Vector Search in Modern Databases

Peter Zaitsev
Founder at PERCONA

About me

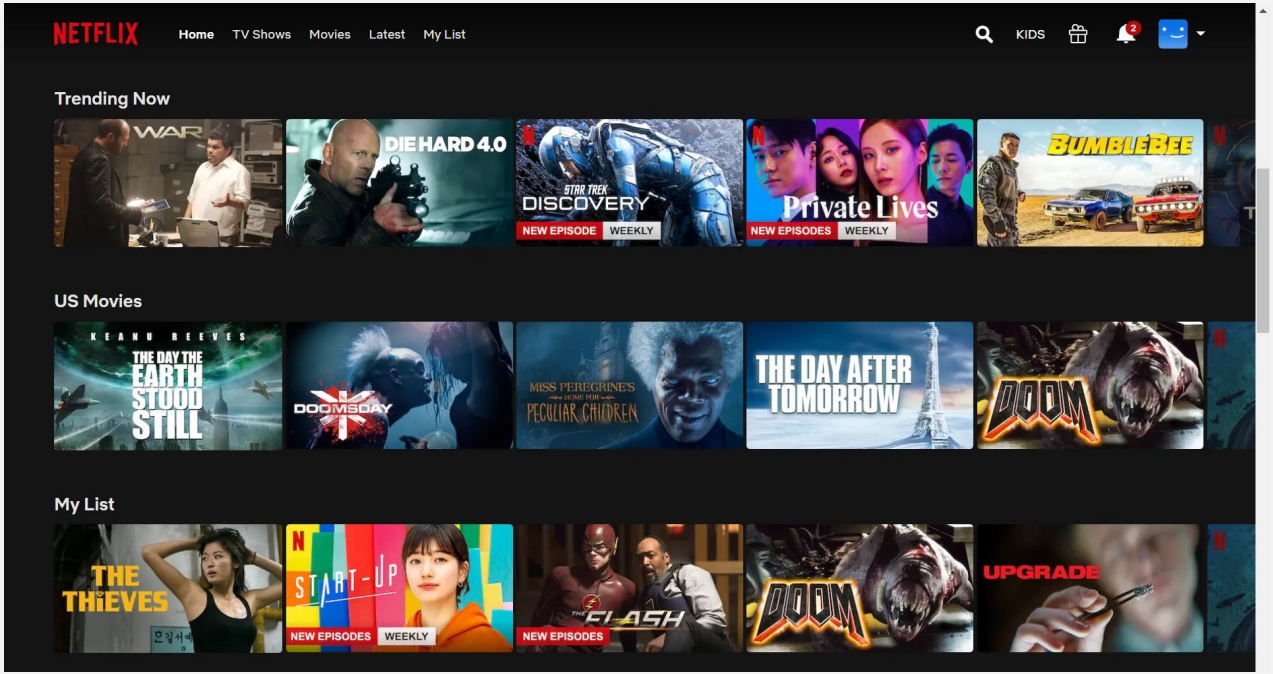
Peter Zaitsev

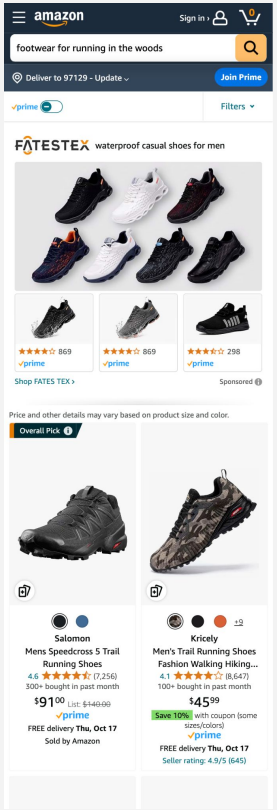
- Founder of Percona, Altinity, FerretDB, Coroot
- Open Source Database Enthusiast
- Open Source Advocate
- Speaker, Advisor, Investor

Vector Search – Not the Only Name

- Vector Databases
- Vector Search Databases
- Vector Similarity Search Databases
- AI Native Database
- Both Standalone Databases as well as Functionality in Existing Databases
- Landscape is Evolving

Why Vector Search?





Vector Search support in databases

Open-source vector dbs

Milvus	2019
Vespa	2020
Weaviate	2021
Qdrant	2022

Open-source databases and search engines

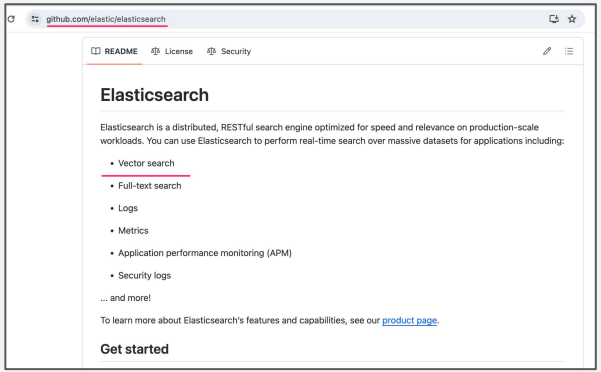
Elasticsearch	2019
PostgreSQL	2021
Lucene	2021
OpenSearch	2022
Redis	2022
SOLR	2022
Cassandra	2023
Typesense	2023
Clickhouse	2023
Manticore Search	2023
Meilisearch	2023
MariaDB	2024
MySQL	Non-Open Source

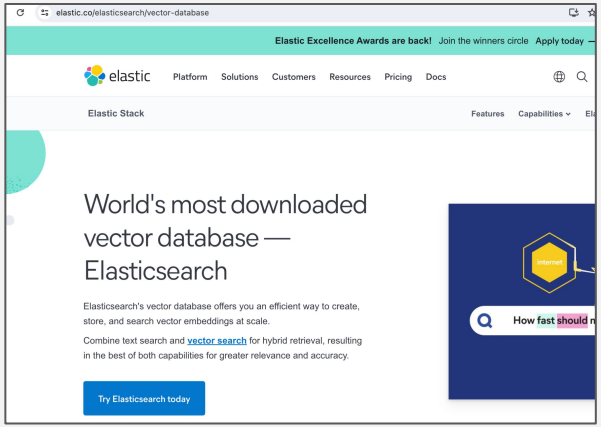
Non-open-source dbs

Oracle	2023
MongoDB	2023

Clouds

Pinecone	2019
Amazon Elasticsearch / OpenSearch	2020
Google Cloud Platform	2021
Alibaba Cloud AnalyticDB	2023
Azure	2023
Amazon DocumentDB	2023
Cloudflare Vectorize	2023
Planetscale	2024





"World's most downloaded vector database"

—Elasticsearch

Introducing EDB Postgres® AI

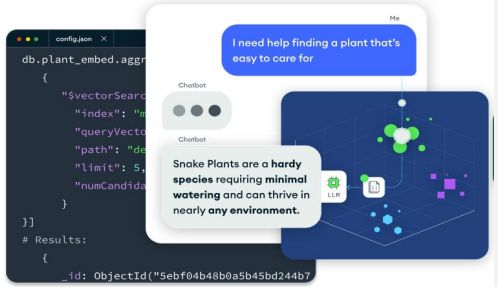
An intelligent platform for unified management of transactional, analytical and AI workloads — powered by Postgres

Read the Press Release

Loved by developers. Built for

Vector Search

You don't need a separate database to support transactions, rich search, or gen AI. The world's most popular document database is now the world's most versatile developer data platform.



Building AI Capability

MongoDB. News & Events Corporate Governance Financial Information Stock Information Shareholder Services

Release Details

MongoDB Announces Acquisition of Voyage AI to Enable Organizations to Build Trustworthy AI Applications

February 24, 2025

PDF Version

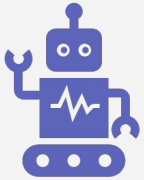
MongoDB to integrate Voyage AI's industry-leading embedding and reranking models, delivering highly accurate and relevant information retrieval to power sophisticated AI use cases

NEW YORK, Feb. 24, 2025 /PRNewswire/ -- MongoDB, Inc. (NASDAQ: MDB), the leading database for modern applications, today announced it has acquired Voyage


Any Sufficiently Advanced
Technology Is
Indistinguishable From Magic.
~ ARTHUR C CLARKE ~
Statustown.com

Let's Talk about Vectors

Vectors and AI

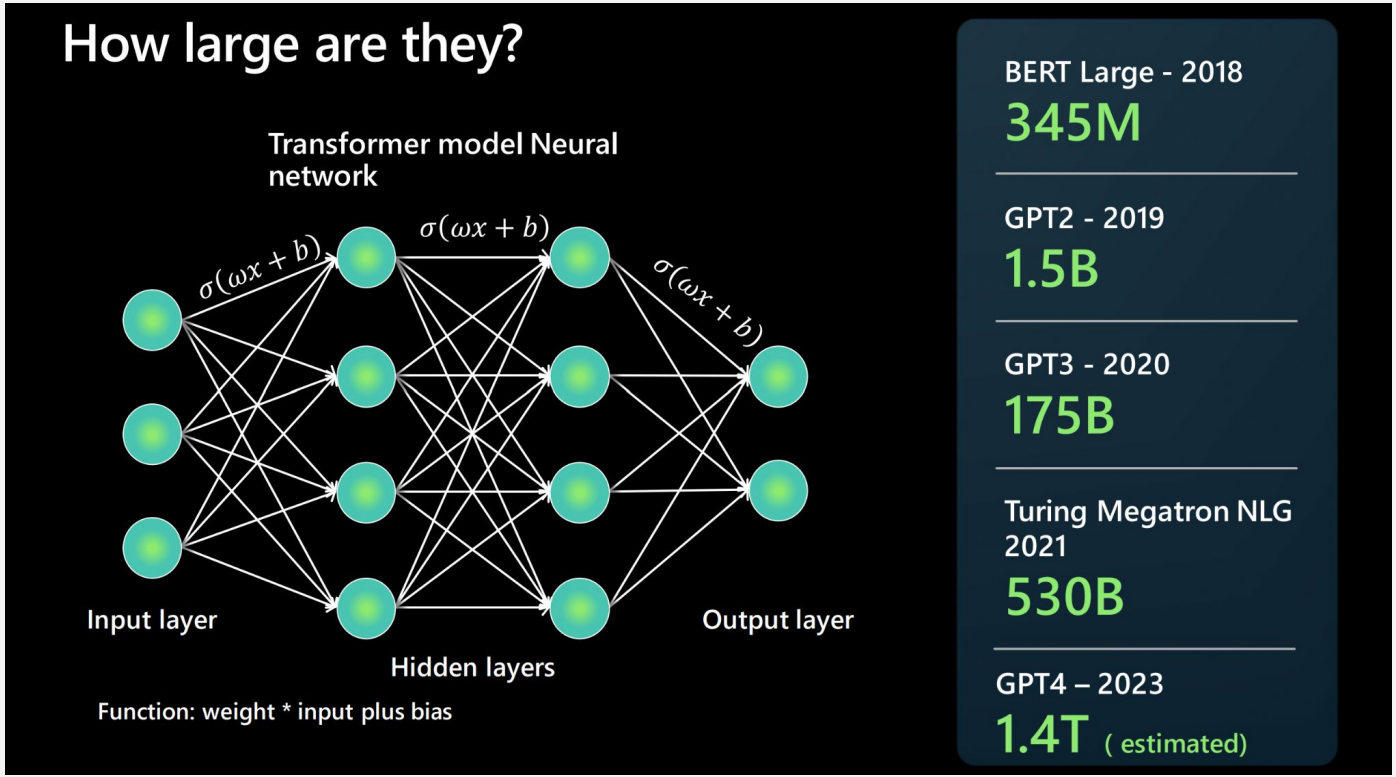


AI (Artificial Intelligence) and ML (Machine Learning) are using a lot of “Linear Algebra” Math

Linear Algebra deals with Matrices and Vectors

Real world objects (texts, images, videos) are represented by Vectors for Processing

LLMs (Large Language Models)



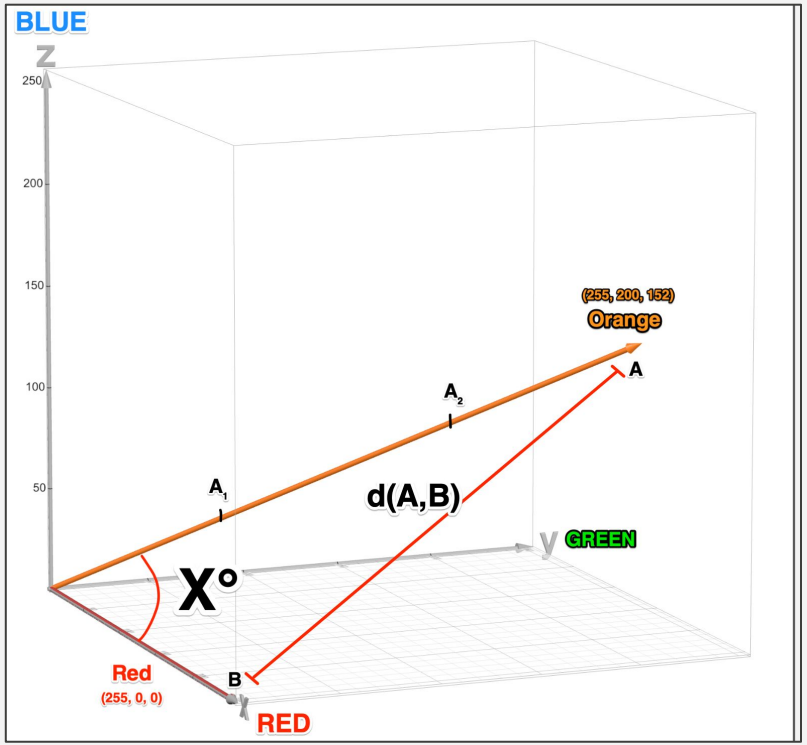
https://microsoft.github.io/Workshop-Interact-with-OpenAI-models/llms/

Vector space and vector similarity

- x : 0 .. 90°
- cos(x): 0 .. 1
- cos(x) is the same between B an A_1, A_2 and A
- Cosine similarity doesn't account vector lengths: 0 .. 1

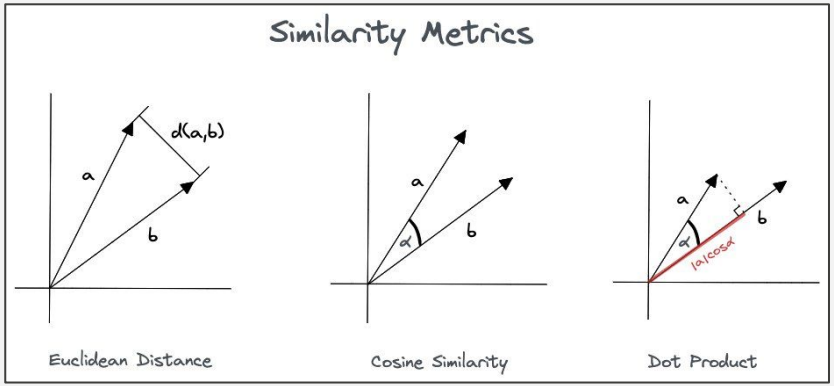
$$\text{Cosine Similarity} = \frac{A \cdot B}{\|A\|\|B\|}$$

- Euclidean distance d(A,B) does



Distance between Vectors

- Cosine similarity
- Euclidean distance
- Dot product



OpenAI API docs:

Which distance function should I use?

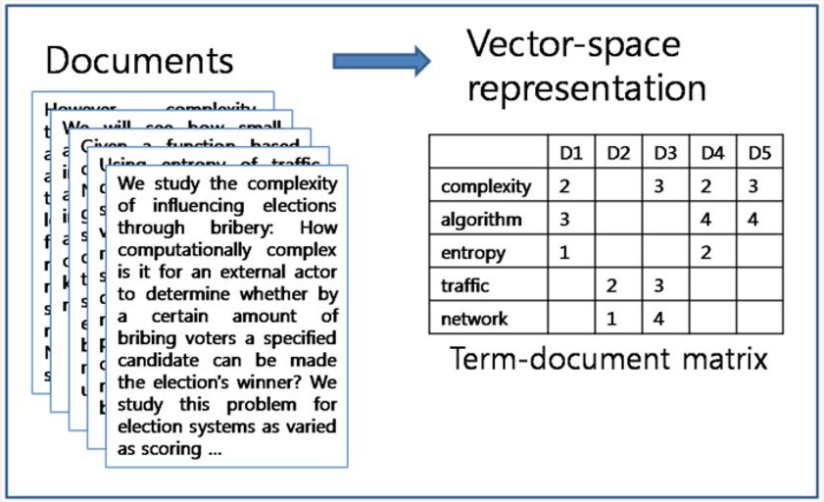
We recommend cosine similarity. The choice of distance function typically doesn't matter much.

OpenAI embeddings are normalized to length 1, which means that:

- Cosine similarity can be computed slightly faster using just a dot product
- Cosine similarity and Euclidean distance will result in the identical rankings

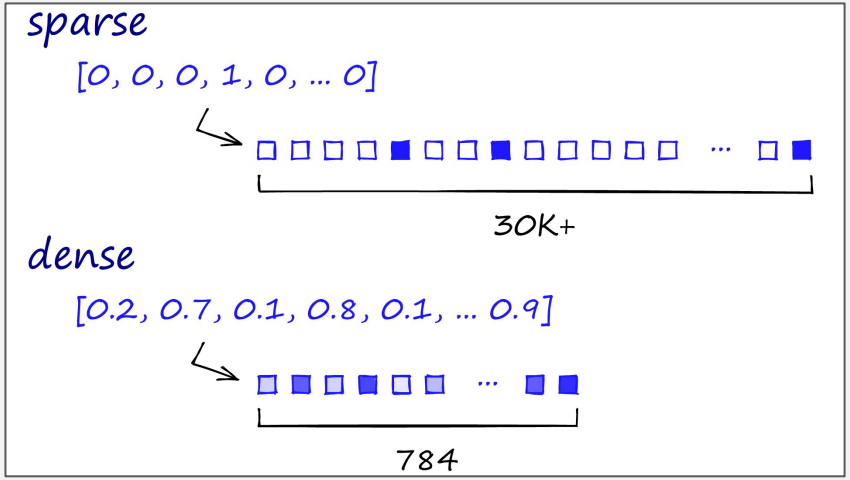
Sparse vectors

- [Green, Red, Blue] is easy
- More dimensions?
- Bag of words sparse vectors:
- [Has word “Hello”, has word “World”, …]
- [Number of words “Hello”, number of words “World”, …]
- [TF-IDF of word word “Hello”, TF-IDF of word word “World”, …]



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Dense vectors

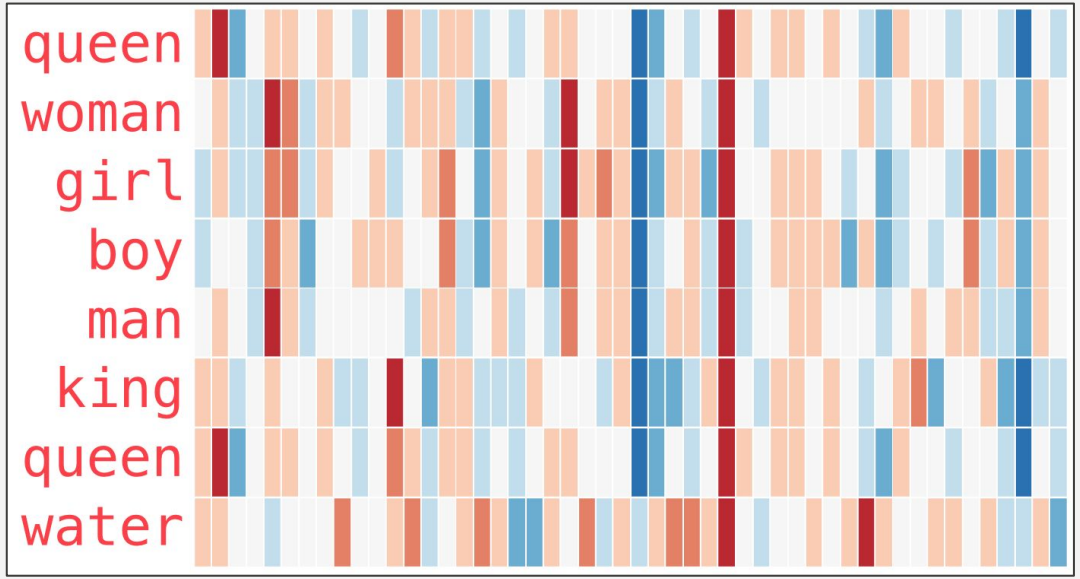


- What's closer: a cat and a dog, or a cat and a car?
- Deep learning => embeddings:
  - Accounts contexts for texts: Word2vec, BERT, GPT
  - Vectors from images
  - Vectors from sounds

Embeddings

Jay Alammar's Experiment: GloVe model to visualize word vectors:

- Red column shows similarity along one dimension (unknown though).
- "Woman" and "girl" are similar across many dimensions; same for "man" and "boy."
- "Boy" and "girl" share some similarities that differ from "woman" or "man" (youth?).
- "Water" added to highlight differences between people-related words and unrelated categories.
- Clear similarities between "king" and "queen," suggesting royalty.

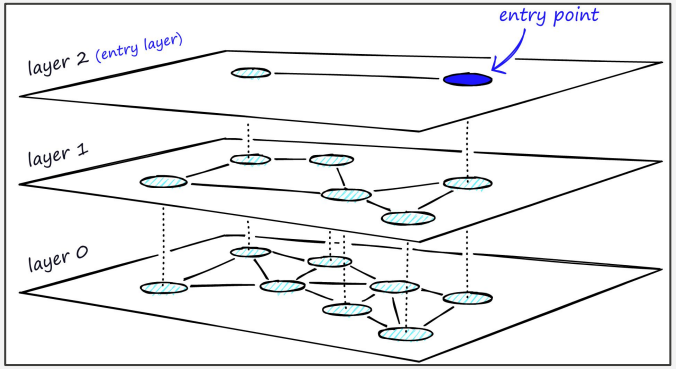


Embeddings are Magic

- Embeddings are properties of the AI Model
- There is no “right” answer for Embeddings
- Embeddings depend on the Model and Configuration used
- You can’t reason about specific values in Embedding Vectors

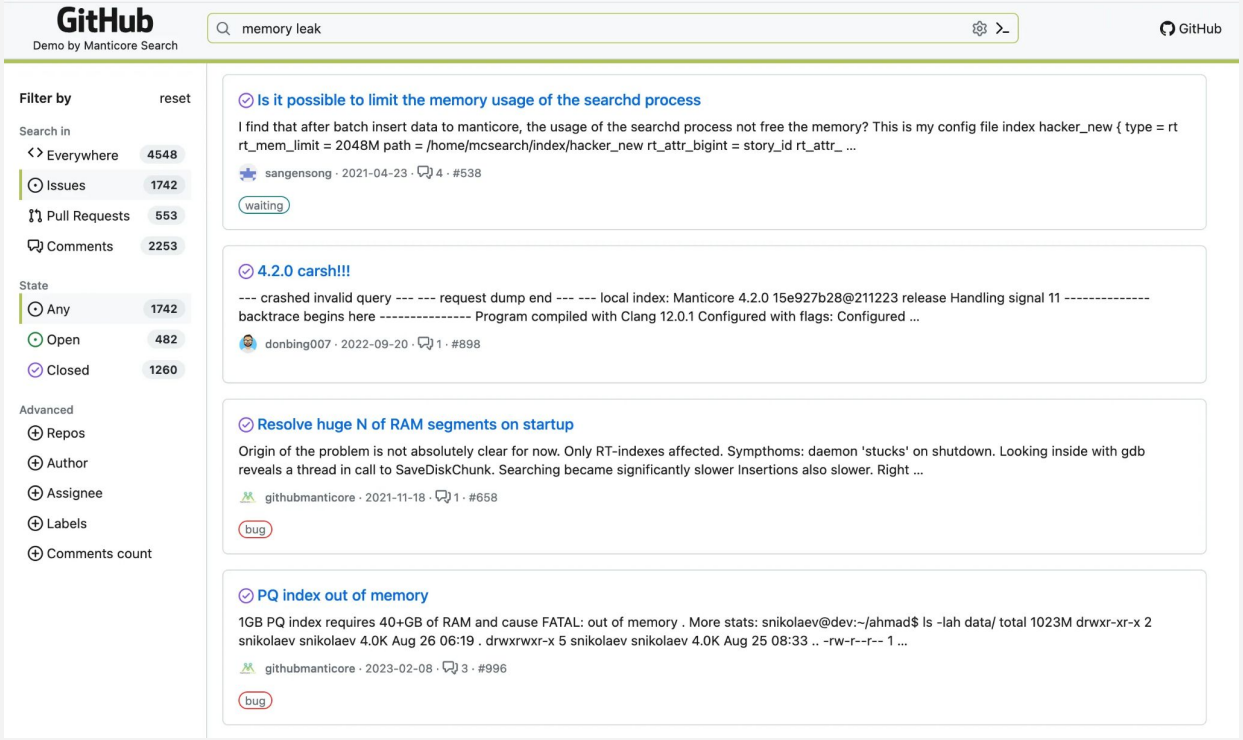
Dealing with embeddings

HNSW index



- Dense embeddings from deep learning pose indexing challenges.
- Traditional methods like inverted indexes are ineffective for non-sparse vectors.
- Dense vectors require comparison with all vectors in the dataset.
- Specialized indexes (KD-trees, LSH, HNSW, Annoy) enable:
  - Faster search
  - Insignificant accuracy loss.
- HNSW is used by most dbs and search engines: Postgres, Lucene, OpenSearch, Redis, SOLR, Cassandra, Manticore Search, OpenSearch and Elasticsearch, Typesense, Meilisearch, Clickhouse

GitHub Issues Vector Search Demo by Manticore Search



https://github.manticoresearch.com/manticoresoftware/manticoresearch?search=semantic-search;query=memory%20leak;filters[state]=any;filters[index]=issues

Machine Learning tasks and apps

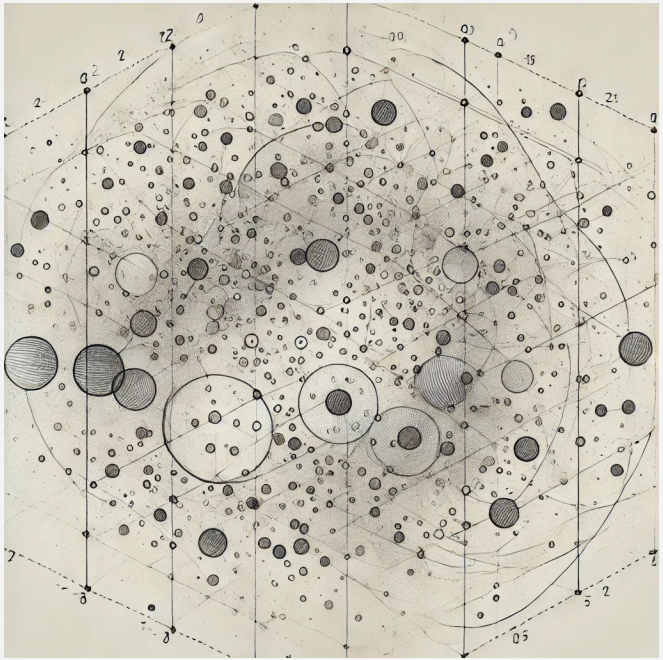
- Many tasks in ML, e.g.:
  - Clustering
  - Classification
  - KNN/ANN search and more
- Apps:
  - Recommendation Systems
  - Image Retrieval
  - Customer Segmentation and more

ML tasks: Clustering

- **Definition**: Clustering involves grouping similar data points based on their distance in a multi-dimensional space.

- **Purpose**: Identify patterns in a dataset without predefined labels.

- **Examples**:
  - User Behavior Grouping
  - Image Organization
  - Customer Segmentation



ML tasks: classification

- **Definition**: Classification involves assigning predefined labels to data points based on their features.
- **Purpose**: To categorize data into specific classes for better organization and decision-making.
- **Examples**:
  - Product Categorization
  - Spam Filtering
  - Medical Diagnosis

ML tasks: KNN/ANN search

- KNN and ANN - most attractive task in databases
  - Enhances databases with search engine-like features.

- Morphology + synonyms + stopwords + ... VS embeddings and vector similarity

Vector Search in PostgreSQL

pgvector – Extension to enable “core” Vector Search

pg_vectorize - Interfacing with LLM models, including Embedding creation

PostgresML – Platform to build AI Apps

Not Just pgvector

PostgreSQL Ecosystem sees a lot of innovation for Vector Search

VectorCord (AGPL)
https://github.com/tensorchord/VectorChord/

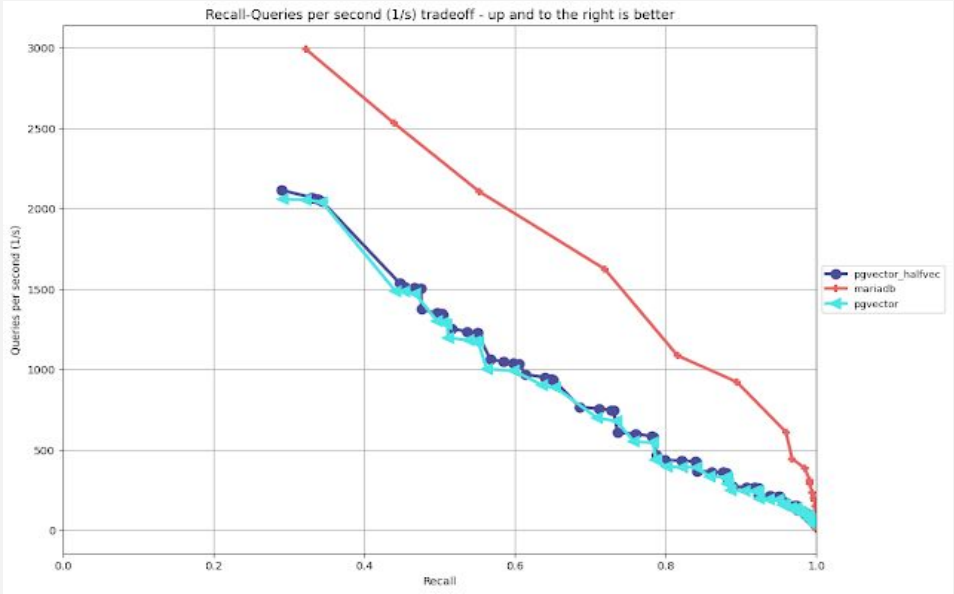
Pg_embedding (now Archived)
https://github.com/neondatabase-labs/pg_embedding

Using pgvector for KNN in works

```
vectordb=# CREATE TABLE items (
  id SERIAL PRIMARY KEY,
  title TEXT,
  image_vector vector(4)  -- This defines a vector with 4 dimensions
);
CREATE TABLE
vectordb=# CREATE INDEX ON items USING hnsw (image_vector vector_cosine_ops);
CREATE INDEX
vectordb=# INSERT INTO items (title, image_vector) VALUES
  ('yellow bag', '[0.653448, 0.192478, 0.017971, 0.339821]'),
  ('white bag', '[-0.148894, 0.748278, 0.091892, -0.095406]');
INSERT 0 2
vectordb=# SELECT id, title, image_vector <=> '[0.286569, -0.031816, 0.066684, 0.032926]' AS distance
FROM items
ORDER BY distance
LIMIT 2;
 id |   title    |      distance
----+------------+--------------------
  1 | yellow bag | 0.14651147524503616
  2 | white bag  |  1.2753469873582017
(2 rows)
```

MariaDB as an Alternative ?

Mark Callaghan: "MariaDB gets between 2.5X and 3.9X more QPS than Postgres for recall >= 0.95"



https://smalldatum.blogspot.com/2025/01/vector-indexes-mariadb-pgvector-large_28.html

Understanding Vector Search Benchmarks

- Vector Search Key Operation - finding K vectors closest to the given vector
- Exact Match is possible but expensive and not needed as embeddings themselves are “lossy”
- **Recall** – The portion of elements in result set which actually match query
- Updates
  - Is the “Index” updatable? How expensive is it to update?
- HNSW Has configuration Parameters, which are hard to reason about
  - M – Maximum Connections in the Graph
  - EF Construction – number of candidates considered during index build
  - EF Search – number of candidates to consider during search

Disadvantages of KNN vs Full-Text Search

- Higher Computational Cost
  - KNN needs more processing power, especially with large datasets.
- Resource-Heavy
  - More memory and storage are required for vectors and indexes.
- Less Explainable
  - Results can be harder to interpret compared to exact keyword matches.
- No Exact Matching
  - Can miss precise results expected in keyword-based queries.
  - Highlighting issues

Hybrid search approaches

Typical solutions:

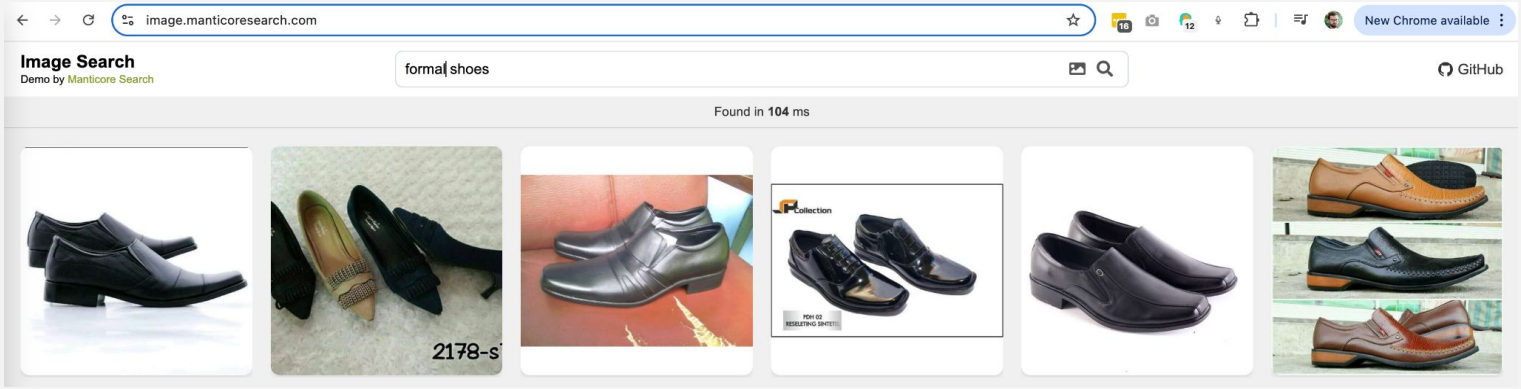
- Reciprocal Rank Fusion

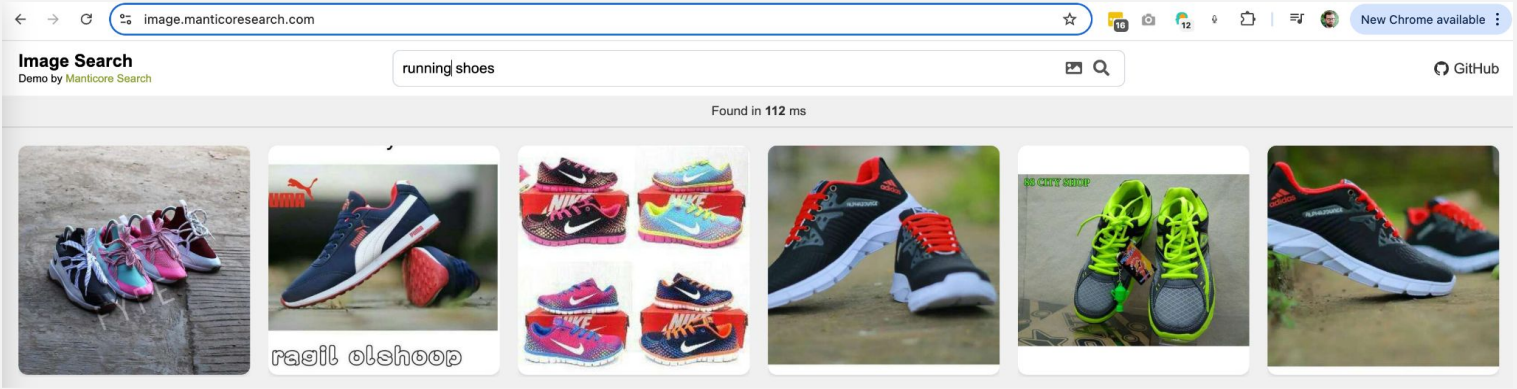
$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)}$$

- Multi-phase

BEIR Dataset	Vespa BM25	Vespa ColBERT	Vespa Hybrid
MS MARCO (*in-domain*)	0.228	**0.401**	0.344
TREC-COVID	0.690	0.658	**0.750**
NFCorpus	0.313	0.304	**0.350**
Natural Questions (NQ)	0.327	0.403	**0.404**
HotpotQA	0.623	0.298	**0.632**
FiQA-2018	0.244	0.252	**0.292**
ArguAna	0.393	0.286	**0.404**
Touché-2020 (V2)	0.413	0.315	**0.415**
Quora	0.761	0.817	**0.826**
DBPedia	0.327	0.281	**0.365**
SCIDOCS	0.160	0.107	**0.161**
FEVER	0.751	0.534	**0.779**
CLIMATE-FEVER	**0.207**	0.067	0.191
SciFact	0.673	0.403	**0.679**
Average nDCG@10 (excluding MS MARCO)	0.453	0.363	**0.481**

Image Search Demo





Generative AI

- Creates new content (text, images, music)
- Examples:
  - ChatGPT
  - MidJourney

Retrieval-Augmented Generation

- Combines Retrieval and Generation in AI Models
  - Merges search capabilities with language generation
- Enhances Responses with Relevant, Up-to-Date Information
  - Accesses external data sources in real-time
- Applications:
  - Advanced chatbots and virtual assistants
  - Real-time customer support
  - Dynamic content creation and summarization

What's Next? Embedding computation

- Non-vector databases typically integrate external embeddings.
- Elasticsearch, OpenSearch, Typesense enable automatic embedding generation.
- Nice libraries available:
  - Microsoft's ONNX Runtime library - used by Vespa, Typesense
  - HuggingFace's Candle - used by Manticore Search
- External embedding creation adds complexity for users.
- Automatic embedding generation simplifies operations and reduces dependency on external tools.
- Future trend: databases will evolve to not only store data but also understand and process it with built-in AI and machine learning, enabling smarter data handling and advanced search.

Future of Vector Search: Smarter Databases

- From Storage to Intelligence
  - Databases may move beyond just storing data to understanding it.
- AI-Powered Insights
  - Built-in machine learning may allow databases to predict and adapt.
- New Search Possibilities
  - Advanced search and analysis could unlock deeper insights.
- Simplified Operations
  - Automatic embedding generation may reduce complexity.
- The Next Frontier
  - Databases might transform into intelligent systems for real-time insights.

Interesting Developments

- **Improving Quantization** – working with smaller vector values rather than 1-bit
  - https://www.elastic.co/search-labs/blog/rabitq-explainer-101
- **RAG vs Large Context**
  - https://medium.com/@ethanbrooks42/rag-is-dead-why-retrieval-augmented-generation-is-no-longer-the-future-of-ai-27734ba456a1
- **Running LLMs Locally is getting more feasible** – DeepSeek R1 671b can be ran on $2K machine
  - https://digitalspaceport.com/how-to-run-deepseek-r1-671b-fully-locally-on-2000-epyc-rig/

Conclusions

- Vector search is transforming data retrieval, now standard in modern databases.
- Database Evolution:
  - Emergence of vector-specific and enhanced traditional databases.
  - Growing demand for smarter, context-aware search.
- Performance & Future:
  - HNSW indexing boosts speed.
  - Shift to internal embedding generation simplifies use and adds intelligence.
  - RAG: Combines search and AI for accurate, context-aware results.
- Key Takeaway:
  - Vector search and RAG signify a major shift in data management, paving the way for future advancements.

https://www.linkedin.com/in/peterzaitsev/

https://twitter.com/PeterZaitsev

http://www.peterzaitsev.com

Q&A
Thank you!