

Vector search in modern databases

Peter Zaitsev,
entrepreneur and co-founder of Percona

Sergey Nikolaev,
CEO, Manticore

Vector search support in databases

Opensource vector dbs

Milvus	2019
Vespa	2020
Weaviate	2021
Qdrant	2022

Opensource dbs and search engines

PostgreSQL	2021
Lucene	2021
Opensearch	2022
Redis	2022
SOLR	2022
Cassandra	2023
Typesense	2023
Clickhouse	2023
Manticore Search	2023
Meilisearch	2023
MariaDB	In progress
MySQL	Not yet

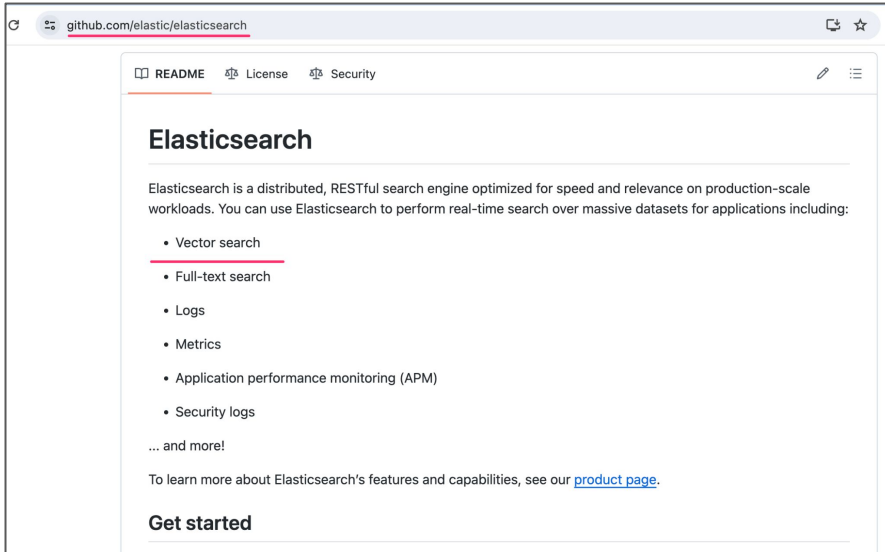
Non-opensource dbs

Elasticsearch	2019
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

Vector search support in Elasticsearch



The screenshot shows the GitHub repository page for Elasticsearch. The browser address bar displays `github.com/elastic/elasticsearch`. The repository navigation bar includes links for `README`, `License`, and `Security`. The main heading is **Elasticsearch**. Below the heading, a paragraph describes Elasticsearch as a distributed, RESTful search engine optimized for speed and relevance on production-scale workloads. A bulleted list highlights key features, with **Vector search** underlined in red. Other features listed include Full-text search, Logs, Metrics, Application performance monitoring (APM), and Security logs. The text concludes with "... and more!" and a link to the [product page](#). A **Get started** section is visible at the bottom.

github.com/elastic/elasticsearch

README License Security

Elasticsearch

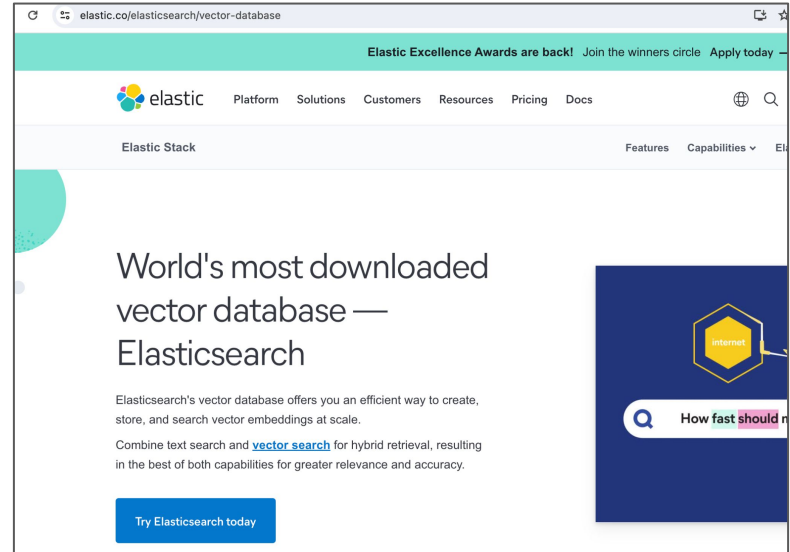
Elasticsearch is a distributed, RESTful search engine optimized for speed and relevance on production-scale workloads. You can use Elasticsearch to perform real-time search over massive datasets for applications including:

- **Vector search**
- Full-text search
- Logs
- Metrics
- Application performance monitoring (APM)
- Security logs

... and more!

To learn more about Elasticsearch's features and capabilities, see our [product page](#).

Get started



The screenshot shows the Elastic website page for the vector database. The browser address bar displays `elastic.co/elasticsearch/vector-database`. A teal banner at the top reads "Elastic Excellence Awards are back! Join the winners circle Apply today". The navigation bar includes the Elastic logo and links for Platform, Solutions, Customers, Resources, Pricing, and Docs. Below the navigation bar, the page features a large heading: "World's most downloaded vector database — Elasticsearch". A sub-heading states: "Elasticsearch's vector database offers you an efficient way to create, store, and search vector embeddings at scale." Below this, a paragraph explains: "Combine text search and [vector search](#) for hybrid retrieval, resulting in the best of both capabilities for greater relevance and accuracy." A blue button labeled "Try Elasticsearch today" is positioned at the bottom left. On the right side, there is a dark blue graphic with a yellow hexagon containing the word "Internet" and a search bar with the text "How fast should n".

elastic.co/elasticsearch/vector-database

Elastic Excellence Awards are back! Join the winners circle Apply today

elastic Platform Solutions Customers Resources Pricing Docs

Elastic Stack Features Capabilities

World's most downloaded vector database — Elasticsearch

Elasticsearch's vector database offers you an efficient way to create, store, and search vector embeddings at scale.

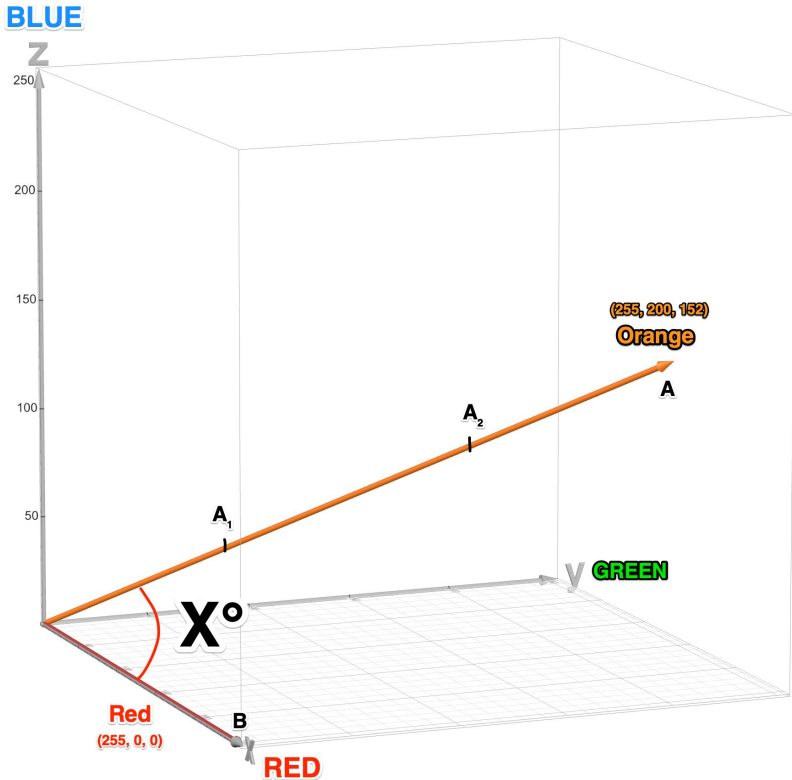
Combine text search and [vector search](#) for hybrid retrieval, resulting in the best of both capabilities for greater relevance and accuracy.

Try Elasticsearch today

Internet

How fast should n

Vector space and vector similarity



- $x : 0 \dots 90^\circ$
- $\cos(x) : 0 \dots 1$
- $\cos(x)$ is the same between B and A_1 , A_2 and A
- Cosine similarity accounts vector lengths: $0 \dots 1$

$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

Vector space and vector similarity

From OpenAI API:

Which distance function should I use?

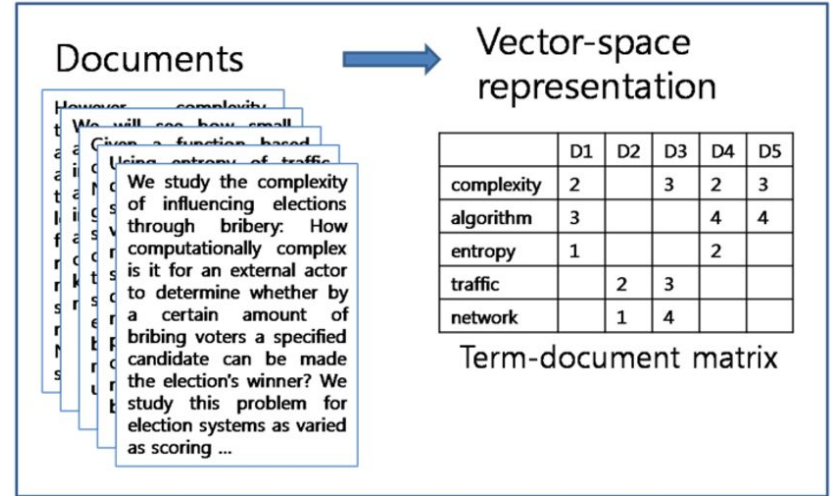
We recommend [cosine similarity](#). The choice of distance function typically doesn't matter much.

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

Vector features: sparse vectors

- [Green, Red, Blue]
- 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”, ...]

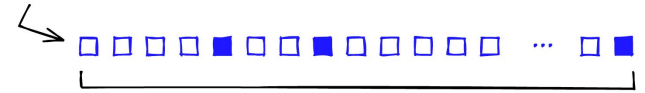


Vector features: dense vectors

- 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

sparse

$[0, 0, 0, 1, 0, \dots 0]$



30K+

dense

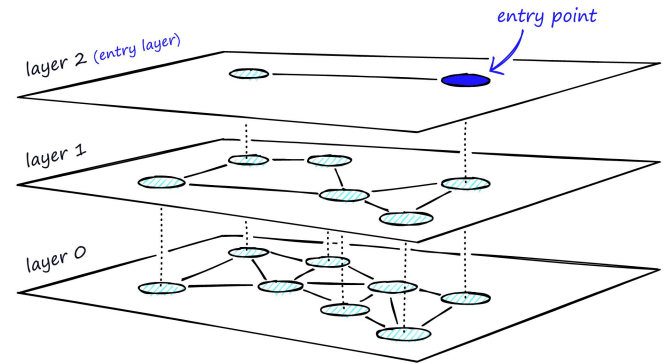
$[0.2, 0.7, 0.1, 0.8, 0.1, \dots 0.9]$



784

Dealing with embeddings

- 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



K-nearest neighbours

- Vector Search:
 - Clustering
 - Classification
 - KNN/ANN and more
- KNN and ANN - most attractive task in databases
 - Enhances databases with search engine-like features.

Vector search in dbs: typical implementation

```
mysql -P9306 -h0
Last login: Wed Jan 24 17:04:10 on ttys000
~ mysql -P9306 -h0
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 23358
Server version: 6.2.13 267b05c3a@24012222 dev (columnar 2.2.5 1d1e432@231204) (secondary 2.2.5 1d1e432@231204) (knn 2.2.5 1d1e432@231204) git branch master...origin/master

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create table test ( title text, image_vector float_vector knn_type='hnsw' knn_dims='4' hnsw_similarity='l2' );
Query OK, 0 rows affected (0.00 sec)

mysql> insert into test values ( 1, 'yellow bag', (0.653448,0.192478,0.017971,0.339821) ), ( 2, 'white bag', (-0.148894,0.748278,0.091892,-0.095406) );
Query OK, 2 rows affected (0.01 sec)

mysql> select title, knn_dist() from test where knn ( image_vector, 5, (0.286569,-0.031816,0.066684,0.032926) );
+-----+-----+
| title      | knn_dist() |
+-----+-----+
| yellow bag | 0.28146550 |
| white bag  | 0.81527930 |
+-----+-----+
2 rows in set (0.00 sec)
--- 2 out of 2 results in 0ms ---

mysql> █
```

Embedding computation

- Non-vector databases typically integrate external embeddings.
- Elasticsearch, Opensearch, Typesense enable automatic embedding generation.
- Microsoft's ONNX Runtime library can be used for integration (used by Vespa, Typesense)
- External embedding creation is challenging for users.
- So others are to catch up in embedding support.

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 (<i>in-domain</i>)	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

Conclusions

- Vector search is revolutionizing data retrieval, becoming common functionality of databases.
- Database Landscape Evolution:
 - Emergence of new vector-focused databases.
 - Established databases integrating vector search capabilities.
 - Reflects a growing demand for advanced search functions.
- Indexes like HNSW enhance speed.
- Future of Databases:
 - Transition from just supporting to internally generating embeddings.
 - Simplifying operations, enhancing power and intelligence.
 - Evolving from basic storage to systems that understand and analyze data.
- Paradigm Shift:
 - Vector search is a significant advancement in data management and retrieval.
 - Marks a new, exciting phase in the field.