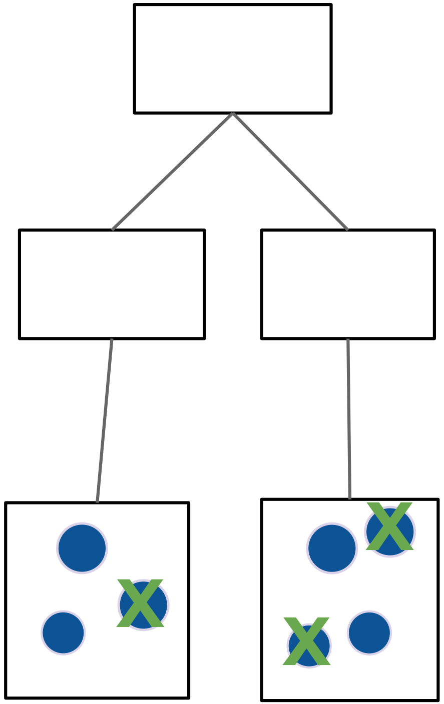




Graph remains connected **at the cost of more distance computations**

[https://weaviate.io/blog/speed-up-filtered-vector-search]

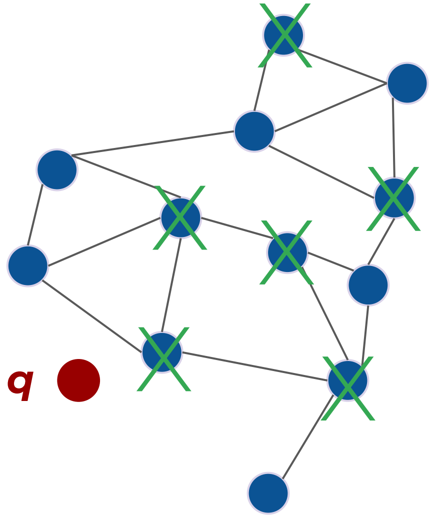
Tree/Hashing Inline-Filtering



Internal node navigation does not change

Data vectors are only in the leaves, filter here

Indices are built on unfiltered data



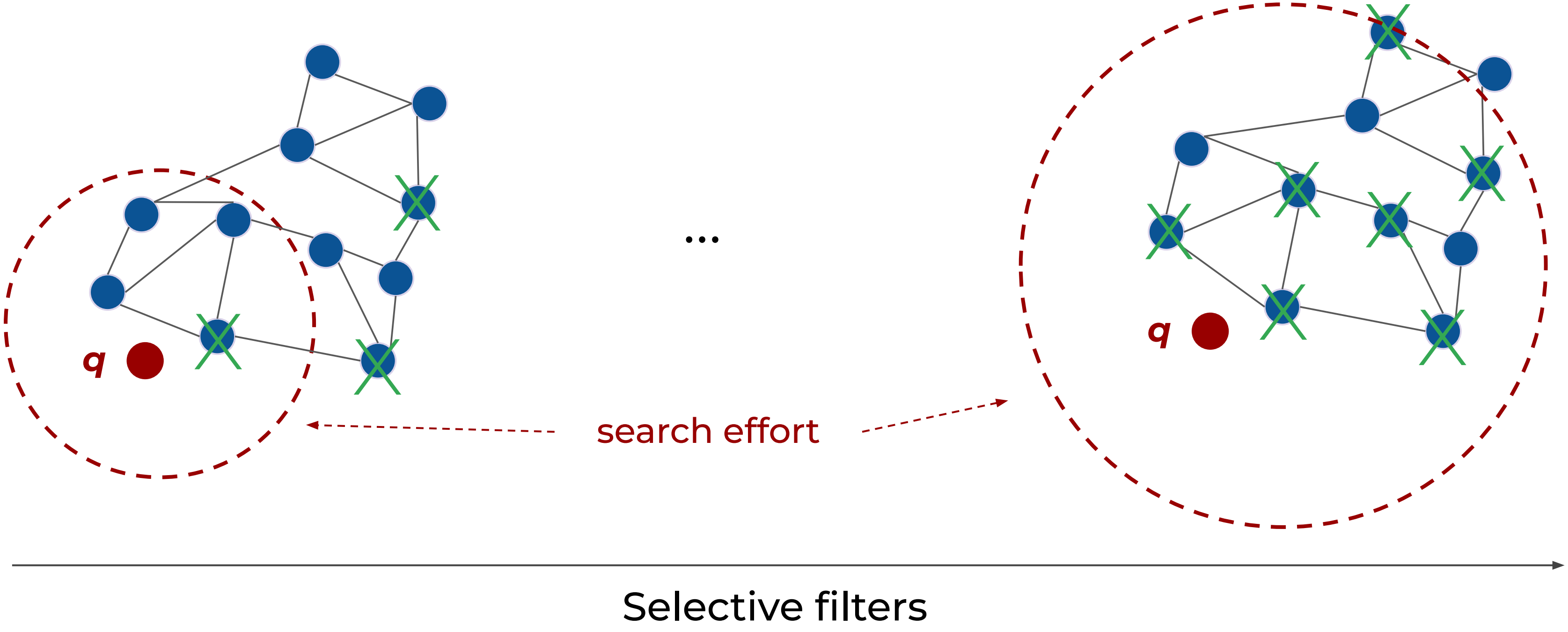
ANNS_search(q, K, *search_effort_params*)

How do we tune the search_effort_params?

Filters increase search effort and the challenge of tuning the search parameters

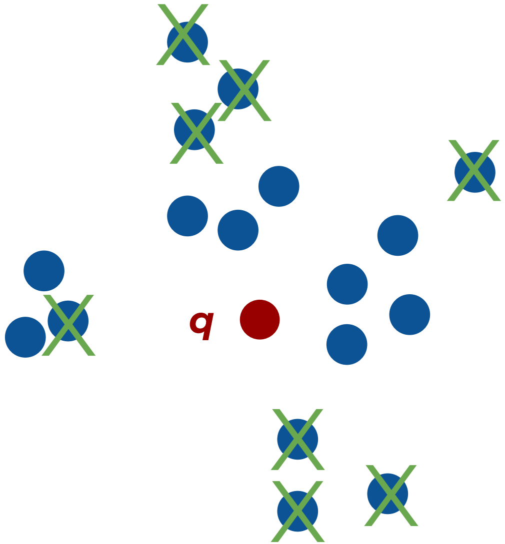
Filter Selectivity

Selective filters make it harder to find valid nodes



search effort increases

Value-Vector Correlation

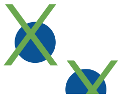


Positive correlation

Captures the relationship between
the **probability of satisfying the filters**
and
the **distance from the query vector.**

[Liana Patel et al SIGMOD 2024]

Value-Vector Correlation



Captures the relationship between the **probability of satisfying the filters**

Definition: Query Correlation. We will consider the query-to-target distances for the *given dataset* compared to the expected query-to-target distances for a *hypothetical dataset*, under which no clustering is present. Formally, we define the *query correlation* of the hybrid search workload Q over dataset D as:

$$C(D, Q) = \mathbb{E}_{(x_i,\, p_i) \in Q}\left[\, \mathbb{E}_{R_i}[g(x_i, R_i)] - g(x_i, X_{p_i}) \,\right]$$

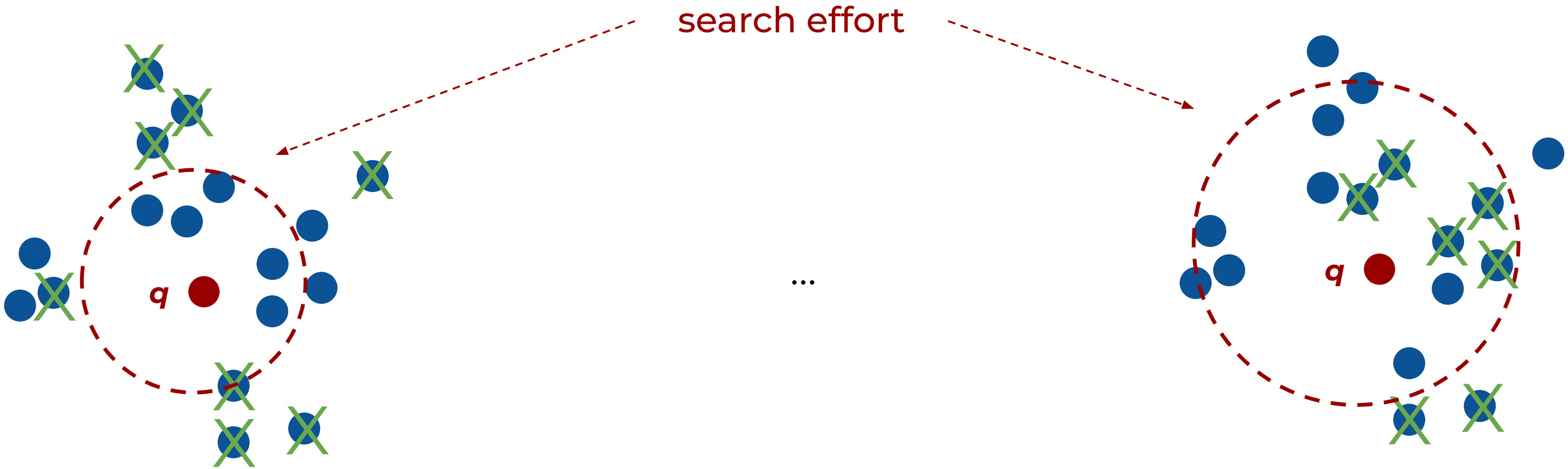
We let R_i be a random set variable of $|X_{p_i}|$ vectors drawn *uniformly* from X, defined for each hybrid query $(x_i, p_i) \in Q$. We define $g(x, S) = \min_{y \in S} dist(x, y)$ to be the function mapping the query vector x to the minimum distance of neighbors from the given vector set $S \subseteq R^d$. Note that $g(x_i, X_{p_i})$ is the ground-truth hybrid-search target of the query (x_i, p_i).

Positive correlation

Filter and Vector Correlation

K=5

Is using an index a good idea?



Positive correlation | **No correlation** | **Negative Correlation**

Items lik  and category= ”summer clothes”

Items lik  and price > 10K $

search effort increases

Performance challenge & Query Optimization

COST = f(**#distance computations** * **cost_distcomp**, **#filter evaluations** * **cost_filter**)

Multiple access paths

Tree/graph/hash indices
ANN vs KNN
Batch vs one-at-time eval (*trees)
Index memory access pattern

heap/columnar/b+tree
*not covering in-mem index

Multiple execution methods

Choose: Pre-/post-/inline-filtering

- access path/query complexity affect costs
- selectivity and correlation impact # of filters/#dist comps

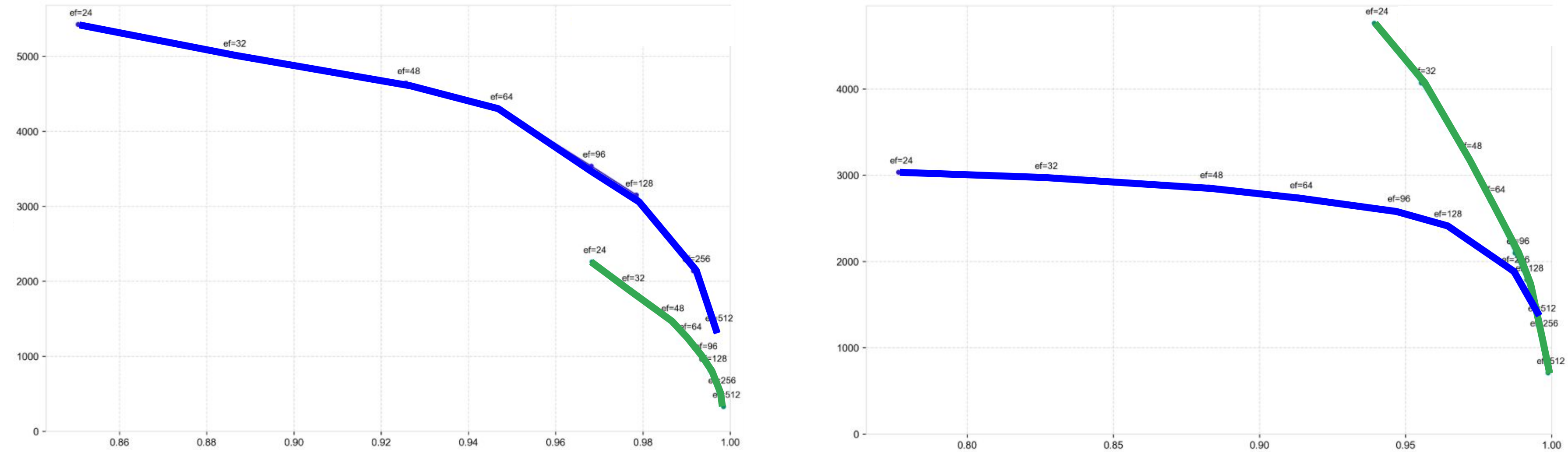
[https://weaviate.io/blog/speed-up-filtered-vector-search]

Performance is query dependent

Filter-first vs Distance compute-first

20% Selectivity (filter-first wins) **50% Selectivity (distcomp-first wind)**

QPS



Recall

VS: filter first VS: distance comp first

Dataset: beir-cohere-500k-filtered-dot-X

Quantization + dimensionality reduction

Distance computation and access cost is relative to vector size

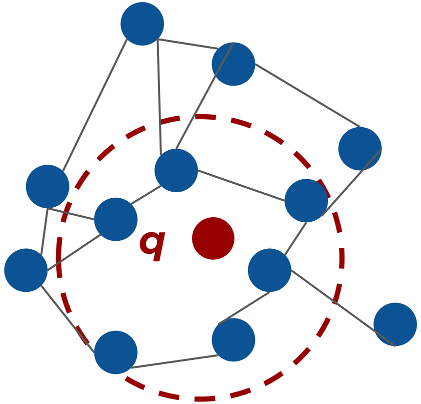
- Dimensionality reduction (PCA, ...)
- Quantization
  - Reduces precision
  - Trees offer more opportunities with residualization
  - Best of both worlds: Score fast with quantized vectors, then, re-score with full precision

Additional tuning knob : How much score vs re-score to do?

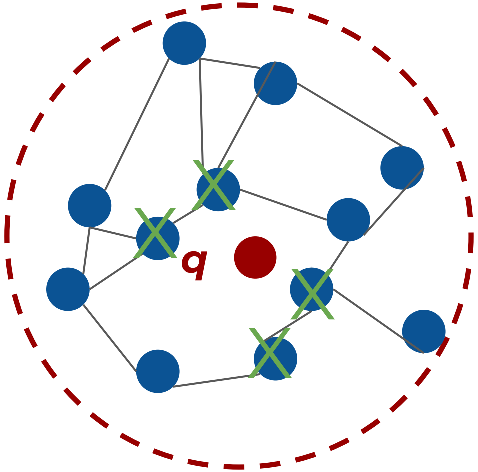
[Philip Sun et al NeurIPS 2023, RaBitQ Jianyang Gao et al SIGMOD 2024]

Not all datasets+queries are equally easy

and filters change hardness



Easy Query



Filters make it a Hard Query

[Zeyu Wang et al VLDB 2024, Martin Aumüller et al Information Systems 2020]

Not all datasets+queries are equally easy

and filters change hardness

Local Intrinsic Dimensionality (LID) / Local Relative Contrast (LRC)

"How hard is it to distinguish kNN points from other points wrt the distance to the query?"

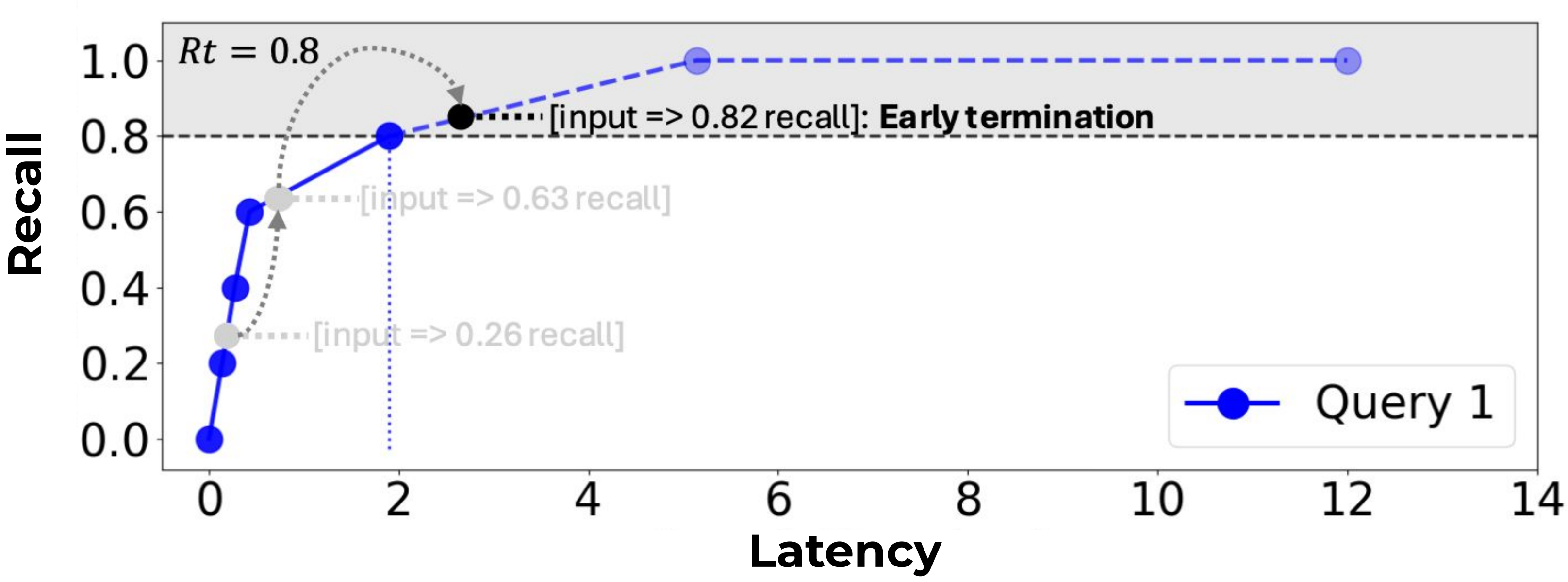
Steiner-Hardness

Minimum Effort (ME) for graphs: Search effort specific to graphs.

Adapt for FVS queries?

[Zeyu Wang et al VLDB 2024, Martin Aumüller et al Information Systems 2020]

Avoid predictions: Adaptive Execution
DARTH



[Manos Chatzakis et al SIGMOD 2026]

FVS with specialized indices

Composite Indexes Classification

- **Index shape**
  - Hashing
  - Graph
  - Tree (multi/single level)
  - IVF

Already covered
(and not specific to filtered search)

- **Filter agnosticism**
- **Index types**
  - Value-induced neighborhood
  - Predicate subgraph traversal
- **Supported Filters**
  - Limited values (i.e. labels)
    - Limited operations
  - Limited filter cardinality ranges

Sometimes inherent to the composite index' nature

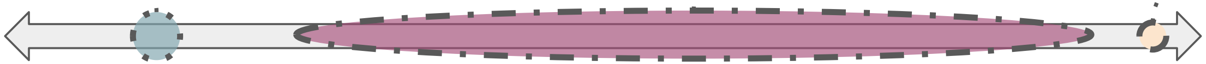
Filter agnosticism is a spectrum

Unmodified Index, Modified search
Inline filtering

Composite indexes
Search + filter simultaneously

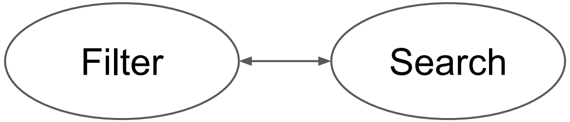
Prepartitioned Indexes
Search a partition

Filters unknown at build time

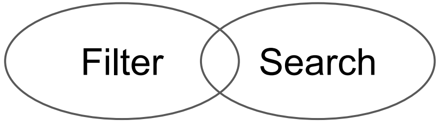


Filters (values, ops) known at build time

More general
(reuses off-the-shelve indexes)
More robust against new query filters



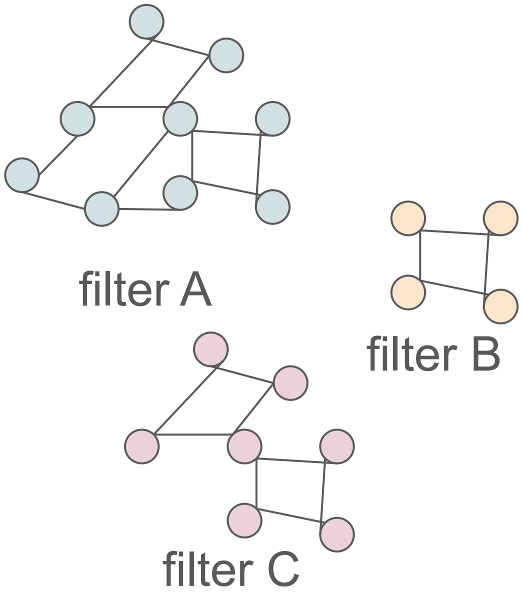
Tightly coupled
filter+search operators



Index by filter
Search

Value-induced neighborhood

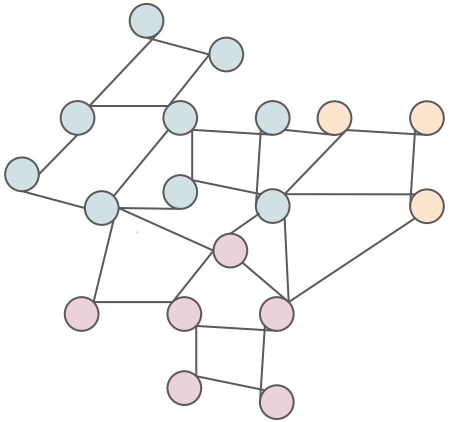
Ideally, we would build an index per filter



But…

Footprint is too high because of duplicated nodes!
or
Not enough data to build an index on small partitions

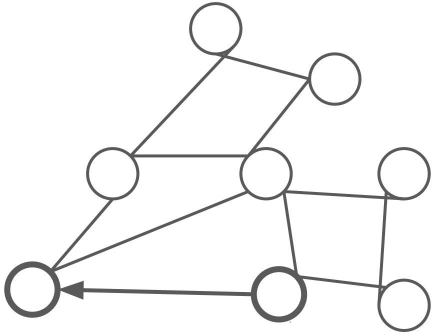
Emulate this partitioning with efficient footprint



Monolithic graph with pruning

Value-induced neighborhood

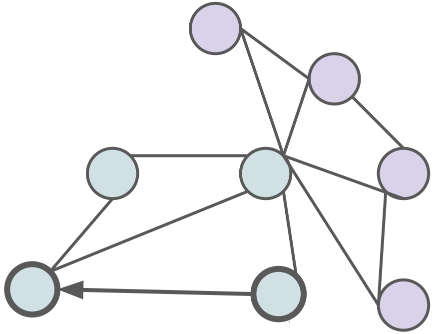
Typically, similarity refers to embedding distance



dist = L2(p1, p2)

Involve attribute values in similarity calculation

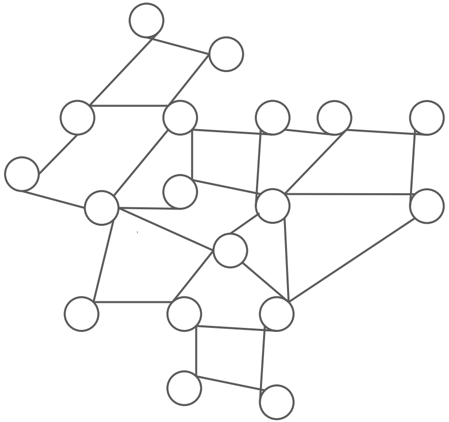
Generate new index based on **attribute + embedding similarity**



dist = f(L2(p1, p2), sim(att1, att2))

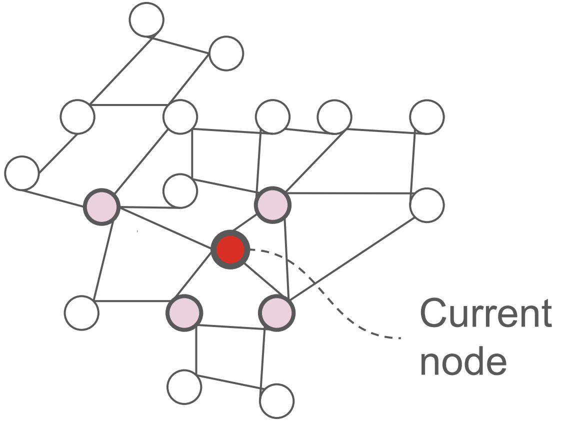
Predicate traversal

Alternatively, we can reuse unfiltered indexes



Use inline filtering to discover filter-passing nodes

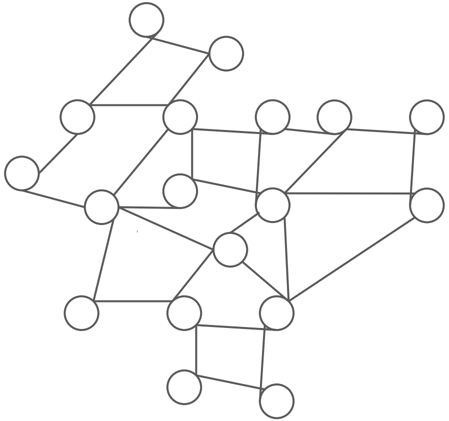
Then, **light up the right neighborhood** at search time!



Predicate traversal

Densified Predicate traversal

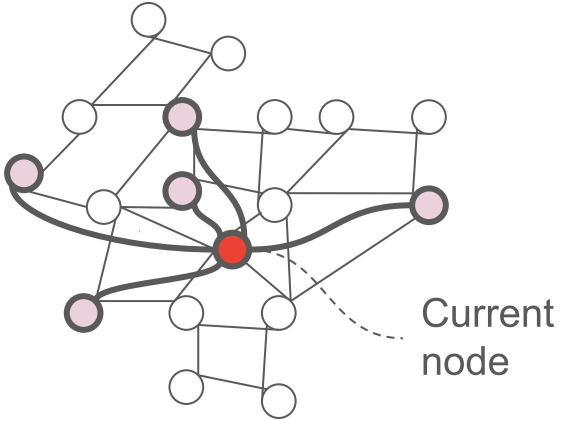
Alternatively, we can reuse unfiltered indexes



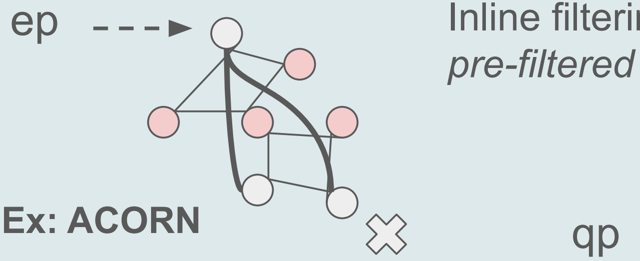
Use inline filtering to discover filter-passing nodes

Add edges to alleviate connectivity issue

Then, **light up the right neighborhood** at search time!



The *archetypes* of composite (graph) indexes



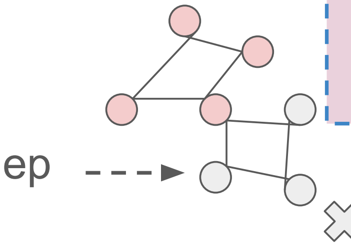
Inline filtering to emulate
pre-filtered subgraphs

Patel@SIGMOD'24

Ex: ACORN

Densified predicate traversal

Prefilter + stitch

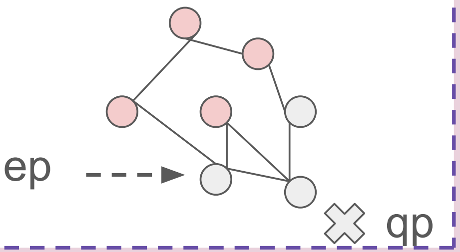


Ex: StitchedVamana
Gollapudi@WWW'23

Approximate prefiltered graph

distance = f(attribute,embeddings)

Ex: NHQ [2]
Wang@NIPS'23



Composite distance

Value-induced neighborhood

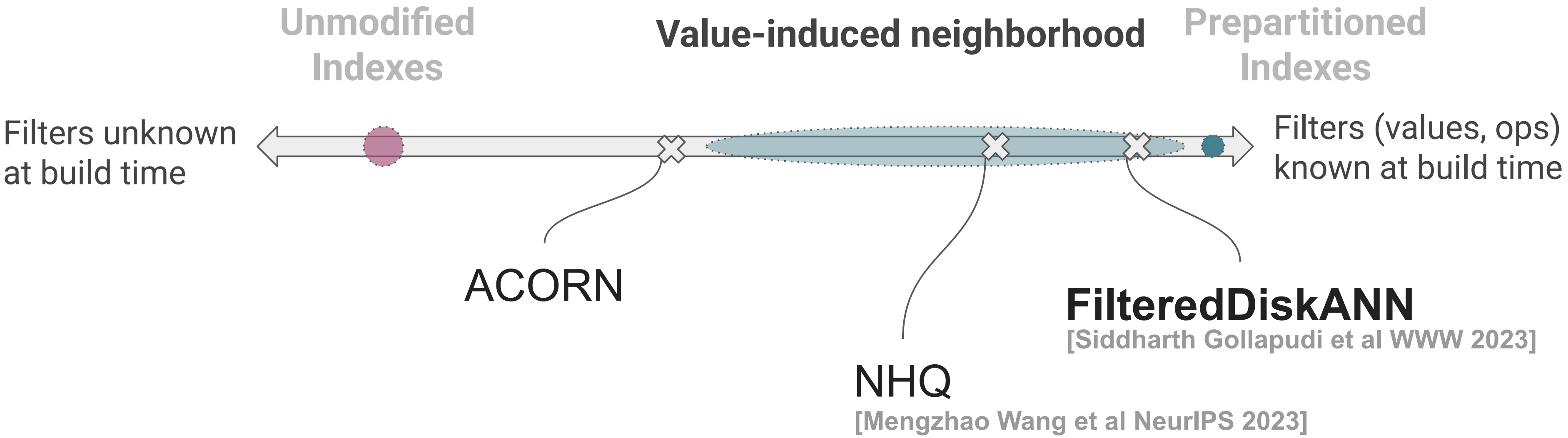
More general
More robust

Filters **unknown** at build time

Filters **known** at build time

Tightly coupled
filter+search

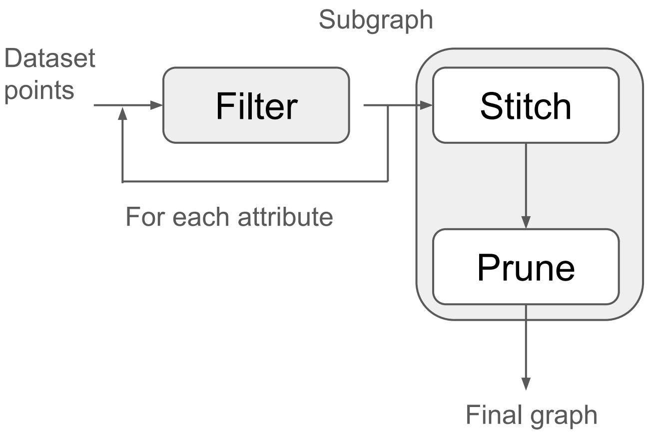
Approximate subgraph traversal



Deep Dive on Filtered DiskANN (Gollapudi WWW '23)

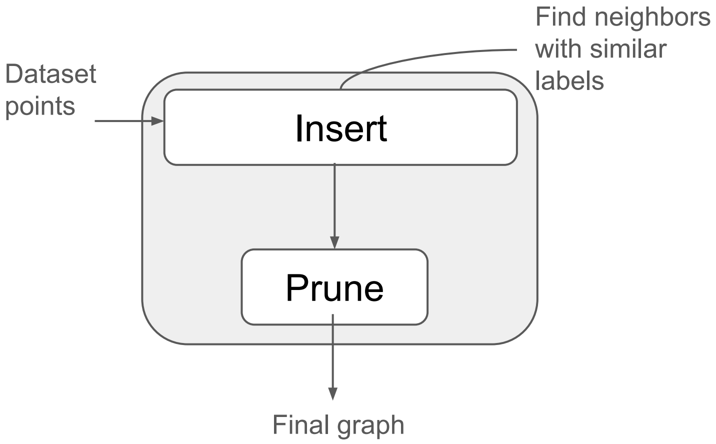
Based on DiskANN (Vamana).

StitchedVamana



Better QPS @same recall

FilteredVamana



Faster index building
&
More amenable to incremental updates

Deep Dive on [Filtered DiskANN](#) (Gollapudi WWW '23)

Dataset	Dim	# Pts.	# Queries	Source Data	Filters	Filters per Pt.	Unique Filters	100pc.	75pc.	50pc.	25pc.	1pc.
Turing	100	2,599,968	996	Text	Natural	1.09	3070	0.127	$1.56x10^{-4}$	$4.15x10^{-5}$	$1.54x10^{-5}$	$7.7x10^{-6}$
Prep	64	1,000,000	10000	Text	Natural	8.84	47	0.425	0.136	0.130	0.127	0.09
DANN	64	3,305,317	32926	Text	Natural	3.91	47	0.735	0.361	0.183	0.167	0.150
SIFT	128	1,000,000	10000	Image	Random	1	12	0.083	0.083	0.083	0.083	0.082
GIST	960	1,000,000	1000	Image	Random	1	12	0.083	0.083	0.083	0.083	0.082
msong	420	992,272	200	Audio	Random	1	12	0.083	0.082	0.082	0.082	0.082
audio	192	53,387	200	Audio	Random	1	12	0.085	0.084	0.083	0.082	0.081
paper	200	2,029,997	10000	Text	Random	1	12	0.083	0.083	0.083	0.083	0.082

Table 1: Datasets used in the evaluation and their statistics. Top 3 rows are real-world datasets; the rest are semi-synthetic.

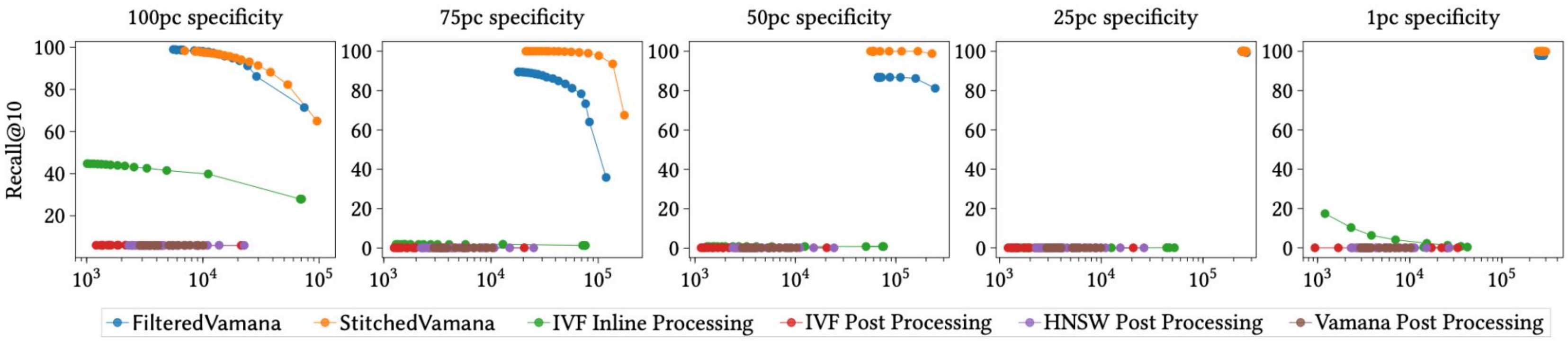


Figure 1: Turing dataset: QPS (x-axis) vs recall@10 for various algorithms with filters of 100, 75, 50, 25 and 1 percentile specificity.

Deep Dive on Filtered DiskANN (Gollapudi WWW '23)

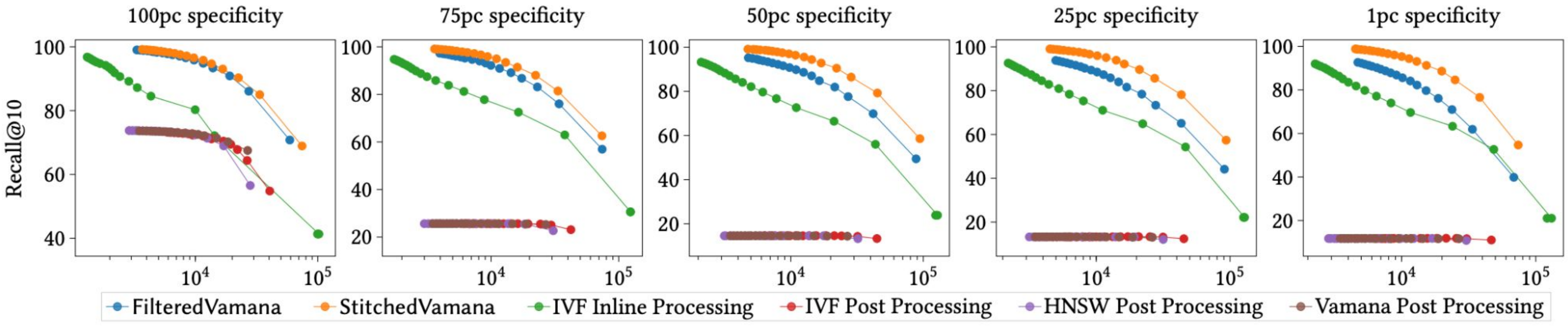


Figure 2: Prep dataset: QPS (x-axis) vs recall@10 for various algorithms with filters of 100, 75, 50, 25 and 1 percentile specificity.

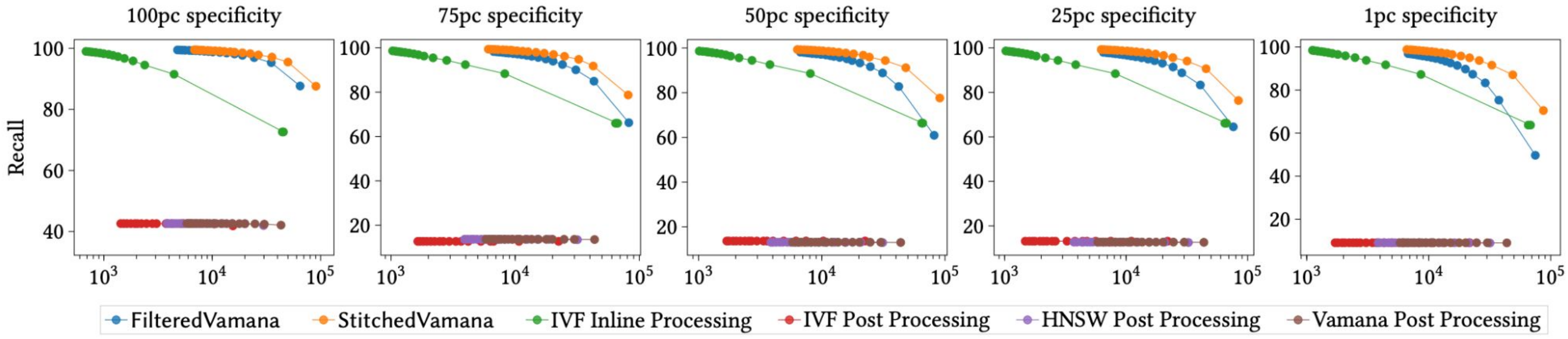
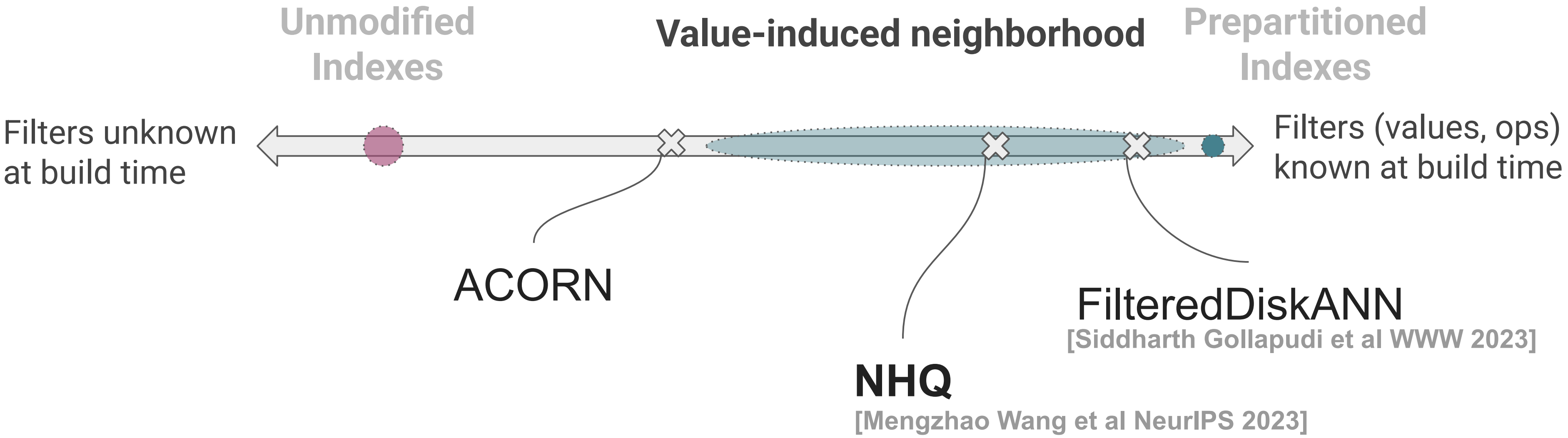
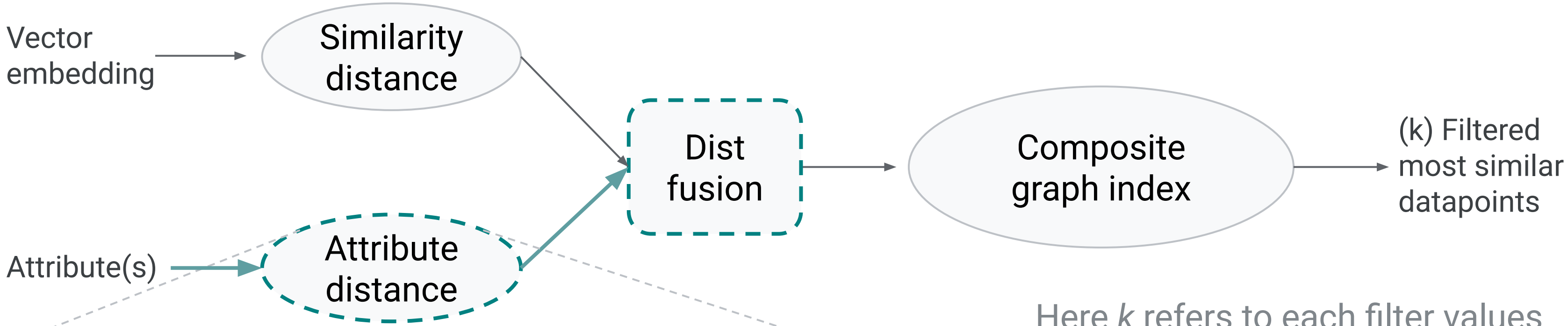


Figure 3: DANN dataset: QPS (x-axis) vs recall@10 for various algorithms with filters of 100, 75, 50, 25 and 1 percentile specificity.

Composite distance: NHQ (Wang NIPS'23)



Deep Dive on NHQ (Wang NIPS'23)



Here k refers to each filter values (m is max # of filter values).

$$\chi(\ell(e_i), \ell(e_j)) = \sum_{k=0}^{m-1} \phi(\ell(e_i)^k, \ell(e_j)^k) \quad ,$$

where $\phi(\ell(e_i)^k, \ell(e_j)^k)$ is

$$\phi(\ell(e_i)^k, \ell(e_j)^k) = \begin{cases} 0 & \ell(e_i)^k = \ell(e_j)^k \\ 1 & \ell(e_i)^k \neq \ell(e_j)^k \end{cases}$$

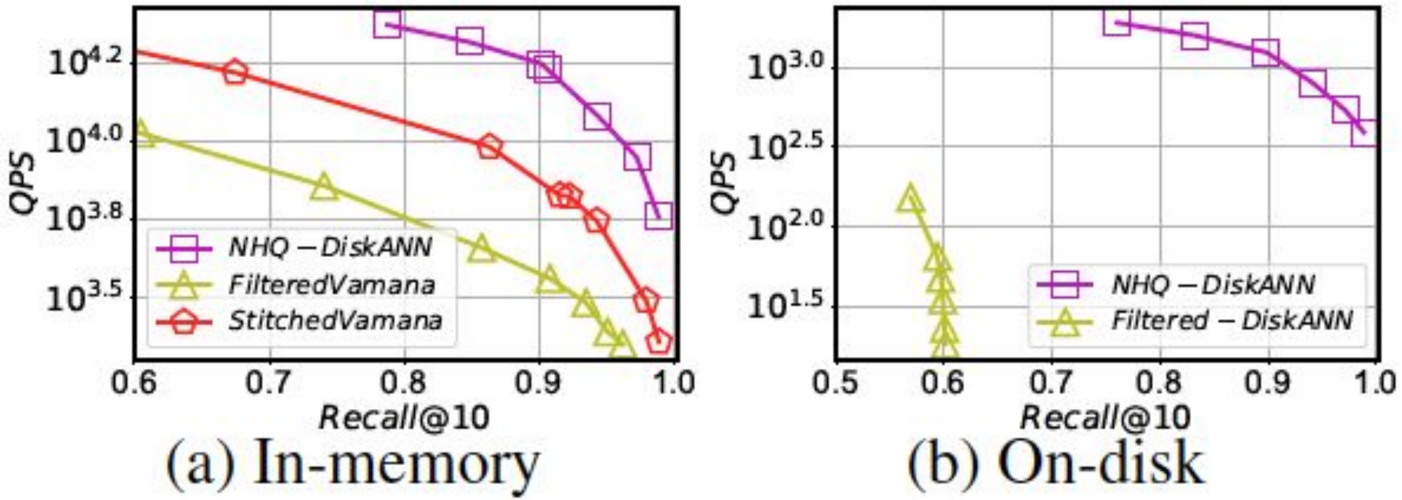
The more filter values in common two edges ei, ej, have, the smaller the distance *X*.

Experimental setup

Table 1: Statistics of real-world datasets.

Dataset	Dimension	# Base	# Query	LID [33, 18]	Type
UQ-V	256	1,000,000	10,000	7.2	Video + Attributes
Msong	420	992,272	200	9.5	Audio + Attributes
Audio	192	53,387	200	5.6	Audio + Attributes
SIFT1M	128	1,000,000	10,000	9.3	Image + Attributes
GIST1M	960	1,000,000	1,000	18.9	Image + Attributes
Crawl	300	1,989,995	10,000	15.7	Text + Attributes
GloVe	100	1,183,514	10,000	20.0	Text + Attributes
Enron	1,369	94,987	200	11.7	Text + Attributes
Paper	200	2,029,997	10,000	-	Text + Attributes
BIGANN100M	128	100,000,000	10,000	9.3	Image + Attributes

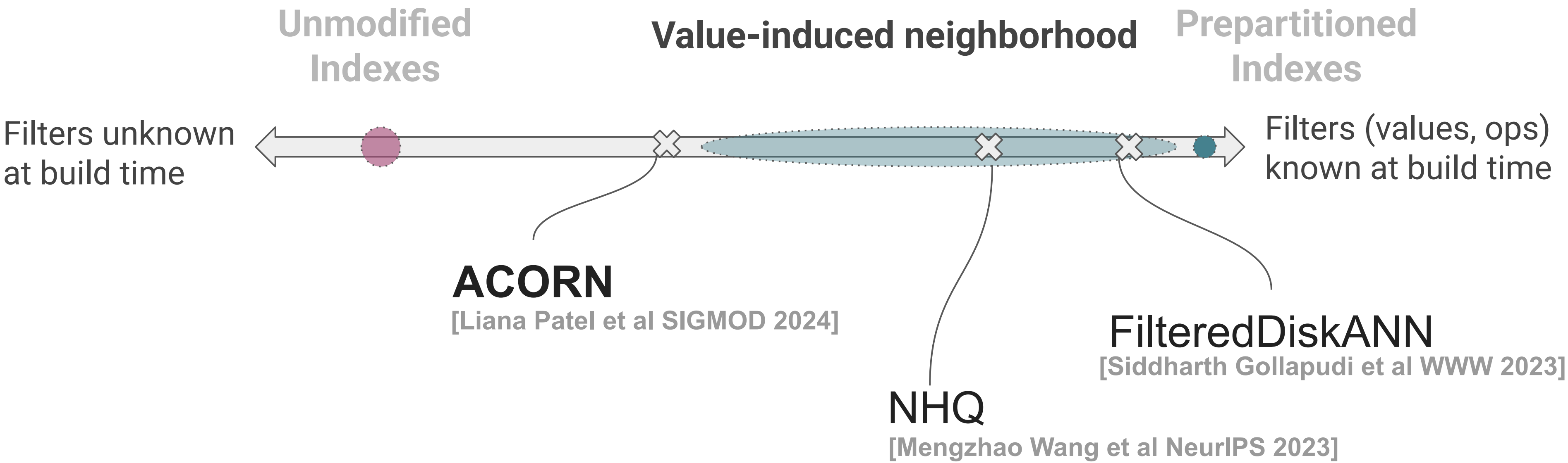
NHQ vs. FilteredDiskANN



NHQ-DiskANN >> Filtered−DiskANN (in-memory or disk)

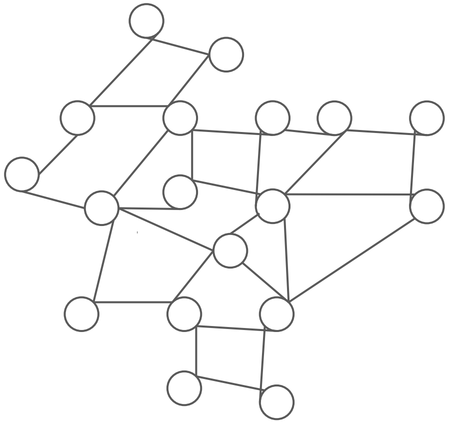
Note: **Equality filter conditions for non-intersecting sets** may benefit from Stitched/FilteredVamana over NHQ.

Densified predicate traversal: ACORN



Deep Dive on ACORN (Patel SIGMOD'24)

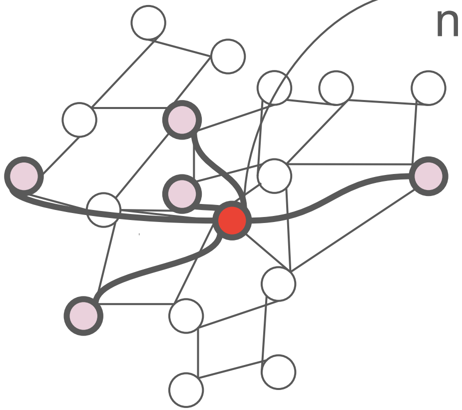
Reuse unfiltered indexes



Use inline filtering to discover filter-passing nodes

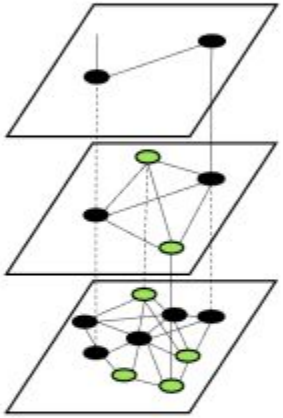
Add edges to alleviate connectivity issue

Then, **light up the right neighborhood** at search time!

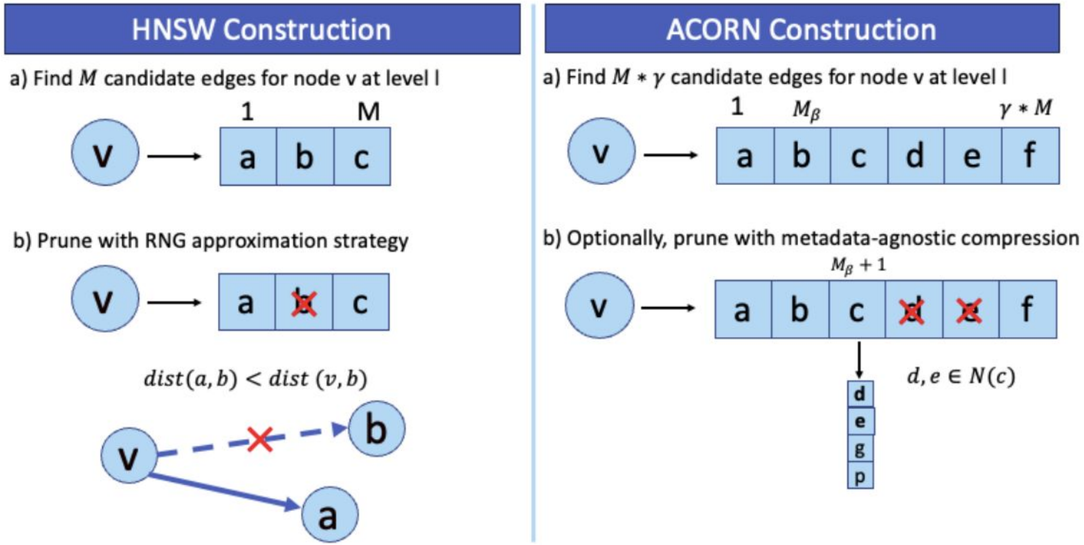


Densified Predicate traversal

Deep Dive on ACORN (Patel SIGMOD'24)



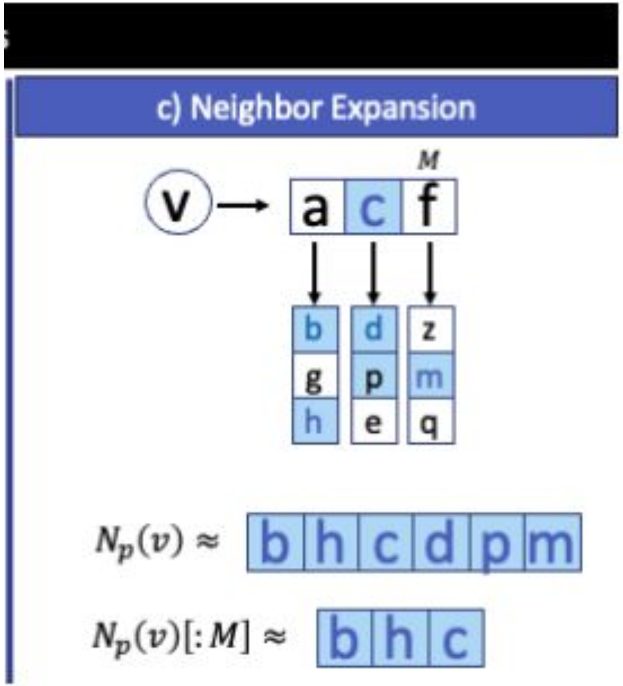
1) Adds edges to avoid islands
2) **Filter Agnostic**
3) Not composite



Deep Dive on ACORN (Patel SIGMOD’24)

Two variants:

1) ACORN-1: faster to build
2) ACORN-γ: faster to search

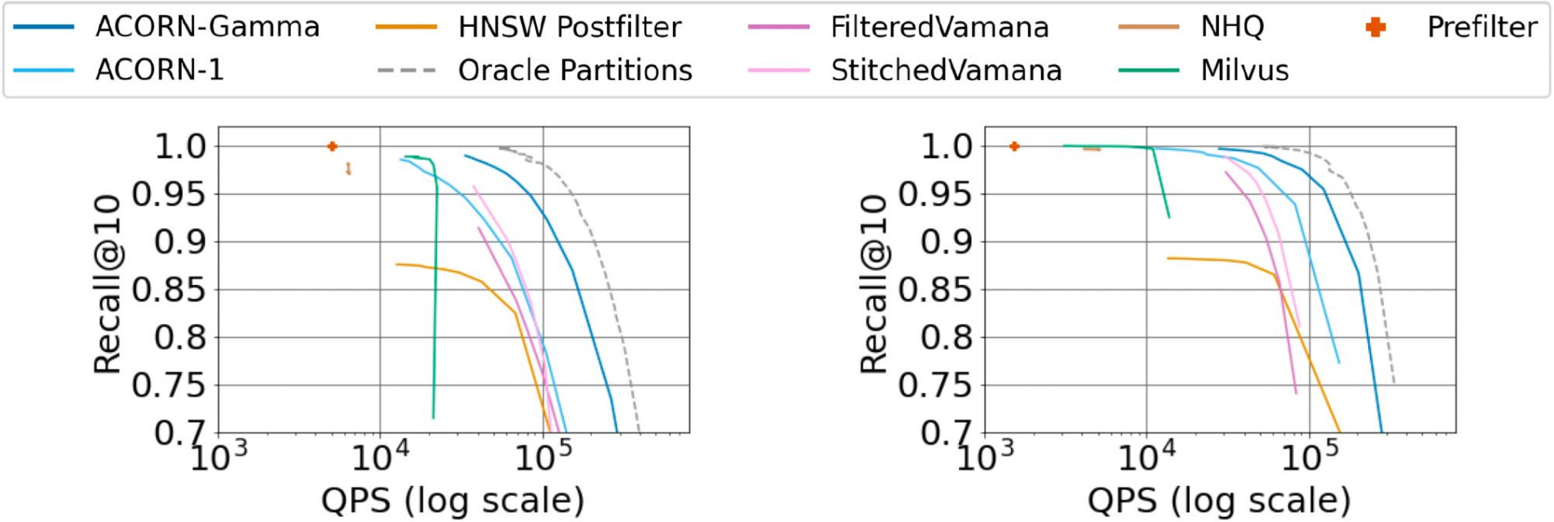


Experimental setup of ACORN (Patel SIGMOD'24)

Table 2: Datasets

	Base Data				Query Workload		
	# Vectors	Vector Dim	Vector Source Data	Structured Data	Predicate Operators	Avg. Query Selectivity	Predicate Cardinality
SIFT1M	1,000,000	128	images	random int.	equals(y)	0.083	12
Paper	2,029,997	200	passages	random int.	equals(y)	0.083	12
TripClick	1,055,976	768	passages	clinical area list & publication date	contains($y_1 \vee y_2 \vee \ldots$) & between(y_1, y_2)	0.17, 0.36 [2]	$> 10^8$
LAION (1M)	1,000,448	512	images	text captions & keyword list	regex-match(y) & contains($y_1 \vee y_2 \vee \ldots$)	0.056 - 0.13 [3]	$> 10^{11}$
LAION (25M)	24,653,427	512	same as above	same as above	same as above	same as above	same as above

Experiments of ACORN (Patel SIGMOD'24)



(a) SIFT1M Dataset (b) Paper Dataset

ACORN outperforms both Filtered DiskANN and NHQ for a fixed recall, while maintaining generality (not specialized to a single filter value).

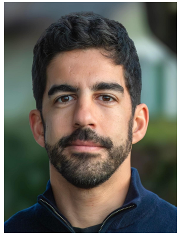
Oracle Partitions = Ideal Filtered HNSW (upper bound for performance)

Future directions

Many research challenges ahead...

- Autotuning: High quality and efficient filtered vector search
  - Index hyperparameter auto-tuning
  - Index data structure: Tree vs graph vs other
  - Search algorithm for inlined FVS: Iterative, sweeping, others…
- Quality metrics better suited for filtered vector search
- Benchmarking
  - Focus on filtered search and correlation between relational columns and vectors

    [In progress effort led by Yannis Chronis @ ETH]



Many research challenges ahead...

- Query optimization
  - Correct choice is highly sensitive to selectivity, which could be erroneous
  - Adaptive execution
  - Correlation within tables, across joins
  - No longer just about latency/throughput but also about high quality
- Embedding similarity metrics
  - Other metrics beyond cosine
  - User-specified metrics
  - Auto-selection based on workloads

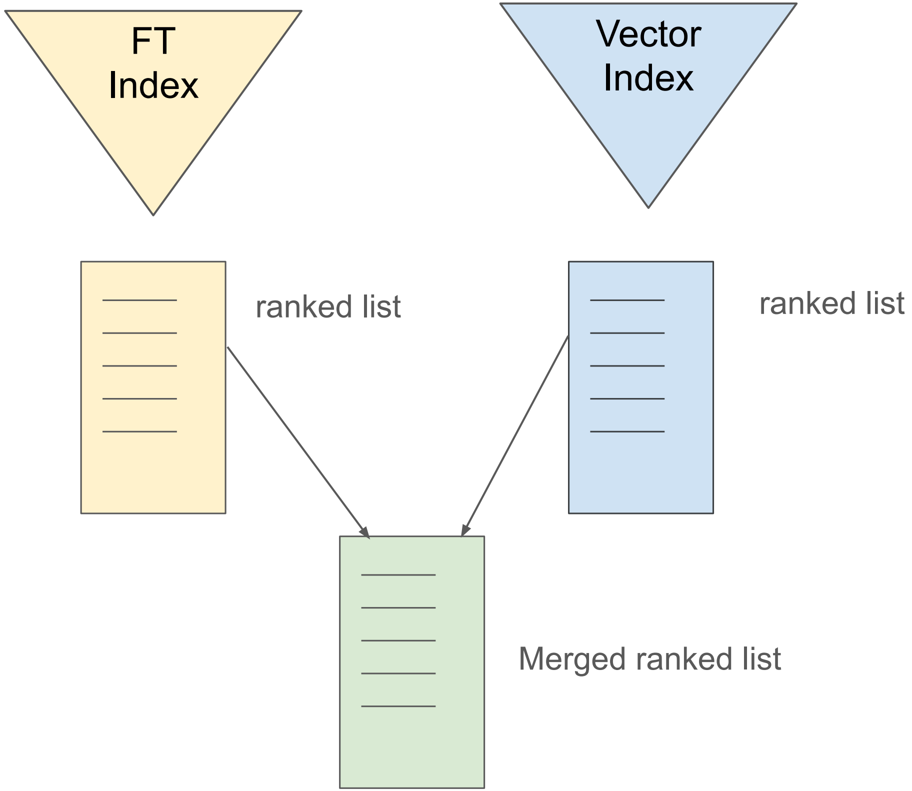
Many research challenges ahead...

Hybrid search: combining full-text keyword search and vector similarity

How to merge the two ranked lists?

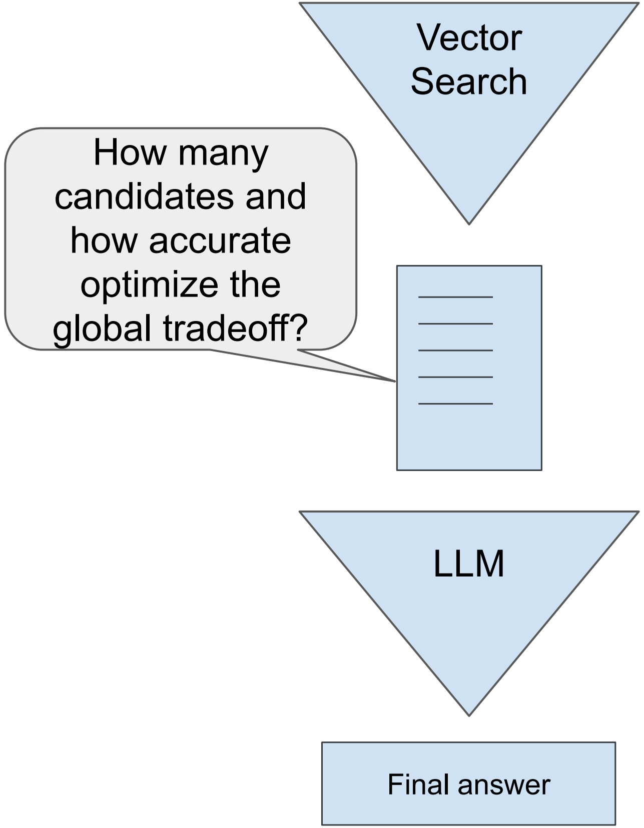
- Fixed weights
- Dynamically adjust to workload

How to add relational filters to the mix?



Optimization of the complete pipeline...

RAG pipeline



End-to-end accuracy is what the customers see

Automating the entire pipeline becomes challenging

References

[1] 2023. Brie Wolfson. Building chat langchain. https://blog.langchain.dev/buildingchat-langchain-2/)).
[2] 2025. Facebook FAISS. https://github.com/facebookresearch/faiss.
[3] 2025. Oracle Vector Search Manual,.
https://docs.oracle.com/en/database/oracle/oracle-database/23/vecse/ai-vector-search-users-guide.pdf.
[4] 2025. Pinecone. https://www.pinecone.io/.
[5] 2025. ScaNN. github.com/google-research/google-research/tree/master/scann.
[6] 2025. ScaNN for AlloyDB,. https://services.google.com/fh/files/misc/scann_for_
alloydb_whitepaper.pdf.
[7] 2025. SPTAG: A Library for Fast Approximate Nearest Neighbor Search. https://github.com/Microsoft/SPTAG.
[8] 2025. Weviate. https://weaviate.io/.
[9] Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. Retrieval-based language models and applications. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts).
[10] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In 2016 IEEE 26th international workshop on machine learning for signal processing (MLSP). IEEE, 1–6.
[11] Fedor Borisyuk, Siddarth Malreddy, Jun Mei, Yiqun Liu, Xiaoyi Liu, Piyush Maheshwari, Anthony Bell, and Kaushik Rangadurai. 2021. VisRel: Media search at scale. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2584–2592.
[12] Cheng Chen, Chenzhe Jin, Yunan Zhang, Sasha Podolsky, Chun Wu, Szu Po Wang, Eric Hanson, Zhou Sun, Robert Walzer, and Jianguo Wang. 2024. SingleStore-V: An Integrated Vector Database System in SingleStore. Proc. VLDB Endow. 17, 12 (Aug. 2024), 3772–3785. https://doi.org/10.14778/3685800.3685805
[13] James C. Corbett and et. al. 2013. Spanner: Google's Globally Distributed Database. ACM Trans. Comput. Syst. 31, 3, Article 8 (Aug. 2013), 22 pages. https://doi.org/10.1145/2491245
[14] Sanjoy Dasgupta and Yoav Freund. 2008. Random projection trees and low dimensional manifolds. In Proceedings of the fortieth annual ACM symposium on Theory of computing. 537–546.

References

[15] Xin Luna Dong. 2024. The Journey to a Knowledgeable Assistant with Retrieval-Augmented Generation (RAG) (SIGMOD/PODS '24). Association for Computing Machinery, New York, NY, USA, 3. https://doi.org/10.1145/3626246.3655999

[16] Ming Du, Arnau Ramisa, Amit Kumar KC, Sampath Chanda, Mengjiao Wang, Neelakandan Rajesh, Shasha Li, Yingchuan Hu, Tao Zhou, Nagashri Lakshminarayana, et al . 2022. Amazon shop the look: A visual search system for fashion and home. In Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining. 2822–2830.

[17] Karima Echihabi, Kostas Zoumpatianos, and Themis Palpanas. 2021. New trends in high-D vector similarity search: al-driven, progressive, and distributed. Proc. VLDB Endow. 14, 12 (July 2021), 3198–3201. https://doi.org/10.14778/3476311.3476407

[18] Karima Echihabi, Kostas Zoumpatianos, and Themis Palpanas. 2021. New trends in high-d vector similarity search: al-driven, progressive, and distributed. Proceedings of the VLDB Endowment 14, 12 (2021), 3198–3201.

[19] Jianyang Gao, Yutong Gou, Yuexuan Xu, Yongyi Yang, Cheng Long, and Raymond Chi-Wing Wong. 2024. Practical and Asymptotically Optimal Quantization of High-Dimensional Vectors in Euclidean Space for Approximate Nearest Neighbor Search. arXiv:2409.09913 [cs.DB] https://arxiv.org/abs/2409.09913

[20] Siddharth Gollapudi and et. al. 2023. Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. In WWW '23 (Austin, TX, USA). Association for Computing Machinery, New York, NY, USA, 3406–3416. https://doi.org/10.1145/3543507.3583552

[21] Martin Grohe. 2020. word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems. 1–16.

[22] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In International Conference on Machine Learning. Https: //arxiv.org/abs/1908.10396

[23] Jiawei Han, Xifeng Yan, and Philip S. Yu. 2006. Mining, Indexing, and Similarity Search in Graphs and Complex Structures. In Proceedings of the 22nd International Conference on Data Engineering (ICDE '06). IEEE Computer Society, USA, 106. https://doi.org/10.1109/ICDE.2006.99

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.

References

[25] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data 7, 3 (2019), 535–547.

[26] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How good are query optimizers, really? Proc. VLDB Endow. 9, 3 (Nov. 2015), 204–215. https://doi.org/10.14778/2850583.2850594

[27] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. IEEE Trans. Pattern Anal. Mach. Intell. 42, 4 (April 2020), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

[28] Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh. 2018. A survey of product quantization. ITE Transactions on Media Technology and Applications 6, 1 (2018), 2–10.

[29] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis, Hossein Ahmadi, Dan Delorey, Slava Min, Mosha Pasumansky, and Jeff Shute. 2020. Dremel: a decade of interactive SQL analysis at web scale. Proc. VLDB Endow. 13, 12 (Aug. 2020), 3461–3472. https://doi.org/10.14778/3415478.3415568

[30] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Survey of vector database management systems. The VLDB Journal 33, 5 (July 2024), 1591–1615. https://doi.org/10.1007/s00778-024-00864-x

[31] Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. 2024. ACORN:Performant and Predicate-Agnostic Search Over Vector Embeddings and Structured Data. Proc. ACM Manag. Data 2, 3, Article 120 (May 2024), 27 pages.https://doi.org/10.1145/3654923

[32] Jianbin Qin, Wei Wang, Chuan Xiao, Ying Zhang, and Yaoshu Wang. 2021. High- dimensional similarity query processing for data science. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 4062–4063.

[33] Parikshit Ram and Kaushik Sinha. 2019. Revisiting kd-tree for nearest neighbor search. In Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining. 1378–1388.

[34] Patrick Schäfer, Jakob Brand, Ulf Leser, Botao Peng, and Themis Palpanas. 2024. Fast and Exact Similarity Search in less than a Blink of an Eye. arXiv preprint arXiv:2411.17483 (2024).

[35] Michael Stonebraker and Greg Kemnitz. 1991. The POSTGRES next generation database management system. Commun. ACM 34, 10 (Oct. 1991), 78–92. https://doi.org/10.1145/125223.125262

References

[36] Suhas Jayaram Subramanya, Devvrit, Rohan Kadekodi, Ravishankar Kr-ishaswamy, and Harsha Vardhan Simhadri. 2019. DiskANN: fast accurate billion-point nearest neighbor search on a single node. Curran Associates Inc., Red Hook, NY, USA.
[37] Philip Sun, David Simcha, Dave Dopson, Ruiqi Guo, and Sanjiv Kumar. 2023. SOAR: Improved Indexing for Approximate Nearest Neighbor Search. In Neural Information Processing Systems. https://arxiv.org/abs/2404.00774
[38] Jianguo Wang and et. al. 2021. Milvus: A Purpose-Built Vector Data Management System. In Proceedings of the 2021 International Conference on Management of Data (Virtual Event, China) (SIGMOD '21). Association for Computing Machinery, New York, NY, USA, 2614–2627. https://doi.org/10.1145/3448016.3457550
[39] Mengzhao Wang, Lingwei Lv, Xiaoliang Xu, Yuxiang Wang, Qiang Yue, and Jiongkang Ni. 2023. An efficient and robust framework for approximate nearest neighbor search with attribute constraint. In NIPS '23 (New Orleans, LA, USA). Curran Associates Inc., Red Hook, NY, USA, Article 692, 14 pages.
[40] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. 2020. Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data. Proceedings of the VLDB Endowment 13, 12 (2020), 3152–3165.
[41] Wei Wu, Junlin He, Yu Qiao, Guoheng Fu, Li Liu, and Jin Yu. 2022. HQANN: Efficient and robust similarity search for hybrid queries with structured and unstructured constraints. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 4580–4584.
[42] Qian Xu, Juan Yang, Feng Zhang, Junda Pan, Kang Chen, Youren Shen, Amelie Chi Zhou, and Xiaoyong Du. 2025. Tribase: A Vector Data Query Engine for Reliable and Lossless Pruning Compression using Triangle Inequalities. Proc. ACM Manag.Data 3, 1, Article 82 (Feb. 2025), 28 pages. https://doi.org/10.1145/3709743
[43] Wen Yang, Tao Li, Gai Fang, and Hong Wei. 2020. Pase: Postgresql ultra-high-dimensional approximate nearest neighbor search extension. In Proceedings of the 2020 ACM SIGMOD international conference on management of data. 2241–2253.
[44] Qianxi Zhang, Shuotao Xu, Qi Chen, Guoxin Sui, Jiadong Xie, Zhizhen Cai, Yaoqi Chen, Yinxuan He, Yuqing Yang, Fan Yang, et al. 2023. {VBASE}: Unifying Online Vector Similarity Search and Relational Queries via Relaxed Monotonicity. In 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23). 377–395

References

[45] Zeqi Zhu, Zeheng Fan, Yuxiang Zeng, Yexuan Shi, Yi Xu, Mengmeng Zhou, and Jin Dong. 2024. FedSQ: A Secure System for Federated Vector Similarity Queries. Proc. VLDB Endow. 17, 12 (Aug. 2024), 4441–4444. https://doi.org/10.14778/3685800.3685895

[46] Chaoji Zuo, Miao Qiao, Wenchao Zhou, Feifei Li, and Dong Deng. 2024. SeRF: Segment Graph for Range-Filtering Approximate Nearest Neighbor Search. Proc. ACM Manag. Data 2, 1, Article 69 (March 2024), 26 pages.

[47] Dmitry Baranchuk, Artem Babenko, and Yury Malkov. 2018. Revisiting the Inverted Indices for Billion-Scale Approximate Nearest Neighbors. In Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part XII. Springer-Verlag, Berlin, Heidelberg, 209–224. https://doi.org/10.1007/978-3-030-01258-8_13

[48] Zeyu Wang, Qitong Wang, Xiaoxing Cheng, Peng Wang, Themis Palpanas, and Wei Wang. 2024. Steiner-Hardness: A Query Hardness Measure for Graph-Based ANN Indexes. Proc. VLDB Endow. 17, 13 (September 2024), 4668–4682. https://doi.org/10.14778/3704965.3704974

[49] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. 2015. Practical and optimal LSH for angular distance. In Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15), Vol. 1. MIT Press, Cambridge, MA, USA, 1225–1233.

[50] Yang T, Hu W, Peng W, Li Y, Li J, Wang G, Liu X. Vdtuner: Automated performance tuning for vector data management systems. In2024 IEEE 40th International Conference on Data Engineering (ICDE) 2024 May 13 (pp. 4357-4369). IEEE.

[51] Gao J, Long C. Rabitq: Quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search. Proceedings of the ACM on Management of Data. 2024 May 29;2(3):1-27.

[52] Jason Ansel, Shoaib Kamil, Kalyan Veeramachaneni, Jonathan Ragan-Kelley, Jeffrey Bosboom, Una-May O'Reilly, and Saman Amarasinghe. 2014. OpenTuner: an extensible framework for program autotuning. In Proceedings of the 23rd international conference on Parallel architectures and compilation (PACT '14). Association for Computing Machinery, New York, NY, USA, 303–316. https://doi.org/10.1145/2628071.2628092

[53] Aumüller M, Bernhardsson E, Faithfull A. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Information Systems. 2020 Jan 1;87:101374.