Vector Databases in RAG: A Critical Analysis

Adam M. Lang
6/19/2024

Research Questions

- Vector DBs role in the LLM-RAG pipeline
- Why do we need Vector DBs (or why not)?
- How do you determine what Vector DB is right for your use case?
  - Does one size fit all?
- Vector DB Algorithms – how are they different?
- Do all embeddings work with every Vector DB?

Research Motivations

- **Recent Customer…..**
  - Building medical domain semantic search application
  - 17,000 documents in MongoDB Atlas (Vector Database)
  - Chose "BGE" embeddings
  - **Problem:** Search results were subpar

- Questions I had (with answers):
  - Why did you choose MongoDB Atlas?
    - "It was conveniently part of my Azure account".
  - Why did you choose the BGE embeddings?
    - "They were in the MTEB leaderboard "Top 10"

Research Motivations

- **Recent arXiv paper**
- Authors are co-founders of Marqo Vector DB
- Methods:
  - Current benchmarks for evaluating ANN systems rely on simple or lower dimensional datasets not representative of real-world data.
  - Datasets DO NOT contain vector embeddings from ML models.
  - **Many popular Vector DBs default search algorithm is HNSW.**

THE IMPACTS OF DATA, ORDERING, AND INTRINSIC DIMENSIONALITY ON RECALL IN HIERARCHICAL NAVIGABLE SMALL WORLDS

A PREPRINT

Owen P. Elliott
Marqo
Melbourne, Australia
owen@marqo.ai

Jesse Clark
Marqo
Melbourne, Australia
jesse@marqo.ai

May 29, 2024

ABSTRACT

Vector search systems, pivotal in AI applications, often rely on the Hierarchical Navigable Small Worlds (HNSW) algorithm. However, the behaviour of HNSW under real-world scenarios using vectors generated with deep learning models remains under-explored. Existing Approximate Nearest Neighbours (ANN) benchmarks and research typically has an over-reliance on simplistic datasets like MNIST or SIFT1M and fail to reflect the complexity of current use-cases. Our investigation focuses on HNSW's efficacy across a spectrum of datasets, including synthetic vectors tailored to mimic specific intrinsic dimensionalities, widely-used retrieval benchmarks with popular embedding models, and proprietary e-commerce image data with CLIP models. We survey the most popular HNSW vector databases and collate their default parameters to provide a realistic fixed parameterisation for the duration of the paper.

We discover that the recall of approximate HNSW search, in comparison to exact K Nearest Neighbours (KNN) search, is linked to the vector space's intrinsic dimensionality and significantly influenced by the data insertion sequence. Our methodology highlights how insertion order, informed by measurable properties such as the pointwise Local Intrinsic Dimensionality (LID) or known categories, can shift recall by up to 12 percentage points. We also observe that running popular benchmark datasets with HNSW instead of KNN can shift rankings by up to three positions for some models. This work underscores the need for more nuanced benchmarks and design considerations in developing robust vector search systems using approximate vector search algorithms. This study presents a number of scenarios with varying real world applicability which aim to better increase understanding and future development of ANN algorithms and embedding models alike.

arXiv paper continued…

Table 1: Default Settings of Various Vector Databases (Approximate Nearest Neighbours Systems)

System	M	$efConstruction$	$efSearch$
MarqoV1[32]	16	128	k
MarqoV2	16	512	2000
HNSWLib[16]	16	200	10
FAISS[38]	32	40	16
Chroma[10]	16	100	10
Weaviate[49]	64	128	100
Qdrant[34]	16	100	128
Milvus[29]	18	240	No Default
Vespa[42]	16	200	k
Opensearch (nmslib)[32]	16	512	512
Opensearch (Lucene)[32]	16	512	k
Elasticsearch (Lucene)[15]	16	100	No Default
Redis[37]	16	200	10
PGVector[33]	16	64	40

Note: In this table, k represents the number of results to return. Systems with $efSearch = k$ do not specify a default $efSearch$ and set it to k at search time.

- Low defaults dominate for HNSW vector DBs.
- M = 16 is default parameter for bidirectional links to form in the graph (Median = 16)
- efConstruction range: 40 - 512 (Median = 128)
- efSearch: not given for every vector DB.

Findings

- Insertion order into graph impacts temporal data in a vector space.
- HNSW recall compared to KNN search strongly influenced by intrinsic dimensionality of vector space and data insertion order.
- **Strong Relationship exists between:**
  - intrinsic dimensionality of a vector space
  - local intrinsic dimensionality (LID) of vectors in a dataset
  - temporal order of insertion into a vector DB.
- If you control the order of insertion of data into a vector DB, recall drops by almost 13%.
- Recall can vary by 8% with varying indexing orders.
- More benchmarks are needed for vector database search algorithms

Research Motivations

- Weaviate Podcast with Nils Reimers
  - Created SBERT Sentence Transformers
  - Neils advice on embeddings:
    - **"Beware of Embedding Soup"**
    - Main topics are often "lost in the soup"
    - More words → More tokens → Low recall?
    - Most embeddings have 1 major topic
      - Smaller topics lost in vector space
      - Bias vs. Variance trade-off
      - Bias to a few major topics – learned vector representations of ANN or KNN represent this in search results

Deep Dive into Vector Databases

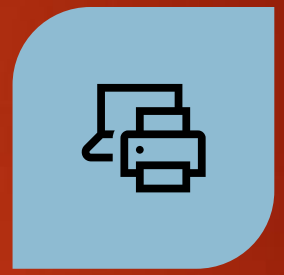
1. General Overview
2. Architecture
3. Indexing
4. How do you select the right vector database for your use case?

1. General Overview

RAG PIPELINE

HISTORICAL TIMELINE

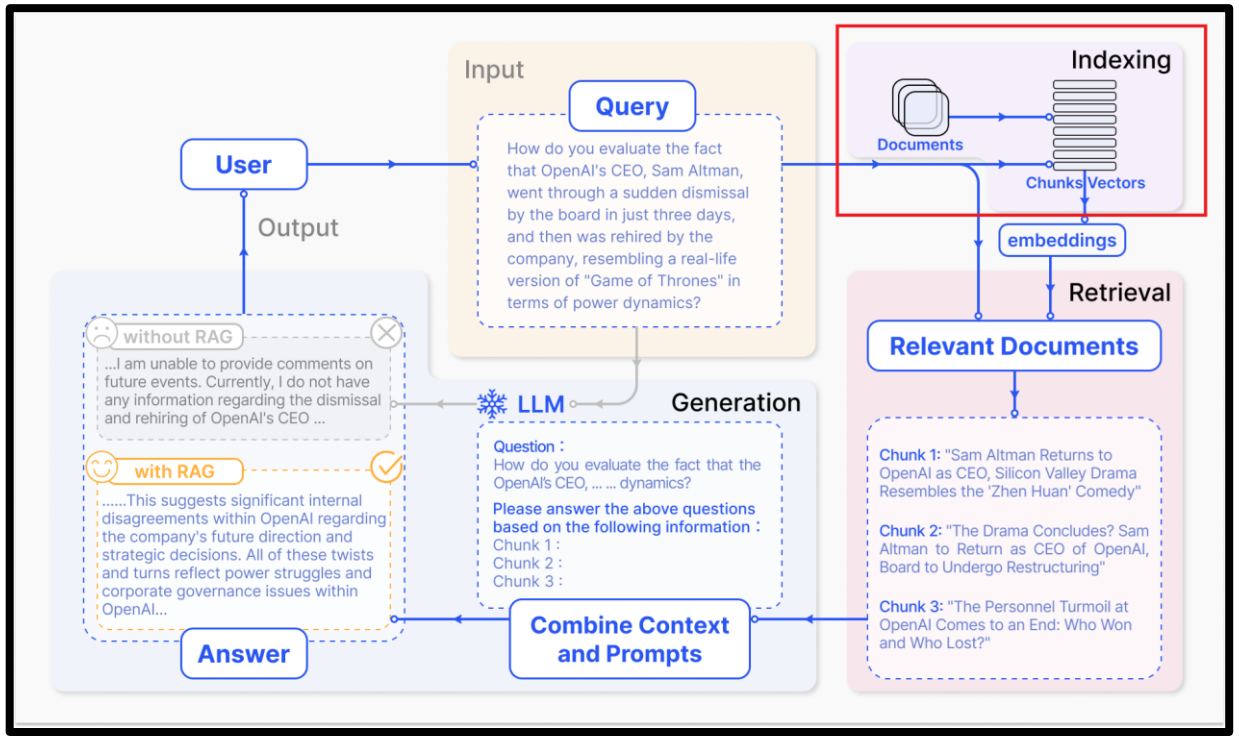
HOSTING METHOD



CLASSES OF VECTOR DBS

“Naïve” RAG Pipeline

- Vector DBs have become a core component of RAG pipelines
  - **Indexing** - chunks stored as vectors in vector database
  - **Retrieval**
    - user queries encoded into vector representation
    - similarity score between query vector + chunks in vector DB
  - **Generation** - synthesized response

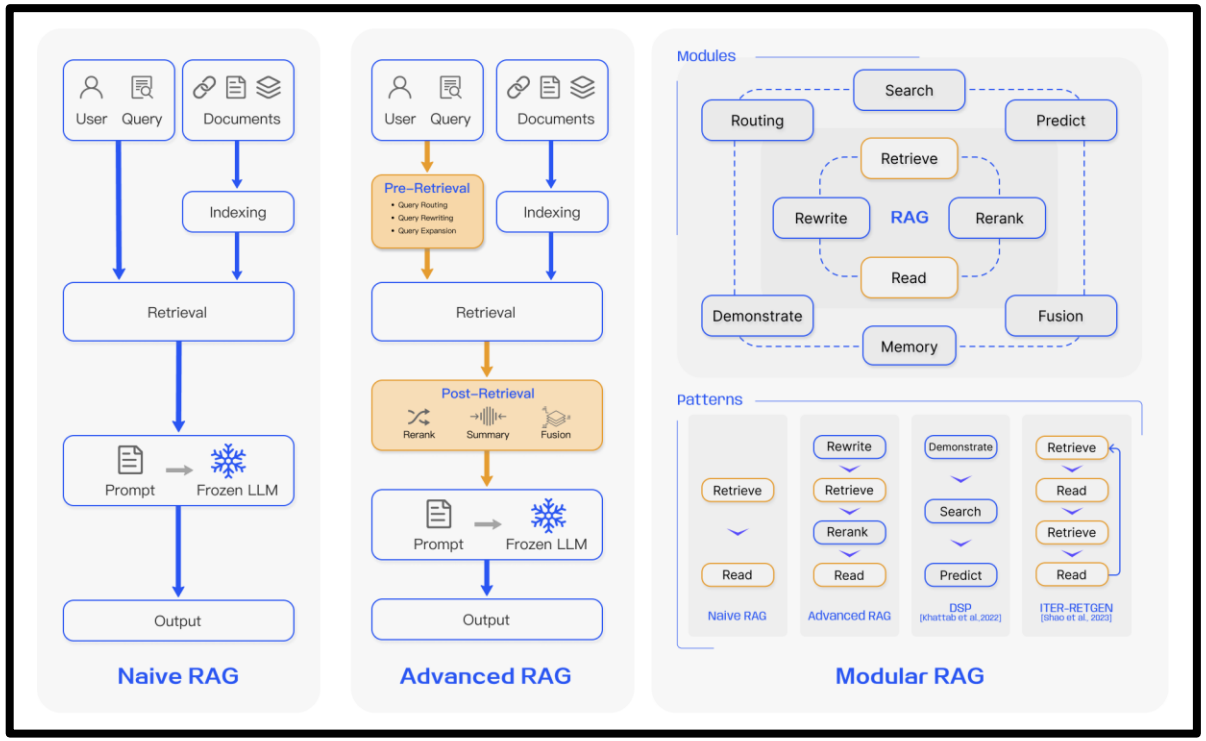


Gao et al. 2024

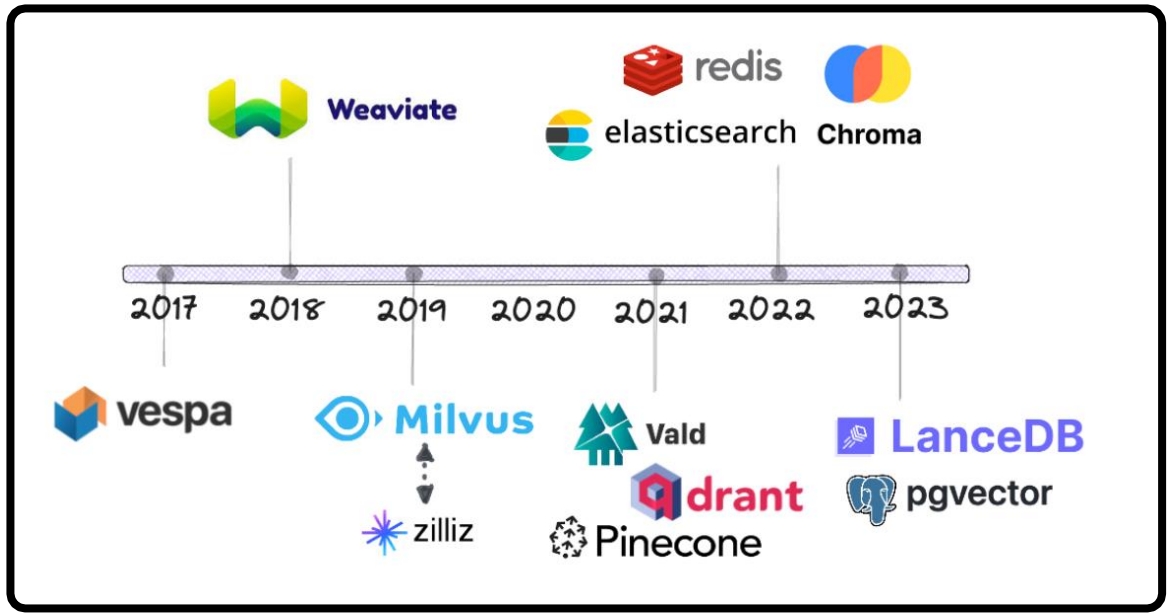
Advanced RAG Pipelines

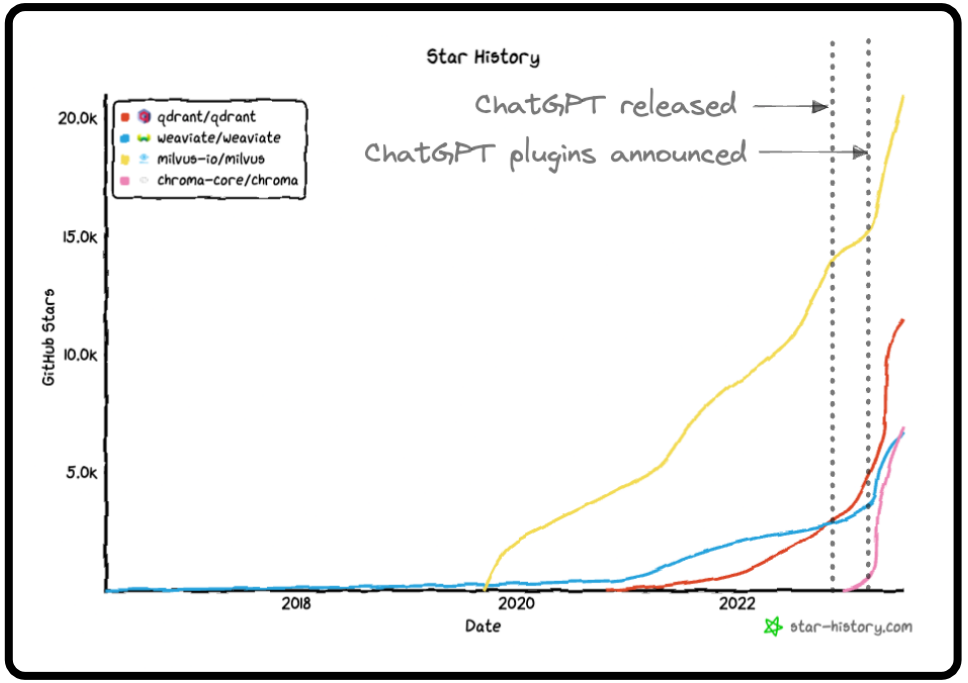
- "Naïve RAG"
- "Advanced RAG"
- "Modular RAG"

- Vector DBs remain vital component of all 3 flavors of RAG.
- *However…..are Vector DBs necessary?*
  - *More on this later…..*



Gao et al. 2024



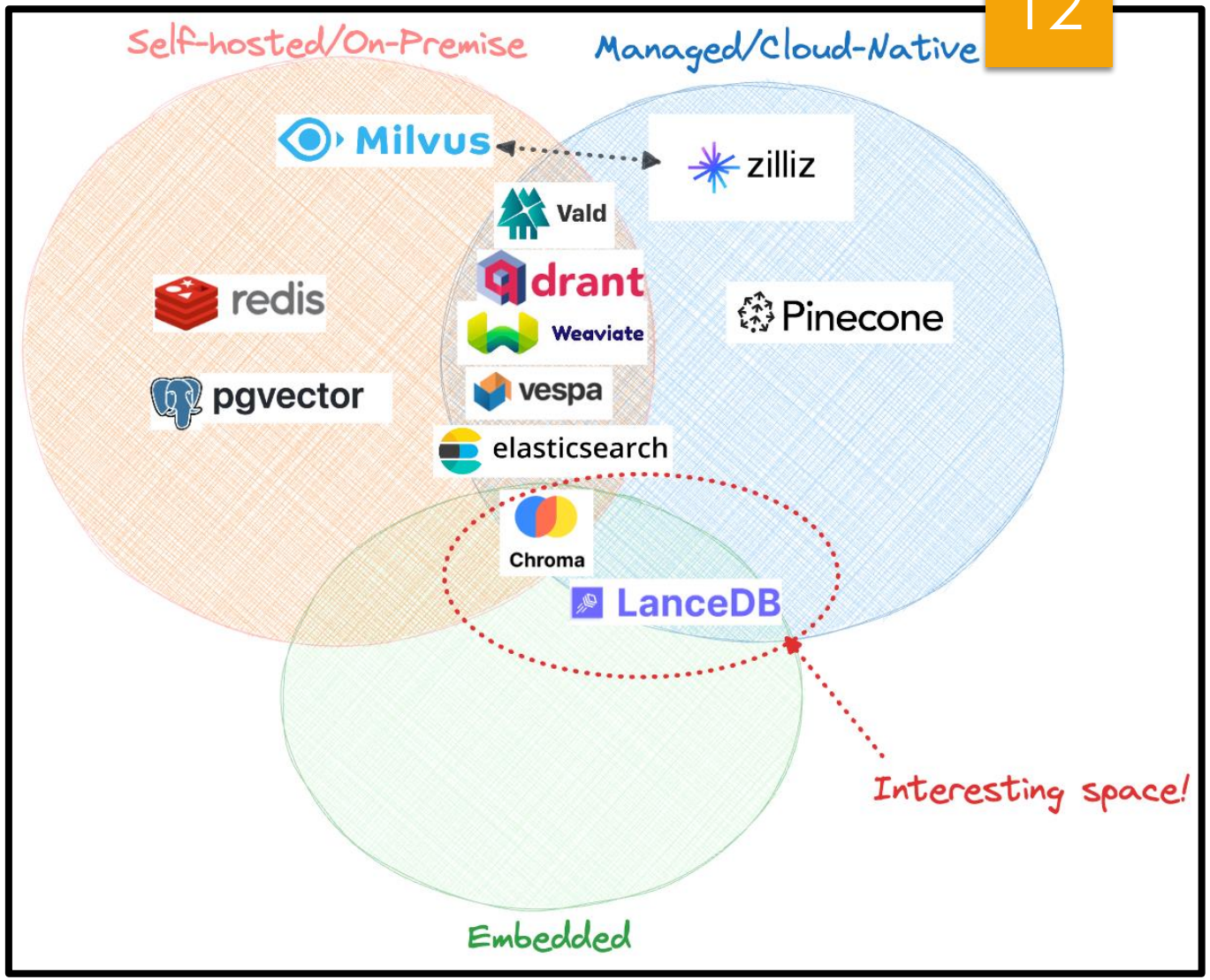


Historical Timeline

SOURCE: THE DATA QUARRY BLOG, 2023

Hosting Method

- Chroma – "all in one"
- LanceDB
  - Embedded
  - Multi-modal
  - Cloud
  - Serverless



Classes of Vector DBs

Vector search libraries

- not databases but in-memory vector indices
- not ideal for enterprise deployment
- nearest neighbor search is fast!
- works well for high dimensional and highly configurable data (e.g. index, params you select)
- **Examples: FAISS, ANNOY**

Lightweight vector databases

- built on search libraries, good for lightweight deployment but poor scalability and performance.
- **Example: Chroma**

Vector search plugins

- "Add-ons" to traditional databases. Often don't scale well and can impact normal performance.
- **Examples: Elasticsearch, Pgvector**

NoSQL databases with vector search

- Add-ons applied to NoSQL databases.
- Good choice if already heavily invested in product.
- **Examples: MongoDB, Redis, Neo4j, Elasticsearch**

SQL databases with vector search

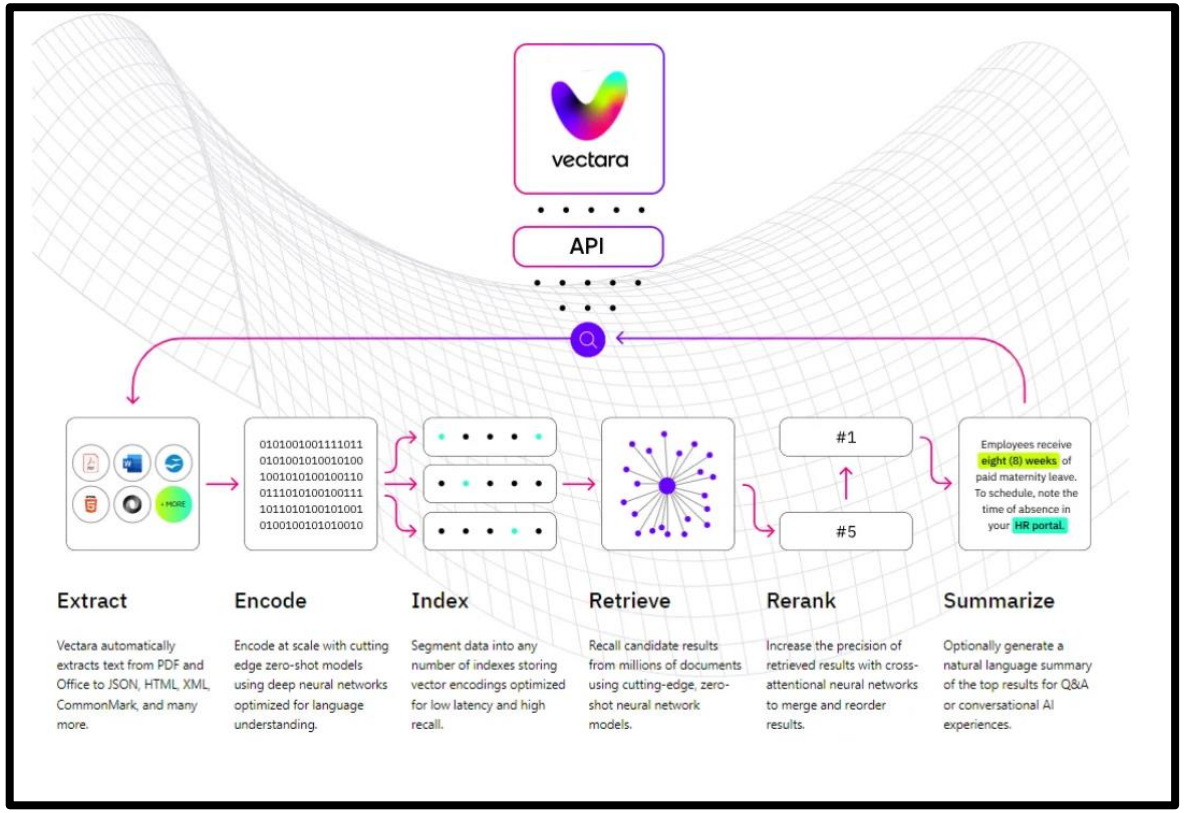
- Similar to above but obviously designed for relational structured data.
- **Examples: PostgreSQL, ClickHouse**

Vector-only databases

- Specific vector databases
- **Example: Weaviate, Pinecone, Milvus**

New Class of Vector DBs – "RAG as a Service"

- End-to-End RAG solutions
  - Generate embeddings
  - Utilize ML and Gen AI solutions all-in-one
- **Examples:**
  - **Vectara (pictured)**
  - **Marqo**



Source: Bussler, 2023

2. Architecture of Vector DBs

- Embeddings
- Similarity Metrics
- Search Algorithms

Embeddings

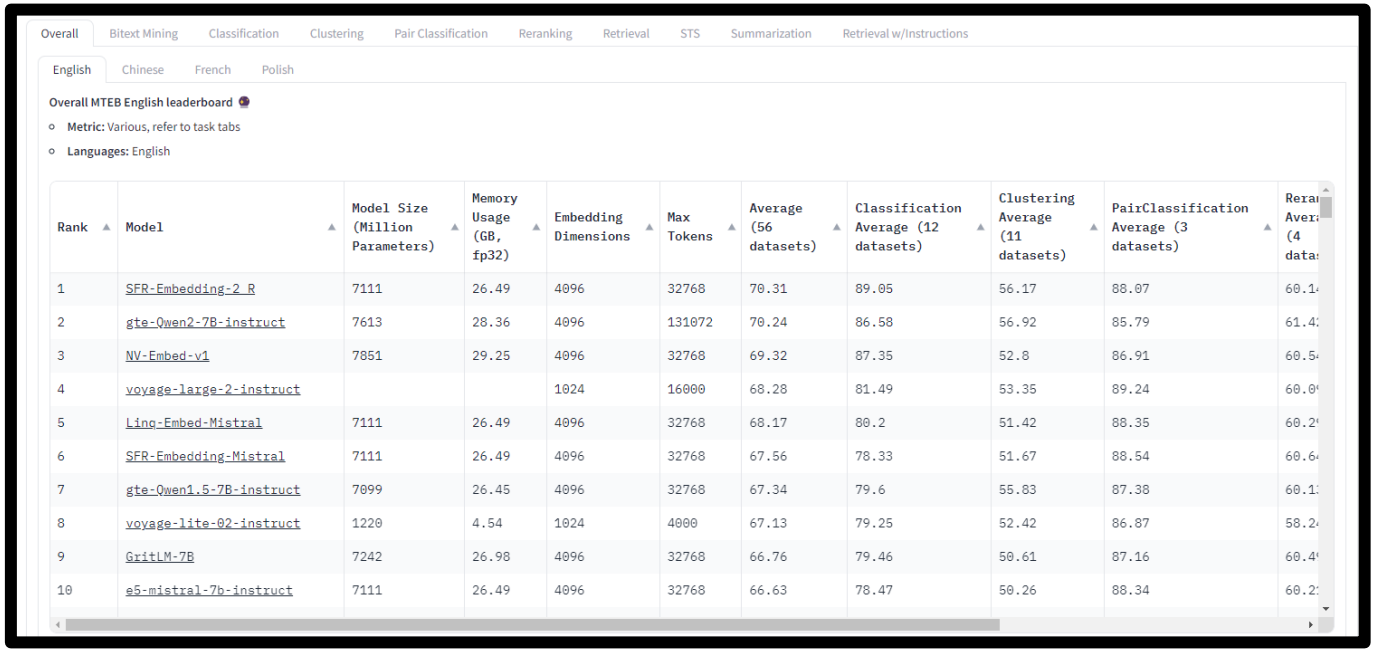
**Note: This could be an entire hour lecture of its own, we will focus on importance to Vector DBs

Benchmarks

- MTEB leaderboard
- BEIR

Evaluate Embeddings

- Tasks
- Score
- Max tokens
- Dimensions
- Model size and Memory Usage
  - "latency-performance trade-off"



Overall | Bitext Mining | Classification | Clustering | Pair Classification | Reranking | Retrieval | STS | Summarization | Retrieval w/Instructions

English | Chinese | French | Polish

Overall MTEB English leaderboard

- Metric: Various, refer to task tabs
- Languages: English

Rank	Model	Model Size (Million Parameters)	Memory Usage (GB, fp32)	Embedding Dimensions	Max Tokens	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)	PairClassification Average (3 datasets)	Rera Aver (4 data
1	SFR-Embedding-2_R	7111	26.49	4096	32768	70.31	89.05	56.17	88.07	60.1
2	gte-Qwen2-7B-instruct	7613	28.36	4096	131072	70.24	86.58	56.92	85.79	61.4
3	NV-Embed-v1	7851	29.25	4096	32768	69.32	87.35	52.8	86.91	60.5
4	voyage-large-2-instruct			1024	16000	68.28	81.49	53.35	89.24	60.0
5	Linq-Embed-Mistral	7111	26.49	4096	32768	68.17	80.2	51.42	88.35	60.2
6	SFR-Embedding-Mistral	7111	26.49	4096	32768	67.56	78.33	51.67	88.54	60.6
7	gte-Qwen1.5-7B-instruct	7099	26.45	4096	32768	67.34	79.6	55.83	87.38	60.1
8	voyage-lite-02-instruct	1220	4.54	1024	4000	67.13	79.25	52.42	86.87	58.2
9	GritLM-7B	7242	26.98	4096	32768	66.76	79.46	50.61	87.16	60.4
10	e5-mistral-7b-instruct	7111	26.49	4096	32768	66.63	78.47	50.26	88.34	60.2

MTEB leaderboard as of 6/18/2024

Embeddings

- Benchmarks are great for "standard tasks" but are "one-dimensional"
- Embeddings for Vector DBs and RAG, consider we use them twice:
  - Offline indexing of data store (vector DB, knowledge base)
  - User queries are embedded for keyword or semantic searches

There are 2 problems with this:

1. Re-indexing and vectorizing existing data when an embedding model is updated or changed.

   a. ** Need to use models that are easy to update and change as needed.

2. Inference latency = number of users

   a. More users will lead to SLOW inference latency.

   b. Latency of more than 100ms is common for models with 1B params or more.

   c. Consider tradeoff between quality vs. latency.

Embedding Dimensionality Analysis

1. Intrinsic Dimensionality Analysis
   a. Minimum number of dimensions to represent data
   b. Higher intrinsic dim → Worse recall in search
   c. Example:
      - Using large pre-trained embedding model on diverse dataset.
      - If only used for small cohort of tasks/topics → lower intrinsic dimensionality

2. Algorithms: PCA, tSNE, UMAP, NMF
3. Topic Modeling
   a. BERTopic
   b. LDA
   c. Top2Vec
4. UMAP with k-means clustering (Zhang et al. 2022)

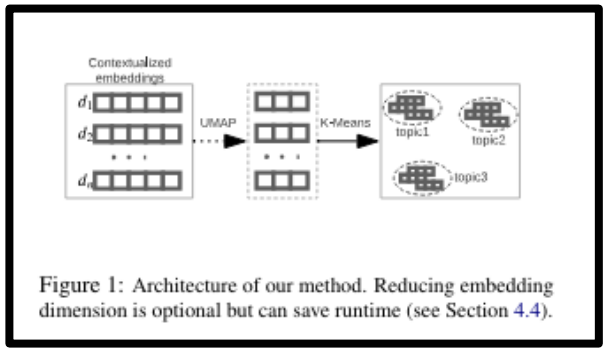


Figure 1: Architecture of our method. Reducing embedding dimension is optional but can save runtime (see Section 4.4).

Intrinsic vs. Extrinsic Embedding Evaluations

Intrinsic Evaluation

- Measures quality by assessing performance on specific NLP tasks related to vector space.
  - Word similarity
  - Analogy
  - Classification
- Metrics for intrinsic evaluation
  - Cosine Similarity
  - Spearman Correlation
  - Accuracy
  - Vec2best Python Library (Principal Component Evaluation)

- Problems:
  - Eval tasks may not represent real-world
  - Metrics are not always correct

Extrinsic Evaluation

- Measures quality by assessing downstream tasks NOT related to embeddings.
  - Machine translation
  - Text classification
- Metrics for extrinsic evaluation
  - F1 score
  - Perplexity

- Problems:
  - Embeddings alone CAN NOT predict downstream performance of LLM-RAG applications

Scalability of Embeddings = Problem!

- **Embeddings often struggle to scale**
  - **More users → Slower inference latency time**
  - **Higher Dimensions → Lower Recall (Elliot et al. 2024)**
- Many "state of the art embeddings" with 1024 dimensions (float32)
  - **Requires 4 bytes per dimension**
- Perform Retrieval/Search over 250 million vectors
  - **Requires 1 TB memory!!!!**

➢ Costs are computed at an estimated $3.8 per GB/mo with x2gd instances on AWS

Embedding Dimension	Example Models	100M Embeddings	250M Embeddings	1B Embeddings
384	all-MiniLM-L6-v2 bge-small-en-v1.5	143.05GB $543 / mo	357.62GB $1,358 / mo	1430.51GB $5,435 / mo
768	all-mpnet-base-v2 bge-base-en-v1.5 jina-embeddings-v2-base-en nomic-embed-text-v1	286.10GB $1,087 / mo	715.26GB $2,717 / mo	2861.02GB $10,871 / mo
1024	bge-large-en-v1.5 mxbai-embed-large-v1 Cohere-embed-english-v3.0	381.46GB $1,449 / mo	953.67GB $3,623 / mo	3814.69GB $14,495 / mo
1536	OpenAI text-embedding-3-small	572.20GB $2,174 / mo	1430.51GB $5,435 / mo	5722.04GB $21,743 / mo
3072	OpenAI text-embedding-3-large	1144.40GB $4,348 / mo	2861.02GB $10,871 / mo	11444.09GB $43,487 / mo

Source: Shakir et al. 2024

Scalability Solutions

- PCA
  - "classic" – but information often lost
- Matryoshka embeddings
  - "cheaper embeddings"
  - OpenAI text-embedding-3-large → 93.1% retention at 12x compression
- Binary Quantization
  - float32 → 1-bit → 32x reduction MEMORY and STORAGE
- Scalar (int8) Quantization
  - float32 → int8 → bits to bytes
- Combined Binary + Scalar Quantization

Source: Shakir et al. 2024

	float32	int8/uint8	binary/ubinary
Memory & Index size savings	1x	exactly 4x	exactly 32x
Retrieval Speed	1x	up to 4x	up to 45x
Percentage of default performance	100%	~99.3%	~96%

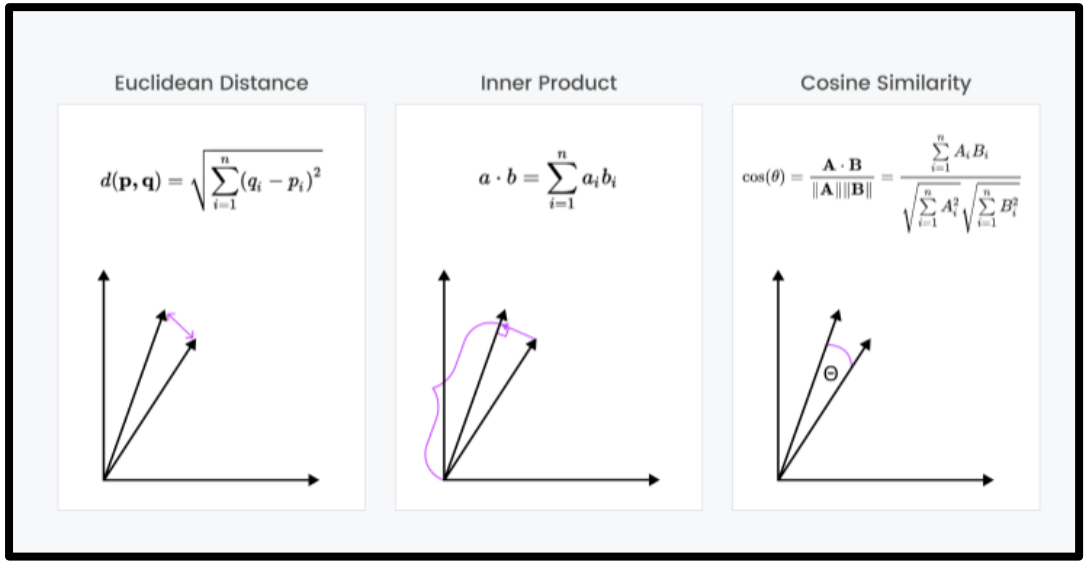
Feature	Float32	Binary Quantization	Scalar Quantization (int8)
Value Representation	32-bit floating-point	1-bit (0 or 1)	8-bit integer (-128 to 127)
Memory Reduction	Baseline (1x)	32x	4x
Retrieval Speed	Moderate	Fastest (due to simple bit comparisons)	Faster than float32
Accuracy	Highest	Lower (may require rescoring for improvement)	Higher than binary, but may be lower than float32
Implementation Complexity	Simplest	Simpler	More complex (requires calibration dataset)

Source: George, 2024

Similarity Metrics/Distance Metrics in Vector DBs

- Cosine Similarity
- Dot Product
- Squared Euclidean distance
- Euclidean distance (L2)
- Manhattan distance
- Minkowski distance
- Hamming distance
- Jaccard distance
- Inner product space
- Angular distance

- **Rule of thumb: Use distance metric that matches model(s) you are using!**
  (Cardenas, 2023)
- Metrics vary by Vector DB
  - Often can be changed at query time.

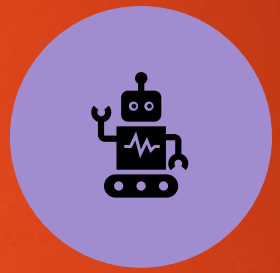


Source: Tang, 2023

3. Indexing in Vector DBs

DATA STRUCTURES

COMPRESSION

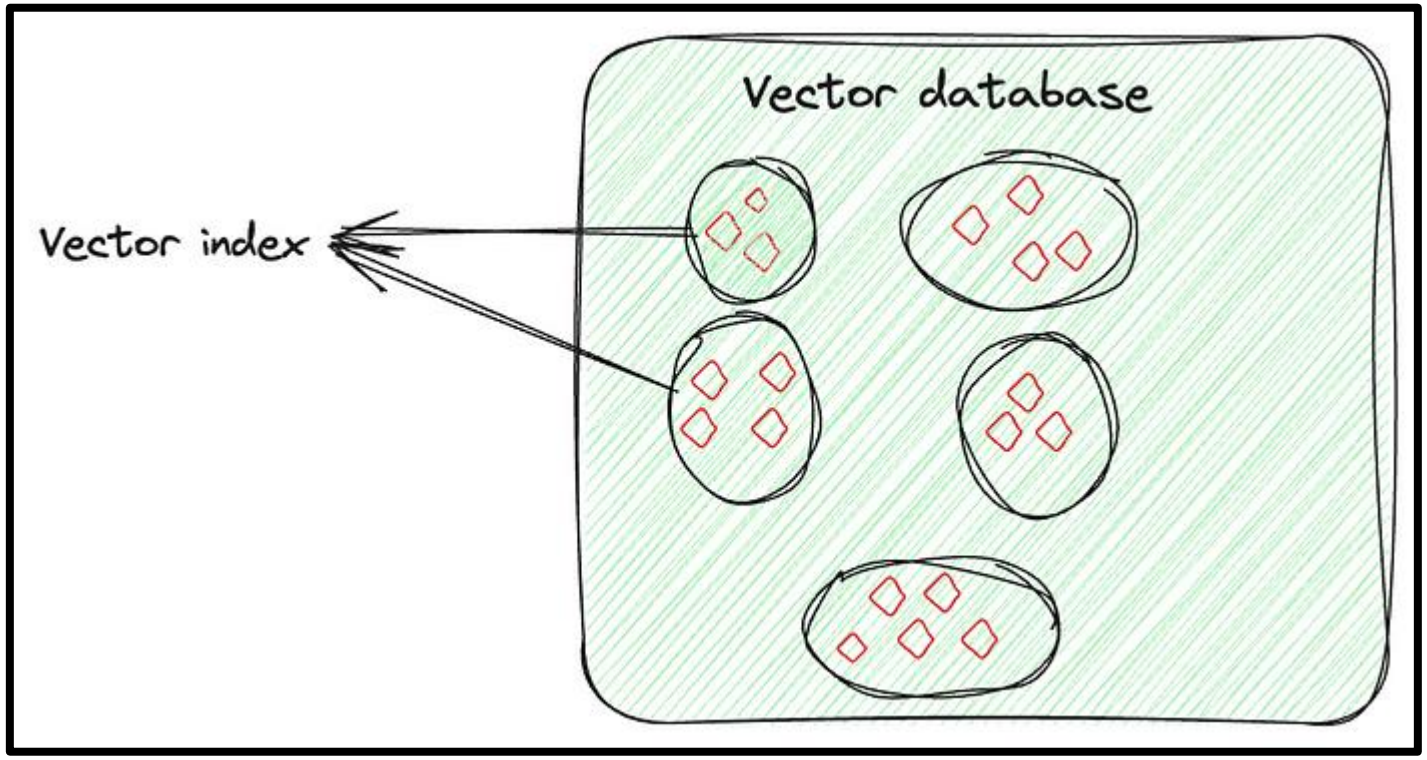


ALGORITHMS

VENDOR SPECIFIC

Indexing

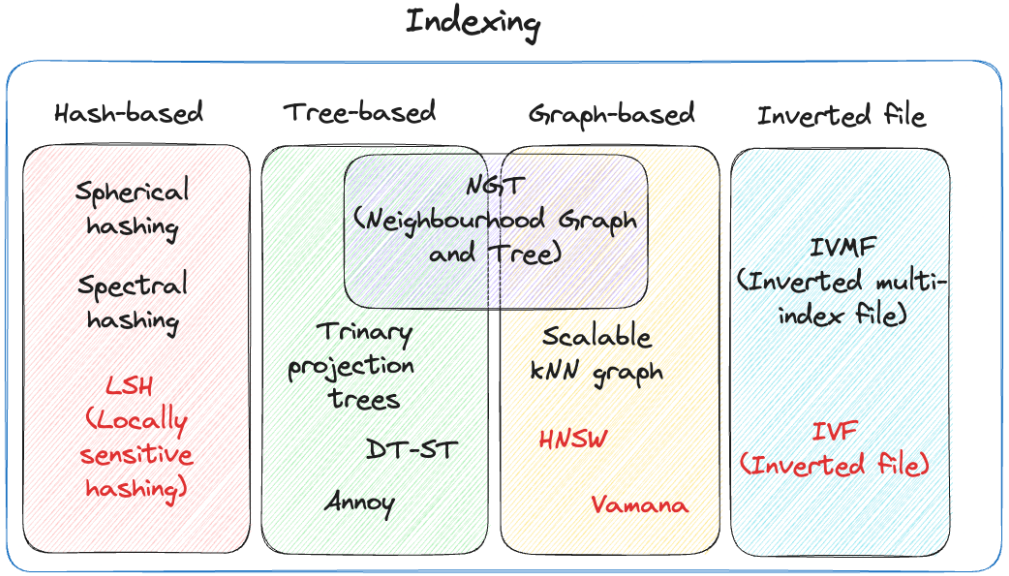
- Vector indexing optimizes the search and retrieval process.
- High-dimensional vectors are grouped together in semantically similar clusters.



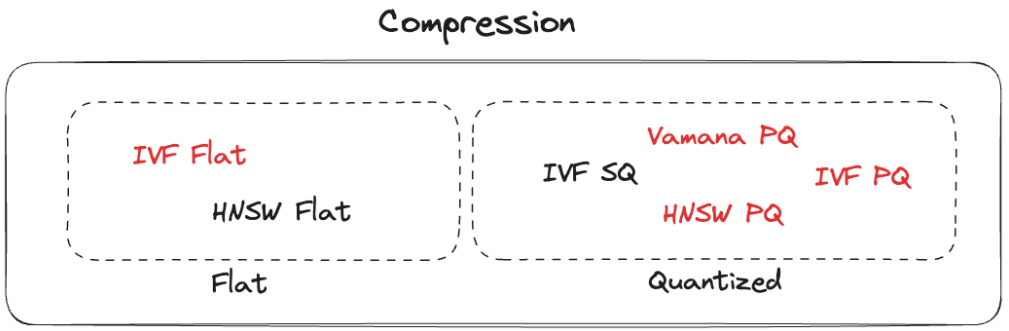
Source: MyScale, 2024

Indexing – Classification

Data Structures



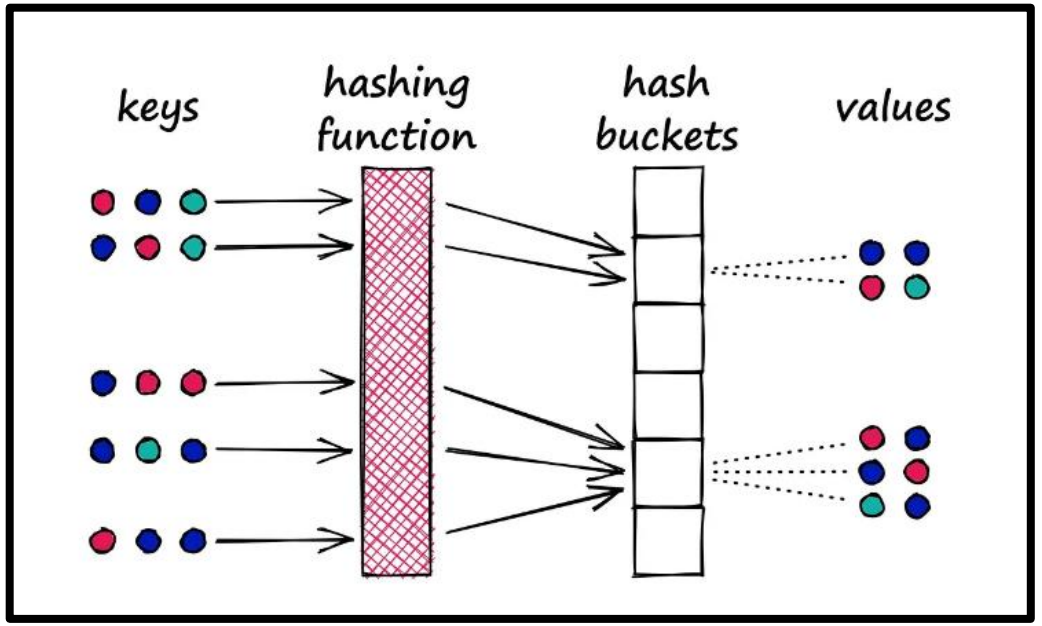
Compression



Source: The Data Quarry Blog, 2023

LSH (Locally Sensitive Hashing)

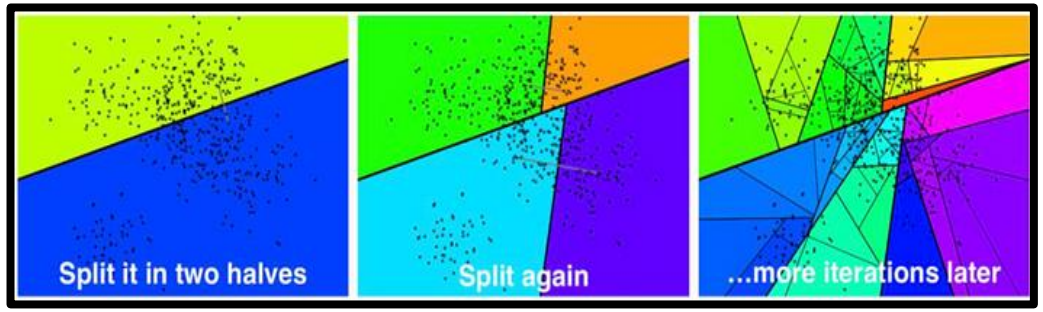
- Hash-Based
- **Algorithm**
  - Transforms high-dim data → Low-dim data
  - Hashes similar vectors to same buckets
- **Use-case**
  - near-duplicate detection
- **Advantages**
  - Approximate similarity search
- **Disadvantages**
  - Requires specific parameter tuning



Sources: Mehta, 2023; Taipalus, 2024

ANNOY (Approximate Nearest Neighbors Oh Yeah)

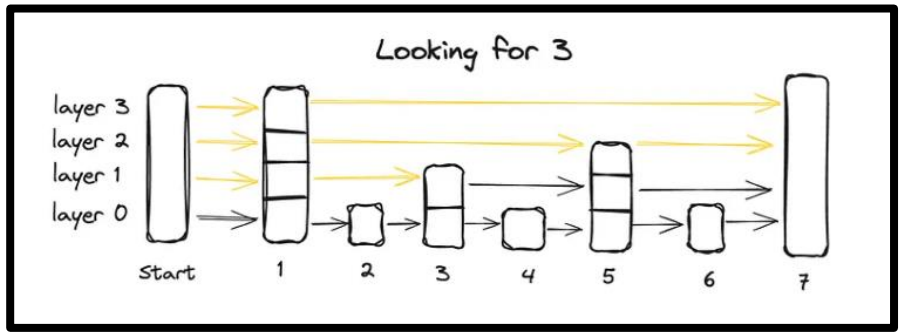
- Tree-based
  - **R-trees:** hierarchical structure with bounding boxes
  - **KD trees:** binary partitions around median point
- **Algorithm**
  - Binary search trees
  - Sub-trees constructed into semantically similar vectors
- **Use-Cases**
  - **R trees**: spatial data (geospatial indexing)
  - **KD trees**: machine learning, clustering
- **Advantages**
  - **R trees:** efficient range queries, updates
  - **KD trees**: balanced tree structure, good for low dimensions
- **Disadvantages**
  - **R trees:** slower nearest neighbor search
  - **KD trees:** Inefficient in high dimensions, complex to build



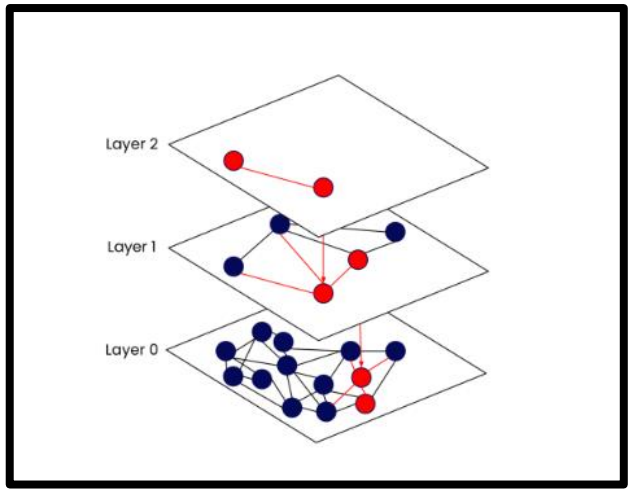
Source: Mehta, 2023; Taipalus, 2024

HNSW (Hierarchical Navigable Small Worlds)

- Graph-based
- **Algorithm**
  - Skip list: linked list quick insertion + array rapid retrieval
  - NSW – similar nodes, edges
  - Hierarchical graph → Searches shortest distance between nodes in each layer
- **Use-Cases**
  - Recommendation systems, text search
- **Advantages**
  - fast "neighborhood" searches
- **Disadvantages**
  - Complex index structure



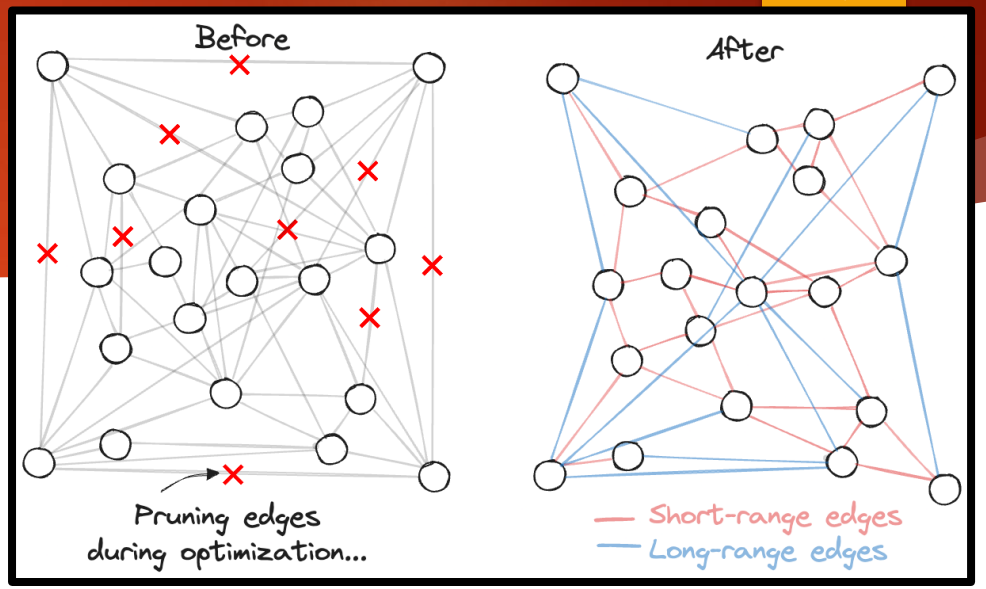
Source: MyScale, 2024

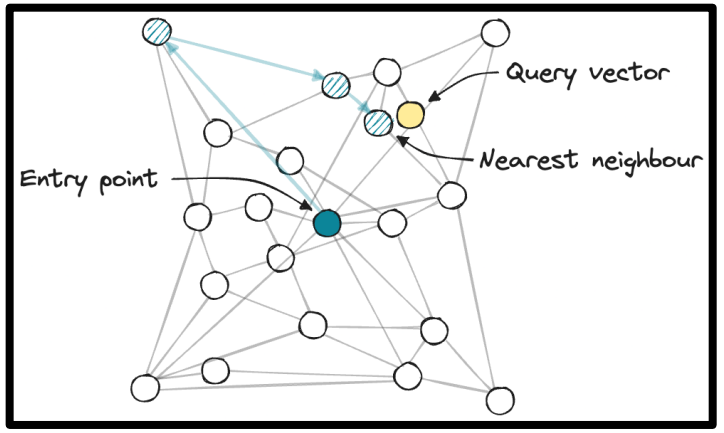


Source: Elliot, 2024; Taipalus, 2024

Vamana (DiskANN)

- Graph-based (newest) – Subramanya et al. 2019
- **Algorithm**
  - Greedy search, robust prune
  - Directed relative neighborhood graph (RNG)
- **Use-Cases**
  - Same as HNSW
- **Advantages**
  - **High-recall, low query latency, high density in points indexed per node**
  - **Inside-out search (vs. outside-in for HNSW)**
- **Disadvantages**
  - **Algorithmically expense: O(n2) for dataset with size n.**
  - **High memory cost can make in RAM use cases expensive and on disk it can have high update overheads.**

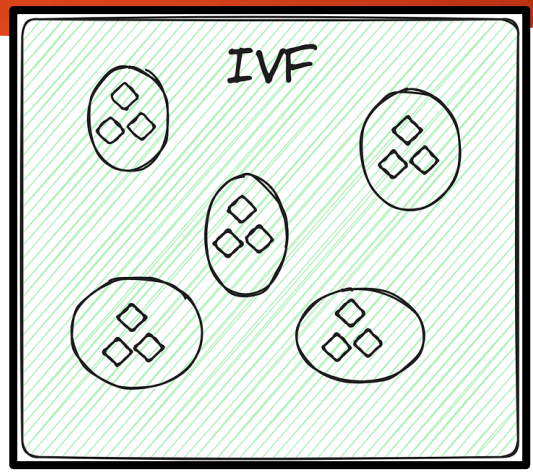


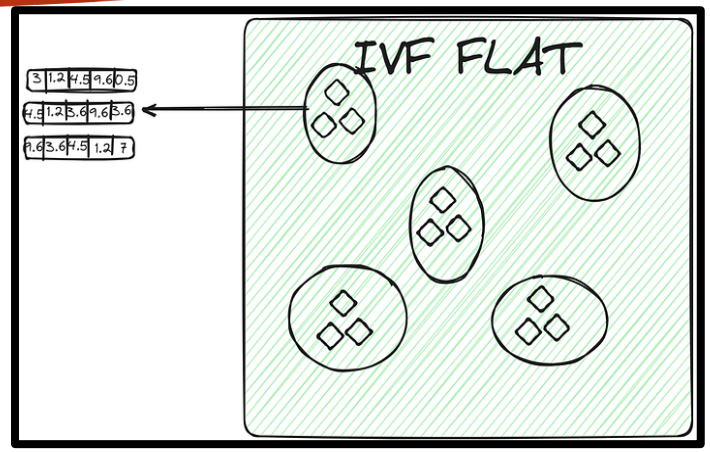


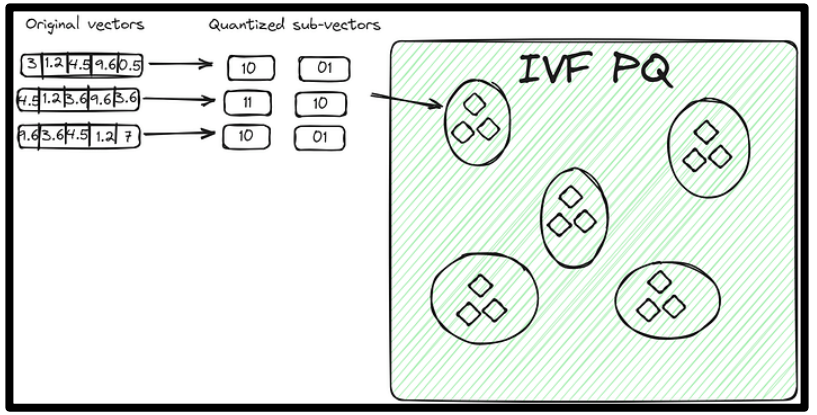
Source: The Data Quarry Blog, 2023

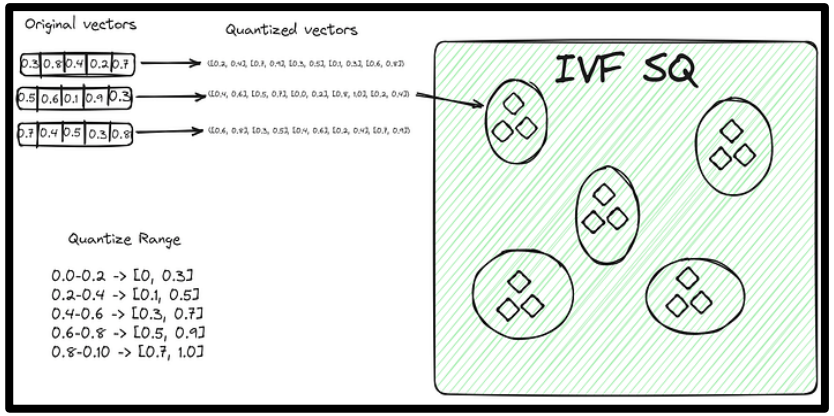
IVF and variants: IVFLAT, IVFPQ, IVFSQ

- Inverted File
- **Algorithms**
  - Simplest → splits data into several clusters similar to K-means clustering
- **Use-Cases**
  - Image search, text search
- **Advantages**
  - Reduces dimensionality
- **Disadvantages**
  - Lossy compression may reduce accuracy

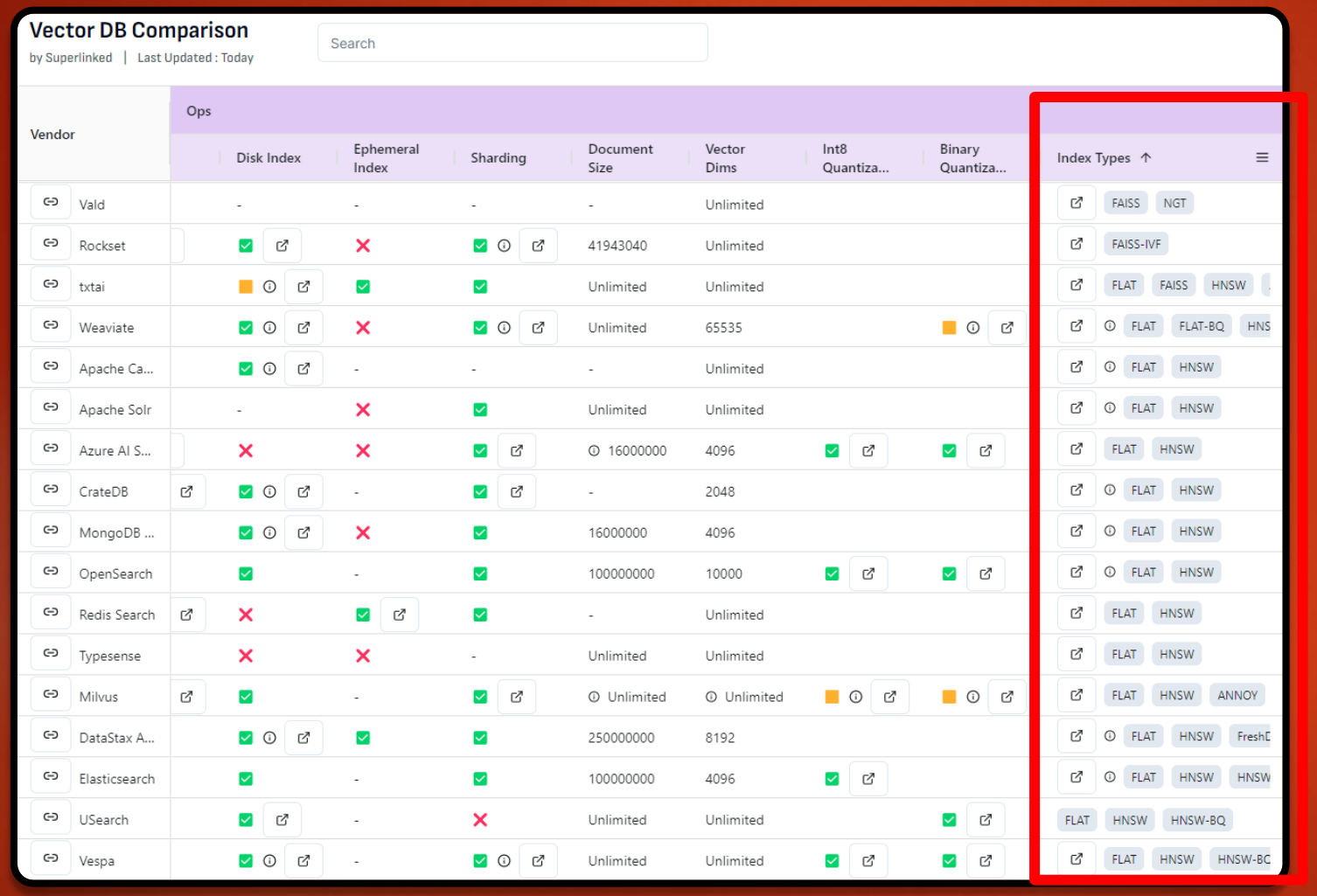








Vendor Indexing Algorithms



Vector DB Comparison

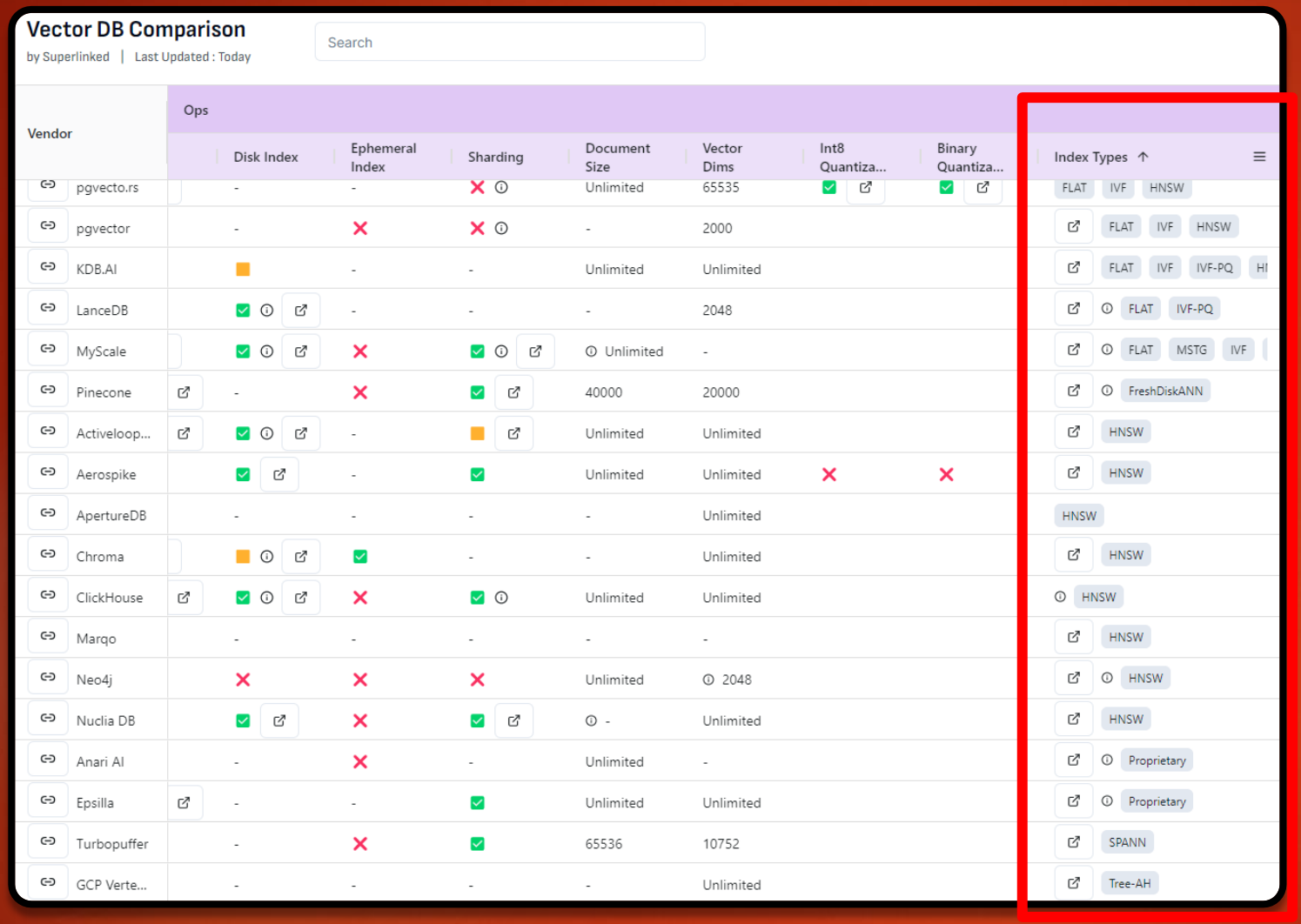
by Superlinked | Last Updated : Today

Search

Vendor	Ops							
	Disk Index	Ephemeral Index	Sharding	Document Size	Vector Dims	Int8 Quantiza...	Binary Quantiza...	Index Types
Vald	-	-	-	-	Unlimited			FAISS NGT
Rockset	✓	✗	✓	41943040	Unlimited			FAISS-IVF
txtai	■	✓	✓	Unlimited	Unlimited			FLAT FAISS HNSW
Weaviate	✓	✗	✓	Unlimited	65535		■	FLAT FLAT-BQ HNS
Apache Ca...	✓	-	-	-	Unlimited			FLAT HNSW
Apache Solr	-	✗	✓	Unlimited	Unlimited			FLAT HNSW
Azure AI S...	✗	✗	✓	16000000	4096	✓	✓	FLAT HNSW
CrateDB	✓	-	✓	-	2048			FLAT HNSW
MongoDB ...	✓	✗	✓	16000000	4096			FLAT HNSW
OpenSearch	✓	-	✓	100000000	10000	✓	✓	FLAT HNSW
Redis Search	✗	✓	✓	-	Unlimited			FLAT HNSW
Typesense	✗	✗	-	Unlimited	Unlimited			FLAT HNSW
Milvus	✓	-	✓	Unlimited	Unlimited	■	■	FLAT HNSW ANNOY
DataStax A...	✓	✓	✓	250000000	8192			FLAT HNSW FreshD
Elasticsearch	✓	-	✓	100000000	4096	✓		FLAT HNSW HNSW
USearch	✓	-	✗	Unlimited	Unlimited		✓	FLAT HNSW HNSW-BQ
Vespa	✓	-	✓	Unlimited	Unlimited	✓	✓	FLAT HNSW HNSW-BC

Source: Superlinked, 2024

Vendor Indexing Algorithms



Vector DB Comparison

by Superlinked | Last Updated : Today

Search

Vendor	Ops							
	Disk Index	Ephemeral Index	Sharding	Document Size	Vector Dims	Int8 Quantiza...	Binary Quantiza...	Index Types
pgvecto.rs	-	-	✗	Unlimited	65535	✓	✓	FLAT IVF HNSW
pgvector	-	✗	✗	-	2000			FLAT IVF HNSW
KDB.AI	■	-	-	Unlimited	Unlimited			FLAT IVF IVF-PQ HI
LanceDB	✓	-	-	-	2048			FLAT IVF-PQ
MyScale	✓	✗	✓	Unlimited	-			FLAT MSTG IVF
Pinecone	-	✗	✓	40000	20000			FreshDiskANN
Activeloop...	✓	-	■	Unlimited	Unlimited			HNSW
Aerospike	✓	-	✓	Unlimited	Unlimited	✗	✗	HNSW
ApertureDB	-	-	-	-	Unlimited			HNSW
Chroma	■	✓	-	-	Unlimited			HNSW
ClickHouse	✓	✗	✓	Unlimited	Unlimited			HNSW
Marqo	-	-	-	-	-			HNSW
Neo4j	✗	✗	✗	Unlimited	2048			HNSW
Nuclia DB	✓	✗	✓	-	Unlimited			HNSW
Anari AI	-	✗	-	Unlimited	-			Proprietary
Epsilla	-	-	✓	Unlimited	Unlimited			Proprietary
Turbopuffer	-	✗	✓	65536	10752			SPANN
GCP Verte...	-	-	-	-	Unlimited			Tree-AH

Source: Superlinked, 2024

4. How to choose the right Vector DB for your use case?

BENCHMARKS

SEARCH GOAL(S)

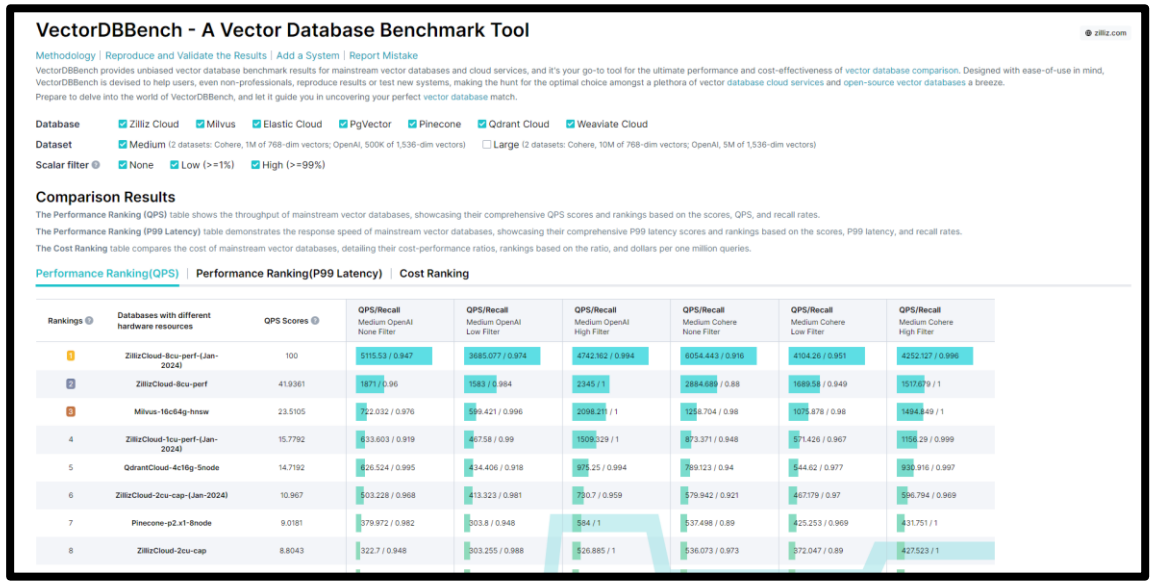
FUNCTION VS. PERFORMANCE "TRADE-OFF"

LOCATION

DATABASE TYPE

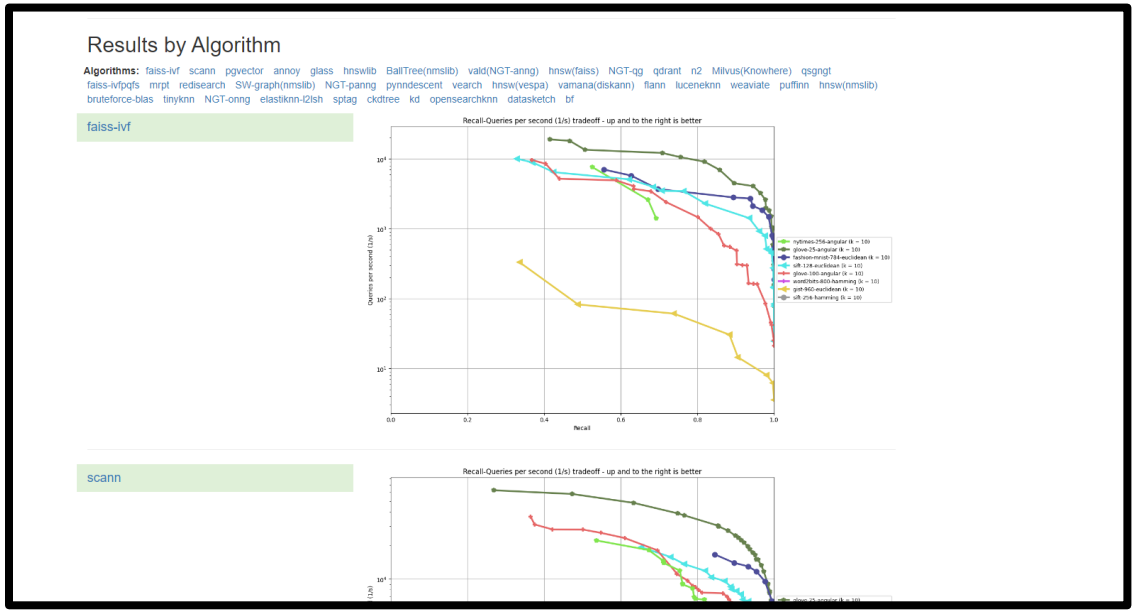
Vector Database Benchmarks

- **VectorDBBench**
  - Open-source benchmarking tool created at Zilliz
  - Link: VectorDBBench homepage
  - Designed for open-source vector databases like Milvus and Weaviate and fully-managed services like Zilliz Cloud and Pinecone.
  - Many fully managed vector search services do not expose their parameters for user tuning,
    - VectorDBBench displays QPS and recall rates separately.



Vector Database Benchmarks

- ANN-Benchmark
  - Link: ANN-Benchmark
  - Evaluates vector search algorithms on benchmark datasets.
  - Graphs results of recall/queries per second of various algorithms based on any number of precomputed datasets.
  - Plots recall rate on the x-axis against QPS on the y-axis, illustrating each algorithm's performance at different levels of retrieval accuracy.



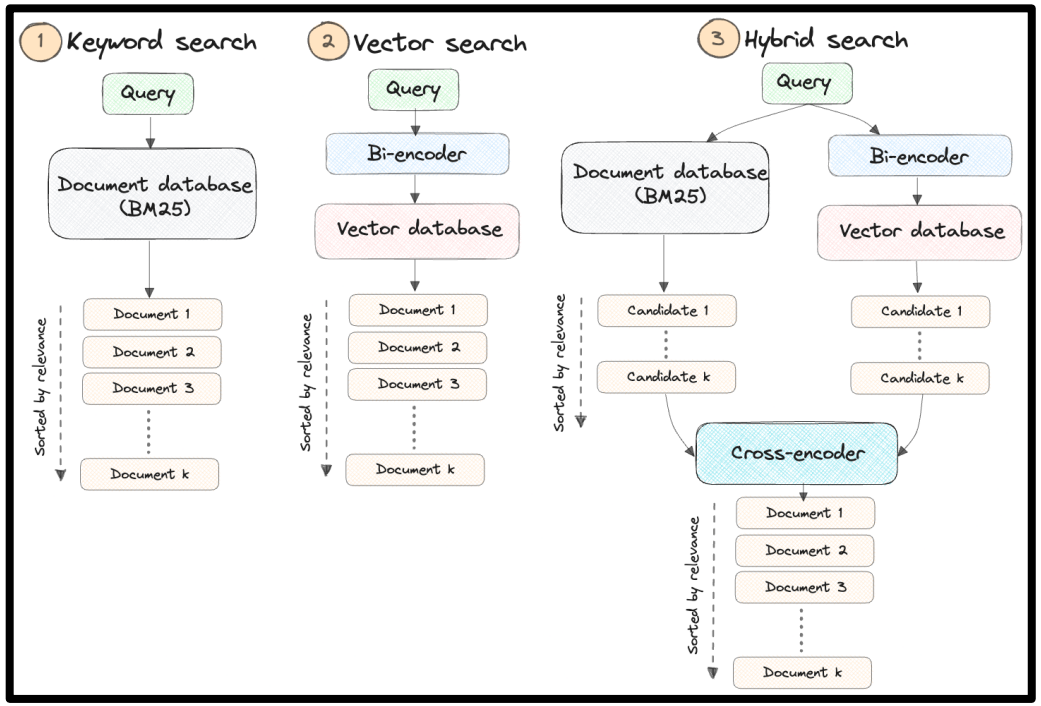
➢ Disadvantages:

- Only compares vector search algorithms
- **Does not evaluate: resource consumption, data loading capacity, and system stability.**
- ANN-Benchmark misses many common scenarios, such as filtered vector searching.

Vector Database Benchmarks

- **Vector DB Comparison**
  - **Link:** [Vector DB Comparison]
  - Developed by Superlinked

What are your search goal(s)?



- Keyword Search
  - BM25 algorithm is best!
- Vector Search
  - Bi-encoders, vector search
- Hybrid Search
  - Sparse + Dense Vector search

RAG Evaluation Metrics for Vector DBs

- NDCG (normalized discounted cumulative gain)
- Precision
- Recall
- F1 score
- Cosine Similarity
- Mean Reciprocal Rank (MRR)
- Mean Average Precision (MAP)
- Hit Rate
- …etc…..

Source: Gao et al. 2024

TABLE III
SUMMARY OF METRICS APPLICABLE FOR EVALUATION ASPECTS OF RAG

	Context Relevance	Faithfulness	Answer Relevance	Noise Robustness	Negative Rejection	Information Integration	Counterfactual Robustness
Accuracy	✓	✓	✓	✓	✓	✓	✓
EM					✓		
Recall	✓						
Precision	✓			✓			
R-Rate							✓
Cosine Similarity			✓				
Hit Rate	✓						
MRR	✓						
NDCG	✓						
BLEU	✓	✓	✓				
ROUGE/ROUGE-L	✓	✓	✓				

TABLE IV
SUMMARY OF EVALUATION FRAMEWORKS

Evaluation Framework	Evaluation Targets	Evaluation Aspects	Quantitative Metrics
RGB†	Retrieval Quality Generation Quality	Noise Robustness Negative Rejection Information Integration Counterfactual Robustness	Accuracy EM Accuracy Accuracy
RECALL†	Generation Quality	Counterfactual Robustness	R-Rate (Reappearance Rate)
RAGAS‡	Retrieval Quality Generation Quality	Context Relevance Faithfulness Answer Relevance	* * Cosine Similarity
ARES‡	Retrieval Quality Generation Quality	Context Relevance Faithfulness Answer Relevance	Accuracy Accuracy Accuracy
TruLens‡	Retrieval Quality Generation Quality	Context Relevance Faithfulness Answer Relevance	* * *
CRUD†	Retrieval Quality Generation Quality	Creative Generation Knowledge-intensive QA Error Correction Summarization	BLEU ROUGE-L BertScore RAGQuestEval

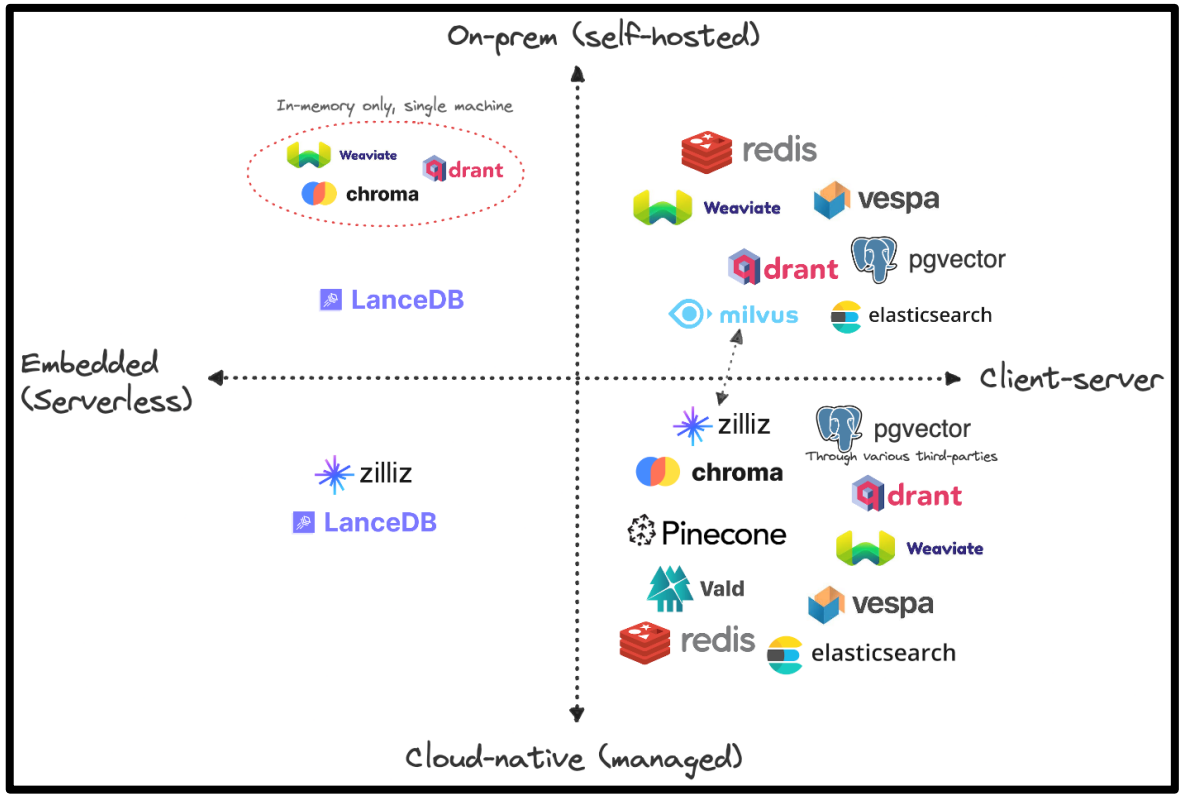
*† represents a benchmark, and ‡ represents a tool. * denotes customized quantitative metrics, which deviate from traditional metrics. Readers are encouraged to consult pertinent literature for the specific quantification formulas associated with these metrics, as required.*

Function vs. Performance "Trade-off"

- Insertion Speed
  - Insert and index 1,000+ vectors/minute
  - Ideal for real-time/streaming apps (e.g. video, financial, etc.)
  - **Leader:** Milvus/Zillis
- Query Speed (QPS)
  - A vector database's capability to handle concurrent queries per second. Higher QPS values indicate better vector database performance.
  - **Leader:** ZillizCloud-8cu-perf
- Recall
  - Vector search accuracy of a vector database. Higher recall rates correspond to more accurate vector search results.
  - **Leader:** Zilliz/Milvus
- Latency
  - Comprehensive scoring results demonstrating a vector database's vector retrieval speed.
  - **Leader:** Zilliz/Milvus

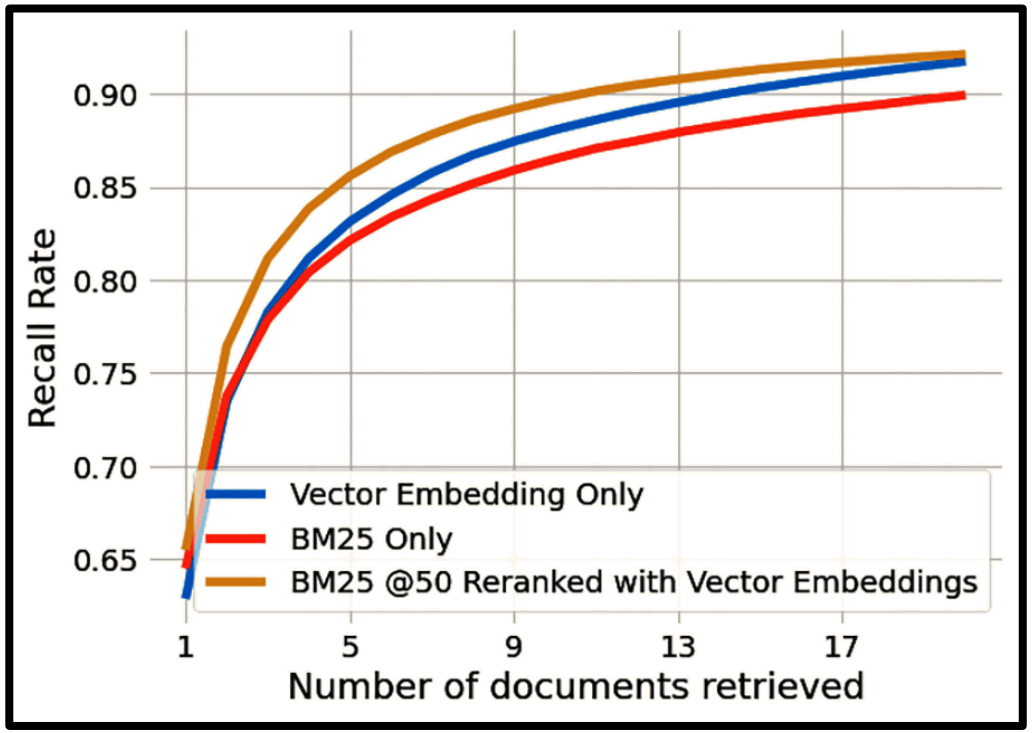
Location

- Cloud-native (managed) + client-server
- On-prem (self-hosted) + embedded
- Cloud-native (managed) + embedded



Do you really need a Vector DB?

- Article in December 2023 by Yucheng Low from xethub: *"You don't need a vector database for RAG."*
- **Low's argument:** "The task of finding a small subset of relevant documents to answer a question is a well studied area of information retrieval (IR), a field of computer science. **The solutions here predate vector databases."**
- Experiment
  - OpenAI Vector Embeddings ONLY
  - BM25 ONLY
  - BM25 → re-rank with vector embeddings



Source: Low, 2023

- Conclusion:
  - **Without a Vector DB, you can use Hybrid search with rerank to obtain a high percentage of documents.**

Hybrid Search: Sparse vs. Dense Vectors

	Sparse Vector Search	Dense Vector Search
Core principle	Keyword based Search	Semantic Search
Applicable for	Fully /Semi Structured data	Unstructured data
Preferable with	Advance Search Filters	Natural language Interface
Best suited to capture	Explicit context in the query	Implicit context in the query
Frameworks	ElasticSearch, Solr, Pinecone etc	FAISS, Vespa, Solr, Pinecone etc
	Hybrid Search	
Core principle	Semantic Search	
Applicable for	Fully / Semi Structured as well as Unstructured data	
Preferable with	Natural language interface alongside Advanced search filters	
Best suited to capture	Both Explicit and implicit context in the query	
Frameworks	Solr, Pinecone etc	

Source: Vm, 2023

Sparse

Pros	Cons
+ Typically faster retrieval	• Performance cannot be improved significantly over baseline
+ Good baseline performance	• Performance cannot be improved significantly over baseline
+ Don't need model fine-tuning	• Suffers from vocabulary mismatch problem
+ Exact matching of terms	

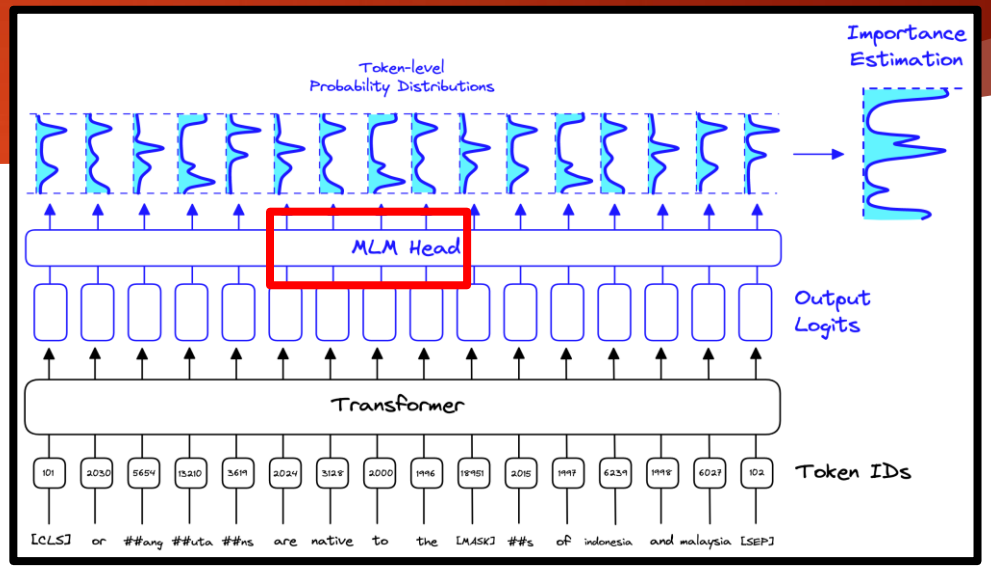
Dense

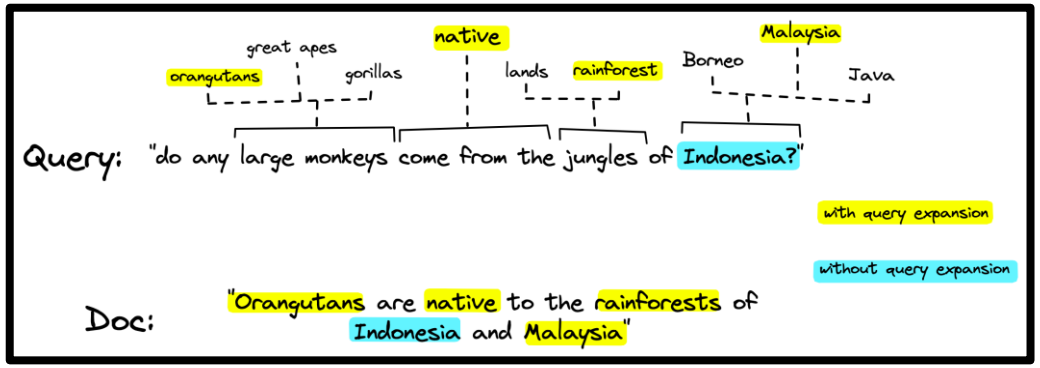
Pros	Cons
+ Can outperform sparse with fine-tuning	• Requires training data, difficult to do in low-resource scenarios
+ Search with human-like abstract concepts	• Does not generalize well, particularly for niche terminology
+ Multi-modality (text, images, audio, etc.) and cross-modal search (e.g., text-to-image)	• Requires more compute and memory than sparse
	• No exact match
	• Not easily interpretable

Source: Briggs, 2023

SPLADE (Sparse Vectors)

- How can we more efficiently leverage 1 stage retrieval?
- **Sp**arse **L**exical **and** **E**xpansion models utilize a pretrained language model like BERT
- SPLADE can do the following:
  - **learn term expansions**
  - before LLMs, "classical" search methods relied on rule-based term or vocabulary expansions
  - We all know how time consuming and extremely limiting rule-based NLP systems are.
  - SPLADE can thus help with OOV (out of vocab) or the "vocabulary mismatch problem"
- MLM head is retained and used (usually HEAD is removed or fine-tuned)

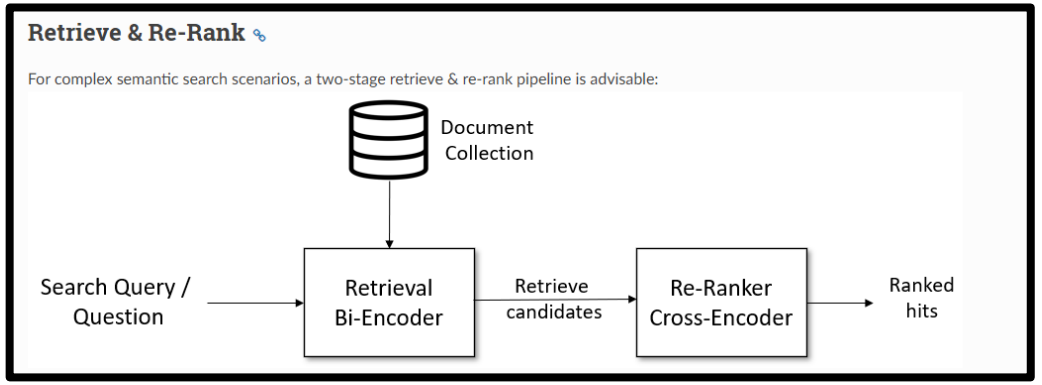


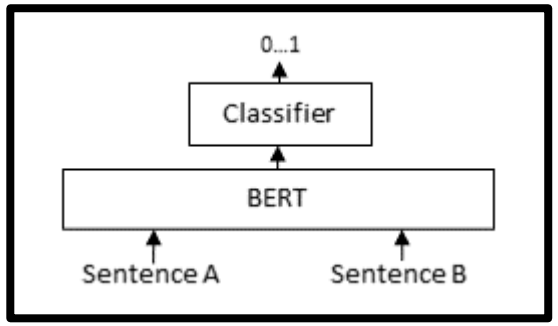


Source: Briggs, 2023

Revisit the customer….

- Vector DB Benchmark Reports
  - Hybrid search capabilities (BM25)
  - DOES NOT support Sparse vectors
  - Indexing: FLAT, HNSW
- Decided to stay with MongoDB for now
  - Recommended to change vector database when scaling up
- Embeddings?
  - Removed BGE embeddings
  - Utilized SBERT Sentence Transformers with 2 stage retrieval
    - Bi-encoder → Cross-encoder → **Re-rank**





Summary

- "One size DOES NOT fit all" for Vector Databases
- Carefully evaluate your use case(s) before choosing your vector database solution.
- Avoid "embedding soup" in your vector database
- Indexing algorithms are important to understand for each vector DB
- Keyword vs. Semantic vs. Hybrid Search
- Consider Vector DB benchmarks when necessary
- Do you really need a vector database?
  - Depends on your use case(s), LLMs, RAG pipeline, embeddings, search methods
  - Controversial….?

References

- Briggs, 2023. "SPLADE for Sparse Vector Search Explained." Retrieved from: https://www.pinecone.io/learn/splade/
- Bussler, 2023. "Vector Databases (are All The Rage)". Retrieved from: https://medium.com/google-cloud/vector-databases-are-all-the-rage-872c888fa348
- Cardenas et al. 2023. "An Overview on RAG Evaluation". Retrieved from: https://weaviate.io/blog/rag-evaluation#generation-metrics
- Cardenas, 2024. "Distance Metrics in Vector Search." Retrieved from: https://weaviate.io/blog/distance-metrics-in-vector-search
- Elliot et al. 2024. "THE IMPACTS OF DATA, ORDERING, AND INTRINSIC DIMENSIONALITY ON RECALL IN HIERARCHICAL NAVIGABLE SMALL WORLDS". arXiv:2405.17813v1 [cs.IR] 28 May 2024
- Gao et al. "Retrieval-Augmented Generation for Large Language Models: A Survey." arXiv:2312.10997v5 [cs.CL] 27 Mar 2024
- George, 2024. "Faster and cheaper RAG with embedding quantization." Retrieved from: https://medium.com/@stephygeorge_33545/faster-and-cheaper-rag-with-embedding-quantization-8ceef694acf8
- Low, 2023. "You Don't Need a Vector Database." Retrieved from: https://about.xethub.com/blog/you-dont-need-a-vector-database
- Mehta, 2023. Understanding Vector DB's Indexing Algorithms. Retrieved from: https://gagan-mehta.medium.com/understanding-vector-dbs-indexing-algorithms-ce187dca69c2
- MyScale, 2024. Understanding Vector Indexing: A Comprehensive Guide. Retrieved from: https://medium.com/@myscale/understanding-vector-indexing-a-comprehensive-guide-d1abe36ccd3c
- Shakir et al., 2024. "Binary and Scalar Embedding Quantization for Significantly Faster & Cheaper Retrieval". Retrieved from: https://huggingface.co/blog/embedding-quantization#binary-quantization-in-sentence-transformers

References

- Subramanya et al. 2019. "DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node" NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems. December 2019Article No.: 1233Pages 13766–13776
- Tang, 2023. "Similarity Metrics for Vector Search". Retrieved from: https://zilliz.com/blog/similarity-metrics-for-vector-search
- Taipalus, 2024. "Vector database management systems: Fundamental concepts, use-cases, and current challenges".
- Cognitive Systems Research. Vol. 85, June 2024, https://doi.org/10.1016/j.cogsys.2024.101216
- The Data Quarry Blog, 2023. "Vector databases (1): What makes each one different?" Retrieved from: https://thedataquarry.com/posts/vector-db-1/
- The Data Quarry Blog, 2023. "Vector databases (2): Understanding their internals." Retrieved from: https://thedataquarry.com/posts/vector-db-2/
- The Data Quarry Blog, 2023. "Vector databases (3): Not all indexes are created equal." Retrieved from: https://thedataquarry.com/posts/vector-db-3/
- The Data Quarry Blog, 2023. "Vector databases (4): Analyzing the trade-offs." Retrieved from: https://thedataquarry.com/posts/vector-db-4/#8-filtering-strategy
- Zhang et al. 2022. "Is Neural Topic Modeling Better than Clustering? An Empirical Study on Clustering with Contextual Embeddings for Topics."
- Zilliz, 2023. "Vector Search Best Practices." Retrieved from: https://zilliz.com/event/vector-search-best-practices/success
- Vm, 2023. "Hybrid Search — Amalgamation of Sparse and Dense vector representations for Active Content Discovery." Retrieved from: https://medium.com/@maithri.vm/hybrid-search-amalgamation-of-sparse-and-dense-vector-representations-for-active-content-107828b30be5