

Vector search engines

George Panchuk
HSE AI Fall 2025

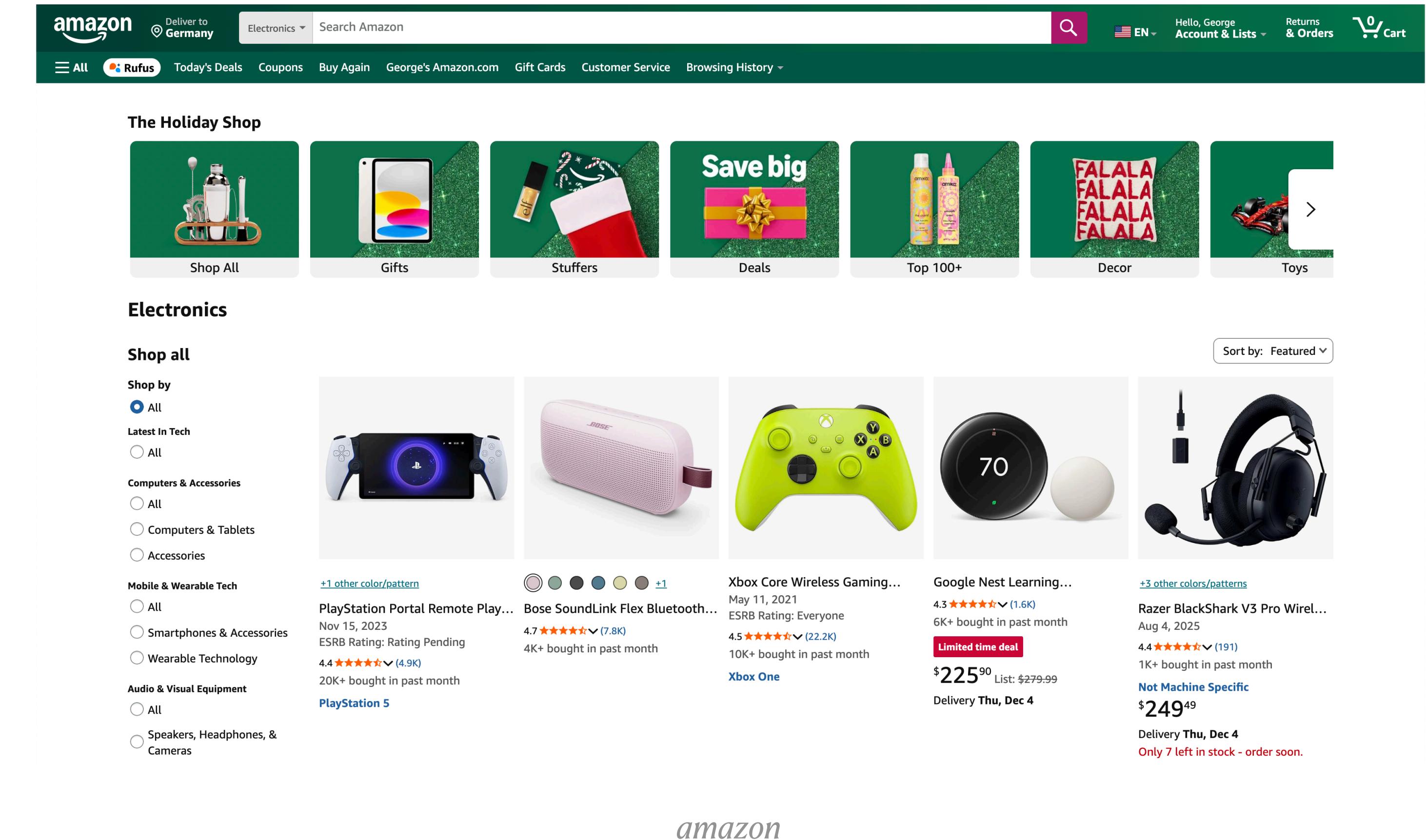


github.com/joein/vector-search-course

From ANN libraries to vector search engines

Are pure ANN libraries enough to build a real world app?

- ANN libraries are good at finding nearest neighbours to the query embeddings
- Can handle millions of records
- Have little to no overhead for compute
- But vectors and their nearest neighbours do not sound like enough to build a full-fledged product..



From ANN libraries to vector search engines

What else is required for the apps?

- Metadata filtering capabilities
- CRUD operations without full index rebuild
- Horizontal scaling
- Access control (RBAC)
- And many more..

The screenshot shows the Amazon search results for "gaming monitors". The search bar at the top has "gaming monitors" typed in. Below the search bar, there are several filters on the left side, each with a red box around it:

- Display Refresh Rate:** Options include 240, 165, 160, 144, 120, 100, and 75. The 165 option is checked.
- Resolution Standard:** Options include 5K UHD 2160p Ultra Wide, 4K UHD 2160p, 4K UHD 1600p Ultra Wide, 2K DCI 1080p, QHD Wide 1440p, QHD Ultra Wide 1440p, and FHD 1080p. The 5K UHD 2160p Ultra Wide, 4K UHD 2160p, and QHD Wide 1440p options are checked.
- Screen Size:** Options include 14.0 to 17.9 in and 26.0 in & above. Both are unchecked.
- Brands:** A search bar with "Search brands" placeholder text and a dropdown menu showing brand names like Samsung, LG, Dell, AOC, GIGABYTE, and Pixio. None are selected.

The main results section shows two products:

- LG 48GQ900-B 48" Ultragear UHD OLED Gaming Monitor:** Description: "LG UltraGear DCI-P3 99% (Typ.) with HDR 10, .1ms (GtG) 120Hz Refresh Rate, HDMI 2.1". Rating: 3.8 stars. Price: \$2,110.35 (1 new offer).
- ViewSonic Elite XG321UG-S 32" 4K IPS 144Hz Gaming Monitor - Certified Refurbished:** Description: "ViewSonic | GAMING ELITE MINI LED 4K 144Hz NVIDIA G-SYNC". Rating: 4.7 stars.

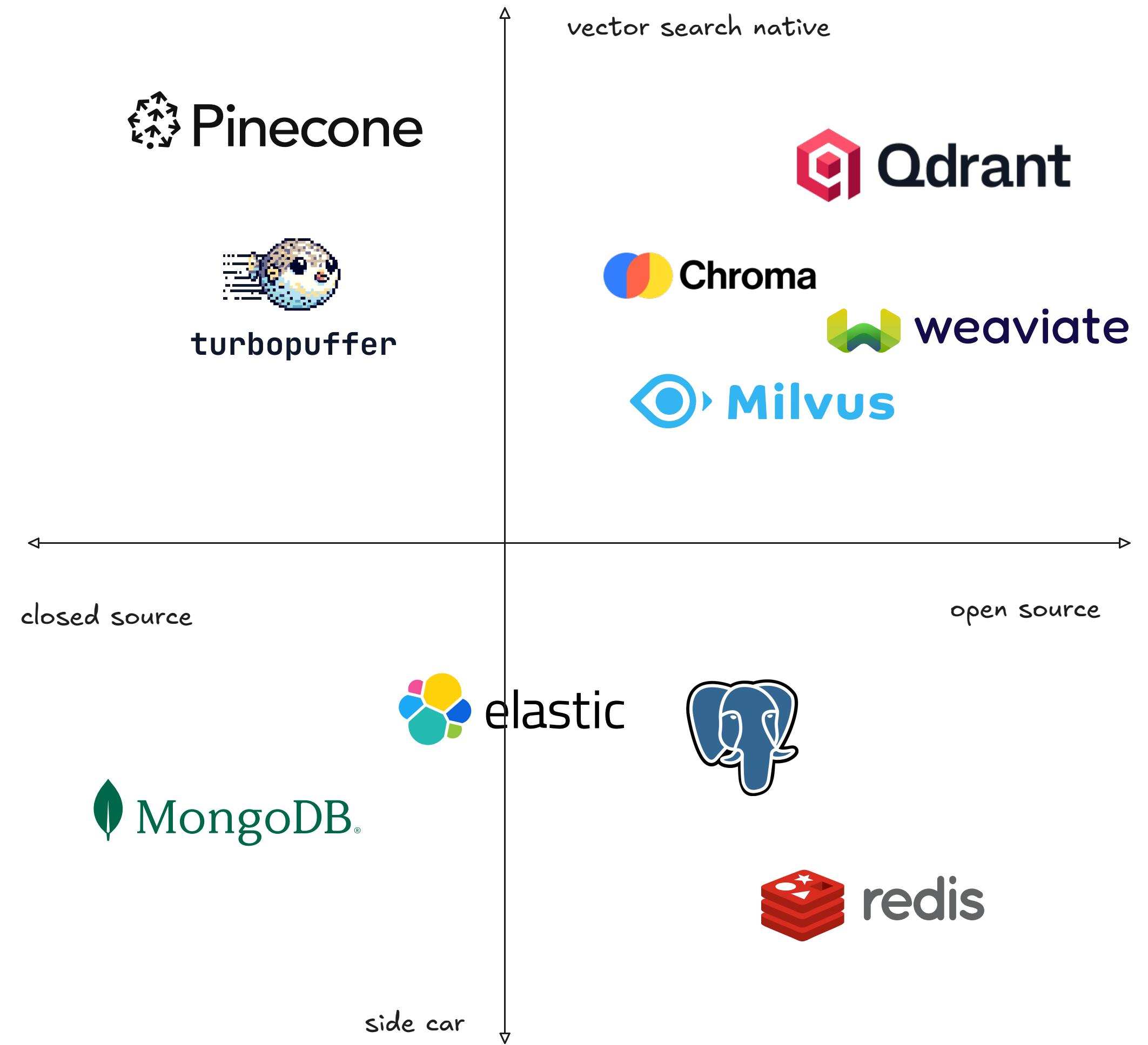
On the right side, there's a "Customers ask" section with a red box around it, containing a link to "Ask Rufus". Below that are "More results" sections for "Gaming monitors for PS5 compatibility", "Gaming monitors under \$150", and "Best curved gaming monitors".

amazon

Vector search engine

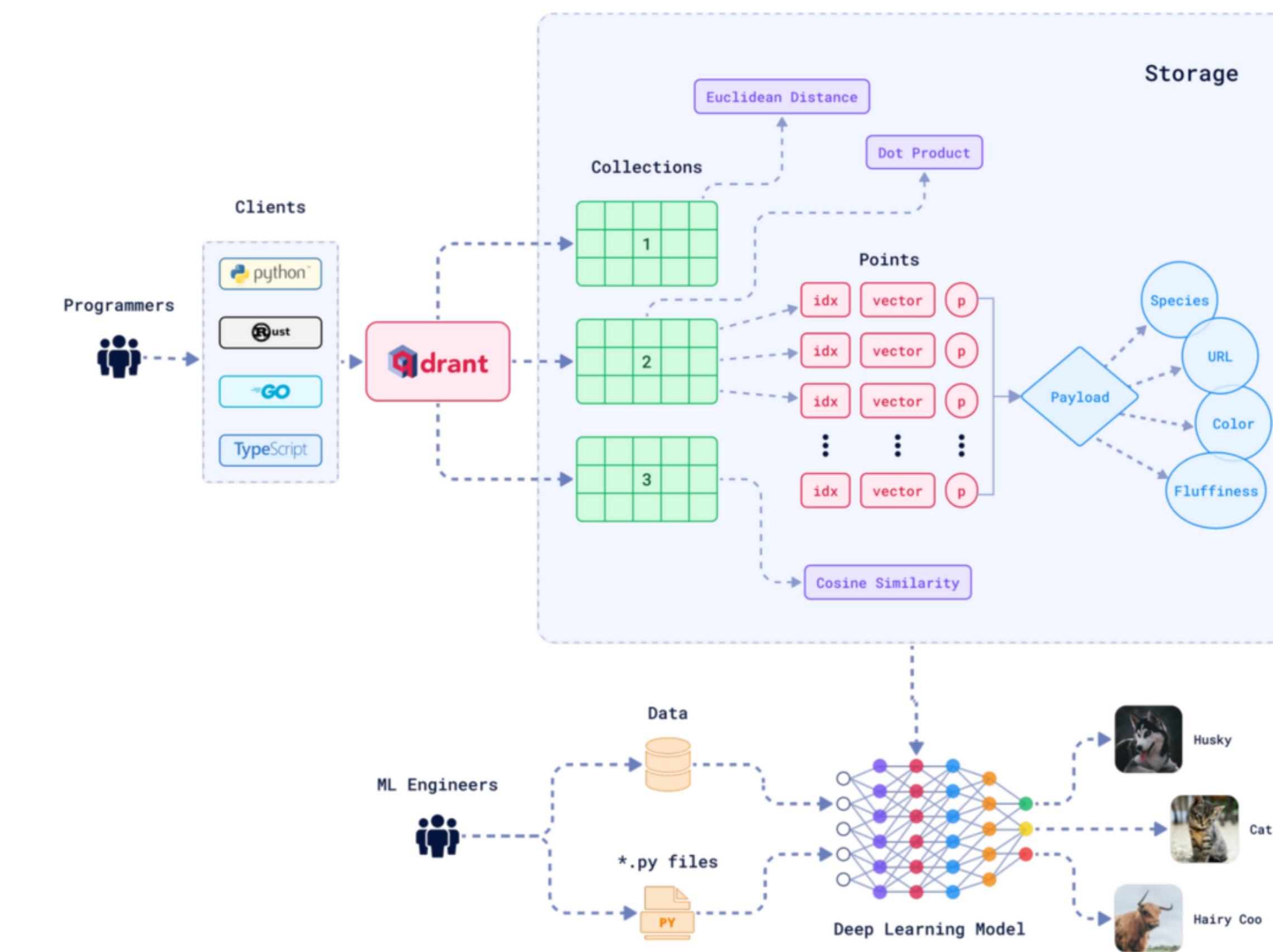
What is a vector search engine?

Vector search engine - is a specialised system which combines ANN indexing with storage, filtering, and scaling to support real-world applications.



A service, not a library

- No need to wrap search layer into an API, HTTP/gRPC/GraphQL APIs are usually available out-of-the-box
- Deploy easily (e.g. with docker), scale vertically and horizontally, or use managed cloud versions
- SDK for various languages, not limited to python and C++
- Built-in schemas, collections, payloads, backup tools



What is a vector database?

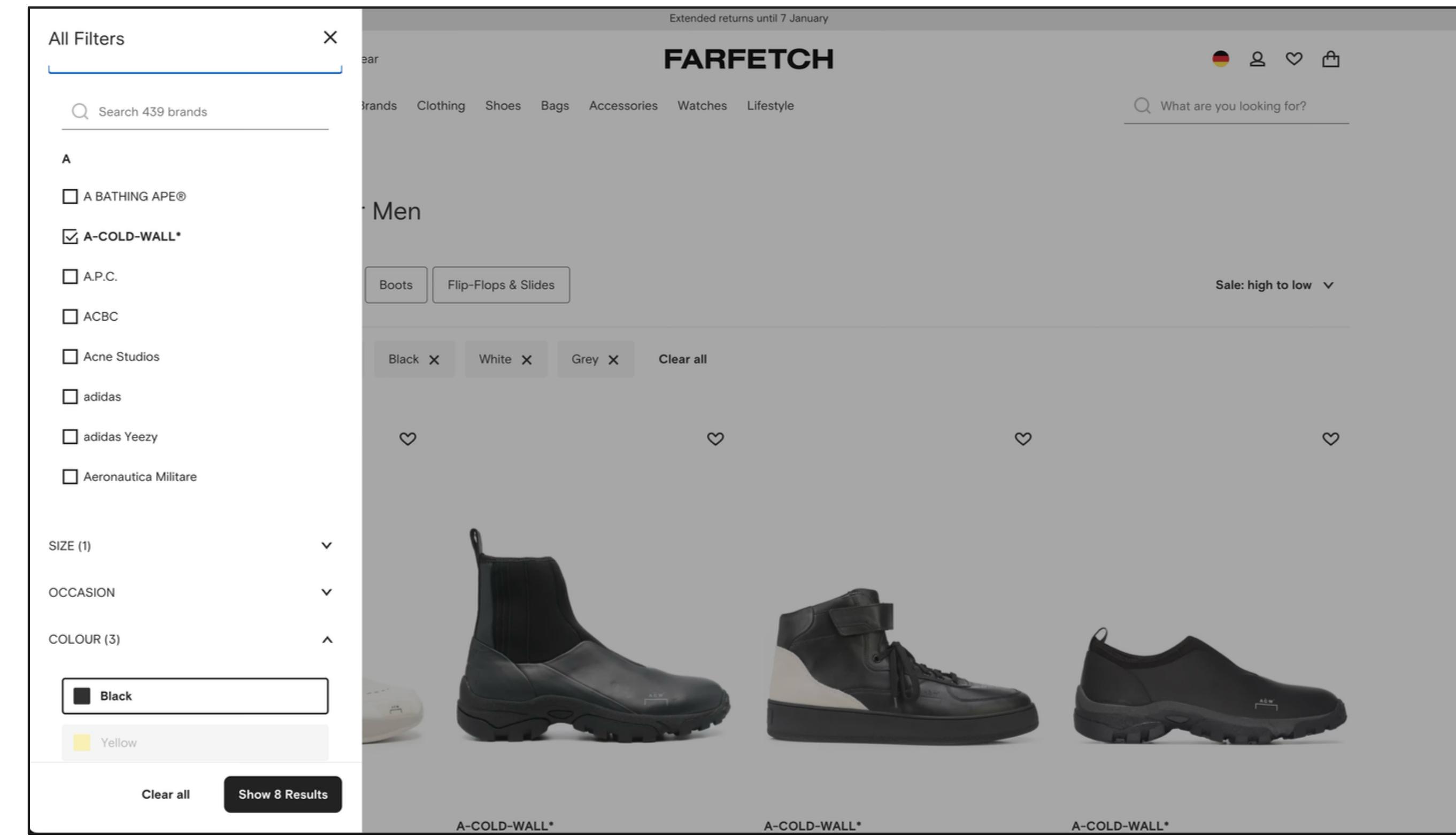
CRUD operations

- Insert new vectors without rebuilding the entire index
- Update payload or replace a vector
- Delete documents (soft delete + background cleanup)
- Consistency provided via WAL/raft, etc.
- Real-time indexing and searching, data is available for search as soon as it is inserted

```
● ● ●  
import numpy as np  
from qdrant_client import QdrantClient, models  
  
client = QdrantClient()  
client.create_collection(  
    collection_name,  
    vectors_config=models.VectorParams(  
        size=512,  
        distance=models.Distance.COSINE  
    )  
)  
client.upload_collection(  
    collection_name,  
    ids=range(10),  
    vectors=np.random.random((10, 512)),  
    payload=[{"index": i} for i in range(10)],  
)  
client.delete(collection_name, points_selector=[0, 1, 4])  
client.update_vectors(  
    collection_name,  
    points=[models.PointVectors(id=2, vector=np.random.random(512))]  
)  
client.set_payload(collection_name, payload={"name": "George"}, points=[7])  
client.clear_payload(collection_name, points_selector=[3, 5])
```

Payload filtering

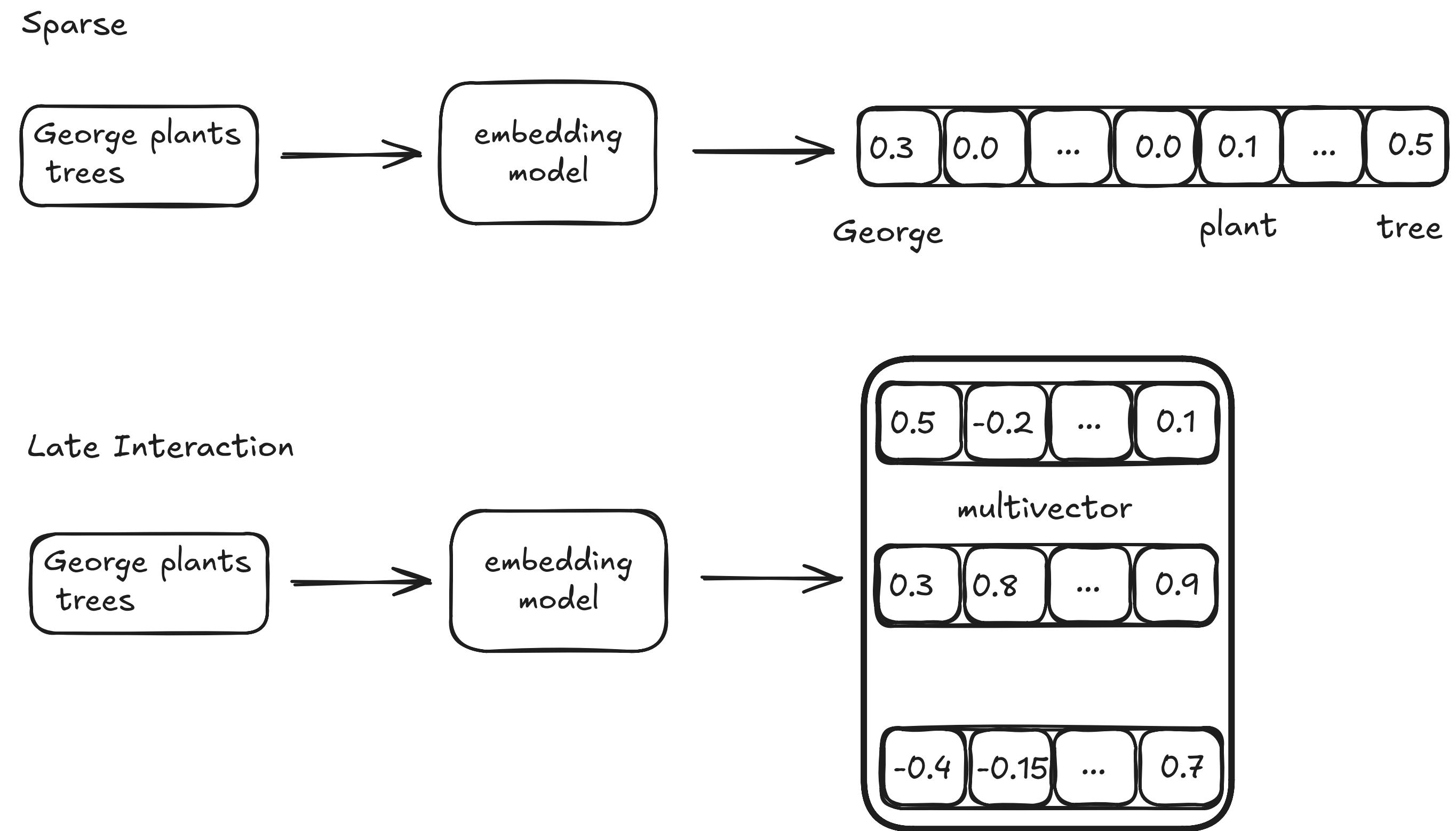
- It's usually not enough to just run nearest neighbours search for a real app, users might want to apply extra rules for the search
- Pre-filtering and brute-force might be too slow if a filter is not strict
- Post-filtering might filter out too many results if filter is too selective
- ANN-aware filtering techniques are required for better performance (filtrable hnsw, ACORN)
- Query planners are essential for balancing workload
- Additional structures and indexes are crucial for an efficient search



Farfetch

Different types of vectors

- Vector search does not rely solely on dense embeddings
- Typical embedding types also include sparse (BM25, SPLADE, etc) and late-interaction embeddings (ColBERT)
- Traditional ANN libraries mainly support only dense vectors



Multistage and Hybrid search

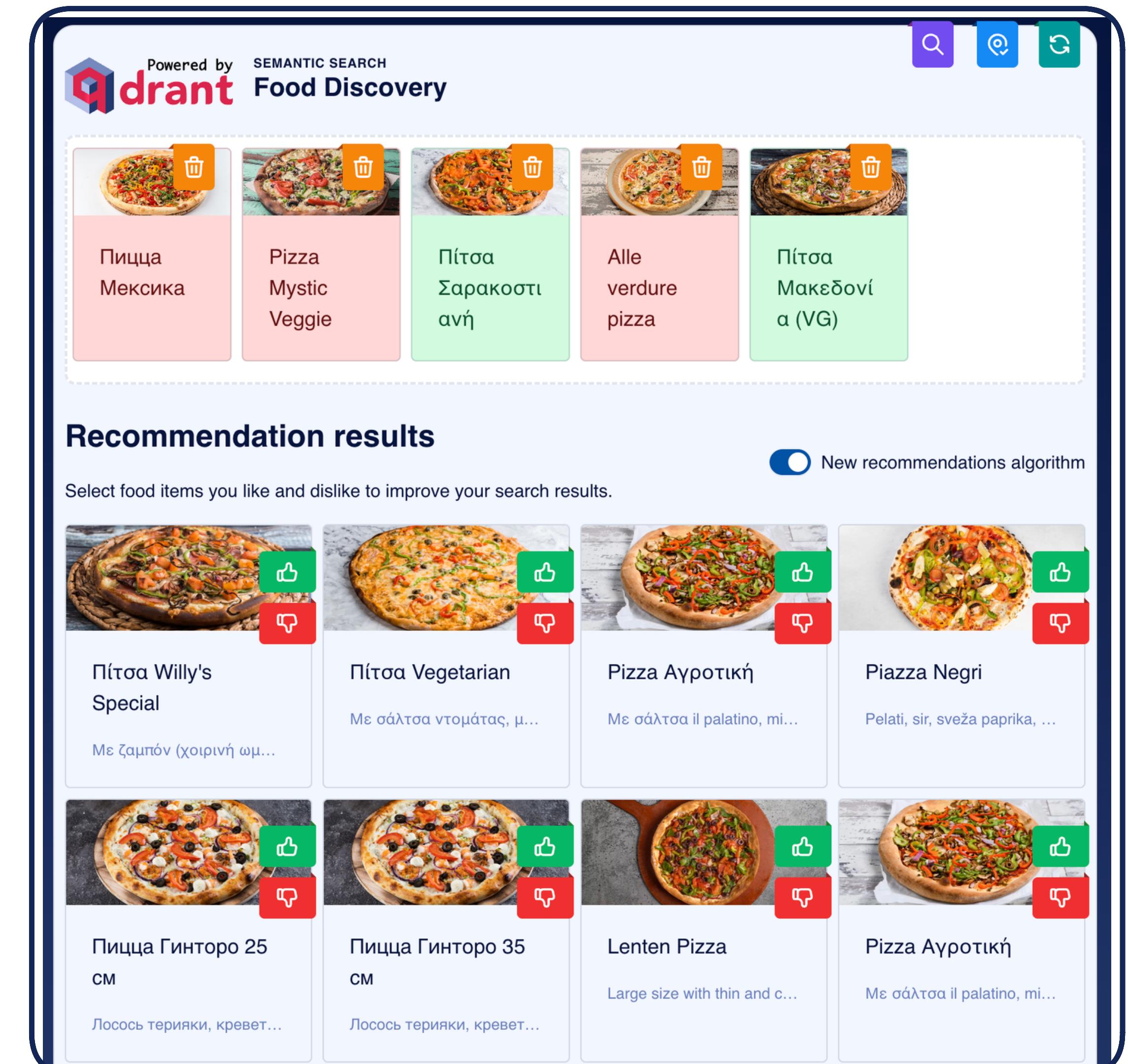
- To balance speed and recall, production systems often adopt two-stage retrieval:
 1. A lightweight model retrieves a broader candidate set,
 2. A larger, more accurate model re-scores those candidates
- Hybrid retrieval might mix dense semantic search with sparse lexical signals (BM25, SPLADE) for better coverage
- Multimodal approaches can search among both textual and image data simultaneously
- Certain results can be boosted or down-weighted using payload-driven decay functions (e.g., time-based scoring).

```
from qdrant_client import QdrantClient, models

client = QdrantClient()
client.create_collection(
    collection_name,
    vectors_config={
        "large-dense": models.VectorParams(size=1024, distance=models.Distance.COSINE),
        "small-dense": models.VectorParams(size=384, distance=models.Distance.COSINE),
    },
    sparse_vectors_config={"bm25": models.SparseVectorParams(modifier=models.Modifier.IDF)},
)
# <upsert>
client.query_points(
    collection_name,
    query=models.Document(
        text="What is the capital of France?", model="intfloat/multilingual-e5-large"
    ),
    using="large-dense",
    prefetch=[
        models.Prefetch(
            query=models.Document(
                text="What is the capital of France?",
                model="sentence-transformers/all-minilm-l6-v2",
            ),
            limit=50,
            using="small-dense",
        ),
        models.Prefetch(
            query=models.Document(
                text="What is the capital of France?",
                model="Qdrant/bm25",
            ),
            limit=50,
            using="bm25",
        ),
    ],
    limit=10,
)
```

Advanced methods: Exploration

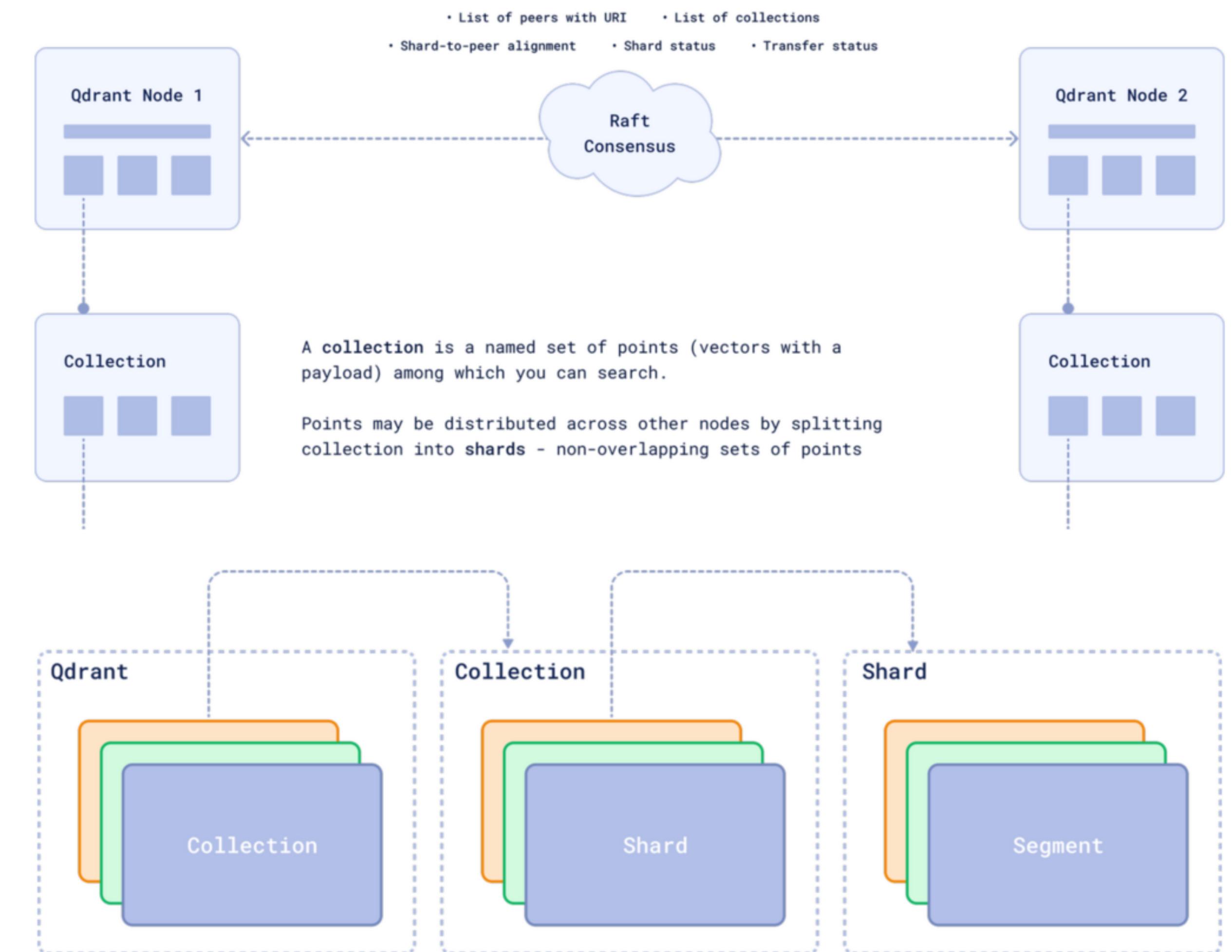
- Vector search is not limited to looking for a point nearest to one query, e.g. it also allows to do a directed traversal of the embedding space
- Neighbourhood exploration: expand from seed points to uncover clusters and local structure
- Discovery / context search: use positive examples to define the target semantic region and negative examples to exclude unwanted directions
- Distance-based exploration tools (distance matrices, local graphs) for analysis and visualisation of the vector space



Qdrant: Food discovery demo

Horizontal scaling

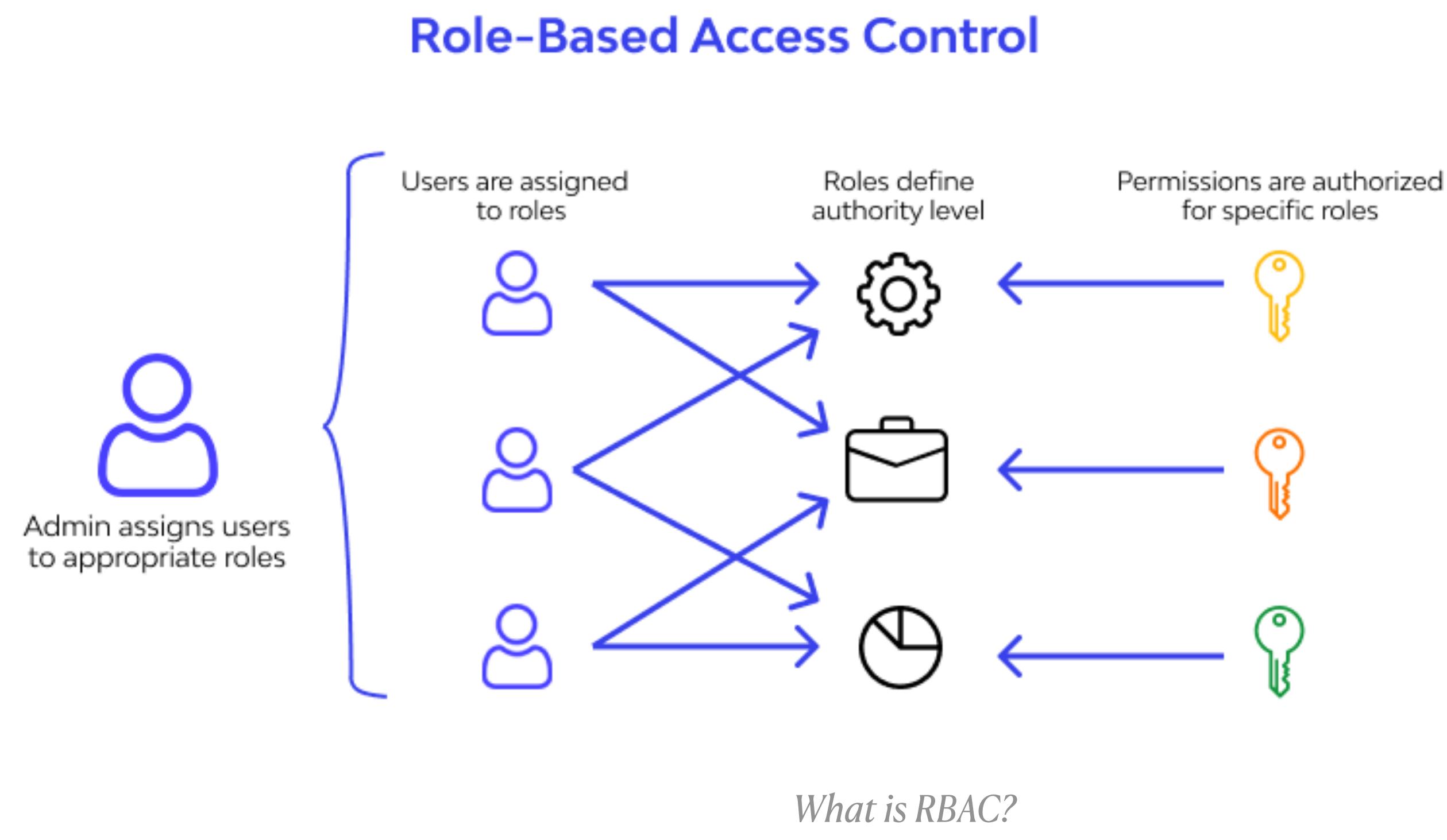
- A single machine eventually hits limits in memory, CPU, storage and I/O
- Distributed search engines split data into shards, each holding a non-overlapping subset of the collection
- Shards can be spread among multiple nodes
- Data is automatically distributed across shard, queries are routed to all relevant nodes
- Replication is used for high availability, failover, and increased read throughput
- Query results from multiple shards are merged into a final ranked list
- Horizontal scaling allows the system to grow with data size and traffic without major architectural changes
- With ANN libraries all this has to be done manually



What is a vector database?

Security and access control

- Data stored in vector search engines is often proprietary and may include sensitive information
- Original data could potentially be reconstructed from the embeddings via “embedding inversion attack”
- Vector search engines are usually not secured by default
- API-keys and TLS encryption can be used if all-or-nothing access is acceptable
- Role-based access control (RBAC) provides more granular approach by defining user roles and corresponding permissions, often with JWT
- Useful for multi-team setups, production environments and managed deployments



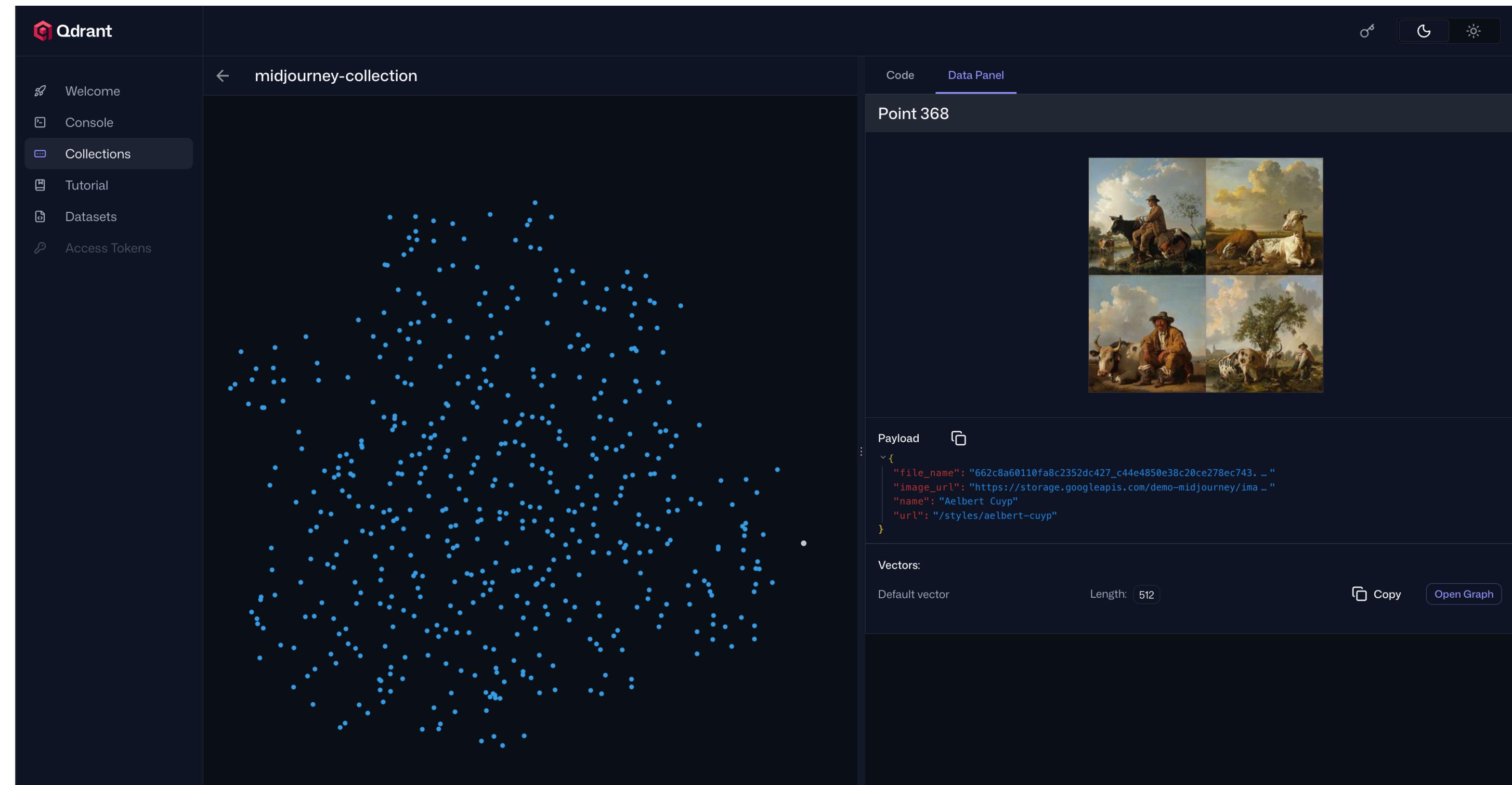
Snapshots & recovery

- Production system require reliable recovery mechanisms to prevent data loss and index corruption
- Write-ahead logs (WAL) record all writes so the system can restore after crashes
- Snapshot is a consistent point-in-time image of the database or a collection
- Can be used for recovery, migration, safe upgrades, and cloning environments
- Combining WAL with snapshots enables restoring the system to an exact previous state

Datasets					
NAME	DATASETS SIZE	VECTORS CONFIG	VECTORS COUNT	ACTIONS	
Qdrant Web Documentation Qdrant Web Documentation. Contains chunks of text from the Qdrant Documentation. Additionally contains Full-Text index.	139 MB	Default 384 Cosine all-MiniLM-L6-v2	18828	Import Dataset	
Prefix Cache Collection, used for caching vectors of small requests. It is possible to bypass the encoder model and use cached vectors. Useful for search-as-you-type applications.	377 MB	Default 384 Cosine all-MiniLM-L6-v2	163075	Import	
Midjourney Styles Collection of Midjourney style samples encoded with CLIP model. Provided by https://midlibrary.io/	15.6 MB	Default 512 Cosine clip-ViT-B-16	5417	Import	

Bells & whistles

- There are lots of other things which vector search engines implement, it's hard to cover them all in one lecture
- Those could include various quantisation mechanisms (PQ, SQ, BQ, etc.) for memory and/or speed optimisations, or targeting particular real world scenarios, such as multitenancy
- Libraries or modules for inference
- Monitoring, telemetry and visualisation tools
- In-memory or local modes for fast experimentation



Engines comparison

- This slide could have been a huge comparison table...
but these tables get outdated quickly, since the vector search engine space evolves fast
- Most engines are open-source, and unique features spread across the ecosystem within weeks or months
- Instead of memorising tables, focus on choosing the right category for your use case

Side-car search (ElasticSearch, Redis, Postgres, Mongo, OpenSearch, etc.)

- Choose this if you app is small, or you already run one of these systems in production
- Good when vector search is a convenience feature, not a core component

Dedicated vector search engine (Qdrant, Milvus, Weaviate, Pinecone, Turbopuffer, Chroma, etc.)

- Prefer this when vector search is central to the product, or your dataset/workload will grow
- Evaluate based on your workload patterns, required features, and unique capabilities available at the moment
- Run your own benchmarks

Questions?