Recently vector databases got a lot of fame with companies raising hundreds of millions of dollars to build them and people calling it a new kind of database for the AI era.

On the otherhand for many projects it might be an overkill solution and traditional databases or even just a numpy array might work.

But there is no doubt the vector databases are extremely fascinating and allow many great applications.

I would like to give a brief overview of what they entail, how they could be useful in LLM based scenarios and give a small glimpse of the current landscape( which is changing probably even while we are talking about it)

Why do we need vector databases. Over 80 percent of the data out there is unstructured such as social media posts images videos or audio data you cannot easily fit them into a relational database.Let's take an image as an example if you want want to put this into a relational database in order to search for similar images what ends up happening is that often we manually assign keywords or tags to it because from the pixel values alone we cannot really search for similar images and the same holds true for unstructured text blobs or audio and video data.

So we either have to assign tags or attributes to it often manually or we can find a different representation to store the data and this brings us to vector embeddings and vector databases .

A vector database indexes and stores Vector embeddings for fast retrieval and similarity search so it has

Two important components first it uses clever algorithms to calculate the so-called vector embeddings This is done by machine learning models.

A vector embedding is just a list of numbers that represents the data in a different way for example you can calculate an embedding for a single word a whole sentence or an image and now we have numerical data that the computer can understand. One easy possibility we get with vectors is to find similar vectors by calculating the distances and doing a nearest neighbor search so we can easily find similar items for.

In reality of course those vectors can have hundreds of Dimensions but just storing the data as embeddings is not enough performing a query across thousands of vectors based on its distance metric would be extremely slow and this is why those vectors also need to be indexed so the indexing process is the second key element of a vector database an index is a data structure that facilitates the search process. So, the indexing step maps the vectors to a new data structure that will enable faster searching this is a whole research field on its own and different ways to calculate indexes exist so indexes are needed for efficient search.

So let's go over some Use cases.

Vector databases and LLM complement each other very well. These are the three most efficient ways in which vector databases amplify LLM use cases and ensure better ROI.

1. As a knowledge base to provide ‘context’ from the enterprise aka RAG ( Retrieval Augmented Generation). In this case**, Vector DB acts as a knowledge extension for LLM’s** and can be queried to retrieve existing similar information (context) from the knowledge base. **This also eliminates the need to use sensitive enterprise data to train or fine-tune LLM. Every time a question is asked:**

· Question gets converted to LLM-specific embedding.

· Embedding is used to retrieve relevant context (or documents) from the Vector database

· LLM Prompt is created with the help of this context.

· Response is generated. Enterprise-specific context helps LLM to provide accurate output.

Use cases: Document discovery, Chatbots, Q&A.

Key benefits: Avoids using sensitive data for model training/fine-tuning. Cheaper than fine-tuning LLMs. Almost real-time updated knowledge base.

2. **Acting as long-term LLM Memory. This helps to retrieve the last N messages relevant to the current message from the entire chat history which can encompass multiple simultaneous sessions and historical interactions. This also helps to bypass context length (tokens) limitations of LLM and gives more control in your hand.** Here key steps are:

· User asks a query.

· System retrieves stored embedding from the vector database and pass on to query LLM

· LLM response is generated and shared with the Use. Also, response embedding (with history) is stored in a vector database.

Use cases: Knowledge discovery, Chatbots.

Key benefits: Bypass token length limitations of LLM and help with conversation topic changes.

3. Cache previous LLM queries and responses. When a query is fired, create embedding and do a cache lookup before invoking the LLM. This ensures quick response and money saved on computation as well as LLM usage. Here key steps are:

· User asks a question.

· Embedding created and Cache lookup performed.

· If information is available in Cache, a response is provided.LLM not invoked.

· If information is unavailable in Cache, LLM is invoked, and the response is stored in the Cache.

Use cases: All use cases such as Document discovery, Information retrieval, Chatbots, and Q&A.

Key benefits: Speeds up performance, optimizes computational resources and LLM invocation cost.

This list doesn’t end here.

We can also use it for similarity search for images audio or video data so we can say hey find me a similar image to this one and we don't need to use some keywords or text to describe the image.

We can use vector database as a ranking and recommendation engine for example for online retailers it can be used to suggest items similar to past purchases of a customer since we can simply identify the nearest neighbors of an item in our database.

So now that you know some use cases let's go over some options you can use as a vector database

there are a number of vector databases available.

In cases where a company possesses a strong technological foundation and faces a substantial workload demanding advanced vector search capabilities, its ideal solution lies in adopting a specialized vector database. Prominent options in this domain include Chroma (having raised $20 million), Zilliz (having raised $113 million), Pinecone (having raised $138 million), Qdrant (having raised $9.8 million), Weaviate (having raised $67.7 million), LanceDB (YC W22), Vespa, Marqo, and others. Many of these players have secured significant funding in recent years and are well-positioned to capture notable market share. These vector databases offer efficient storage, indexing, and similarity search functionalities for vectors. They often incorporate specific optimizations tailored for vector data, such as similarity search based on inverted indexes and efficient vector computations. As a result, they cater to the requirements of companies operating in areas like recommendation systems, image search, and natural language processing.

On the other hand, if a company has already adopted commercial databases like Elastic, Redis, SingleStore, or Rockset and does not necessitate highly advanced vector search capabilities, they can fully utilize the existing functionality of these databases. These commercial databases excel in processing non-vector data and are suitable for various use cases and scenarios. While their performance in vector data processing may be satisfactory rather than exceptional, they can still fulfill the general requirements of most users. Moreover, the field of database technology is constantly evolving, and many databases are considering incorporating vector search capabilities to meet the demands of their current user base. For databases that currently lack vector search functionality, it is only a matter of time before they implement these features.