How to Use Large Language Models (LLMs) on Private Data: A Data Strategy Guide

Sanjeev Mohan · Follow
10 min read · Jun 11

I want to thank Madhukar Kumar, CMO, SingleStore for educating me on the nuances of vector databases. He is probably the most technically-savvy marketing person I know who codes for the sheer fun of exploring the depths of AI. A longer version of this document includes an evaluation criteria and can be found here.

The fury and frenzy of AI is all around us. In just a matter of four months, we have gone from worrying about the next AI winter to fretting over AI dictating every aspect of our lives. Every day brings a new AI application that pushes the boundary of possibilities even further — we were still grappling with ChatGPT when AutoGPT and LangChain introduced new levels of automation.

Despite all the attention AI is garnering, some high-profile missteps have reminded the world once again of "garbage in, garbage out." If we ignore the underlying data management principles, then the output can't be trusted. AI adoption will boost significantly once we can guarantee underlying training data's veracity.

However, the future has to contend with reality! Most business data today sits within corporate data sources — inside its firewall or outside, and not in the public domain internet. If we leverage large language models (LLMs) on this corpus of data, new possibilities emerge.

But, how does one exploit the power of LLMs on private data?

This blog explores how technical professionals should evolve their data strategy and select a data infrastructure to leverage the LLMs along with the enterprise data. This document is not an exploration of LLMs, like OpenAI's GPT-3/4, Facebook's LLaMa and Google's PaLM2.

Rebooting Data Strategy

Organizations have to make a fundamental decision — whether to create their own LLM, tune a general-purpose LLM on private data or leverage a general-purpose LLM's API. Each approach requires a unique set of skills and commitments:

- **Train custom LLM**

Enables purpose-built models for specific tasks, e.g. classify Slack messages to identify PII. This approach requires deep AI skills within an organization and is better suited for organizations with large and sophisticated IT teams. Training an LLM like GPT-4 also requires a massive infrastructure.

The field of generative AI is too early to support this option cost-effectively at the time of writing. However, this space may be the most exciting space with many service providers emerging to develop domain-specific LLMs in the future.

- **Tune a general-purpose LLM**

This option uses model weights to fine-tune an existing model on a specific training set. It also requires deep knowledge of AI and an investment in infrastructure resources which can be quite high depending upon the size of your data. In addition, it has led to the creation of a new category, called *LLMOps*.

Like the previous option, this option is also too early in its maturity curve. Low-Rank Adaptation (LoRA) is one of the recent developments to help with fine-tuning and expect to see rapid development in this space.

- **Prompt general-purpose LLMs**

This option uses model input, whereby context is inserted into an input message that is sent via APIs to an LLM. The model inputs need to be converted into vectors, which are explained in the following section. For organizations with modest IT skills and resources, this option is often the first foray into the space of leveraging generative AI.

A new role, called *prompt engineering*, has emerged to develop accurate and relevant text prompts for AI models. The process of leveraging external content to augment the LLM is called **Retrieval Augmented Generation (RAG)**. Facebook and Hugging Face open-sourced their RAG model in September 2020.

The table below shows the trade-offs:

Rebooting Data Strategy using LLMs SanjMo

Option	Pros	Cons
Train custom LLM	• Specialized, with high accuracy • Confidentiality	• Requires deep AI skills • Cost and time of training may be high due to the use of GPUs (or TPUs), batch
Tune an LLM ("model weight")	• Specialized, with high accuracy • Faster than building LLM	• Inaccurate tuning can degrade performance and accuracy • Batch, time consuming and expensive
Prompt an LLM ("model input")	• Delivers data freshness • Ability to chain LLMs • Lower AI skills needed • Lower cost and fast onboarding	• High vectorization embedding latency • Expensive (pay per token) • Lower precision in exchange for faster onboarding of AI workloads

The focus of this paper is on the **prompt LLM option**, because most organizations will not have the skills needed to train or tune LLMs. This approach, involving vectorizing data and creating embeddings, only requires coding skills, like Python. This option also consumes resources but at significantly lower levels than the two former options. Its biggest benefit is that it allows contextual data to be fed to the LLM in real time.

Persisting Data for LLMs

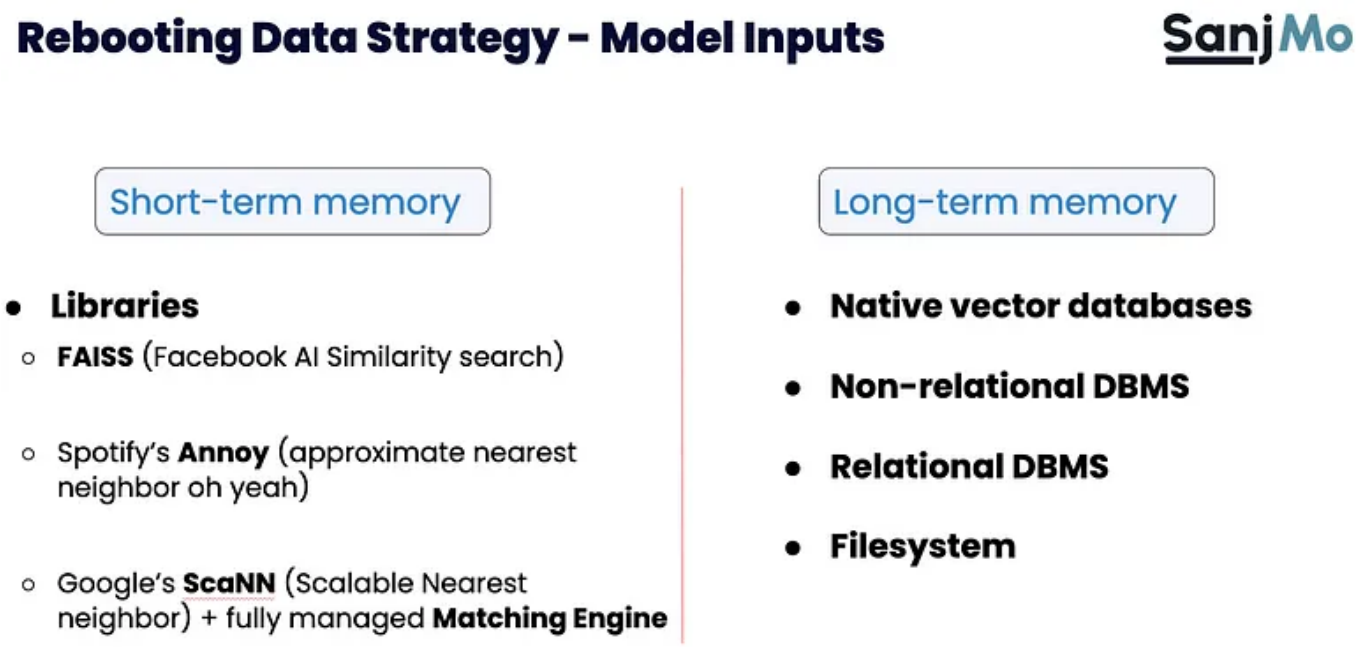
The second part of the data strategy is to identify what technologies to use to enable AI workloads. Does this require *an altogether new tech stack, or can existing technologies be repurposed?*

As this document is focused on using the option to leverage public LLMs, our focus is on persisting, managing and querying model input data and its vector embeddings. For

leveraging prompts, there are two options:

- Short-term memory for LLMs that use APIs for model inputs
- Long-term memory for LLMs that persist the model inputs

The figure below shows examples of the two options:



Short-term memory is ephemeral while long-term memory introduces persistence.

- **Libraries with built-in embedding and vectorization classes**

All the libraries mentioned above, such as FAISS, are open-source and being used widely by several products. However, the onus is on the developer to build the pipeline using the libraries to deliver the outcomes. Google's Matching Engine is a full managed option that has been optimized for model inputs and also provides persistence.

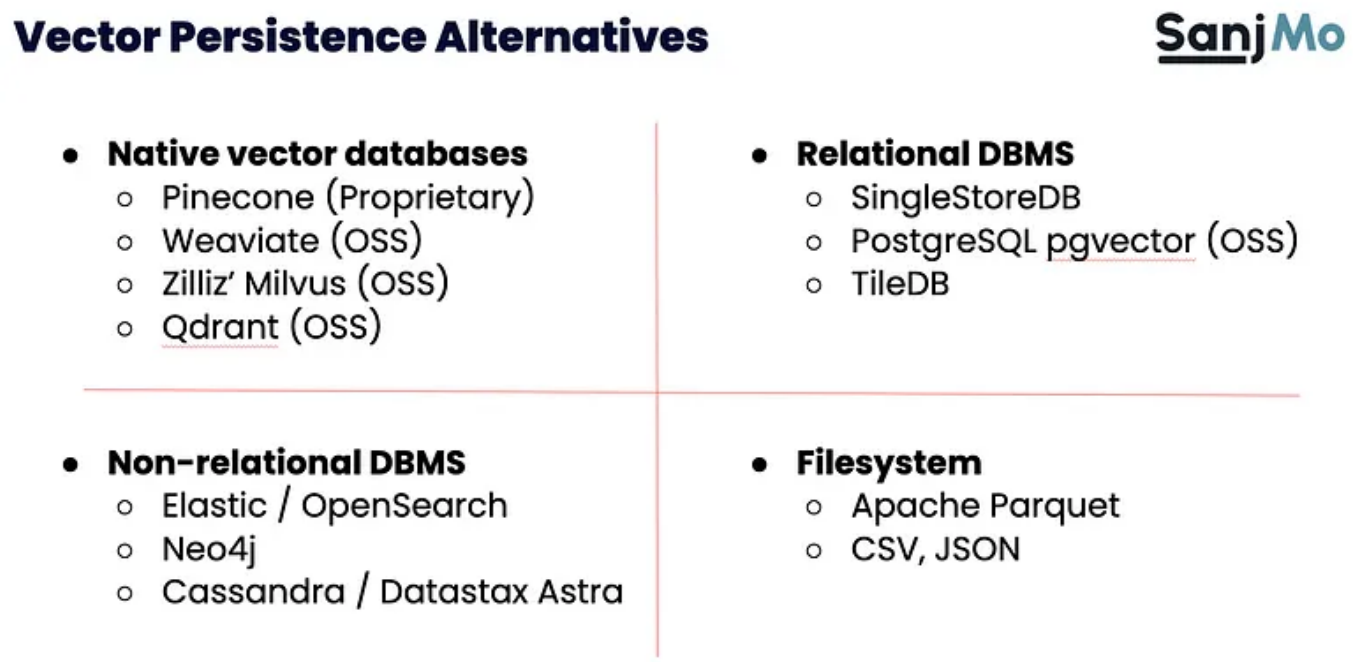
- **Long-term memory**

Native vector databases are specialty databases built specifically to handle vectors. Many non-relational DBMS and relational databases are also adding support to handle vectors. Search data stores like Elastic that already offered 'inverted search' are now being explored as an option to provide vector search.

Databases, like SingleStoreDB and many others already support vector embeddings with support for native semantic search functions — although these capabilities were not heavily emphasized in the past for traditional workloads. Now they are.

Finally, there is no reason vectors can't be stored in files, especially the ones that have support for columnar data, like Apache Parquet. However, the problem is that querying vectors sequentially in a file can be very slow unless indexes are used. You will see more on indexes later in this paper.

The figure below shows examples for long-term memory, although this list is simply representative as most database vendors are adding support for vectors:



The modern data stack was already bursting at the seams when generative AI became the talk of the town. There was already a rallying cry for simplification. *So, rebooting the data strategy must support complexity reduction.*

This means exploring whether and how currently deployed data and analytical technologies can be utilized for the vector searches on private data. Hence, this paper focuses on the last option of using an RDBMS.

Finally, it is time to demystify several new concepts and terms that have been mentioned thus far, such as vectors and embeddings. To do so, we need to understand

how exactly an AI search works.

Value Chain to Use LLMs on Private Data

Enabling natural language search of enterprise data using a chatbot can significantly expand the number of data consumers and use cases. Besides the search, LLMs, which use deep neural network algorithms, can be used for advanced tasks, such as summarizing documents, ranking, and recommendations, etc.

Let's say, for example, you search for a very specific product on a retailer's website, and the product is not available. An additional API call to an LLM with your request that returned zero results may result in a list of similar products. This is an example of a vector search, which is also known as a similarity or semantic search.

The current state of applications, like ChatGPT, use GTP-3 and GPT-4 LLMs, which have been trained on public data until Sep 2021. So, the LLM has no information about World Cup Soccer, which concluded in December 2022. Also, training LLMs is a very expensive and a batch process requiring expensive infrastructure like 10,000 GPUs for ChatGPT.

To circumvent these limitations and leverage more recent data, the user can insert, say the Wikipedia page on World Cup Soccer, in the API call to the GPT-3 LLM. Now, the LLM can use this "model input" or "prompt" to answer your question. However, the size of this input is limited to 4K tokens for GPT-3 (almost 5 pages) to 32K for GPT-4 (almost 40 pages). A token could be a word, or a segment of text or code and is the model input to the LLM.

Now, let's pivot to business' need where the requirement is to search enterprise data and generate fresh new insights. We will look at a marketing example to increase customer conversion. Your app should analyze all incoming data in real time, apply models to generate personalized offers and execute them while your users are in your app. To do so, you'd have to first extract the data from your transactional DB, extract, load and transform using a batch operation, run analytics in your OLAP engine and then finally create segments and generate offers.

In the new AI model, you ingest the data in real time, apply your models by reaching to one or multiple GPT services and action on the data while your users are in the online

experience. These GPT models may be used for recommendation, classification personalization, etc., services on real-time data. Recent developments, such as LangChain and AutoGPT, may further disrupt how modern applications are deployed and delivered.

To enable this, the following **three-step process** comprises preparing the data for vector search and enabling users.

Step 1. Prepare data for vector search

Eventually, we get to dwell on what a vector is. Conventional search works on keys. However, when the ask is a natural query, that sentence needs to be converted into a structure so that it can be compared with words that have similar representation. This structure is called an ***embedding***. An embedding uses ***vectors*** that assign coordinates into a graph of numbers — like an array. *An embedding is high dimensional as it uses many vectors to perform semantic search.*

Without the embedding vectors, the LLM cannot extract the context of the prompt and relevantly respond.

When a search is made on a new text, the model calculates the "distance" between terms. For example, searching for "king" is closer to "man," than to "woman." This distance is calculated on the "nearest neighbors" using functions like, cosine, dot product and Euclidean.

So far so good, but how many nearest neighbors must the algorithm look up? What if there are millions of neighbors? This is where "approximate nearest neighbors" (ANN) algorithms are used to reduce the vector search space. A very popular way to index the vector space is through a library called 'Hierarchical Navigable Small World (HNSW).' Many vector databases and libraries like FAISS use HNSW to speed up vector search.

Databases used for generative AI workloads must enable the ability to convert their data into embedding vectors, persist them, and index them for fast lookup.

Let's look at a workflow:

1. **Ingest data into a database.** You can do this using the standard ingestion techniques. In a RDBMS, it will load this data into tables. This ingest mechanism

may be batch loads, but streaming ingest will give the ability to operate on the latest data. The destination may be an array or a JSON data type.

2. **Harmonize data**. This is a lightweight data transformation step to help with data quality and consistent content formatting. It is also where data may need to be enriched. The data may be converted into a list.

3. **Encode data**. This step is used to convert the ingested data into embeddings. One option is to use an external API. For example, OpenAI's ADA and sentence_transformer have many pre-trained models to convert unstructured data like images and audio into vectors.

4. **Load embedding vectors**. In this step, data is moved to a table that mirrors the original table but has an additional column of type 'vector,' JSON or a blob that stores the vectors.

5. **iPerformance tuning**. Like any other database use case, a search has to be fine tuned. Some available options include: **compression** for faster searching in memory. SingleStoreDB provides JSON_ARRAY_PACK. And **indexing vector** using HNSW as mentioned earlier. This allows parallel scans using SIMD.

This completes the back-end tasks for preparing the data to be used for vector search.

Step 2. Perform vector search

The action now shifts to the front end — or to the consumer who is using a chat bot like ChatGPT — to ask a question. The question, or the prompt is in natural language, which needs to be converted into a vector first. This is done through a call to an LLM like GPT-3.

Next, you want to search enterprise data first to find matches, enrich it with additional context and leverage the LLM for the second time. As we saw earlier, the payload, or tokens, that can be passed to the LLM are limited (with GPT-4, which is still not publicly available to everyone at the time of writing this article, you can send up to ~40 pages of data as context). So doing a vector search of the user input with the corporate database will reduce the amount of data we need to send to the LLM.

By using a traditional database that supported vectors, it is possible to mix a traditional keyword search that joins various tables with a vector search before sending that request to the LLM.

Step 3. Leverage the LLM

With the matches made from our databases, data warehouses or the lake houses, we now want the LLM to perform, say, a recommendation. The matches are sent to the LLM APIs.

When ChatGPT was first introduced, many organizations banned it as the model inputs were being used by OpenAI to train or improve models. In a new updated data usage policy, effective March 2023, OpenAI no longer uses users' data for training purposes. It does retain the prompts for 30 days but only for legal reasons, after which the data is deleted.

The LLM completes users' requests and sends the response back.

Summary

Generative AI space is nascent and a work in progress. This document captures the essence of what is needed to accomplish the promises of semantic searches and the technologies that undergird the ecosystem. The focus, however, is primarily on the

guiding principles that apply to other data management disciplines should still be adhered to. We hope this document helps in demystifying the concepts needed to leverage AI workloads

The next document on databases for generative AI will provide an evaluation criteria on how to choose the optimal database technologies.

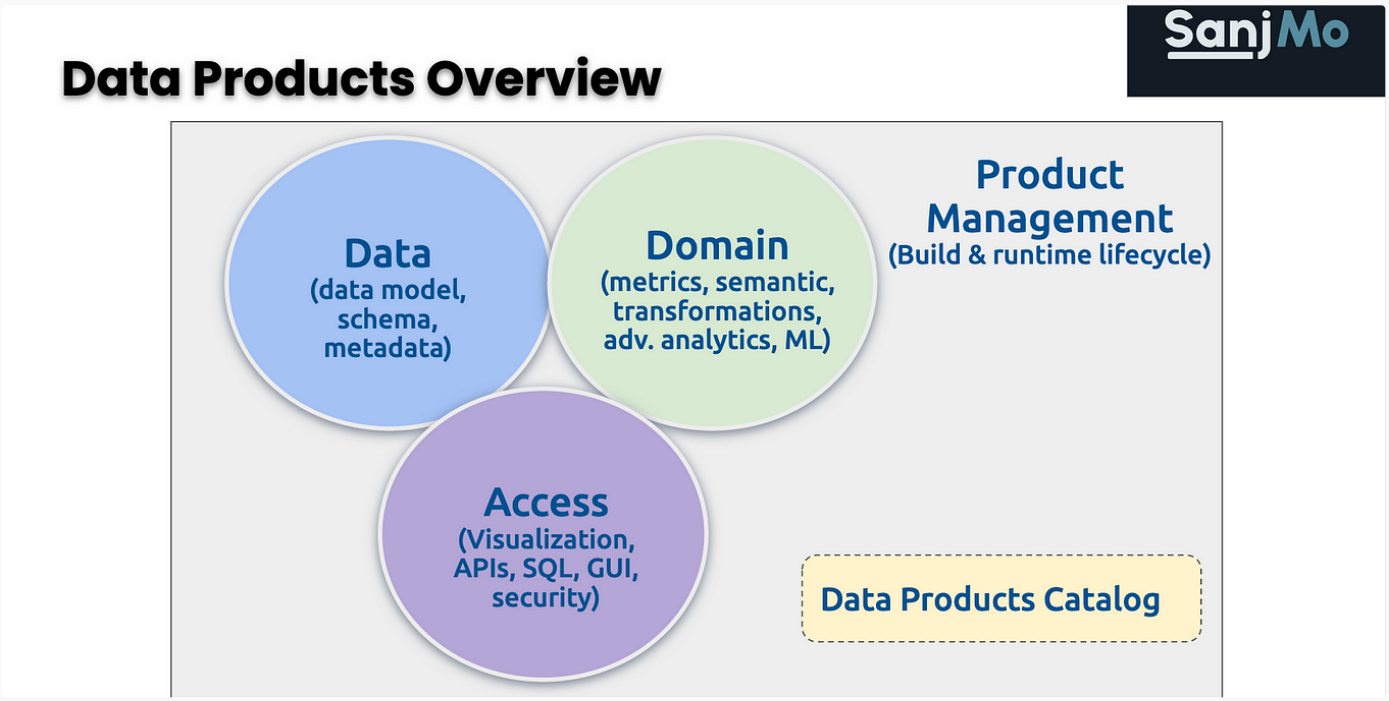
Follow

Written by Sanjeev Mohan

423 Followers

Sanjeev researches the space of data and analytics. Most recently he was a research vice president at Gartner. He is now a principal with SanjMo.

More from Sanjeev Mohan



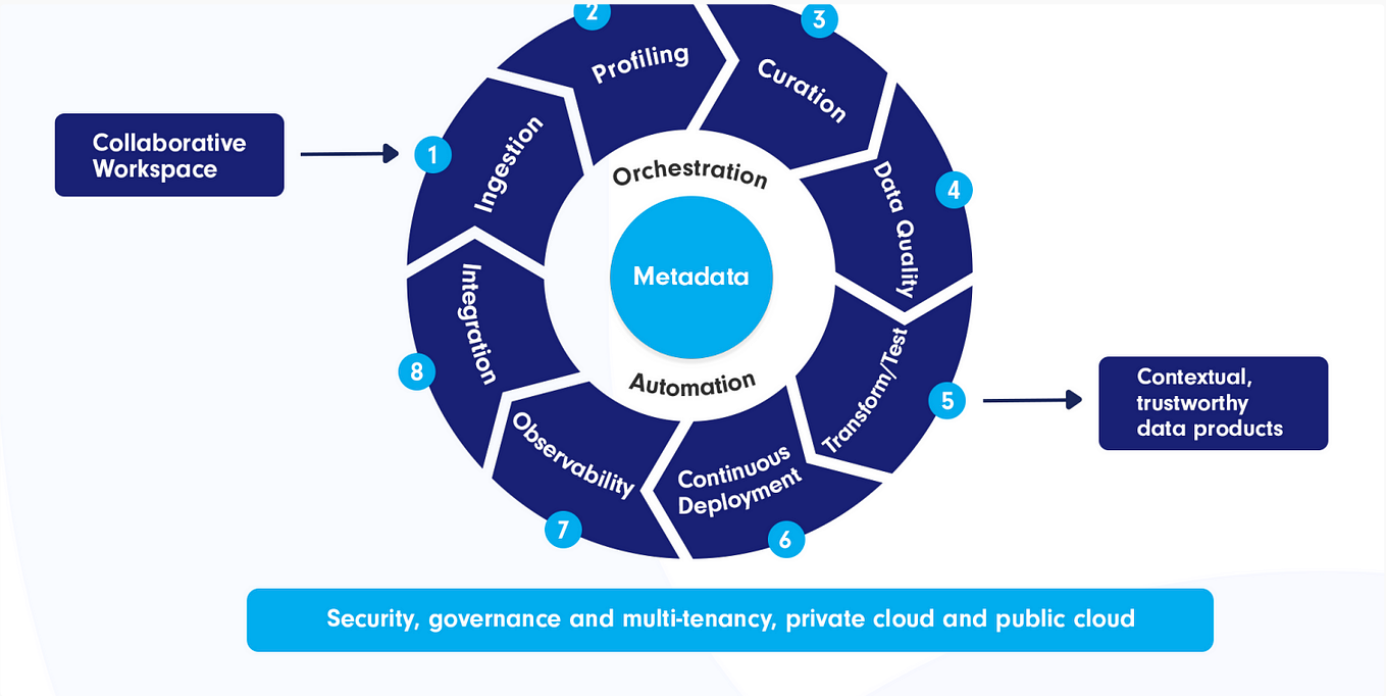
Sanjeev Mohan

What Exactly is a Data Product?

This blog has taken much longer to write than I had expected. Everyone in my circles who read an initial version had a different take on...

10 min read · Mar 17

70 5



Sanjeev Mohan

Delivering Modern Data Solutions Faster

With the launch of Microsoft Fabric last week, the discussion between vertically integrated ecosystem versus a composable ecosystem of best...

12 min read · May 28

5

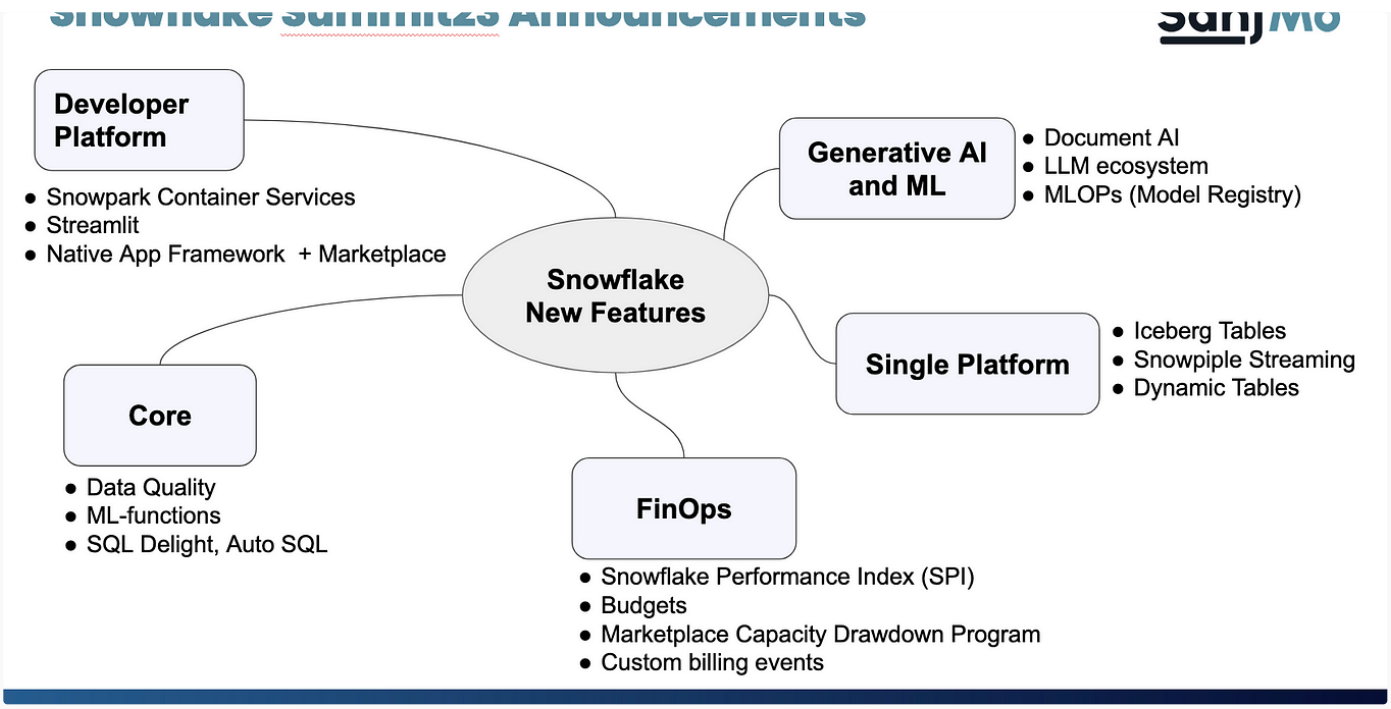
Sanjeev Mohan

SanjMo: 2nd Anniversary Newsletter

Two years ago, I set out on an unknown journey brimming with trepidation. I started SanjMo to continue pursuing my role as an industry...

4 min read · Aug 4

4 1



Sanjeev Mohan

Snowflake Summit 2023 Announcements

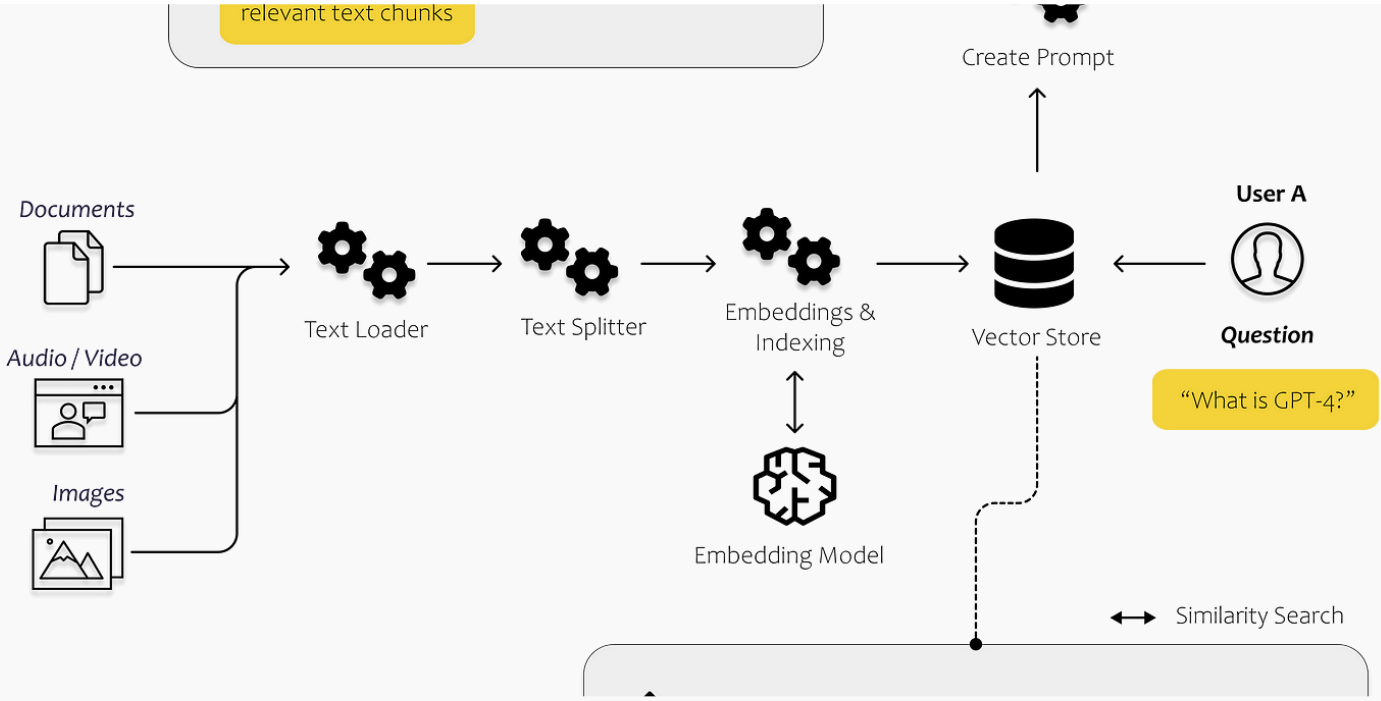
This document showcases new developments—some are GA, many are in private or public preview. As the status is constantly changing, this...

5 min read · Jun 30

31

See all from Sanjeev Mohan

Recommended from Medium



Dominik Polzer in Towards Data Science

All You Need to Know to Build Your First LLM App

A step-by-step tutorial to document loaders, embeddings, vector stores and prompt templates

· 26 min read · Jun 21

4.5K 42

Yingjun Wu in Data Engineer Things

Why You Shouldn't Invest In Vector Databases?

Delving into the vector database field from three perspectives: technology, applications, and the market.

8 min read · Jun 7

637 12

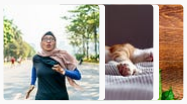
Lists

Staff Picks

408 stories · 230 saves

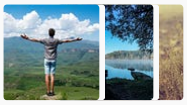
Stories to Help You Level-Up at Work

19 stories · 178 saves



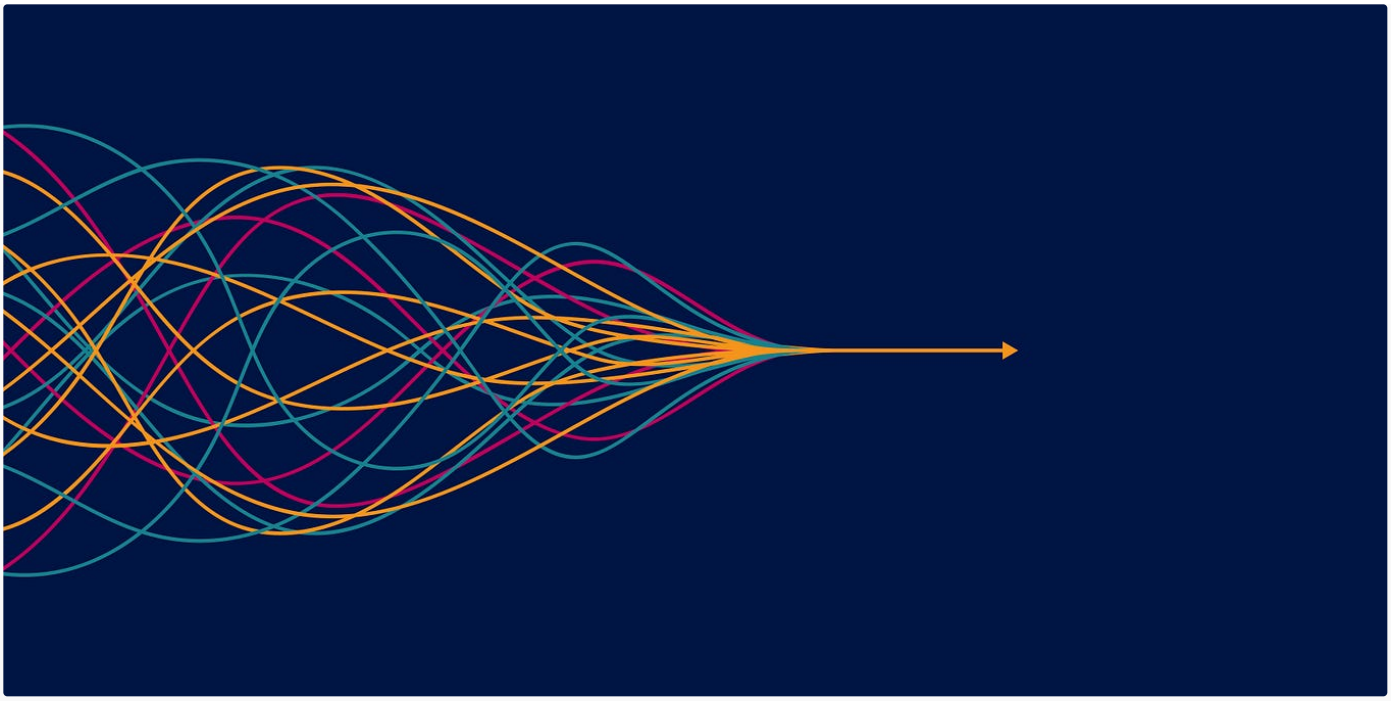
Self-Improvement 101

20 stories · 444 saves



Productivity 101

20 stories · 445 saves



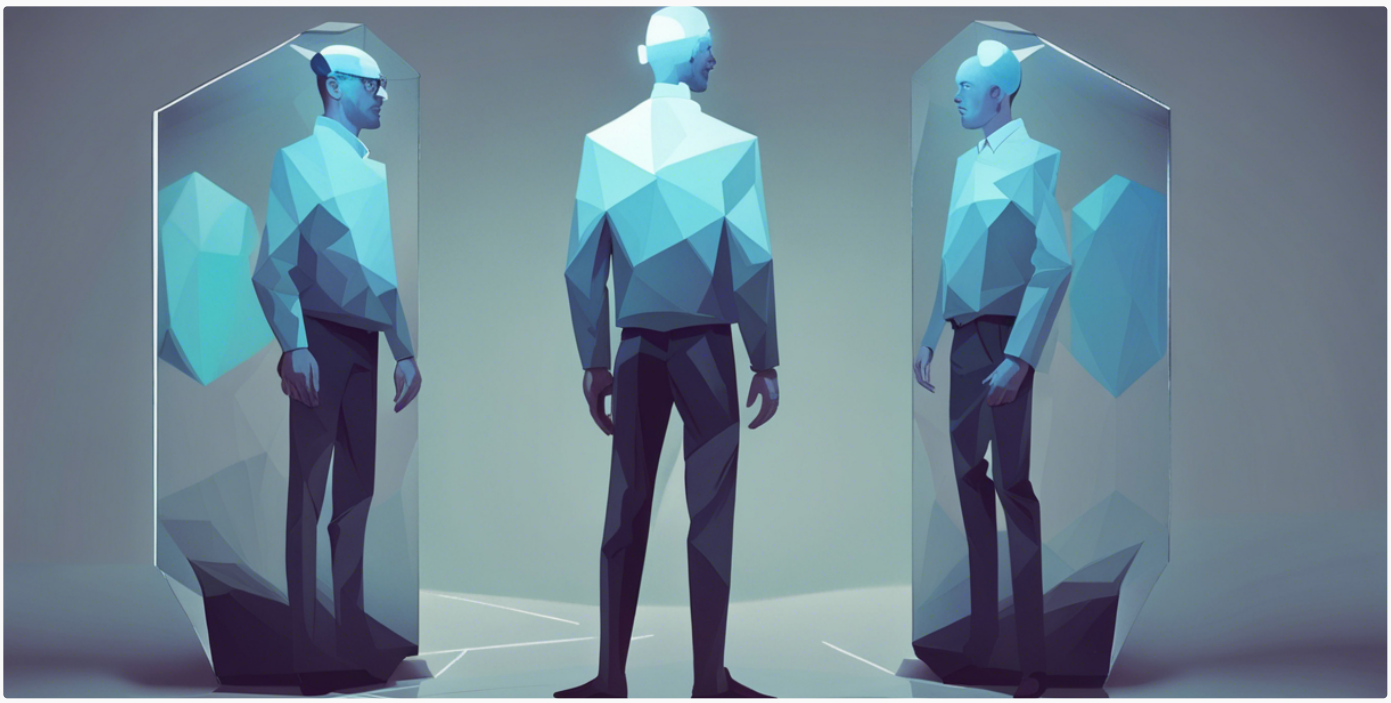
Analytics at Meta

The future of the data engineer—Part I

Introduction

10 min read · Apr 3

647 9



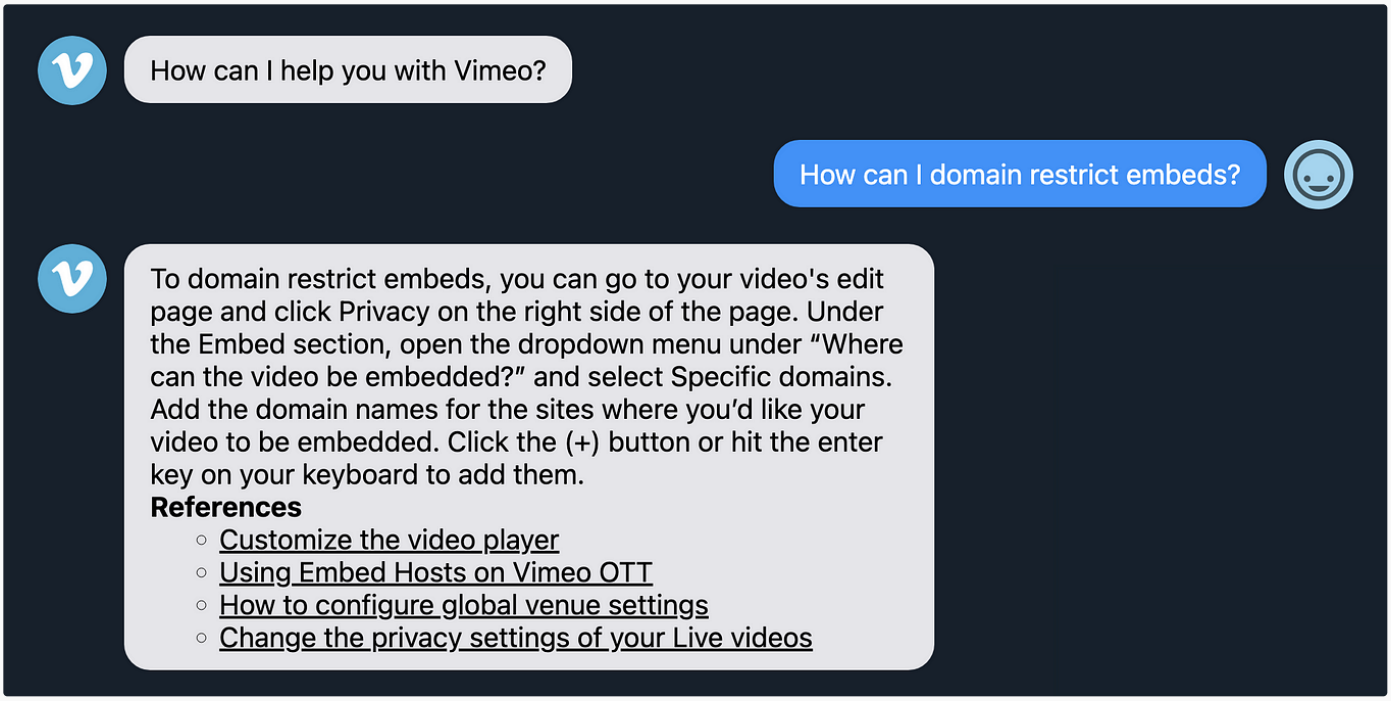
Sergei Savvov in Better Programming

Create a Clone of Yourself With a Fine-tuned LLM

Unleash your digital twin

11 min read · Jul 27

1.8K 14



Jon Ruddell in Vimeo Engineering Blog

From idea to reality: Elevating our customer support through generative AI

How we prototyped and enhanced the Vimeo Help Desk through rigorous testing.

13 min read · Aug 11

279 6