#### **Integrating Vector Databases with LLMs**

**Topics Covered**

* + **Introduction to Vector Databases**
  + **Applications and Use Cases of Vector Databases**
  + **Embedding Textual Data for Vector Databases**
  + **Building Semantic Search Applications**
  + **Enhancing LLM Responses with Vector Database Queries**

**Introduction to Vector Databases**

**Definition and Overview:**

A vector database indexes, stores, and provides access to structured or unstructured data (e.g., text or images) alongside its [vector embeddings](https://weaviate.io/blog/vector-embeddings-explained), which are the data's numerical representation. It allows users to find and retrieve similar objects quickly at scale in production.

Because of its search capabilities, a vector database is sometimes also called a vector search engine.

**Characteristics:**

* Stores data in vectorized form, where each data point is represented as a high dimensional vector.
* Enables efficient similarity search and retrieval based on vector distances or similarities.
* Often used in applications requiring complex data analysis, such as machine learning, natural language processing, and recommendation systems.

### 

### 

### **Vector vs. Traditional Databases**

**How Does it Differ from Traditional Relational Databases?**

*Structure:*

* Relational databases organise data into tables with predefined schemas, consisting of rows and columns.
* Vector databases store data as vectors, allowing for flexible and dynamic representation of complex data structures.

*Querying:*

* Traditional relational databases use SQL (Structured Query Language) for querying and manipulation of data.
* Vector databases employ specialised algorithms for similarity search and retrieval based on vector distances or similarities.

*Use Cases:*

* Relational databases are suitable for transactional and structured data, such as financial transactions and customer records.
* Vector databases excel in applications requiring similarity search, such as image recognition, recommendation systems, and text analysis.

**Key features and benefits of vector databases**

*Scalability:*

* Vector databases are designed to handle large scale datasets with millions or even billions of vectors efficiently.
* They offer horizontal scalability, allowing for distributed storage and processing of data across multiple nodes or clusters.

*Real Time Querying:*

* Vector databases enable real time querying and retrieval of similar items or objects, making them ideal for interactive applications and services.

*Versatility:*

* Vector databases can accommodate diverse types of data, including text, images, audio, and structured data, making them versatile for various use cases.

*Enhanced Performance:*

* By leveraging vector representations and specialised indexing techniques, vector databases offer faster query performance compared to traditional databases for similarity search tasks.

*Advanced Analytics:*

* Vector databases support advanced analytical operations, such as clustering, classification, and recommendation, enabling deeper insights and personalised experiences for users

**Applications and Use Cases of Vector Databases**

**1. Similarity Search:**

Vector databases excel in finding data points similar to a given query in high dimensional spaces, enabling various applications such as:

**Ecommerce:** Enhancing product discovery through visually similar product searches.

**Music Streaming:** Recommending songs based on audio features similar to a user's favourites.

**Healthcare Imaging:** Assisting radiologists by retrieving medical images with similar pathologies for analysis.

**2. Recommendation Systems:**

Vector databases support personalised recommendations by matching users with similar items, benefiting industries like:

**Streaming Platforms**: Personalising viewing experiences by recommending content based on viewing history.

**Online Retailers:** Suggesting products based on browsing and purchase history.

**News Aggregators:** Delivering personalised news feeds by matching articles with reader preferences.

**3. ContentBased Retrieval:**

Vector databases enable searching based on content rather than metadata, useful in scenarios such as:

**Digital Asset Management**: Facilitating search and retrieval of media based on content characteristics.

**Legal and Compliance:** Searching documents for contextually related information.

**Academic Research:** Finding scholarly articles similar to researchers' work, even without specific keywords.

**4. Natural Language Processing (NLP):**

Vector databases play a crucial role in NLP tasks by storing and querying text embeddings, supporting applications like:

**Chatbots:** Enhancing conversational agents' responses by retrieving relevant information from large text datasets.

**Document Similarity:** Identifying documents with similar content for summarization or plagiarism detection.

**Sentiment Analysis:** Analysing text data to determine sentiment trends and opinions.

**5. Image and Video Analysis:**

Vector databases facilitate efficient retrieval of images and videos based on visual features, powering applications such as:

**Visual Search:** Allowing users to search for visually similar images or videos.

**Content Moderation:** Identifying and filtering inappropriate content based on visual characteristics.

**Object Detection:** Locating and recognizing objects within images or video frames for various purposes.

**6. Graph Analytics:**

Vector databases support graph data analysis by storing node and edge embeddings, enabling applications like:

**Social Networks:** Finding communities or influential nodes within social graphs for targeted marketing or network analysis.

**Recommendation Systems:** Incorporating social connections or user interactions in recommendation algorithms for improved accuracy.

**Fraud Detection:** Identifying anomalous patterns or suspicious behaviour within transaction networks.

These applications highlight the versatility and importance of vector databases across various industries and domains, enabling advanced data analysis and enhancing user experiences.

**Embedding Textual Data for Vector Databases**

### **Vector Embeddings**

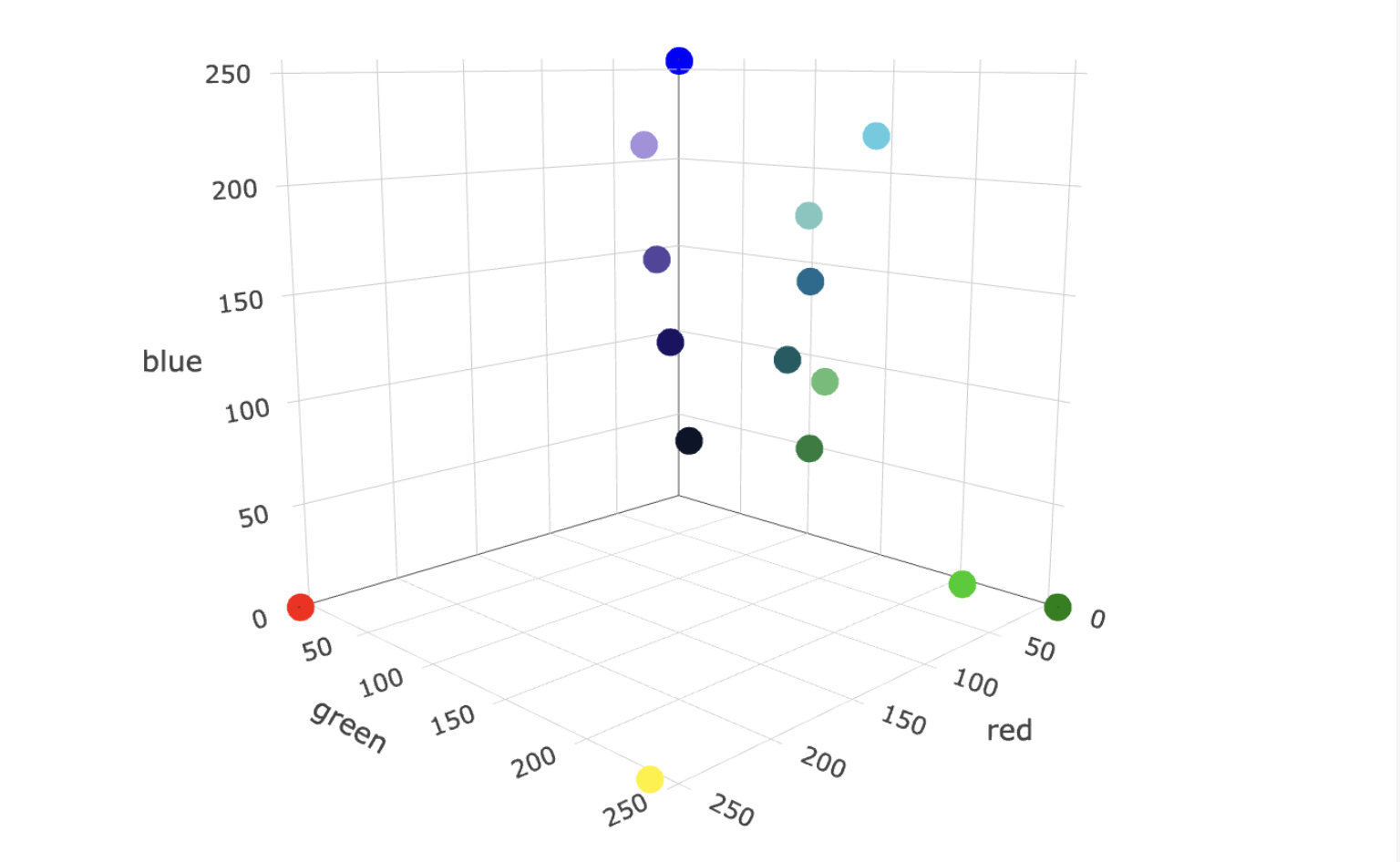
Vector embeddings enable numerical representation of unstructured data, such as images, text, or time series data, preserving semantic meaning. They consist of lists of numbers describing features of the data object. Machine Learning models, like embedding models or vectorizers, generate these embeddings by learning contextual relationships, allowing complex data, like words or sentences, to be represented as vectors.

An example is how we numerically represent colours [in the RGB system](https://huggingface.co/spaces/jphwang/colorful_vectors), where each number in the vector describes how red, green, or blue a colour is. E.g., the following green colour can be represented as [6, 205, 0] in the RGB system.



But fitting more complex data, such as words, sentences, or text, into a meaningful series of numbers isn’t trivial. This is where Machine Learning models come in: Machine Learning models enable us to represent the contextual meaning of, e.g., a word as a vector because they have learned to represent the relationship between different words in a vector space. These types of Machine Learning models that can generate embeddings from unstructured data are also called embedding models or vectorizers.

Below, you can see an example of the three-dimensional vector space of the RGB system with a few sample data points (colours).



Depending on the used embedding model, the data can be represented in different vector spaces, which can differ in the number of dimensions, ranging from tens to thousands, where the precision of the representation increases with increasing dimensions. In the colour example, there are also different ways to represent a colour numerically. E.g., you can represent the previous green colour in the RGB system as [6, 205, 0] or in the CMYK system as [97, 0, 100, 20].

This is why it is important to use the same embedding model for all your data to ensure it is in the respective vector space.

# **Turning words or sentences into embeddings**

A word or sentence can be turned into an embedding (a vector representation) using the OpenAI API. To get an embedding, send your text string to the [embeddings API endpoint](https://platform.openai.com/docs/api-reference/embeddings) along with a choice of embedding model ID (e.g., text-embedding-ada-002). The response will contain an embedding you can extract, save, and use.

In my case, I’m using the Python API. Using this API, you can simply use the code below to turn the word hamburger into an embedding.



If you have a text document, you would turn all the words or sentences from that document into embeddings. Once you’ve done that, you essentially have a semantic representation of the document as a series of vectors. These vectors capture the meaning and context of the individual words or sentences.

**Building Semantic Search Applications Using Open Source Vector database ChromaDB**

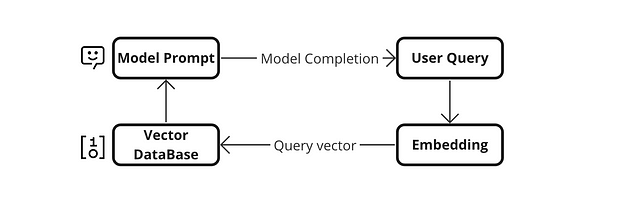
[Link to learn about Building Semantic Search Applications](https://medium.com/ai-science/build-semantic-search-applications-using-open-source-vector-database-chromadb-a15e9e7f14ce)

# 

# **Integrating the results of semantic searches into the model**

The integration of results from semantic searches into a language model involves the use of a technique known as “Retrieval Augmented Generation” (RAG). This technique combines semantic search and text generation to produce more accurate and contextualised responses.

First, a semantic search is performed, using a vector database (VectorDB) or a similar method, based on the question or context provided to the model. This semantic search returns a list of documents or text snippets considered relevant to the question.



The retrieved documents are combined with the original question and presented as input to the language model, such as GPT-3.5-turbo, for example. This model takes information from both the question and the documents to generate a coherent and contextualised response.

The language model generates the final answer based on the information from the question and the document retrieval. By understanding the context and word relationships in the retrieved documents, the model can generate more accurate and appropriate answers. Integrating semantic search with the language model allows leveraging the knowledge contained in the relevant documents, significantly improving the quality of the model-generated responses.

This approach is particularly useful when you want the model to have access to specific and up-to-date information to answer questions in a more informed and contextual manner

**Enhancing LLM Responses with Vector Database Queries**

#### **Importance of Enhancing LLM Responses with Vector Database Queries**

1. **Improved Response Quality:**
   * Description: By integrating vector databases, LLMs can access a richer set of contextual information beyond the input data, allowing for responses that are not only accurate but also contextually deeper.
   * Impact: This leads to enhanced user experiences, as the responses are more informative and nuanced, reducing misunderstandings and follow-up queries.
2. **Scalability in Knowledge Management:**
   * Description: Vector databases enable efficient storage and retrieval of large-scale data embeddings, supporting LLMs in handling expanding data volumes without loss of performance.
   * Impact: Essential for scaling applications such as digital assistants, e-commerce platforms, and data-driven research tools where the volume and variety of queries are high.
3. **Real-time Information Access:**
   * Description: The integration allows LLMs to perform dynamic queries on vector databases that can be continually updated with new information, keeping the responses relevant to current events or data.
   * Impact: Particularly beneficial in sectors like news, finance, and social media, where timeliness and accuracy of information are critical.

#### **Strategies for Enhancing LLM Responses with Vector Database Queries**

1. **Optimal Embedding Techniques:**
   * Choice of Model: Select embedding models that best capture the nuances of the specific data types handled, such as using BERT for natural language or specialised models for image or sound data.
   * Continual Learning: Implement strategies to continually update the embeddings as new data is acquired, ensuring that the vector database remains relevant.
2. **Efficient Query Mechanisms:**
   * Indexing Strategies: Utilise advanced indexing techniques in vector databases to speed up query response times, ensuring that the retrieval of relevant embeddings is efficient.
   * Batch Queries: Where possible, batch similar queries to reduce the number of database hits, which can improve response times and reduce computational overhead.
3. **Integration Architecture:**
   * Hybrid Models: Develop a hybrid model where initial responses are generated by the LLM and then refined through queries to the vector database, balancing speed and depth of responses.
   * Microservices Approach: Implement the vector database query as a separate microservice, allowing for scalability and easier maintenance of the overall system.
4. **Use of Caching and Prefetching Techniques:**
   * Caching Common Queries: Store the results of common queries or high-frequency embeddings in a cache to speed up response times.
   * Predictive Fetching: Analyse usage patterns to predictively fetch data that is likely to be needed soon, especially during high-load scenarios.
5. **Monitoring and Feedback Systems:**
   * Performance Monitoring: Regularly monitor the performance impact of vector database queries on LLM responses to identify bottlenecks and opportunities for optimization.
   * User Feedback Integration: Use feedback loops from users to understand how well the enhanced responses meet their needs and adjust the system accordingly.

[Link to view Building a Closed-QA Bot with Falcon-7B and ChromaDB](https://docs.google.com/document/d/1C8DgfpX3fkvH0_j4j5KaQAL6-HXemzbyn9Sd49LXHPo/edit#heading=h.kqo58p010w3)

[Link to view Enhancing LLM Accuracy Using MongoDB Vector Search and Unstructured.io Metadata](https://www.mongodb.com/developer/products/atlas/llm-accuracy-vector-search-unstructured-metadata/)

**Videos to learn about Integrating Vector Databases with LLMs**

# [Video to learn about Vector Databases: How They Work, Use Cases & LLM Applications](https://www.youtube.com/watch?v=oI9A84NBC3w&ab_channel=Voiceflow)

[Video to learn about Using Langchain and Open Source Vector DB Chroma for Semantic Search with OpenAI's LLM Code](https://www.youtube.com/watch?v=5NG8mefEsCU&ab_channel=PradipNichite)

**References and Further Readings**

* Explore the various applications of vector databases in AI and machine learning, including use cases like semantic search and recommendation systems, available on AWS's official blog.

<https://aws.amazon.com/what-is/vector-databases/>

* Learn how to develop semantic search applications using vector databases through detailed tutorials on the Elasticsearch website, which covers concepts of vector search and practical implementations.

<https://www.elastic.co/search-labs/blog/text-similarity-search-with-vectors-in-elasticsearch>

* Applications and Use Cases of Vector Databases

[From prototype to production: Vector databases in generative AI applications](https://stackoverflow.blog/2023/10/09/from-prototype-to-production-vector-databases-in-generative-ai-applications/)

[**Quiz Questions on Integrating Vector Databases with LLMs**](https://docs.google.com/document/d/1Qsp-kuDgZuKFeG0dCq21_veD1-HWI2cl9hkhOmCcDek/edit)