Medium Search Write Sign up Sign in

# Vector Embedding 101: The New Building Blocks for Generative AI

Fundamental concepts for getting the most from your unstructured data and getting going with Generative AI.

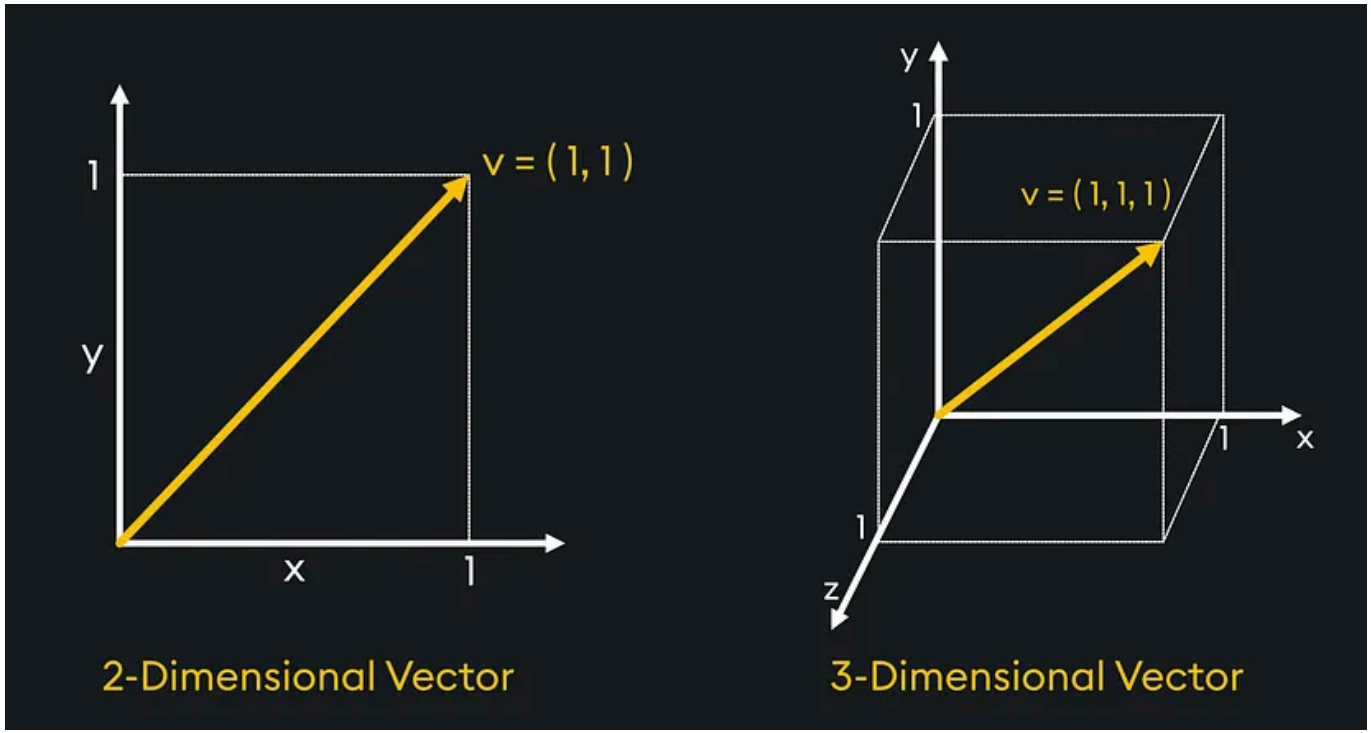
Nathan Crone · Follow
Published in KX Systems · 6 min read · Aug 22, 2023

15 1

Despite their impressive capabilities, computers cannot easily understand text, images, audio, or other human-readable data formats. When analysing these data types, we can instead represent them as numerical “vectors” which can be better processed computationally. In this article, we will explore how this vectorisation process can be done and will equip you with a basic understanding of these concepts and why they matter.

## What is a vector?

Put simply, a vector is a fixed-length array of numbers which mathematically represents a point in space. Each number in this array represents a unique direction in the vector space, called a dimension, and the values populated for each dimension represent the vectors magnitude in that direction. In some machine learning applications, vectors can have thousands of dimensions which is too many to even attempt to visualise, however, simple vectors, such as those with only two or three dimensions, can easily be visualised and understood:



The critical point to note here is once something has been represented as a mathematical vector, it opens it up to mathematical vector operations such as measuring distances, calculating similarities, and performing transformations. These operations become crucial for various tasks, including similarity search, clustering, classification, and uncovering patterns and trends.

## What is a vector embedding?

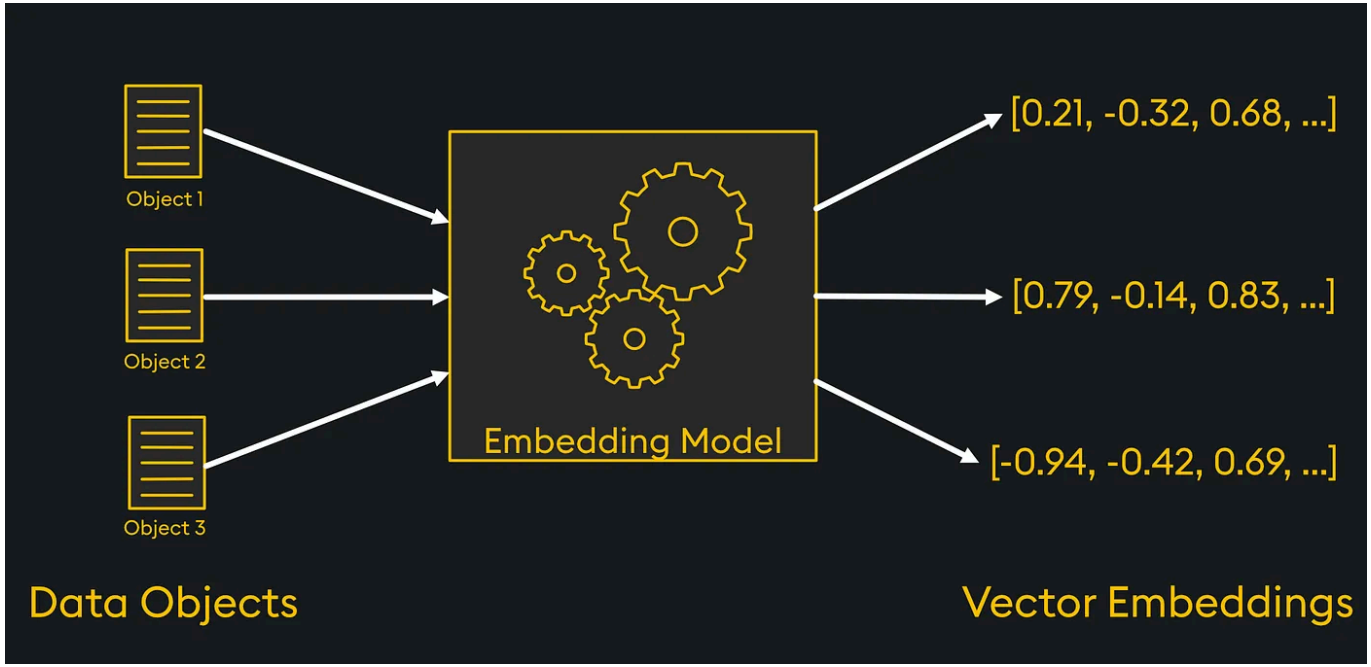
A vector embedding, or simply “an embedding,” is a vector created as the numerical representation of typically non-numerical data objects. Embeddings capture the inherent properties and relationships of the original data in a condensed format and are often used in Machine Learning use cases.

For instance, the vector embedding for an image containing millions of pixels, each with a unique colour, hue, and contrast, may only have a few hundred or thousand numbers. In this way, embeddings are designed to encode relevant information about the original data in a lower-dimensional space, enabling efficient storage, retrieval, and computation. Simple embedding methods can create sparse embeddings, whereby the vector’s values are often 0, while more complex and “smarter” embedding methods

can create dense embeddings, which rarely contain 0's. However, these sparse embeddings are often higher dimension than their dense counterparts, requiring more storage space.

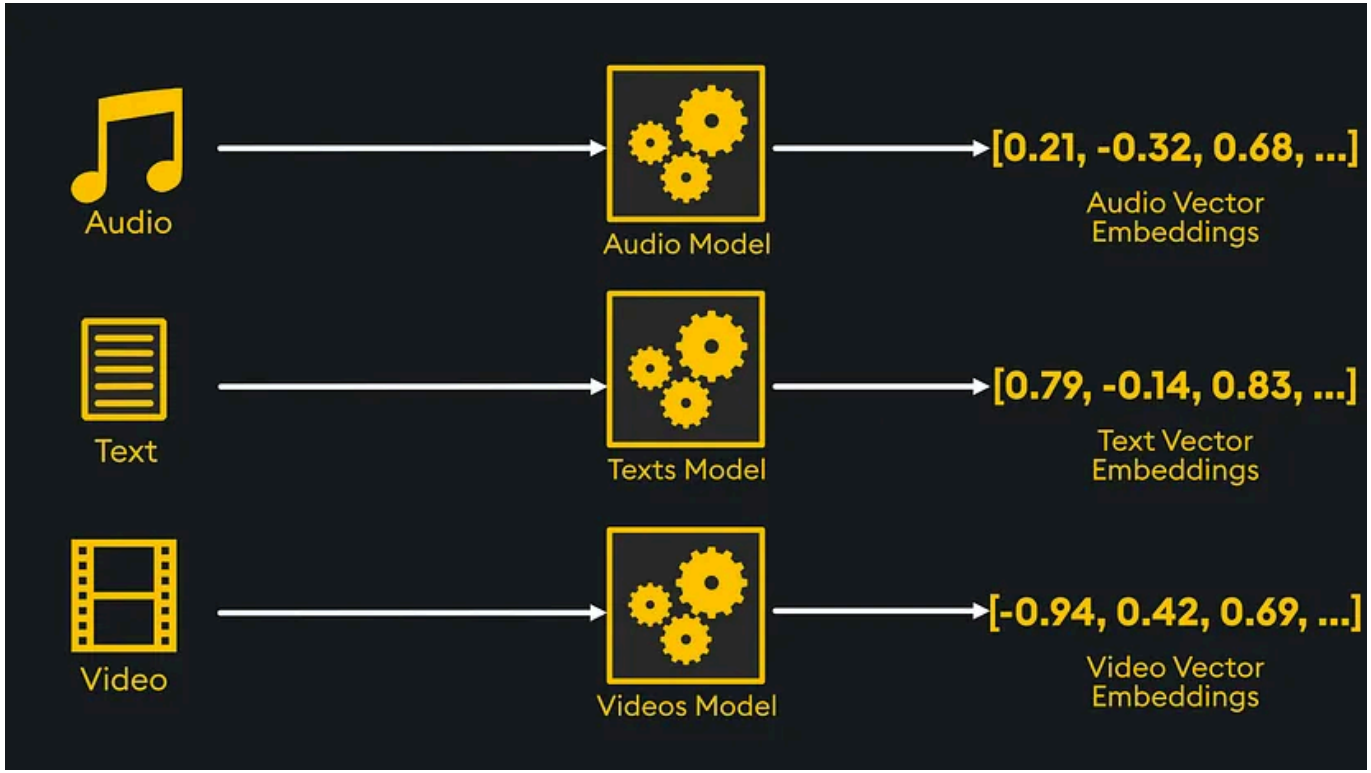
Unlike the original data, which may be complex and heterogeneous, embeddings typically strive to capture the essence of the data in a more uniform and structured manner. This transformation process is performed by what's known as an "embedding model" and often involves complex machine learning techniques.

These models take in data objects, extract meaningful patterns and relationships from this data, and return vector embeddings which algorithms can later use to perform various tasks. Many sophisticated embedding models are openly available online, examples of which we will give in a later section of this article.




## Four types of vector embeddings: text, image, audio, and time

The precise information contained in an embedding depends on the specific data types and the embedding technique employed.




Embeddings aim to capture semantic, contextual, or structural information relevant to the task. Each embedding model utilizes specific techniques and algorithms tailored to the type of data being dealt with and other characteristics of the data being represented.

Here, we can give examples of what features may be encoded for text, image, audio, and temporal data and provide a list of common techniques used to achieve this:

## TEXT EMBEDDINGS

Text embeddings capture the semantic meaning of words and their relationships within a language. For example, they could encode semantic similarities between words, such as “king” being closer to “queen” than to “car”.

Common models used for creating text embeddings include:

- **TF-IDF (Term Frequency — Inverse Document Frequency)** creates sparse embeddings by assigning weights to words based on their occurrence frequency in a document relative to their prevalence across the entire dataset.
- **Word2Vec** creates dense vector representations that capture semantic relationships by training a neural network to predict words in context.
- **BERT (Bidirectional Encoder Representations from Transformers)** creates context-rich embeddings that capture bidirectional dependencies by pre-training a transformer model and using this to predict masked words in sentences.

## IMAGE EMBEDDINGS

Image embeddings capture visual features like shapes, colours, and textures. For example, they might encode the *contrast* between colours — orange objects being more similar to yellow objects than black objects.

Common models used for creating image embeddings include:

- **Convolutional Neural Networks (CNNs)** create dense vector embeddings by passing them through convolutional neural network layers that extract hierarchical visual features from the images.
- **Transfer Learning with Pre-trained CNNs** like ResNet and VGG create embeddings by fine-tune pre-trained CNNs which have already learned complex visual features from large datasets.
- **Autoencoders** are neural network models which are trained to encode and decode images by generating embeddings that capture compact representations of the raw images.

### AUDIO EMBEDDINGS

Audio embeddings capture audio signals like pitch, frequency, or speaker identity. For example, they could encode the sound of a piano and a guitar to

have distinct numerical representations reflecting the acoustic features of each sound, enabling differentiation.

Common models user for creating audio embeddings include:

- **Spectrogram-based Representations** create embeddings by first converting the audio into visual representations, like spectrograms, and then applying image-based methods to embed these images as vectors.
- **MFCCs (Mel Frequency Cepstral Coefficients)** create vector embeddings by calculating spectral features of the audio and using these to represent the sound content.
- **Convolutional Recurrent Neural Networks (CRNNs)** create vector embeddings by combining convolutional and recurrent neural network layers to handle spectral features and the sequential context in creating informative audio representations.

### TEMPORAL EMBEDDINGS

Temporal embeddings capture the patterns and dependencies in time-series data. For example, they might be used to encode time series patterns of

heart rates in medical systems to compare the similarities between a person at rest, sleeping, or running a marathon.

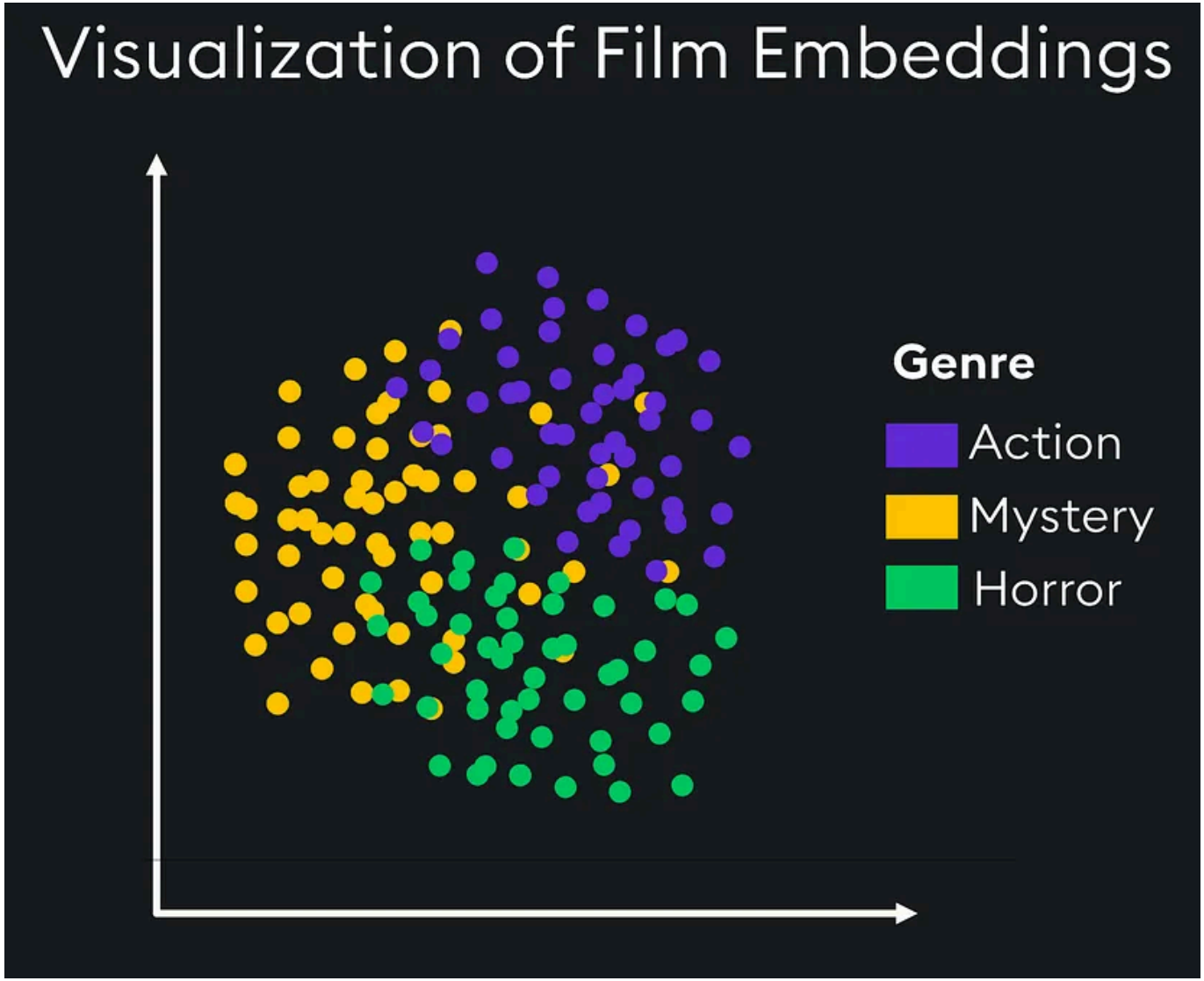
Common models used for creating temporal embeddings include:

- **LSTM (Long Short-Term Memory)** models create embeddings by capturing long-range dependencies and temporal patterns in sequential data using a recurrent neural network (RNN) architecture.
- **Transformer-based Models** create vector embeddings using a self-attention mechanism to capture complex temporal patterns in the input sequence.
- **Fast Fourier Transform (FFT)** creates vector embeddings that capture periodic patterns and spectral information in the temporal data by converting it into its frequency-domain representation and extracting frequency components.

## What can you do with vector embeddings?

So, what can we do with these vector embeddings once we have obtained them?

- **Similarity search:** Use embeddings to measure the similarity between different instances. For example, in Natural Language Processing (NLP), you can find similar documents or identify related words based on their embeddings.
- **Clustering and classification:** Use embeddings as the input features for clustering and classification models to train machine-learning algorithms to group similar instances and classify objects.
- **Information retrieval:** Utilise embeddings to build powerful search engines that can find relevant documents or media based on user queries.
- **Recommendation systems:** Leverage embeddings to recommend related products, articles, or media based on user preferences and historical data.
- **Visualisations:** Visualise embeddings in lower-dimensional spaces to gain insights into the relationships and patterns within the data.
- **Transfer learning:** Use pre-trained embeddings as a starting point for new tasks, allowing you to leverage existing knowledge and reduce the need for extensive training.




**Vector embeddings: the foundation of your new generative AI house**

Vector embeddings bridge the gap between human-readable data and computational algorithms. By representing diverse data types as numerical vectors, we unlock the potential for a wide range of Generative AI applications. These embeddings condense complex information, capture relationships, and enable efficient processing, analysis, and computation.

Armed with vector embeddings, you can explore and transform data to facilitate new ways to understand information, make better decisions, and innovate with generative AI applications.

Data Science | Vector Database | Generative Ai Tools | Vectorization | Vector Embeddings

# Written by Nathan Crone

Follow

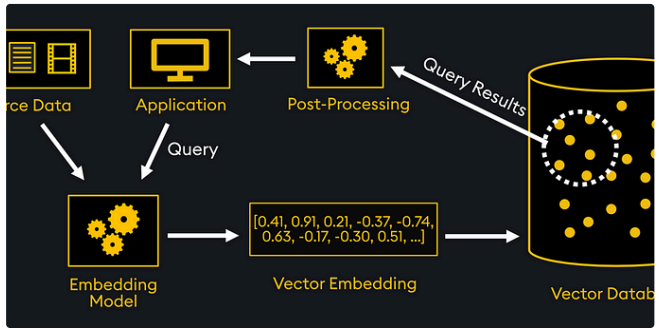
5 Followers · Writer for KX Systems

Data Scientist who likes to explore the latest and greatest data and machine learning concepts

## More from Nathan Crone and KX Systems

Ryan Siegler in KX Systems

### Guide to Multimodal RAG for Images and Text




Ryan Siegler in KX Systems

### Optimizing Vector Search with Metadata Filtering

Multimodal AI stands at the forefront of the next wave of AI advancements. This sample...

12 min read · Feb 12, 2024

91 1

Improve vector similarity searches and RAG applications with metadata filtering.

4 min read · Jan 24, 2024

16

Neil Kanungo in KX Systems

## How Vector Databases Search by Similarity: A Comprehensive...

Similarity search allows for exploration of complex, unstructured data sets, but how is...

8 min read · Aug 25, 2023

17

Neil Kanungo in KX Systems

## Vector Indexing: A Roadmap for Vector Databases

Road maps are useful when traveling because they take dense geographical information...

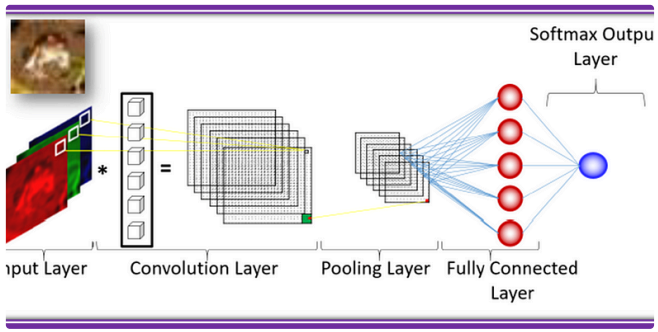
7 min read · Sep 8, 2023

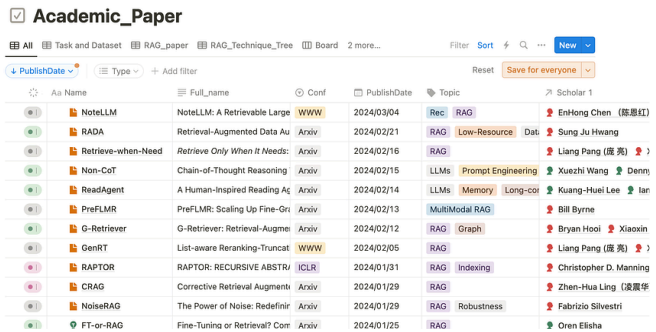
63

See all from Nathan Crone

See all from KX Systems

# Recommended from Medium




Jyoti Dabass, Ph.D in Python in Plain English

## Friendly Introduction to Deep Learning Architectures (CNN, RN...

This blog aims to provide a friendly introduction to deep learning architectures...

8 min read · Apr 2, 2024

474 6

Yunfan Gao

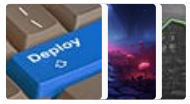
## OpenRAG Base: Your individual RAG Knowledge Base

We're officially launching the RAG Knowledge Base: OpenRAG Base 🎉✨

9 min read · Apr 3, 2024

14

## Lists




## Predictive Modeling w/ Python

20 stories · 1081 saves

## Practical Guides to Machine Learning

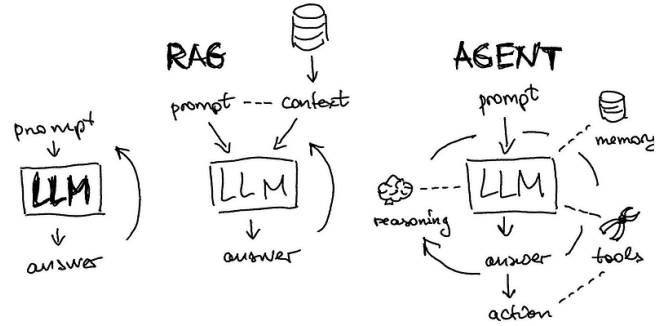
10 stories · 1304 saves

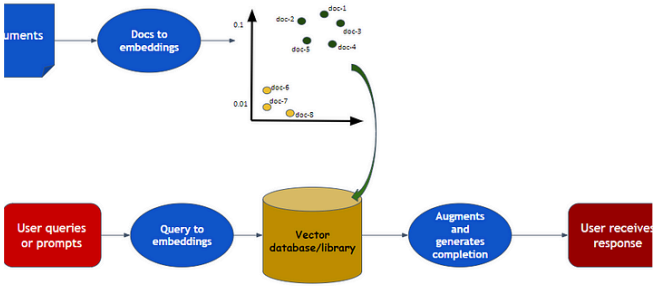
## Coding & Development

11 stories · 555 saves

## ChatGPT prompts

47 stories · 1405 saves




Alex Honchar in Towards Data Science

# Intro to LLM Agents with Langchain: When RAG is Not...

First-order principles of brain structure for AI assistants

7 min read · Mar 15, 2024

1.93K 9

Akriti Upadhyay

# Implementing RAG with Langchain and Hugging Face

Using Open Source for Information Retrieval

9 min read · Oct 16, 2023

1K 13

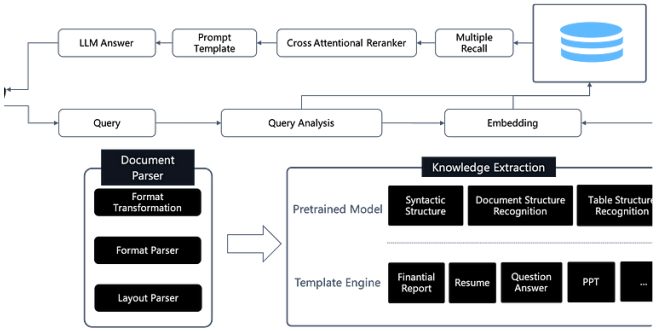
Adam Blum

## Comparing Vector Databases

If you are a developer interested in AI, there is a good chance that you may have started...

10 min read · Nov 10, 2023

185 7




InfiniFlow

## RAGFlow: Customizable, Credible, Explainable RAG engine based on...

Following the official open-sourcing of the AI-native database Infinity at the end of 2023,...

7 min read · Apr 2, 2024

59 2

See more recommendations