Search Medium Write

Member-only story

Vector Database: The Secret Behind Large Language Models Capabilities

What are Vector Databases and Why Are They Important for LLMs?

Youssef Hosni · Following
Published in Level Up Coding · 11 min read · Jul 30

330 1

Have you ever wondered how language models like GPT-3, BERT, and others seem to understand and generate text with astonishing accuracy? The answer lies in their ability to represent words, sentences, and documents as dense numerical vectors, known as vector embeddings. These vector embeddings encode the semantic meaning and contextual information of the language, enabling LLMs to navigate and manipulate language data like never before.

In this blog, we will take you on an exciting journey through the world of Vector Databases, shedding light on their significance in modern language processing and machine learning. Whether you are a seasoned data scientist, a language enthusiast, or simply curious about the inner workings of these powerful models, this article is for you.

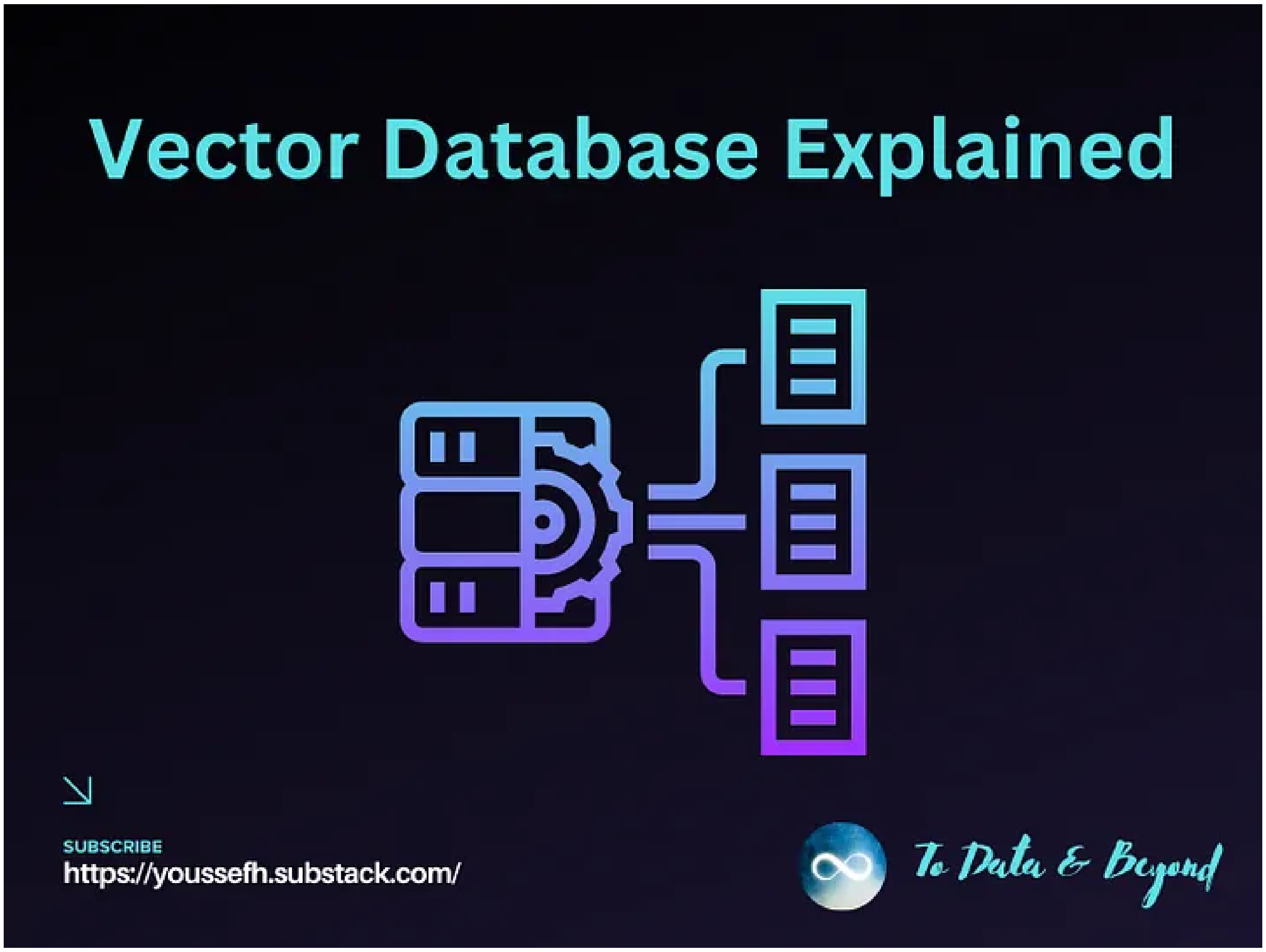


Table of Contents:

Looking to start a career in data science and AI and need to learn how. I offer data science mentoring sessions and long-term career mentoring:

- **Mentoring sessions:** https://lnkd.in/dXeg3KPW
- **Long-term mentoring:** https://lnkd.in/dtdUYBrM

Join Medium with my referral link - Youssef Hosni

As a Medium member, a portion of your membership fee goes to writers you read, and you get full access to every story…

youssefraafat57.medium.co

All the resources and tools you need to teach yourself Data Science for free!

- The best interactive roadmaps for Data Science roles. With links to free learning resources. Start here: https://aigents.co/learn/roadmaps/intro
- The search engine for Data Science learning recourses. 100K handpicked articles and tutorials. With GPT-powered summaries and explanations. https://aigents.co/learn
- Teach yourself Data Science with the help of an AI tutor (powered by GPT-4). https://community.aigents.co/spaces/10362739/

1. Vector Embeddings

Vector embedding is a powerful method for representing data in artificial intelligence and natural language processing. It serves to capture the essence of information, aiding AI systems in gaining deeper insights into the data and fostering long-term memory retention. When learning something new, comprehension and recall are key factors.

AI models, like LLMs, produce embeddings by transforming data into lower-dimensional vectors. This transformation is valuable because it simplifies the data representation, especially when dealing with a substantial number of features. The resulting embeddings encode various aspects of the data, allowing AI models to grasp intricate relationships, detect patterns, and uncover concealed structures. In essence, embeddings serve as a bridge between raw data and the AI system's ability to make sense of it all.

2. Why We Need a Vector Database?

Working with vector embeddings presents a unique set of challenges, particularly when traditional scalar-based databases are employed. These conventional databases struggle to cope with the complexity and scale of vector data, which can hinder the extraction of valuable insights and real-time analysis. However, a solution to this problem lies in the adoption of vector databases, specially engineered to handle this type of data efficiently. By leveraging vector databases, organizations can unlock the full potential of their data, enjoying improved performance, scalability, and flexibility.

1. *Complexity and Scale*: Vector embeddings encode data in multi-dimensional vectors, and as the number of features or dimensions increases, so does the complexity of the data representation. Traditional scalar databases, which are designed for handling simpler, one-dimensional data, can become overwhelmed and inefficient when confronted with the intricacies of vector data. This complexity often leads to difficulties in querying and processing the data effectively.

2. ***Extracting Insights***: Extracting meaningful insights from vector embeddings requires specialized tools and techniques. Traditional databases may lack the necessary capabilities to perform advanced operations on vector data, hindering the ability to uncover patterns, relationships, and similarities within the data.

3. ***Real-time Analysis***: In many AI applications, real-time analysis is crucial for making timely decisions and responding to dynamic changes. However, the computational demands of working with vector embeddings can strain traditional databases, resulting in slow response times and limiting real-time capabilities.

4. ***Vector Databases***: Vector databases are purpose-built to handle vector data efficiently. They are designed to store, index, and query high-dimensional data, enabling faster and more effective operations on vector embeddings. By using specialized indexing techniques optimized for high-dimensional spaces, vector databases can accelerate search and retrieval operations, facilitating quicker insights and analysis.

5. ***Performance***: Vector databases are engineered to deliver superior performance when dealing with large-scale vector data. Their architecture is tailored to take advantage of hardware capabilities, such as GPUs and TPUs, to speed up computation and enhance overall performance.

6. ***Scalability***: As data volumes grow, traditional databases may struggle to keep up with the expanding data size. In contrast, vector databases are designed with scalability in mind, making them capable of handling massive amounts of vector data without compromising performance.

7. ***Flexibility***: Vector databases offer flexibility in how data is represented and queried, allowing users to experiment with different models, algorithms, and approaches without being limited by database constraints.

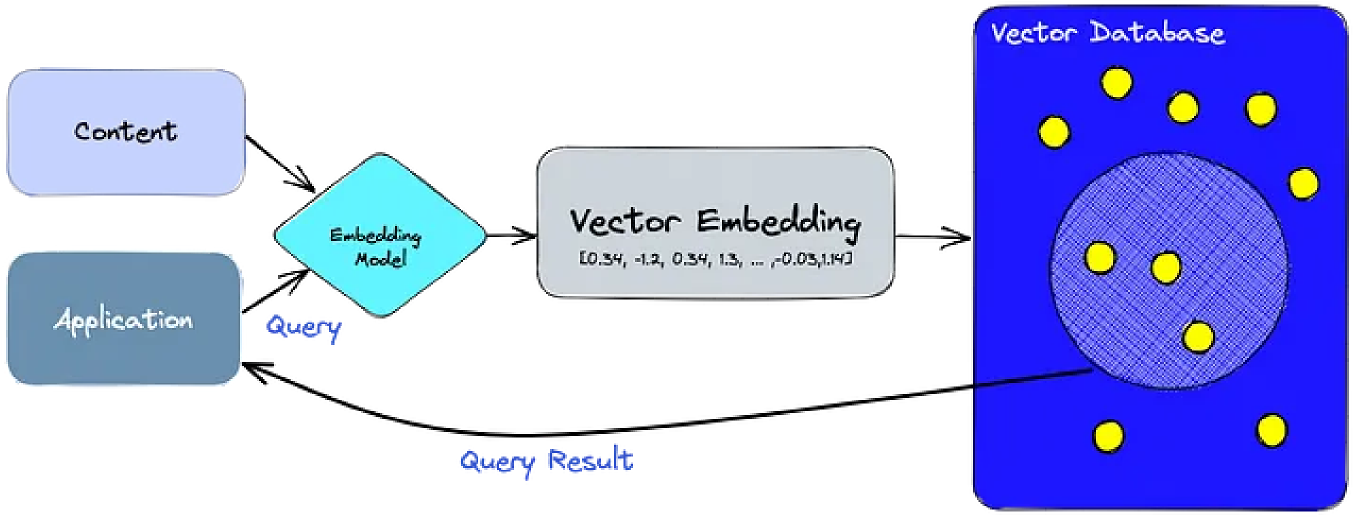
In summary, vector databases provide a powerful solution for efficiently managing and leveraging vector embeddings. By embracing these specialized databases, organizations can unlock the full potential of their data, enabling faster insights, real-time analysis, and improved decision-making capabilities in the realm of artificial intelligence and beyond.

3. How Does Vector Database Work?

To understand how vector database work let's take an example of a chatbot such as chatGPT or bard which are based on LLM. These models have large volumes of data with a lot of content. Here is how vector database is used within the context of this application:

1. The user will input your query into the application.
2. Then the query is inserted into the embedding model, which generates vector embeddings according to the content we wish to index.
3. The vector embedding then moves into the vector database, regarding the content that the embedding was made from.
4. The vector database produces an output and sends it back to the user as a query result.

The diagram below gives us a better understanding of the role of vector databases in this type of application:



In traditional databases strings, numbers, etc are stored in rows and columns. Therefore when we query a certain row or column we are querying a row or column that matches the query statement. On the other hand, vector databases work with vectors rather than strings, numbers, etc. Vector databases also apply a similarity metric which is used to help find a vector most similar to the query.

In the vector database, the similarity search is done using different algorithms which all aid in the Approximate Nearest Neighbor (ANN) search. This is done via hashing, graph-based search, or quantization which are assembled into a pipeline to retrieve neighbors of a queried vector. The results are based on how close or approximate it is to the query, therefore the main elements that are considered are accuracy and speed. If the query output is slow, the more accurate the result.

The three main stages that a vector database query goes through that are:

1. Indexing: Once the vector embedding moves into the vector database, it then uses a variety of algorithms to map the vector embedding to data structures for faster searching.

2. Querying: Once it has gone through its search, the vector database compares the queried vector to indexed vectors, applying the similarity metric to find the nearest neighbor.

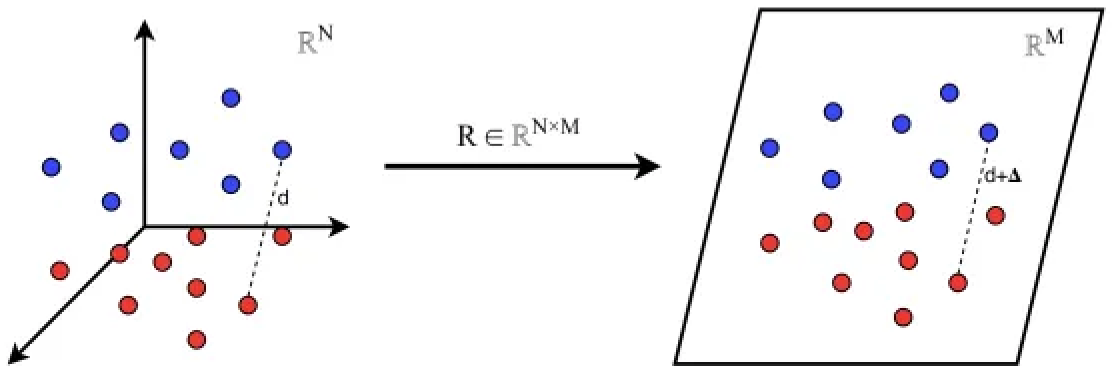
3. Post Processing: Depending on the vector database you use, the vector database will post-process the final nearest neighbor to produce a final output to the query. As well as possibly re-ranking the nearest neighbors for future reference.

4. Vector Index Creation Algorithms

Several algorithms can facilitate the creation of a vector index. Their common goal is to enable fast querying by creating a data structure that can be traversed quickly. They will commonly transform the representation of the original vector into a compressed form to optimize the query process. This section will explore several algorithms and their unique approaches to handling vector embeddings.

4.1. Random Projection

Random projection is a dimensionality reduction technique used to approximate high-dimensional data by projecting it onto a lower-dimensional space. To achieve this, a random projection matrix is created, typically using random numbers, with dimensions corresponding to the desired lower-dimensional space. The dot product of the input vectors and the random projection matrix yields a projected matrix with fewer dimensions than the original vectors, while still preserving their similarities.



During querying, the same projection matrix is applied to project the query vector onto the lower-dimensional space. By comparing the projected query vector to the projected vectors in the database, we can efficiently find the nearest neighbors. The reduced dimensionality of the data allows for a significantly faster search process compared to searching the entire high-dimensional space.

It's important to note that random projection is an approximate technique, and while it retains similarity relationships reasonably well, there is some loss of information due to the dimensionality reduction and randomness involved. Nonetheless, it proves beneficial in various machine learning and data mining tasks, where computational efficiency and memory requirements are critical considerations.

4.2. Product Quantization

Product quantization (PQ) is an effective method for building an index and compressing high-dimensional vectors, such as vector embeddings. Its goal is to represent the original vectors with smaller chunks while preserving vital information for similarity operations. The PQ process involves four main steps: splitting, training, encoding, and querying.

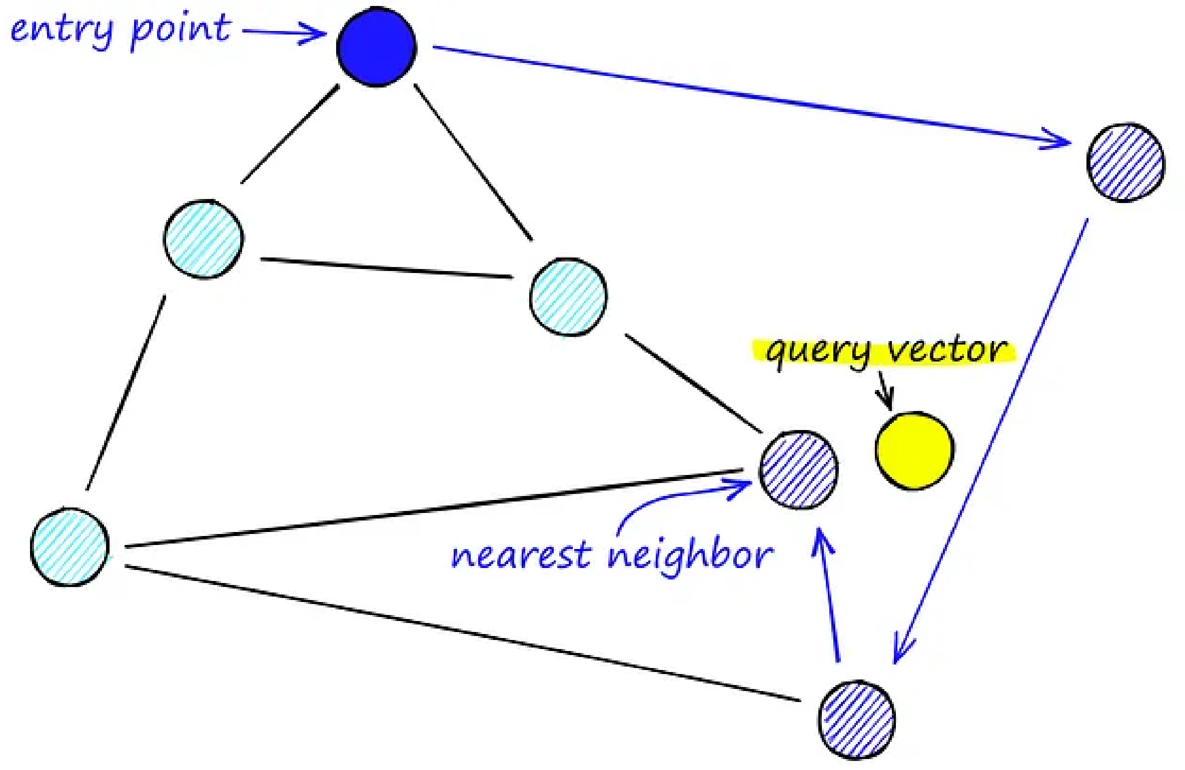
1. ***Splitting***: Initially, the high-dimensional vectors are divided into smaller segments.

2. ***Training***: In the training phase, a "codebook" is constructed for each segment. This involves generating a pool of potential "codes" that can be assigned to vectors. Practically, the codebook is created by finding the center points of clusters through k-means clustering applied to each segment. The number of values in the segment codebook corresponds to the value chosen for k in the k-means clustering.

3. ***Encoding***: During encoding, specific codes are assigned to each segment. The algorithm finds the nearest value in the codebook for each vector segment after the training process is completed. The PQ code for a segment represents the identifier of the corresponding value in the codebook. It's possible to use multiple PQ codes, allowing for the representation of each segment with multiple values from the codebook.

4. **Querying:** During querying, the algorithm breaks down the vectors into sub-vectors and quantizes them using the same codebook. Then, it leverages the indexed codes to find the nearest vectors to the query vector.

4.3. Hierarchical Navigable Small World (HNSW)



HNSW creates a hierarchical, tree-like structure where each node of the tree represents a set of vectors. The edges between the nodes represent the **similarity** between the vectors. The algorithm starts by creating a set of nodes, each with a small number of vectors. This could be done randomly or by clustering the vectors with algorithms like k-means, where each cluster becomes a node.

The algorithm then examines the vectors of each node and draws an edge between that node and the nodes that have the most similar vectors to the one it has.

When we query an HNSW index, it uses this graph to navigate through the tree, visiting the nodes that are most likely to contain the closest vectors to the query vector

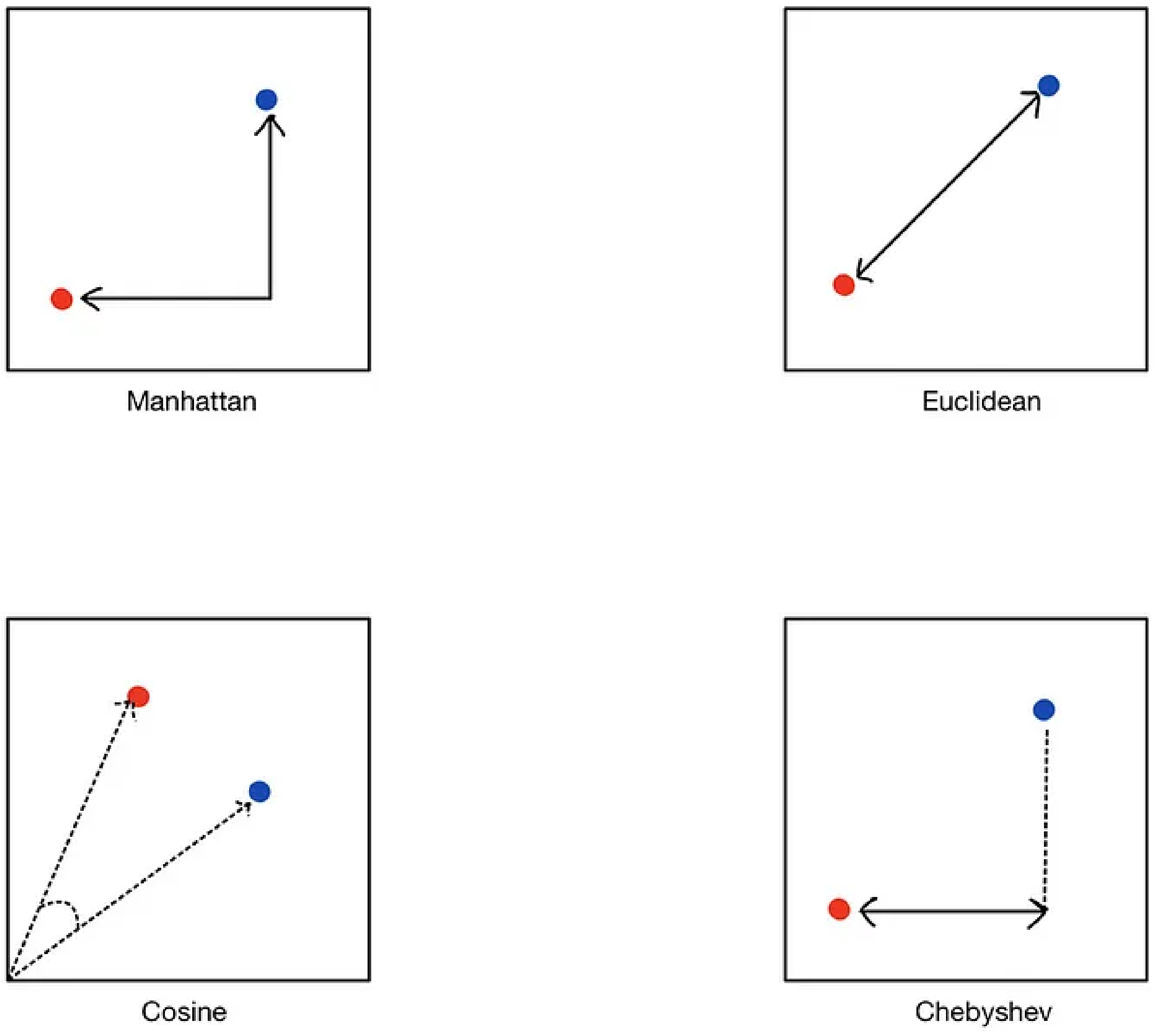
4.4. Locality-Sensitive Hashing

Locality-Sensitive Hashing (LSH) is a technique for indexing in the context of an approximate nearest-neighbor search. It is optimized for speed while still delivering an approximate, non-exhaustive result. LSH maps similar vectors into "buckets" using a set of hashing functions.

To find the nearest neighbors for a given query vector, we use the same hashing functions used to "bucket" similar vectors into hash tables. The query vector is hashed to a particular table and then compared with the other vectors in that same table to find the closest matches. This method is much faster than searching through the entire dataset because there are far fewer vectors in each hash table than in the whole space.

It's important to remember that LSH is an approximate method, and the quality of the approximation depends on the properties of the hash functions. In general, the more hash functions used, the better the approximation quality will be. However, using a large number of hash functions can be computationally expensive and may not be feasible for large datasets.

5. Similarity Measurement Methods



Different similarity measurement methods

Similarity measures are mathematical methods for determining how similar two vectors are in a vector space. Similarity measures are used in vector databases to compare the vectors stored in the database and find the ones that are most similar to a given query vector.

Several similarity measures can be used, including:

- ***Cosine similarity***: Cosine similarity is a mathematical measure that quantifies the similarity between two vectors in a vector space. It assesses the cosine of the angle formed between the vectors. The resulting similarity score falls within the range of -1 to 1.

- ***Euclidean distance***: measures the straight-line distance between two vectors in a vector space. It ranges from 0 to infinity, where 0 represents identical vectors, and larger values represent increasingly dissimilar vectors.
- ***Dot product***: measures the product of the magnitudes of two vectors and the cosine of the angle between them. It ranges from -∞ to ∞, where a positive value represents vectors that point in the same direction, 0 represents orthogonal vectors, and a negative value represents vectors that point in opposite directions.

References:

- What are Vector Databases and Why Are They Important for LLMs?
- What is a Vector Database?

If you like the article and would like to support me, make sure to:

- **👏 Clap for the story (50 claps) to help this article be featured**
- **Subscribe to To Data & Beyond Newsletter**
- **Follow me on Medium**
- **📰 View more content on my medium profile**

- 🔔 Follow Me: LinkedIn |Youtube | GitHub | Twitter

Join the Medium membership program for only 5$ to continue learning without limits. I'll receive a small portion of your membership fee if you use the following link at no extra cost.

Join Medium with my referral link - Youssef Hosni
As a Medium member, a portion of your membership fee goes to writers you read, and you get full access to every story...
youssefraafat57.medium.com

Looking to start a career in data science and AI and do not know how. I offer data science mentoring sessions and long-term career mentoring:

- Mentoring sessions: https://lnkd.in/dXeg3KPW
- Long-term mentoring: https://lnkd.in/dtdUYBrM

Level Up Coding

Thanks for being a part of our community! Before you go:

- 👏 Clap for the story and follow the author 👉
- 📰 View more content in the Level Up Coding publication

🔔 Follow us: Twitter | LinkedIn | Newsletter

🧠 AI Tools ⇒ Become an AI prompt engineer

LIm Vector Database Data Science NLP

More from the list: "NLP"

Curated by Himanshu Birla

Jon Gi... in Towards Data ...

Characteristics of Word Embeddings

✦ · 11 min read · Sep 4, 2021

Jon Gi... in Towards Data ...

The Word2vec Hyperparameters

✦ · 6 min read · Sep 3, 2021

Jon Gi... in

The Word2ve

✦ · 15 min rea

View list

Written by Youssef Hosni

Following

17.7K Followers · Writer for Level Up Coding

Computer Vision Researcher & Data Scientist | My Newsletter: https://youssefh.substack.com/ | Mentoring Services & E-Products: https://topmate.io/youssef_hosni

More from Youssef Hosni and Level Up Coding

Youssef Hosni in Level Up Coding

13 Guided Time Series Projects to Build Your Portfolio

Elevate Your Data Science Portfolio with These 13 Time Series Guided Projects

✦ · 7 min read · Sep 18

291

Comment	Date
WIP	3 days ago
Off for lunch	1 day ago
End of code for today	20 hours ago
I am tired AF	18 hours ago
Happy Weekend Team	16 hours ago
First to commit	14 hours ago
Fixed final bug	10 hours ago
Added a new feature	9 hours ago
Fixed another bug	7 hours ago
Made some changes	5 hours ago
fixed two build-breaking issues	3 hours ago
Bugs are never ending. fixed another bug	[illegible]

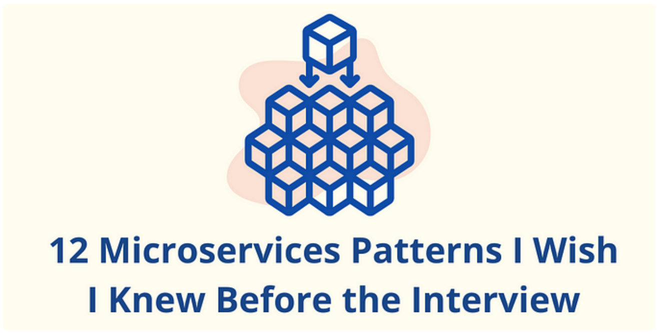
Victor Timi in Level Up Coding

“Good Commit” vs “Your Commit”: How to Write a Perfect Git Commi...

A good commit shows whether a developer is a good collaborator — Peter Hutterer, Linux.

✦ · 8 min read · Sep 5

2.6K 32

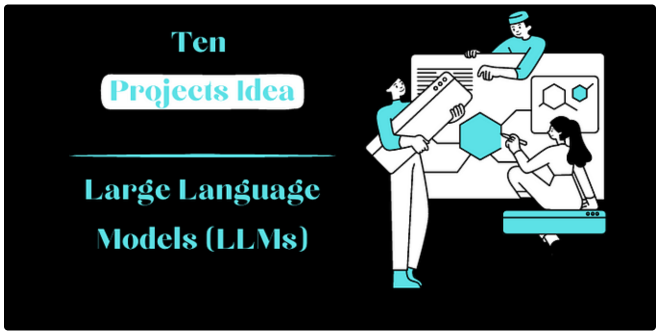


Arslan Ahmad in Level Up Coding

12 Microservices Patterns I Wish I Knew Before the System Design...

Mastering the Art of Scalable and Resilient Systems with Essential Microservices Desig...

13 min read · May 16



Youssef Hosni in Level Up Coding

10 Large Language Models Projects To Build Your Portfolio

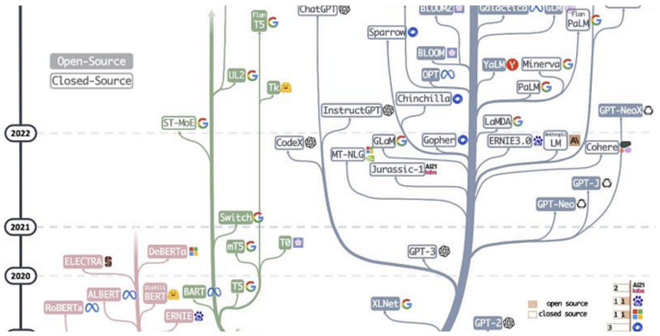
Build end-to-end applications and showcase your skills with large language models (LLMs)

✦ · 13 min read · Sep 3

3.8K 17 292 1

See all from Youssef Hosni See all from Level Up Coding

Recommended from Medium



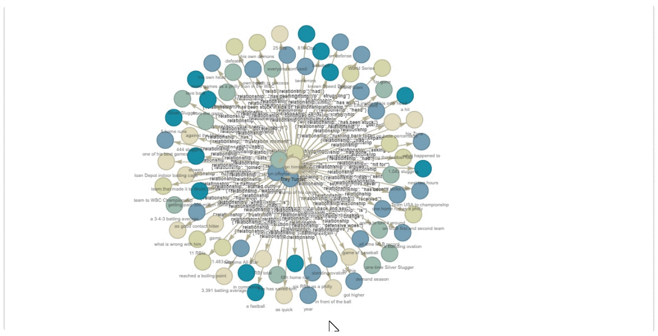
Haifeng Li

A Tutorial on LLM

Generative artificial intelligence (GenAI), especially ChatGPT, captures everyone's...

15 min read · Sep 14

372



Wenqi Glantz in Better Programming

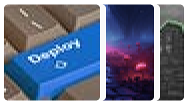
7 Query Strategies for Navigating Knowledge Graphs With...

Exploring NebulaGraph RAG Pipeline with the Philadelphia Phillies

· 17 min read · 4 days ago

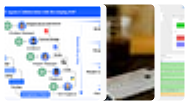
501 4

Lists



Predictive Modeling w/ Python

20 stories · 452 saves



Natural Language Processing

669 stories · 283 saves

New_Reading_List

174 stories · 133 saves

Practical Guides to Machine Learning

10 stories · 519 saves

Ryan Nguyen in Towards AI

So, You Want To Improve Your RAG Pipeline

Ways to go from prototype to production with LlamaIndex

✦ · 9 min read · Sep 27

176 2

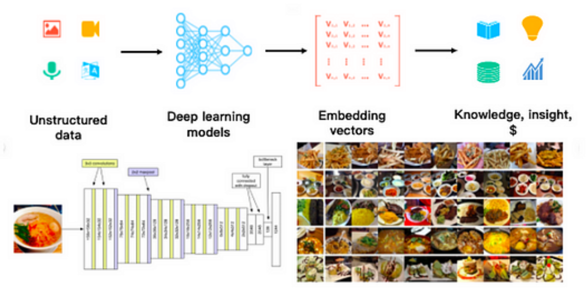
Han HELOIR, Ph.D. in Artificial Corner

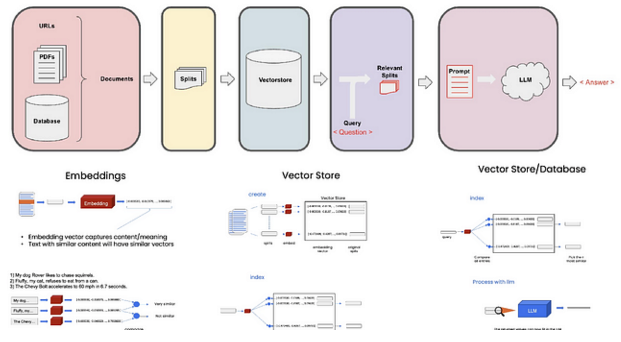
MongoDB and Langchain Magic: Your Beginner's Guide to Setting...

Introduction:

✦ · 7 min read · Sep 12

1.4K 12





Jayita Bhattacharyya in GoPenAI

TeeTracker

Primer on Vector Databases and Retrieval-Augmented Generation…

Vector Databases Generation (RAG) Langchain Pinecone HuggingFace Large…

9 min read · Aug 16

228 1

Chat with your PDF (Streamlit Demo)

Conversation with specific files

4 min read · Sep 15

56

See more recommendations