SEMANTIC SEARCH BASED ON EMBEDDING MODELS

Vanya Lazarova[1]
e-mail: vlazarova@unwe.bg

Abstract

In this paper, very briefly the embedding models and the capabilities they provide for semantic search are introduced. In the paper is also presented the workflow of semantic search and the Semantic search in SQL database that are extended with data type VECTOR and the operations with vectors. The benefit for the end user is that he can find relevant documents in his organization's database by comparing query embeddings and documents.

Key words: LLMs, **vector search, vector databases, embedding vectors**

JEL: C30

1. Embedding models

The embedding models are kinds of large language models (LLMs) that generate for a given text (it could be a single sentence, but also could be a business document, journal paper, chapter of book, and so on) a vector that represents the coordinates of this text in the multidimensional space of the model.
Models are named „language” or „linguistic“, but some LLMs can represent images, audio and video objects, not only text.
We will consider further exclusively models that represent text.

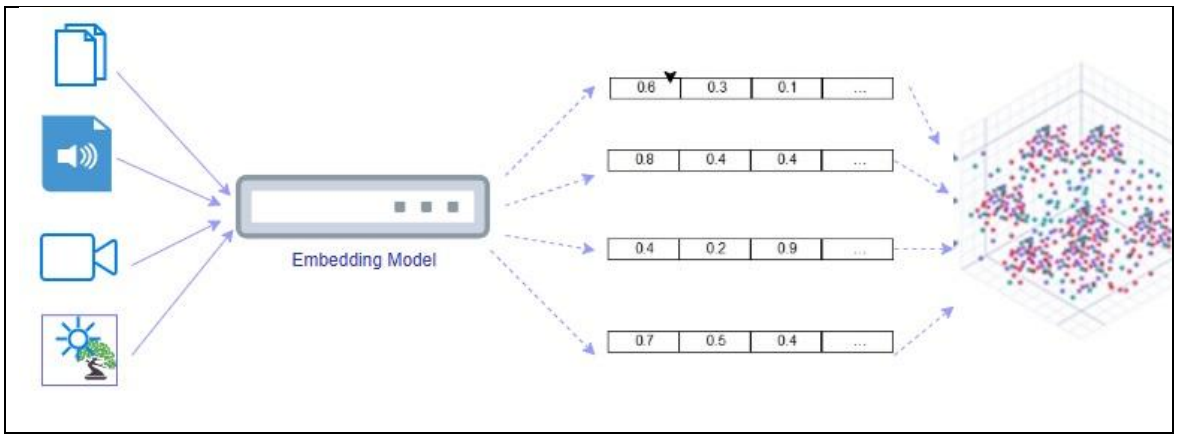


Fig. 1. Embedding models can represent different objects – text, audio, video, images

The embedding vectors capture semantic meaning and context through numerical representations of data. Data with similar semantic meaning have embeddings that are closer together. Embeddings enable a wide range of applications, including:

1) **Search with queries on natural language:** Find relevant documents within large databases, like legal document retrieval or enterprise search, by comparing the embeddings of queries and documents.
2) **Retrieval-Augmented Generation (RAG):** Enhance the quality and relevance of generated text by retrieving and incorporating contextually relevant information into the context of a model.
3) **Clustering and Categorization:** Group similar texts together, identifying trends and topics

[1] Доц. д-р към катедра ИТК, УНСС, email: vlazarova@unwe.bg

within your data.

4) **Classification:** Automatically categorize text based on its content, such as sentiment analysis or spam detection.

5) **Text Similarity:** Identify duplicate content, enabling tasks like web page deduplication or plagiarism detection.

In the paper we will consider the usage of embedding vectors, which can be applied in almost any information systems of small and middle companies - search in databases with queries on natural language.

The leaderboard of the site **huggingface.co** https://huggingface.co/spaces/mteb/leaderboard (8) compares 100+ text embedding models across different languages and task types.

In this paper we recommend the multilanguage model of Google, **gemini-embedding-exp-03-07**. It generates embedding vectors with the following characteristics:

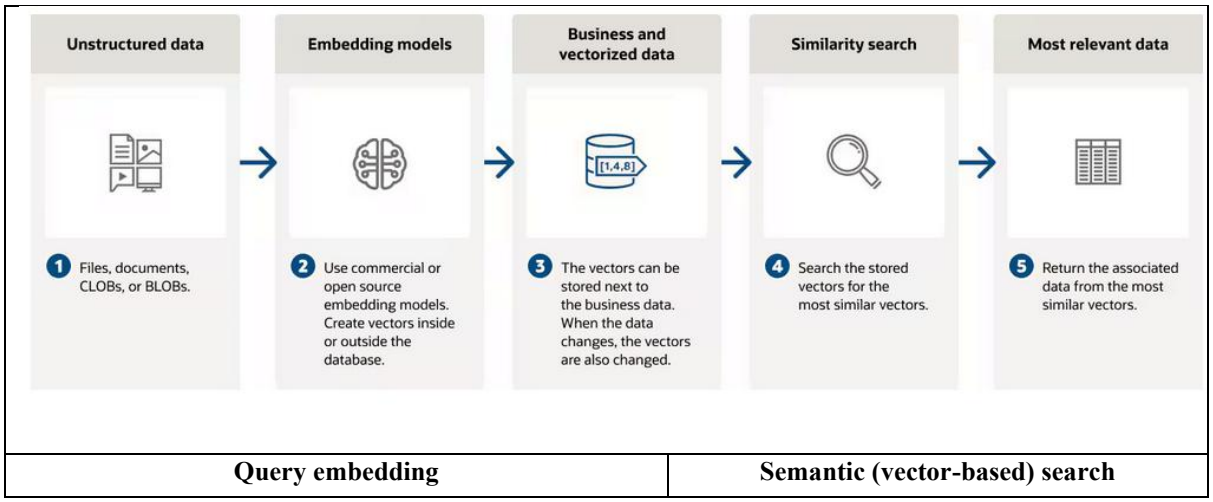
- **Dimensionality:** Each embedding vector consists of **3,072 dimensions**. It allows truncating the original 3,072-dimensional vectors to a smaller size, enabling a balance between accuracy and storage efficiency.
- **Input Token Limit:** The model supports input texts up to **8,192 tokens** in length.
- **Multilingual Support:** The model supports over 100 languages, making it versatile for various linguistic applications.

This model is a**vailable free** through the Gemini API. Details of this embedding model are described on the Google webpage:
https://developers.googleblog.com/en/gemini-embedding-text-model-now-available-gemini-api/.

2. Semantic search based on embedding models

2.1. Workflow of semantic search

Semantic search is more meaningful than simple keyword search, as it seeks to understand the deeper meaning and intent behind the query. LLMs make it possible to go beyond keyword matching. The information about the relationships between words, synonyms, and entities, stored in LLM, allows to interpreter an user’s request articulated in natural language.



Source: https://www.oracle.com/my/database/ai-vector-search/ (7)

Fig.2. The workflow of vector search

The process of vector-based semantic search in database has two phases:
Query embedding: The LLM converts the user's query into an *embedding vecto*r, that is a numerical representation capturing its semantic meaning. If the document is large (e.g. a book), it's typically broken into smaller chunks (e.g. chapters), each mapped as a unique point in a high-dimensional vector space.
Semantic search: The embedding vector of the query is compared to the *embedding* vectors of the documents. The documents with the highest similarity score are provided to the user as matching results.

2.2. Semantic search in SQL database

SQL databases supporting VECTOR type and VECTOR INDEX
Since 2023 most SQL servers are extended with data type VECTOR and the operations with vectors.
The vector functions, vector data type and vector indexes, allow developers to perform efficient searches and analyses on large volumes of vector data.
Embedding models generate typical vectors with 1 536 or 3 072 dimensions.
The embedding model **gemini-embedding-exp-03-07** generates vectors with **3 072** dimensions, but could be truncated to **1 536**

Two SQL databases supporting VECTOR type and VECTOR INDEX

	data type VECTOR(n) n - dimensions of embedding vectors	VECTOR INDEX algorithm
Microsoft SQL Server 2025	VECTOR(n) n: max 32,768 dimensions Optimized for vectors to **2 048** vector elements are of FLOAT32 Each row of Embedding column contains an area of **float32**	HNSW
Oracle database 23ai	VECTOR(n,***element_type***) n: max 32 768 dimensions Optimized for vectors to **4 096** *element_type :* FLOAT32, FLOAT64 Each row of Embedding column contains a **VARBINARY**	HNSW

CREATE table with a column of VECTOR type and a VECTOR INDEX for this column

Microsoft SQL 2025	CREATE TABLE **Documents** (**Id** INT PRIMARY KEY, **Content** NVARCHAR(MAX), **Embedding** VECTOR(1536)); CREATE **VECTOR INDEX** idx_Documents_Embedding ON Documents (Embedding) WITH (DISTANCE_FUNCTION = 'COSINE' -- or 'EUCLIDEAN');
Oracle Database 23ai	CREATE TABLE **Documents** (**Id** NUMBER PRIMARY KEY, **Docuntent** CLOB, **Embedding** VECTOR(1536)); CREATE **VECTOR INDEX** idx_Documents_Embedding ON Documents (Embedding)

	WITH PARAMETERS (dimensions = 1536, metric = 'COSINE' -- or 'L2');

The vector search is SQL is performed by operator (v1) <-> (v2) or function VECTOR_DISTANCE(v1,v2, search_mode).
These function and operator are not standard in SQL. The considered SQL databases differ in the way the choice between exact and approximate search is pointed.

	operator (v1) <-> (v2)	function VECTOR_DISTANCE(v1,v2, search_mode)
• **Microsoft SQL 2025**	(v1) <->(v2) search mode depends on the VECTOR INDEX	search_mode: **COSINE** for cosine distance **EUCLIDEAN** for Euclidean (L2) distance
• **Oracle Database 23ai**	(v1) <-> (v2) search mode depends on the VECTOR INDEX	search_mode: **COSINE** for cosine distance **L2** for Euclidean (L2) distance

In both SQL servers, the **approximate search** requires two conditions:

1) VECTOR INDEX must be created
2) In the SELECT statement, a restriction on the number of the selected rows must be defined:
   - TOP n -- in Microsoft SQL 2025
   - FIRST n -- in Oracle Database 23ai

The choice between exact and approximate search is different in these two SQL servers using books database.

- **SELECT in Microsoft SQL 2025**

```
-- Exact search:  function VECTOR_DISTANCE() must be used
SELECT TOP 5  BookId,  Title,
     VECTOR_DISTANCE ( Book_embedding_vector, @query_vector, 'COSINE') as Distance
FROM                                                                    books
ORDER BY Distance;
```

```
-- Approximate search:  operator <-> must be used.
SELECT TOP 5  BookId,  Title,
FROM books
ORDER BY Book_embedding_vector <-> @query_vector;
```

- **SELECT in Oracle SQL Database 23ai**

```
-- Exact search: In FETCH clause the key word   EXACT must be used.
SELECT  BookId,  Title,
     VECTOR_DISTANCE ( Book_embedding_vector, :query_vector, COSINE ) as Distance
FROM books
ORDER BY Distance
FETCH EXACT FIRST 5 ;
```

```
-- Approximate search: In FETCH clause the key word   APPROX must be used.
SELECT   BookId,  Title,
     VECTOR_DISTANCE ( Book_embedding_vector, :query_vector, COSINE ) as Distance
FROM books
ORDER BY Distance
FETCH  APPROX  FIRST 5 ;
```

3. Conclusion

In the text above, we have indicated a wide range of applications that can use Embeddings. We only looked at a small part – search with queries on natural language and find relevant documents within large databases.

There are some concepts used in the code - Euclidean vector distance and Cosine vector distance - that are not explained in details, but this is beyond the scope of this report. More detailed information on the matter can be obtained from the article „Artificial Intelligence as a Service“ (5).

We hope this will be useful for practical developments in the field of artificial intelligence as a service.

References

1. Hristov, G. (9 2024 r.). Improving the Quality of Financial Information Through Machine Learning. Economic Alternatives, pp. 529-540.
2. Belev, I. (2023). Object-Oriented Ecm Approach, Innovative Information Technologies for Economy Digitalization (IITED), University of National and World Economy, Sofia, Bulgaria, issue 1, pages 7-14.
3. Mihova, V. (2023). Expanding The Possibilities Of Information System For Working With Voice, Innovative Information Technologies for Economy Digitalization (IITED), University of National and World Economy, Sofia, Bulgaria, pages 152-157.
4. Velkova, I. (2023). Reducing Advertising Fraud In Digital Marketing With Artificial Intelligence. Innovative Information Technologies for Economy Digitalization (IITED) (pages 79-88). Sofia: University of National and World Economy, Bulgaria.
5. Lazarova, V. (2025). Artificial Intelligence as a Service. Ikonomiceski i Sotsialni Alternativi, (2), 128-141.
6. Stefanov, Geno (2024). Integrating Amazon Web Services (AWS) and Hadoop for Big Data processing, Innovative Information Technologies for Economy Digitalization (IITED), University of National and World Economy, Sofia, Bulgaria, pages 154-161.
7. oracle.com. Oracle AI Vector Search. Retrieved from https://www.oracle.com/asean/database/ai-vector-search/
8. huggingface.co. Embedding Leaderboard. Retrieved from https://huggingface.co/spaces/mteb/leaderboard