Connect

redis | Red Hat

Connect

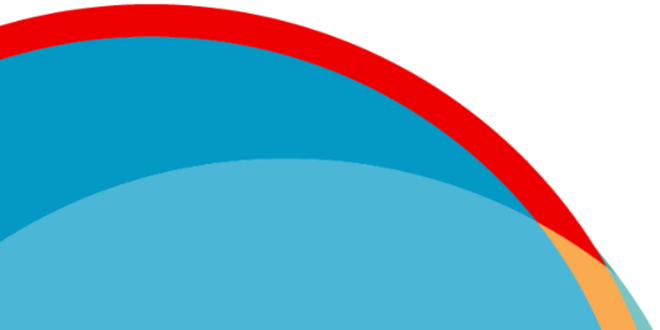
Luigi Fugaro

Sr. Solution Architect
Redis

Redis Enterprise as Vector Database

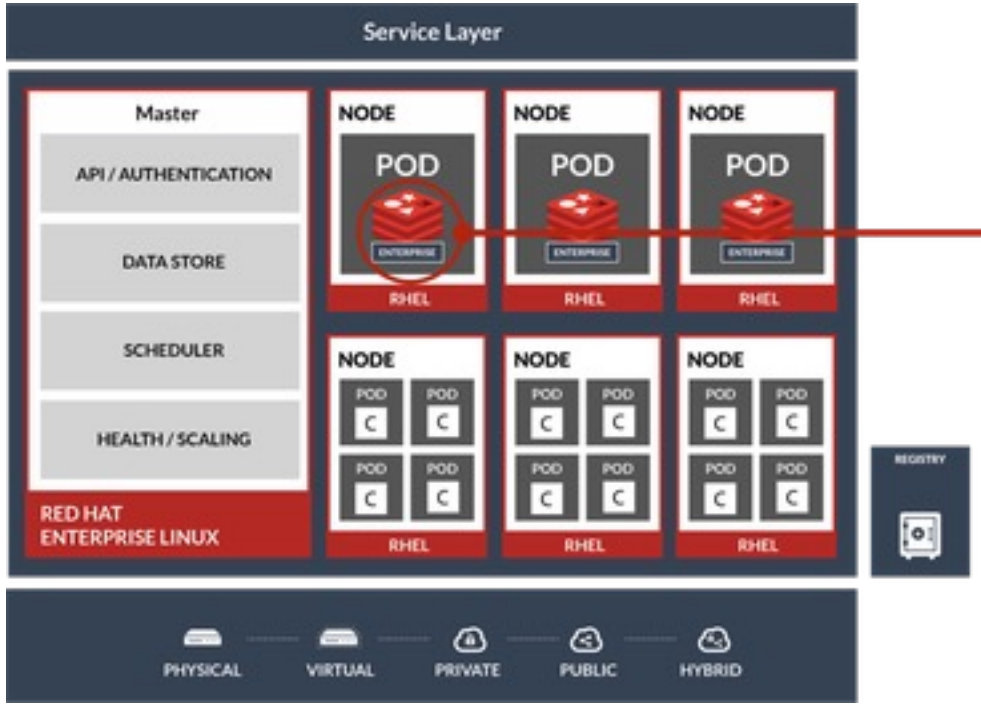
Agenda:

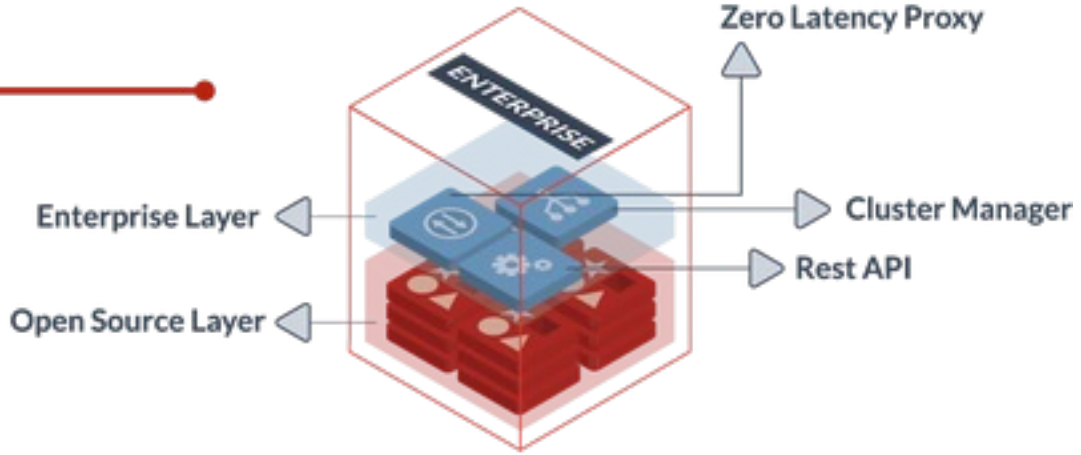
- Native OpenShift Integration
- Introduction to vector embeddings
- Vector Databases
- Redis Enterprise as Vector Database
- The client libraries
- The use cases
  - Text Semantic Search
  - LLM & RAG



redis | Red Hat

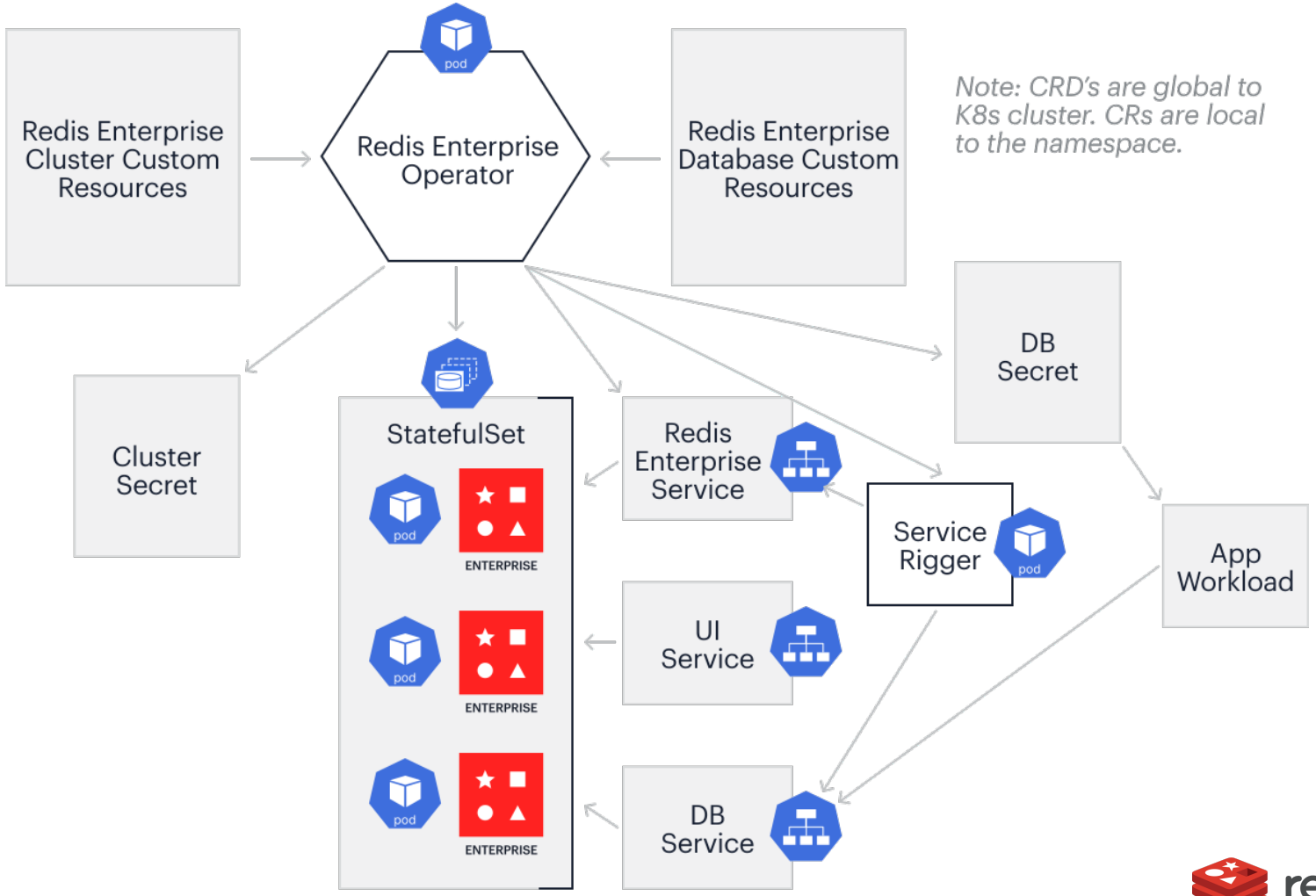
Native OpenShift Integration





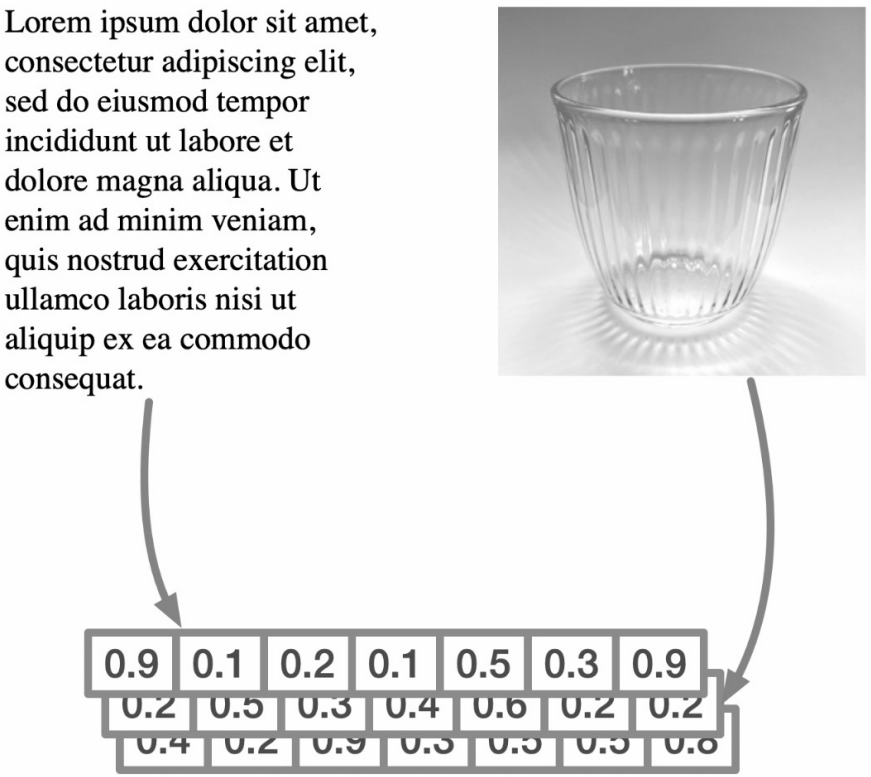
redis | Red Hat

Native OpenShift Integration



Introduction to vector embeddings

- Used to represent unstructured data
- A list of floating-point numbers
- It has fixed size
- Compact and dense data representation
- Produced by feature engineering or deep learning techniques
- Translates perceived semantic similarity to the vector space



redis | Red Hat

Introduction to vector embeddings

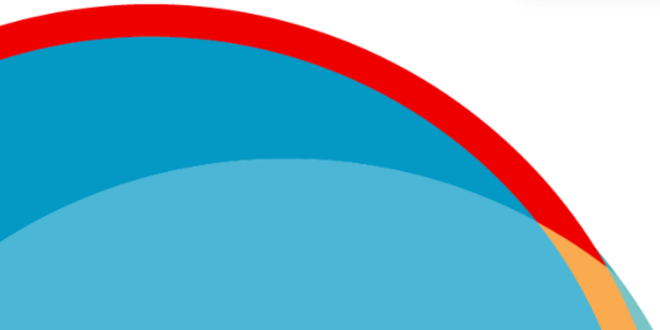
How to create vector embeddings?

Feature Engineering

- Manual creation
- Domain knowledge
- Expensive to scale

Using models

- Models are trained
- Turn objects into vectors
- Dense and high-dimensional

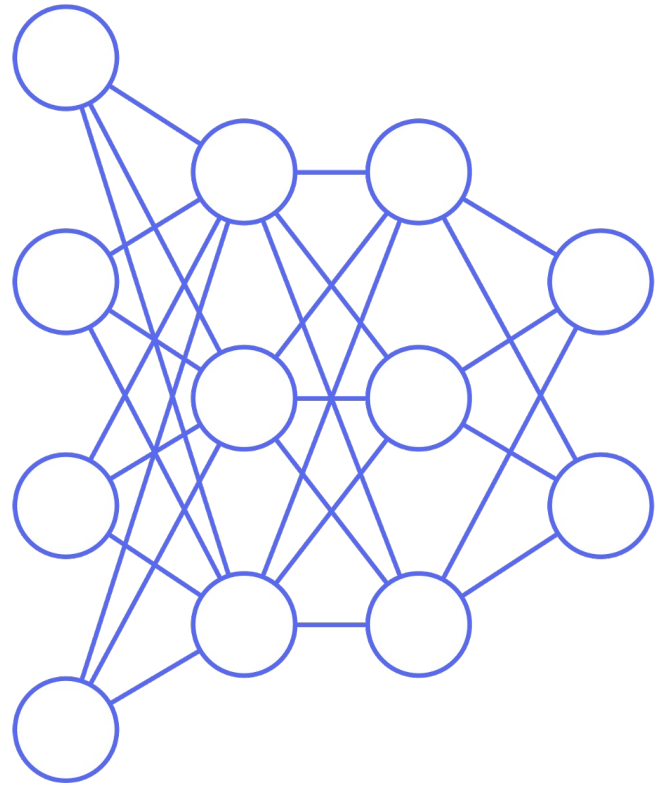


redis | Red Hat

Introduction to vector embeddings

How to create vector embeddings?

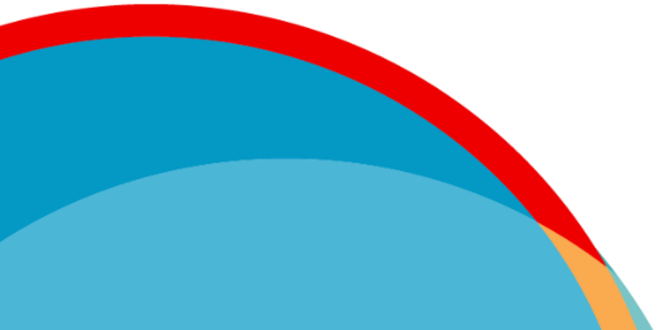
1. The input is transformed into a numerical representation
2. Features are captured by the network
3. A layer is extracted, it provides a dense representation of the features
4. This layer is the embedding and is feasible for similarity search



redis | Red Hat

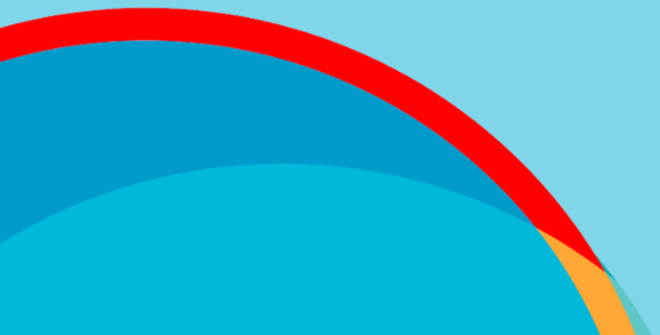
Introduction to vector embeddings

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('sentence-transformers/multi-qa-MiniLM-L6-cos-v1')
embedding = model.encode("This is a technical document, it describes the SID sound chip of the Commodore 64")

print(embedding[:10])
[ 0.00631137 -0.005189 -0.03774299 -0.09026785 -0.05783698  0.01209931 -0.02595172  0.01094836 -0.06051398  0.0521009 ]
```



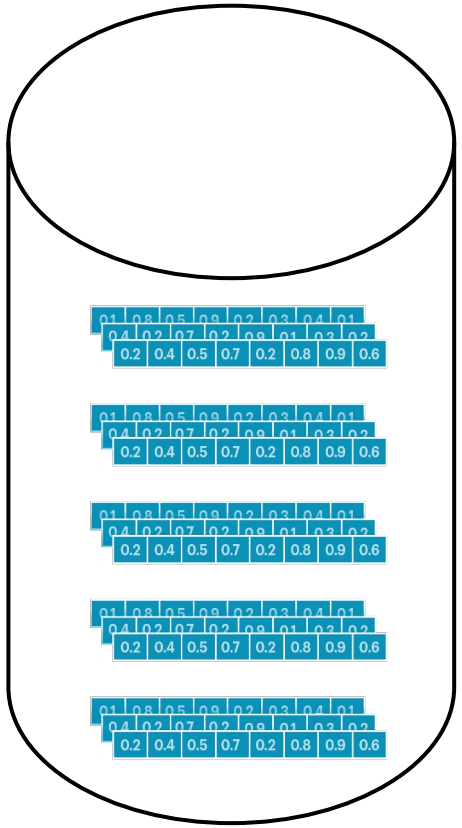
redis | Red Hat

Vector Databases



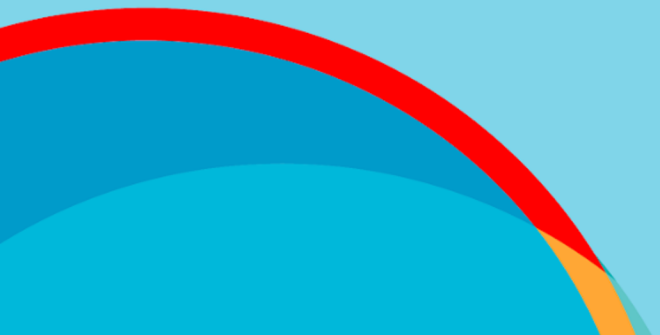
What is a Vector Database?

- A database that can store vectors
- It can index vectors
- It can search the vector space
- Has throughput requirements
- It is scalable and highly available

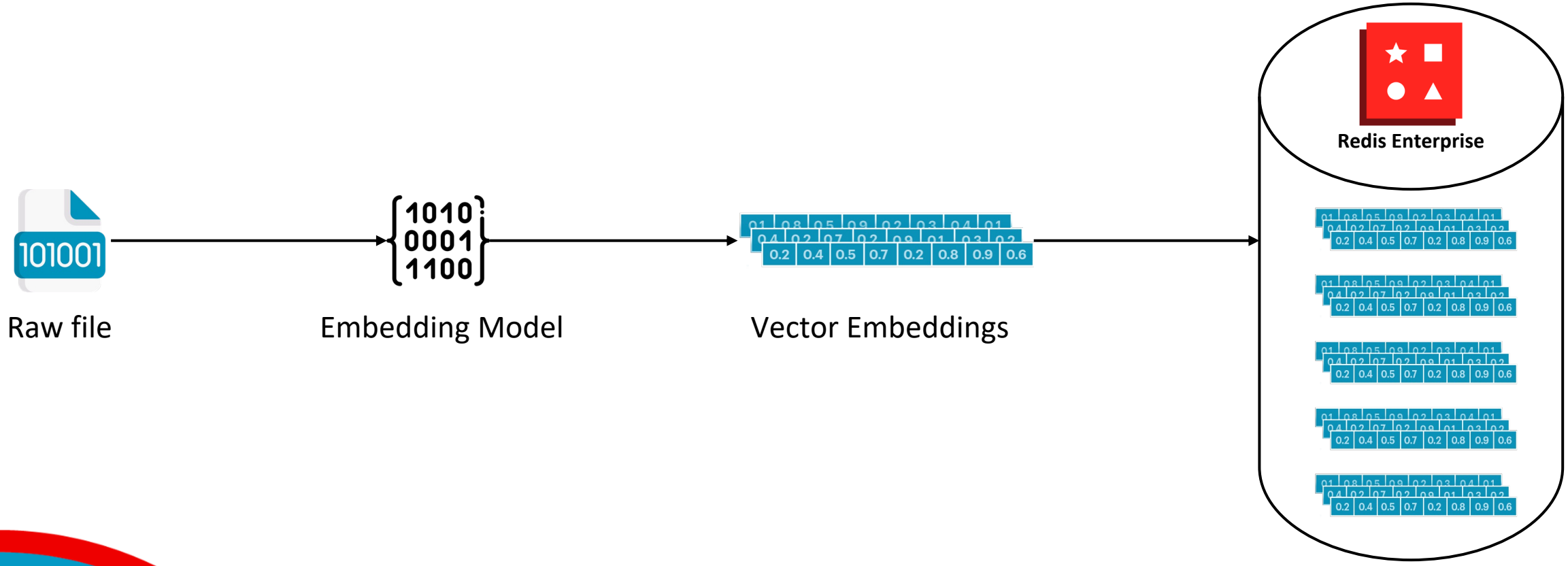


redis | Red Hat

Redis Enterprise as Vector Database



Redis Enterprise as Vector Database



Redis Enterprise as Vector Database

Vector Similarity Search:

- Vector Similarity Search (VSS) is a key feature of a vector database.
- It is the process of finding data points that are similar to a given query vector in a vector database.
- Popular VSS uses include recommendation systems, image and video search, natural language processing, and anomaly detection.

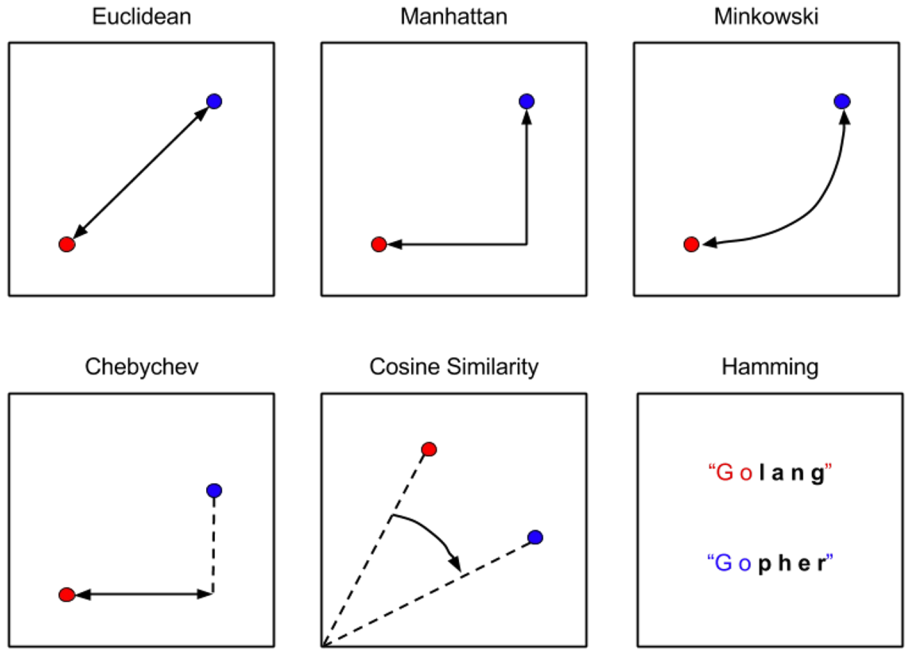
Redis Enterprise

RediSearch
Vector Similarity
Search

redis | Red Hat

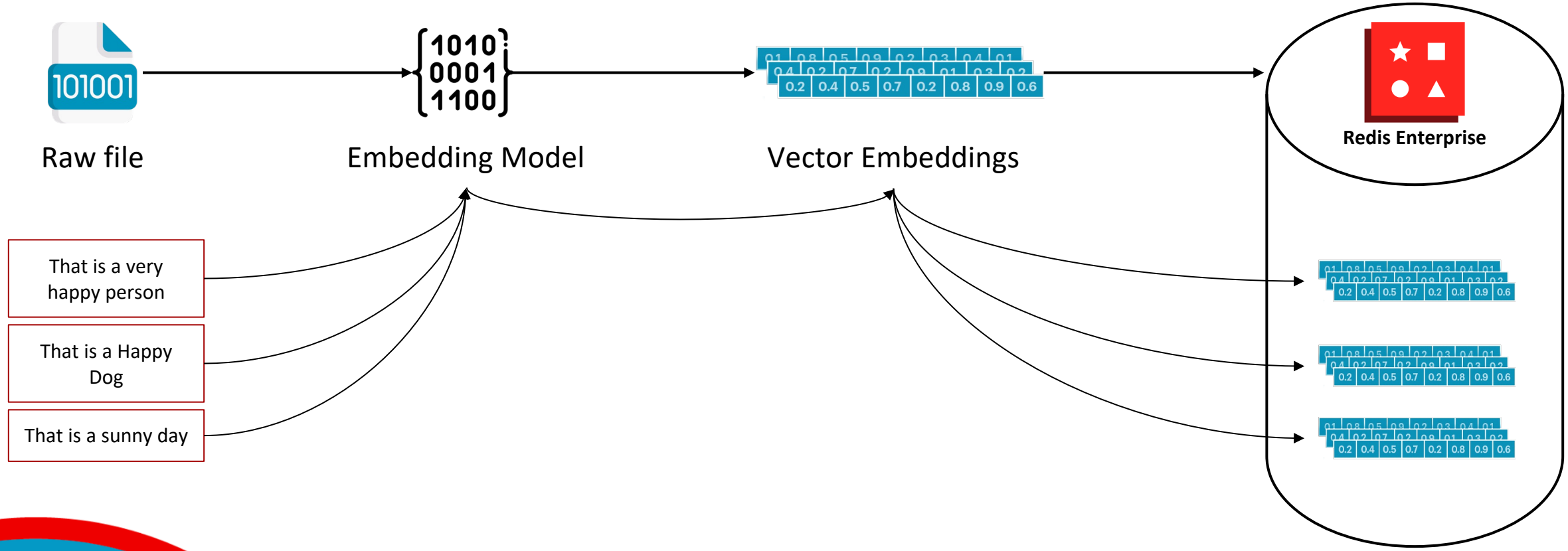
Redis Enterprise as Vector Database

Vector Similarity Search focuses on finding out how alike or different two vectors are. To achieve this in a reliable and measurable way, we need a specific type of score that can be calculated and compared objectively. These scores are known as distance metrics.



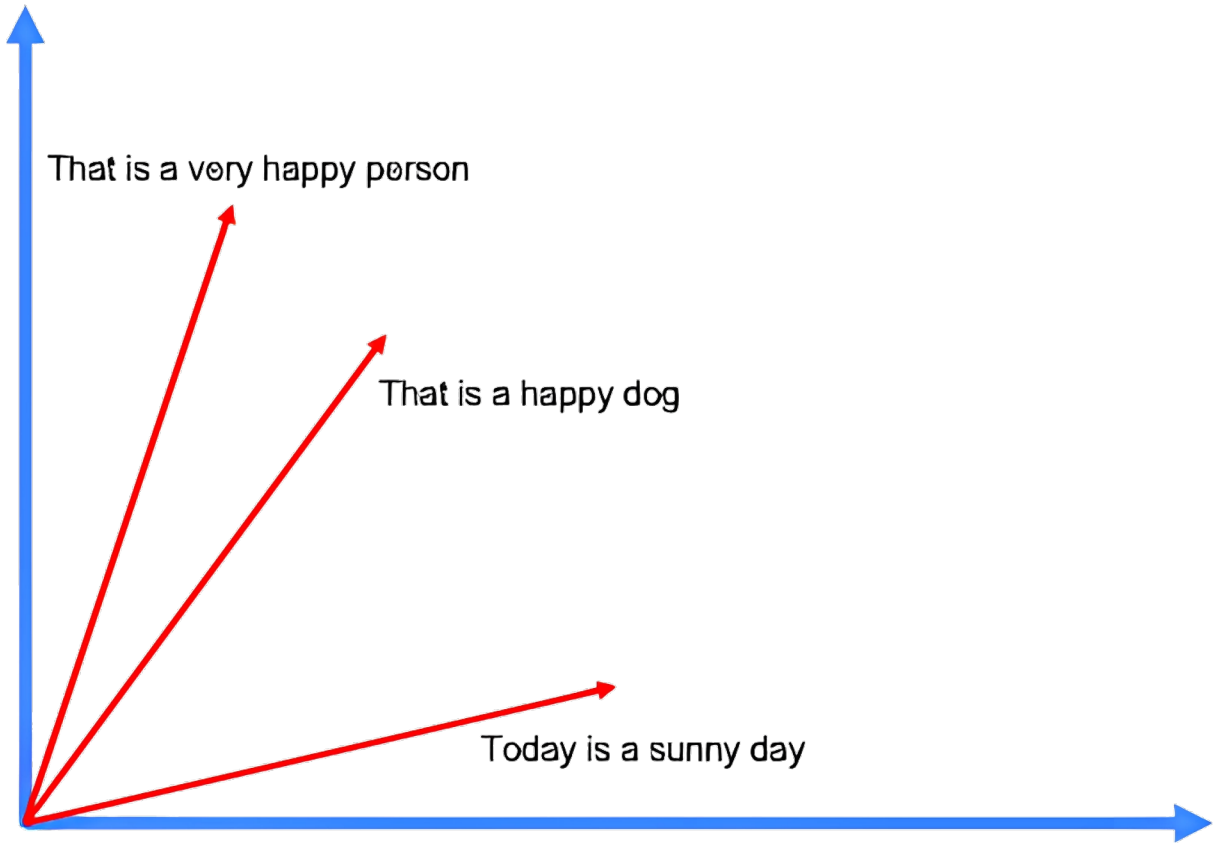
redis | Red Hat

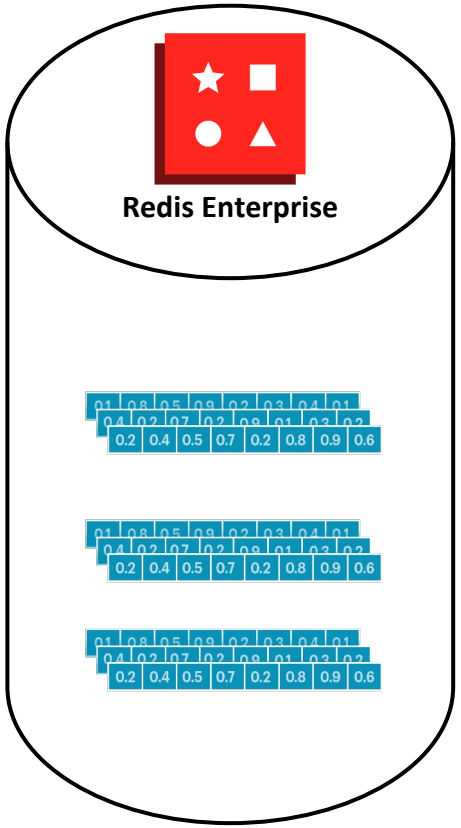
Redis Enterprise as Vector Database



redis | Red Hat

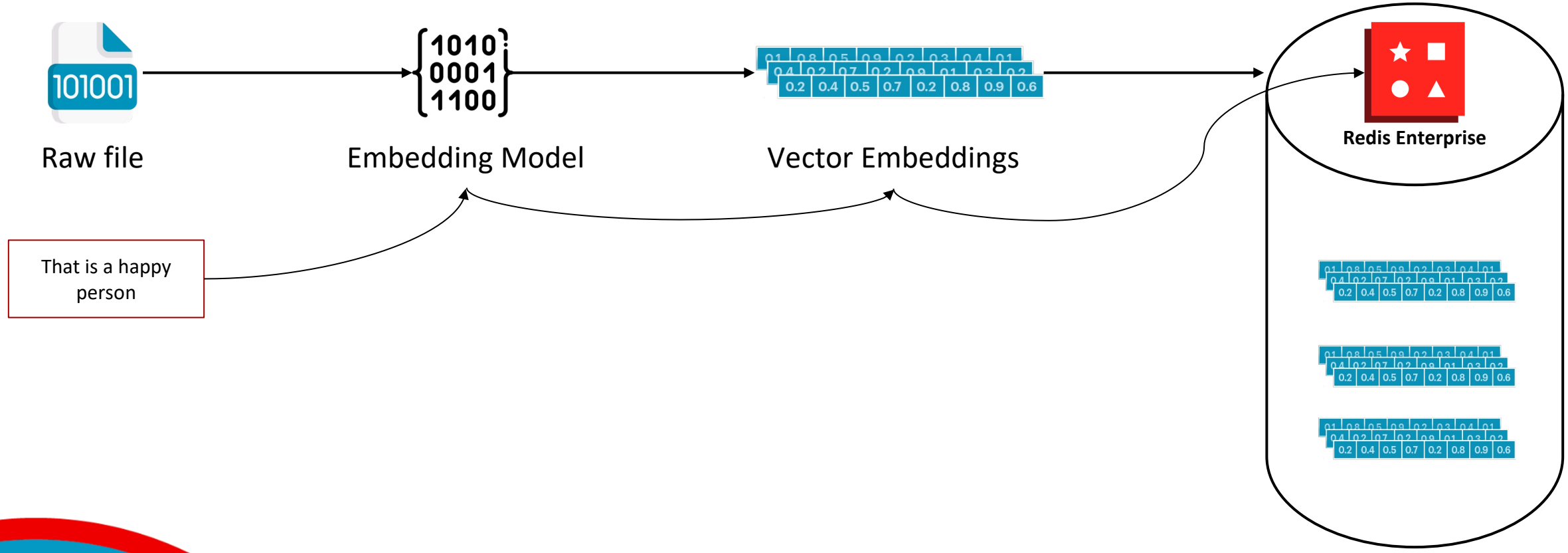
Redis Enterprise as Vector Database





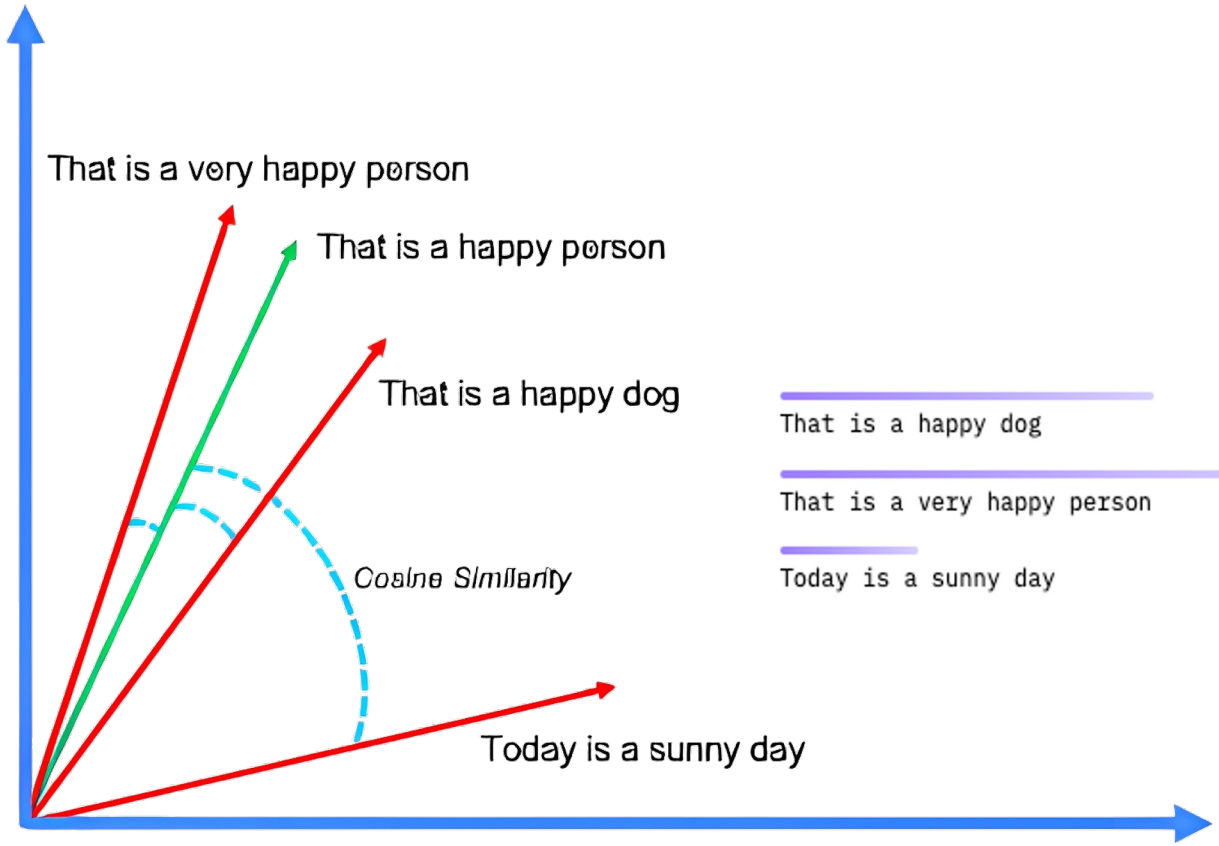
redis | Red Hat

Redis Enterprise as Vector Database



redis | Red Hat

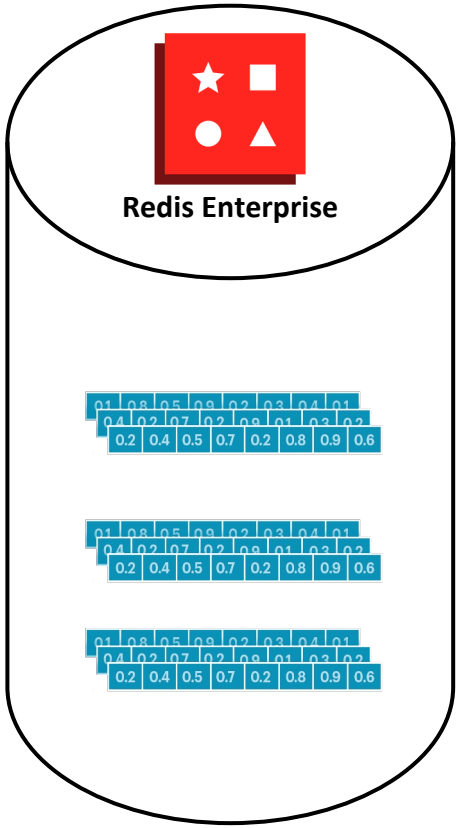
Redis Enterprise as Vector Database



0.695

0.943

0.257



redis | Red Hat

Redis Enterprise as Vector Database

```python
import numpy as np

from numpy.linalg import norm
from sentence_transformers import SentenceTransformer

# Define the model we want to use (it'll download itself)
model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-v2')

sentences = [
 "That is a very happy person",
 "That is a happy dog",
 "Today is a sunny day"
]

sentence = "That is a happy person"
```

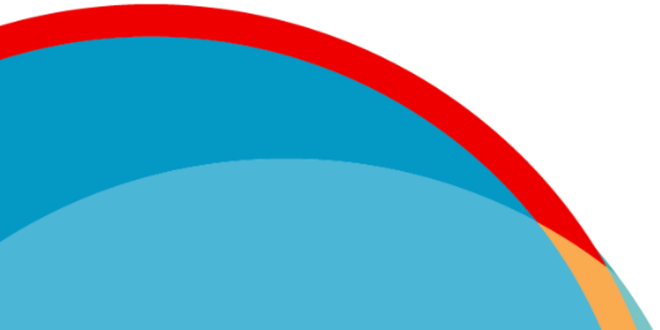
```python
# vector embeddings created from dataset
embeddings = model.encode(sentences)

# query vector embedding
query_embedding = model.encode(sentence)

# define our distance metric
def cosine_similarity(a, b):
  return np.dot(a, b)/(norm(a)*norm(b))

# run semantic similarity search
print("Query: That is a happy person")
for e, s in zip(embeddings, sentences):
  print(s, " -> similarity score = ",
      cosine_similarity(e, query_embedding))
```

```
That is a happy dog                    0.695
That is a very happy person            0.943
Today is a sunny day                   0.257
```

redis | Red Hat

Redis Enterprise as Vector Database

```
from redis import Redis
from redis.commands.search.field import VectorField, TagField

def create_flat_index(redis_conn: Redis, number_of_vectors: int, distance_metric: str='COSINE'):

                image_field = VectorField("img_vector","FLAT", {
              "TYPE": "FLOAT32",
              "DIM": 512,
              "DISTANCE_METRIC": distance_metric,
              "INITIAL_CAP": number_of_vectors,
              "BLOCK_SIZE": number_of_vectors})
                redis_conn.ft().create_index([image_field])
```



redis | Red Hat

Redis Enterprise as Vector Database

```python
def search_redis(
    redis_conn: redis.Redis,
    query_vector: t.List[float],
    return_fields: list = [],
    k: int = 5,
) -> t.List[dict]:
    # Prepare the Query
    base_query = f'*=>[KNN {k} @embedding $vector AS vector_score]'
    query = (
        Query(base_query)
         .sort_by("vector_score")
         .paging(0, k)
         .return_fields(*return_fields)
         .dialect(2)
    )
    params_dict = {"vector": np.array(query_vector, dtype=np.float64).tobytes()}
    # Vector Search in Redis
    results = redis_conn.ft(INDEX_NAME).search(query, params_dict)
    return [process_doc(doc) for doc in results.docs]
```

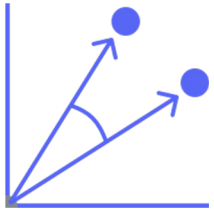
redis | Red Hat

Redis client libraries

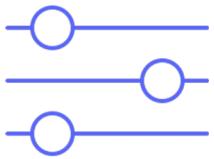


Redis Enterprise as Vector Database

Vector indexing algorithms
Redis Enterprise manages vectors in an index data structure to enable intelligent similarity search that balances search speed and search quality. Choose from two popular techniques, **FLAT** (a brute force approach) and **HNSW** (Hierarchical Navigable Small World - a faster, and approximate approach).



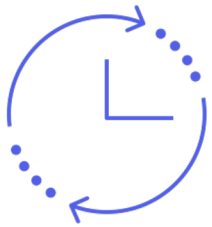
Vector search distance metrics
Redis Enterprise uses a distance metric to measure the similarity between two vectors. Choose from three popular metrics – Euclidean, Inner Product, and Cosine Similarity – used to calculate how "close" or "far apart" two vectors are.



Powerful hybrid filtering
Take advantage of the full suite of search features available in Redis Enterprise query and search. Enhance your workflows by combining the power of vector similarity with more traditional numeric, text, and tag filters. Incorporate more business logic into queries and simplify client application code.

redis | Red Hat

Redis Enterprise as Vector Database

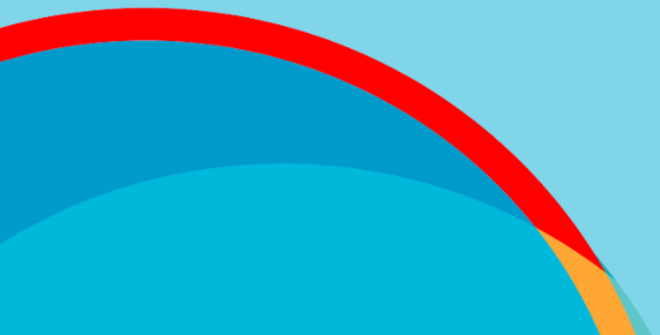


Real-time updates
Real-time search and recommendation systems generate large volumes of changing data. New images, text, products, or metadata? Perform updates, insertions, and deletes to the search index seamlessly as your dataset changes overtime. Redis Enterprise reduces costly impacts of stagnant data.

Vector range queries
Traditional vector search is performed by finding the “top K” most similar vectors. Redis Enterprise also enables the discovery of relevant content within a predefined similarity range or threshold for an alternative, and offers a more flexible search experience.

redis | Red Hat

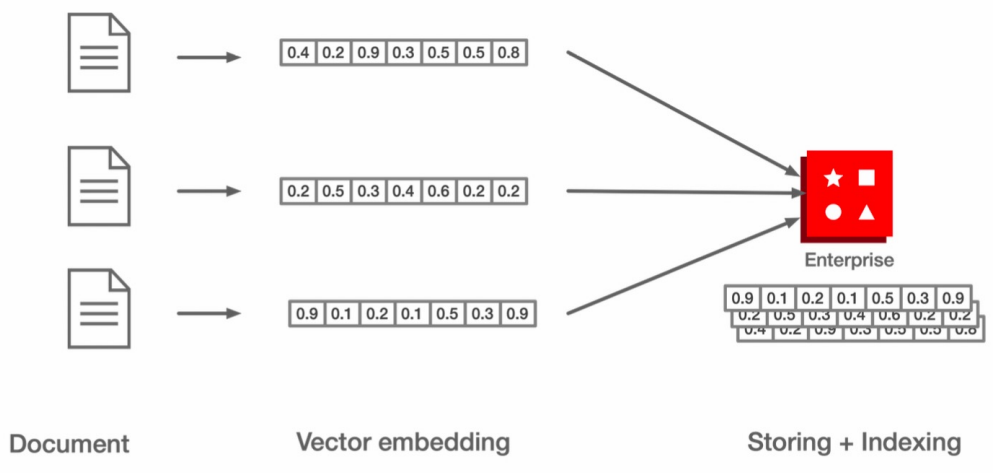
Use cases



Text Semantic Search

Vectorize, store and index your documents

Based on a provided document, I want to get a list of recommendations "you may also want to read..."



- Audit the length of your documents: embedding models consider documents up to a number of words
- Split the documents into chunks if they exceed the supported length
- Store the original documents and its metadata in a hash or JSON
  JSON documents can store and index multiple embeddings

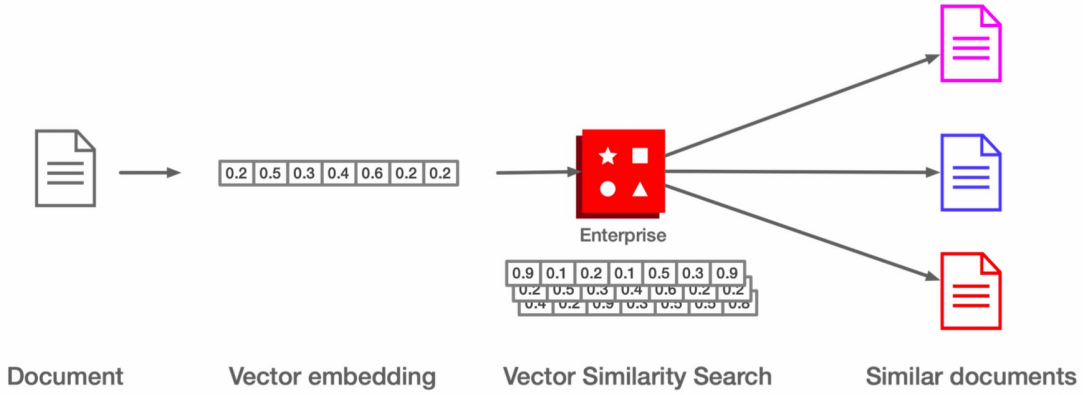
redis | Red Hat

Text Semantic Search

Search your documents

Propose a list of similar documents, books, web pages.

- The current document has already a vector embedding associated
- The embedding is compared to the rest of vector embeddings with VSS
- It is possible to specify the number of results
- It is also possible to perform hybrid search with metadata such and search for recent documents, stock available, categories and more
- You can filter the results by the similarity score using VSS range search

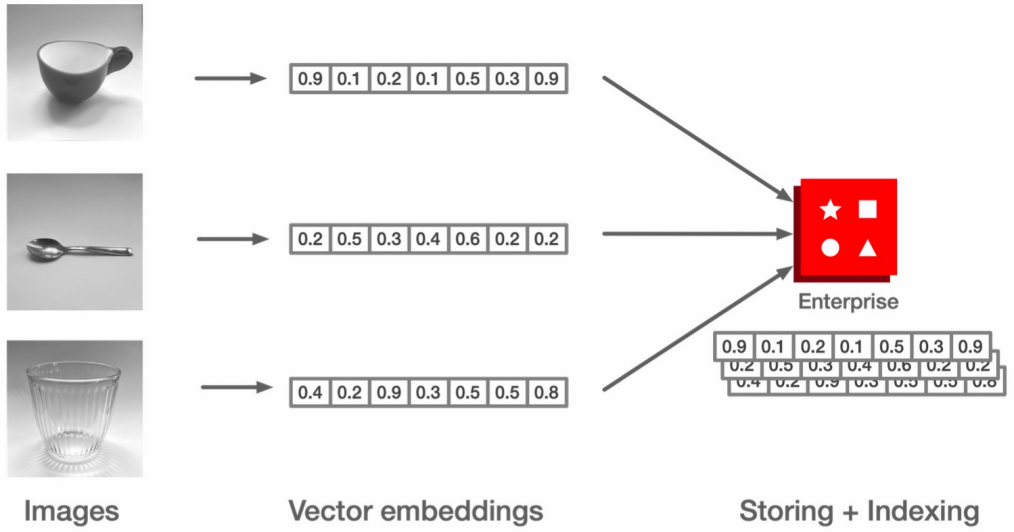


redis | Red Hat

Visual Search

Vectorize, store and index your documents

Based on a provided image, I want to get a list of similar images "check products similar to..."



- Convert the images to the vector embedding using a suitable model (Resnet, Densenet...)
- Store the embedding together with metadata, images are usually in the file system
- You can choose between storing the embeddings in hash or JSON documents.
  JSON documents can store and index multiple embeddings

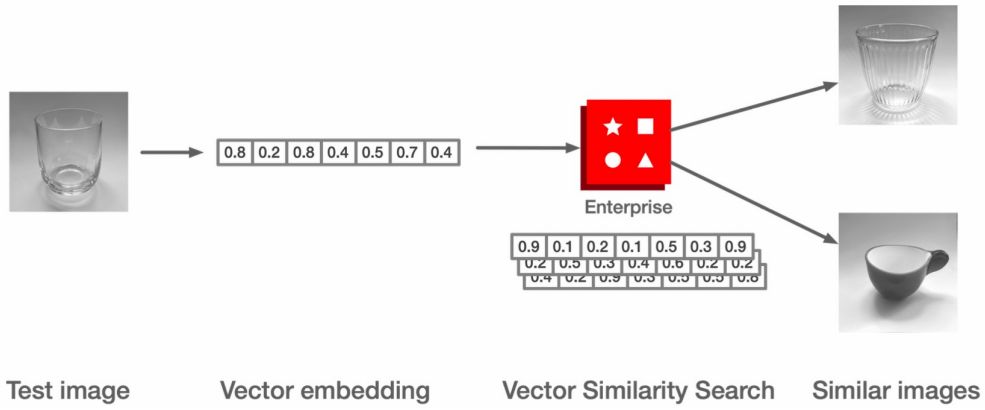
redis | Red Hat

Visual Search

Search your images

Propose a list of similar products, faces, pictures in general.

- Documents store metadata and the embedding for the image
- The embedding is compared to the rest of vector embeddings with VSS
- It is possible to specify the number of results
- It is also possible to perform hybrid search with metadata such and search for recent documents, stock available, categories and more
- You can filter the results by the similarity score using VSS range search



redis | Red Hat

Large Language Models - LLM

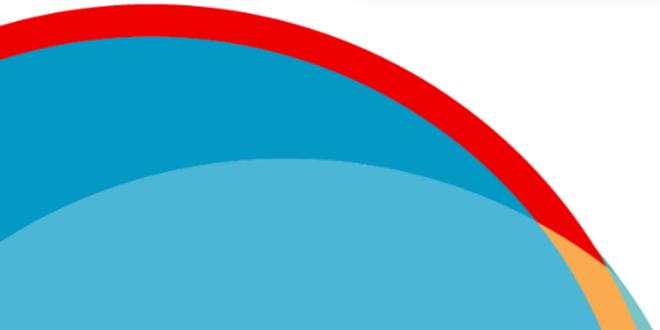
Motivation

Fine Tuning

- Teach the model from your data
- Higher task-specific performance
- Resolves prompts size limitations
- Higher accuracy than RAG
- Fresh knowledge needs retraining

Retrieval Augmented Generation

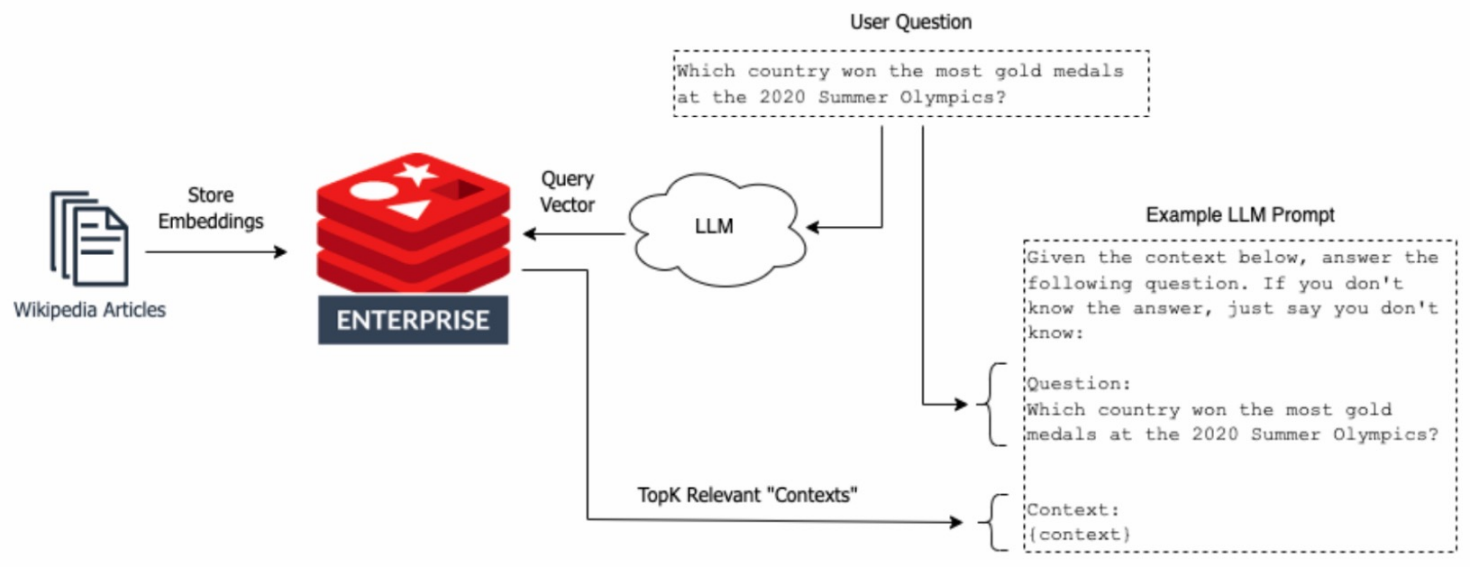
- Incorporate external knowledge sources via retrieval
- Extend the LLM with your knowledge
- Works with your latest data
- Prompt engineering is crucial
- Manages fresh knowledge immediately



redis | Red Hat

Large Language Models - LLM

Context retrieval for Retrieval Augmented Generation (RAG)



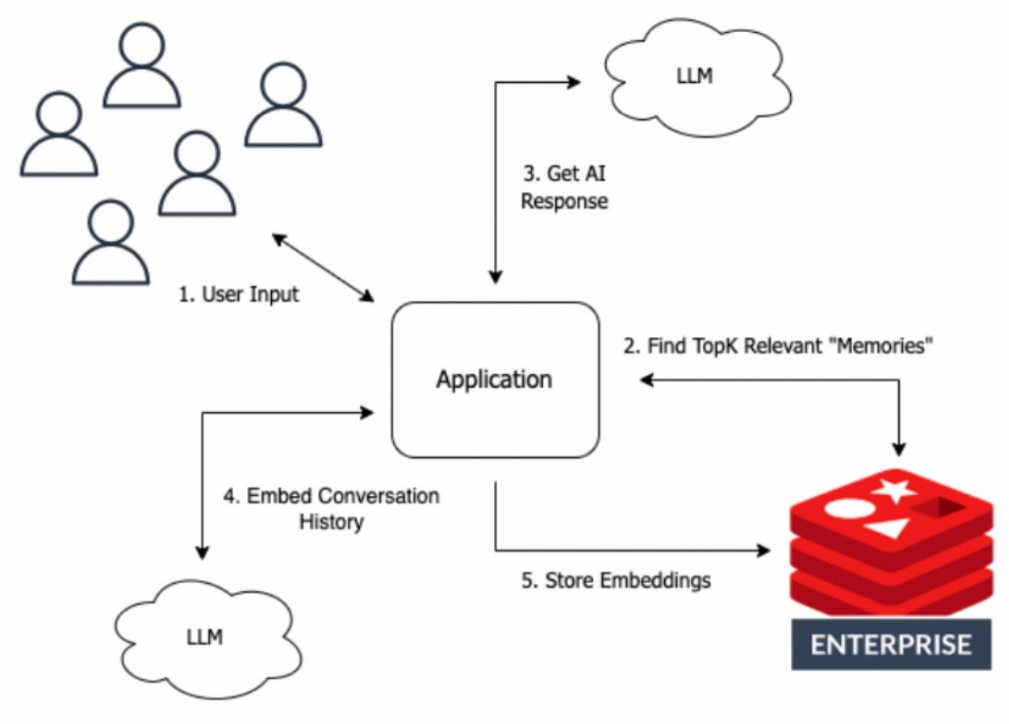
Pairing Redis Enterprise with Large Language Models (LLM) such as OpenAI's ChatGPT, you can give the LLM access to external contextual knowledge.

- Enables more accurate answers and prevents model 'hallucinations'.
- An LLM combines text fragments in a (most often) semantically correct way.

redis | Red Hat

Large Language Models - LLM

LLM Conversion Memory



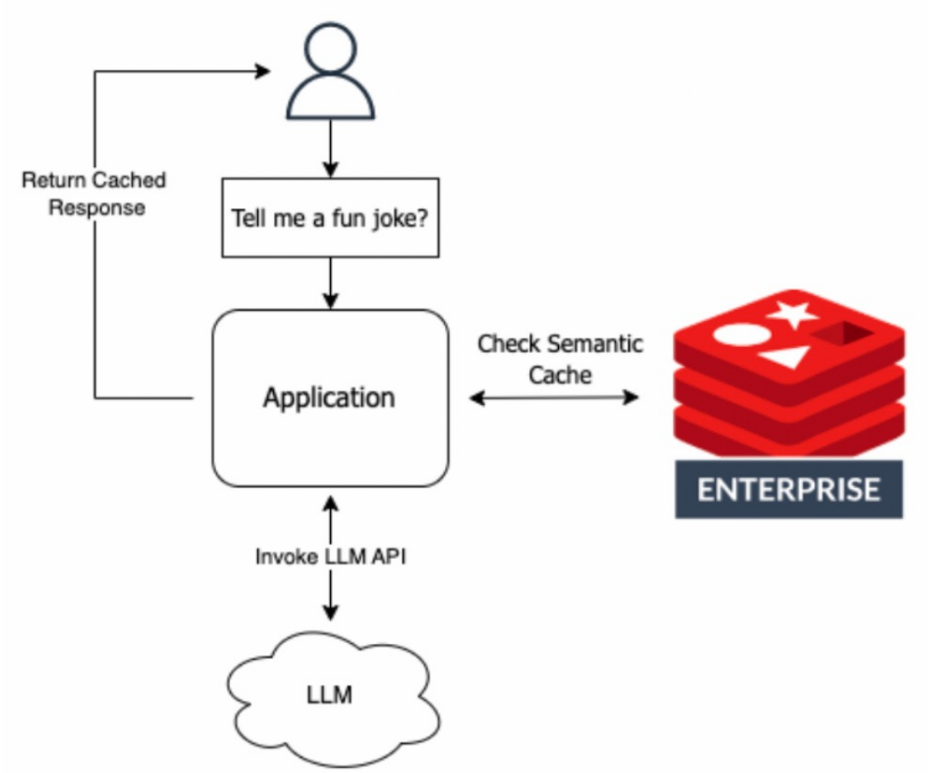
The idea is to improve the model quality and personalization through an adaptive memory.

- Persist all conversation history (memories) as embeddings in a vector database.
- A conversational agent checks for relevant memories to aid or personalize the LLM behaviour.
- Allows users to change topics without misunderstandings seamlessly.

redis | Red Hat

Large Language Models - LLM

Semantic caching



Because LLM completions are expensive, it helps to reduce the overall costs of the ML-powered application.

- Use vector database to cache input prompts
- Cache hits evaluated by semantic similarity

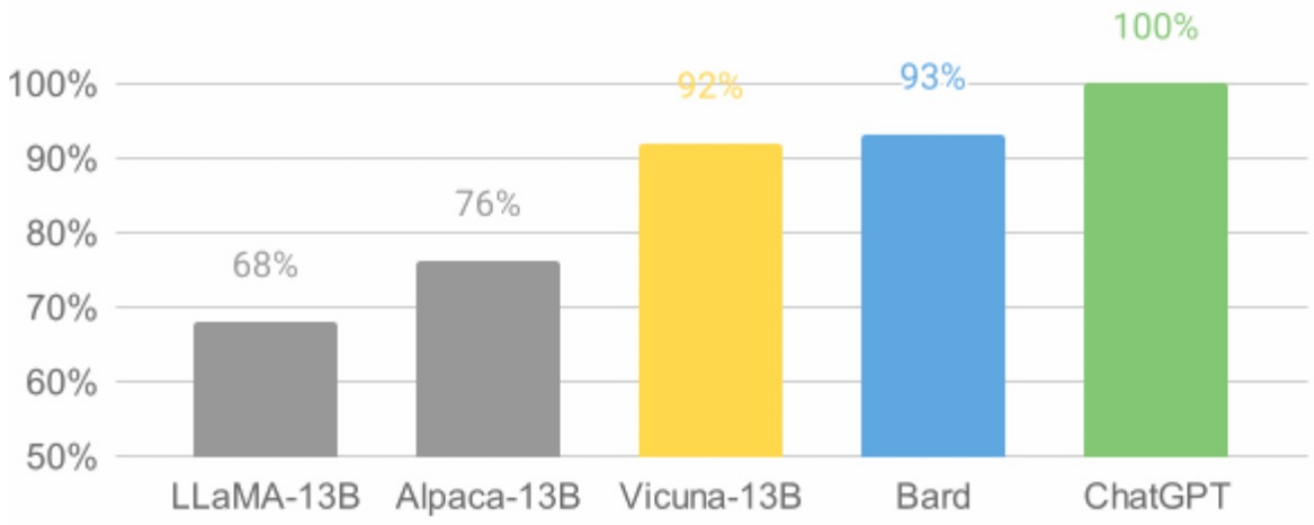
redis | Red Hat

Retrieval Augmented Generation

Choose your domain and prepare your data

Connecting your data is not easy, but Redis comes to the rescue. But before starting, you should answer a few questions.

- Who is the target of your service, what data can you offer?
- What LLM do you plan to use, local or as-a-service?
  - OpenAI
  - Llama
  - Bard
  - Vicuna
- What embedding model are you planning to adopt?
  - HuggingFace
  - OpenAI
  - Cohere
- Planning to use a framework (LangChain, LlamaIndex...)?
- Are you storing and sending the context on every interaction?
- Planning to setup a semantic cache or a conversation memory?



Relative Response Quality Assessed by GPT-4* (Vicuna)

redis | Red Hat

Retrieval Augmented Generation

Choose your domain and prepare your data

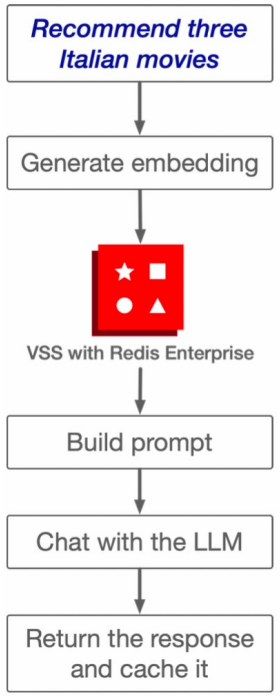
Generation. Chat with your data.

When your data is loaded, indexed and you have completed the integration of your codebase with the chosen LLM, when the user asks a question the following happen:

- The question in natural language is turned into an embedding
- Using VSS and based on the question, related content is retrieved from Redis Enterprise 3. The prompt is built based on the results
- The prompt is sent to the LLM and the response is returned to the user

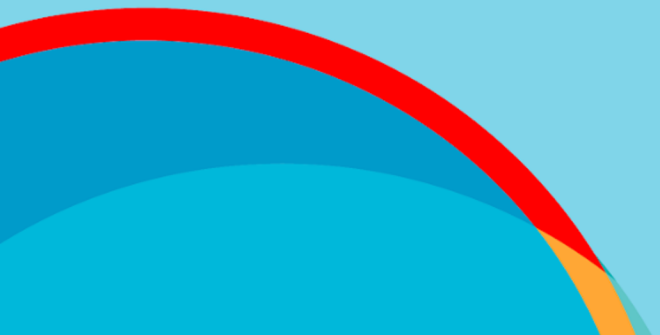
Additionally:

- You may cache the response in Redis Enterprise
- You can store the context in Redis Enterprise and reuse it in a conversation
- You can create complex logic against your entire data set using function calling



redis | Red Hat

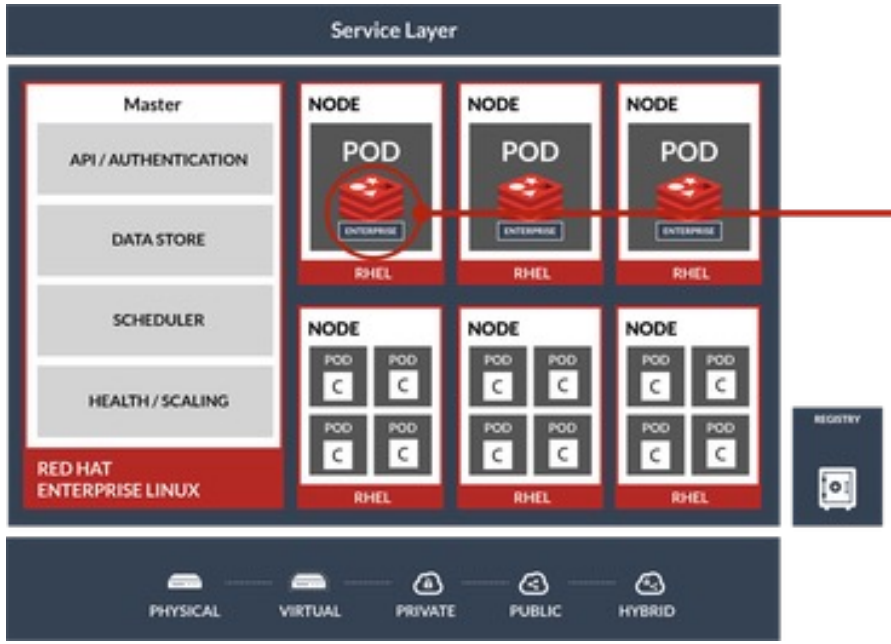
So, why Redis Enterprise?

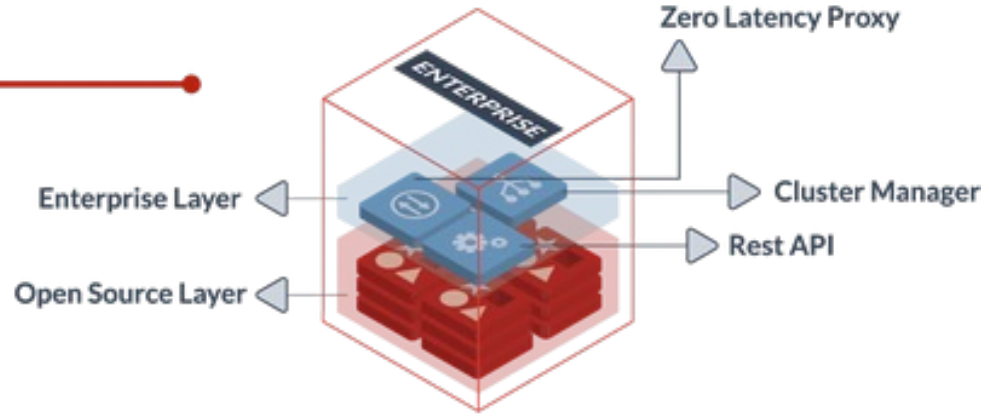


Why Redis Enterprise?

Benefits to Customers:

- Certified to interoperate with Red Hat OpenShift, following best practices for Kubernetes and containerization, and portability across clouds.
- Simple to transact, manage and control enterprise software with one bill from IBM through the Red Hat Marketplace and automated deployment to any cloud.

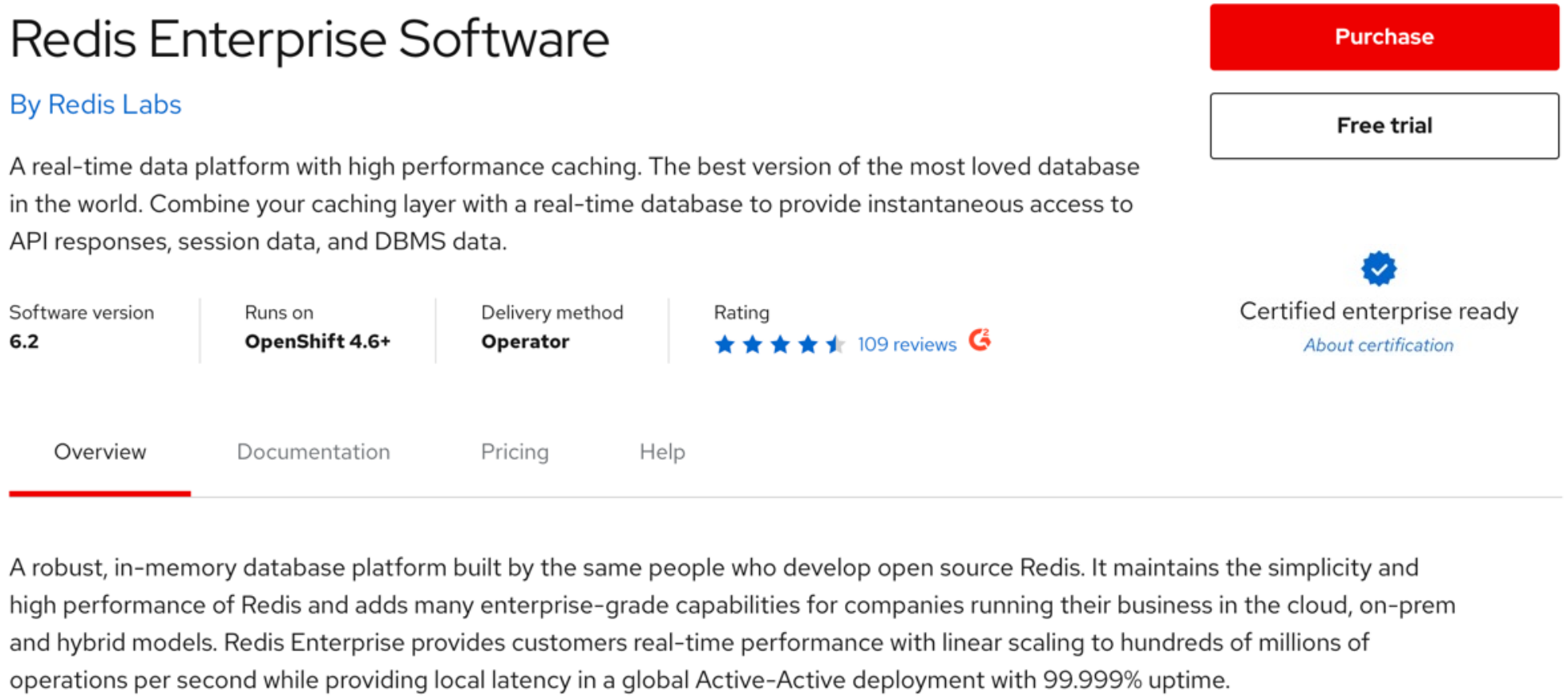




Why Redis Enterprise?

Benefits to Customers:

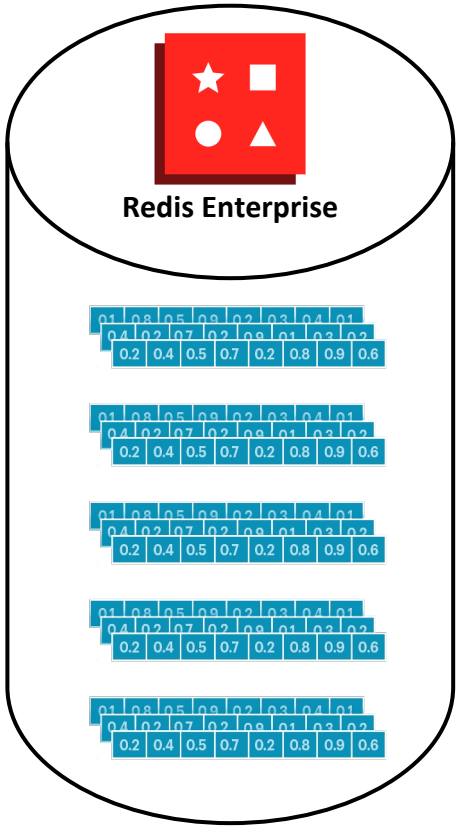
- Certified to interoperate with Red Hat OpenShift, following best practices for Kubernetes and containerization, and portability across clouds.
- Simple to transact, manage and control enterprise software with one bill from IBM through the Red Hat Marketplace and automated deployment to any cloud.



redis | Red Hat

Why Redis Enterprise?

- Native OpenShift Integration
- You already have it
- Simple and efficient
- Highly performant
- Highly scalable
- One solution for vectorizing, indexing, searching and caching
- It's the only database that doesn't need a cache
- Runs everywhere

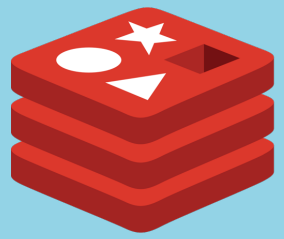


redis | Red Hat

Red Hat Summit

Connect

Q&A?



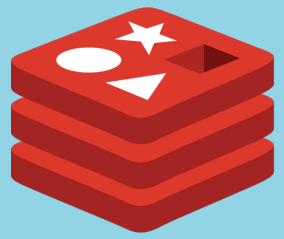
redis

Red Hat

Red Hat Summit

Connect

Thank you



redis

Red Hat