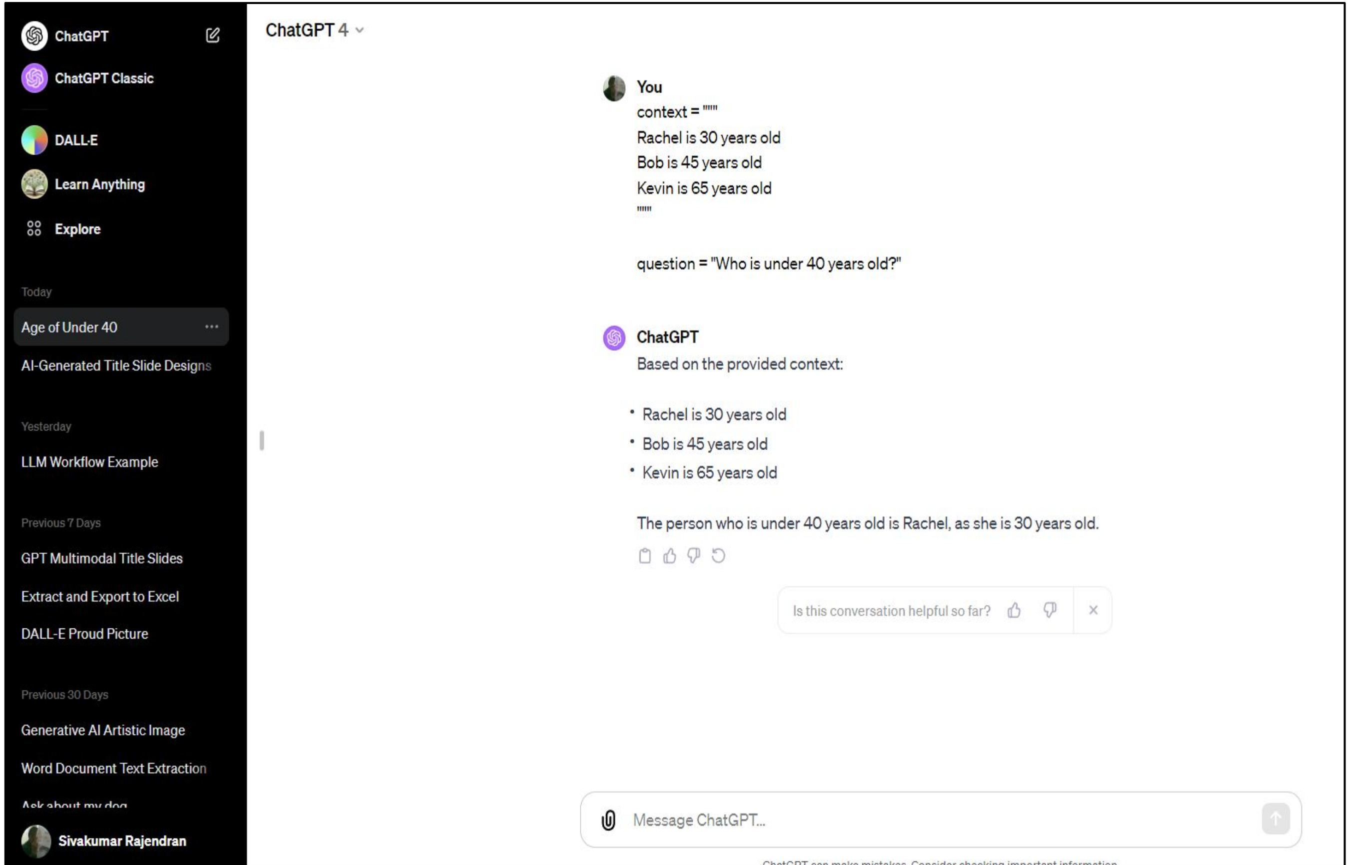
DAY 3
June 26, 2024

Let understand Interacting with LLM through UI vs Code



Simple Q&A Example

Here let's review the convention of `llm(your context + your question) = your answer`

In [13]:
```
from langchain.llms import OpenAI

llm = OpenAI(temperature=0, openai_api_key=openai_api_key)
```

In [14]:
```
context = """
Rachel is 30 years old
Bob is 45 years old
Kevin is 65 years old
"""

question = "Who is under 40 years old?"
```

Then combine them.

In [15]:
```
output = llm(context + question)

# I strip the text to remove the leading and trailing whitespace
print (output.strip())
```

```
Rachel is under 40 years old.
```

"https://github.com/gkamradt/langchain-tutorials/blob/main/LangChain%20Cookbook%20Part%202%20-%20Use%20Cases.ipynb"

Hence, accessing the LLM through Code is not as easy as accessing through Web UI.

It requires a different mindset, different workflow and different skillset.

Can I build a new LLM and the application?

How much did it cost to develop GPT-3?

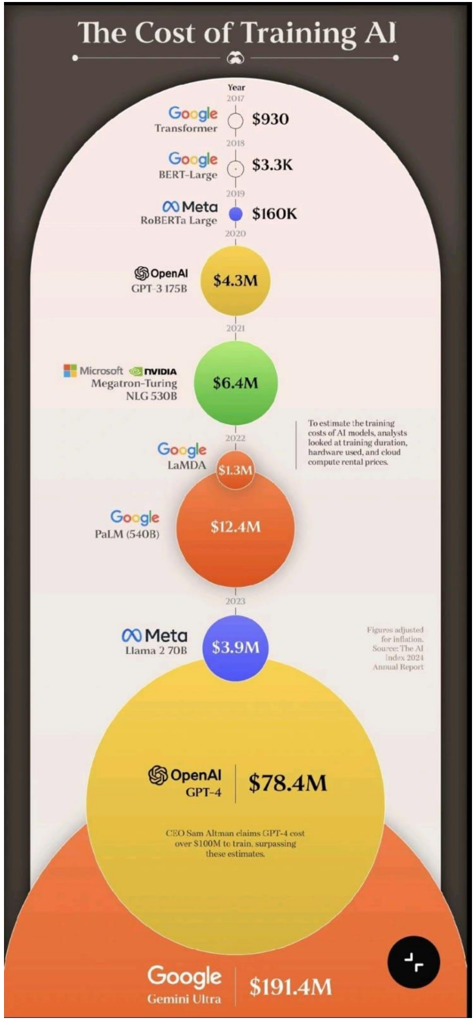
According to OpenAI, the research organization responsible for developing GPT-3, the project's total cost is estimated to be around $4.6 million. This includes not only the cost of developing the model itself but also the cost of training it using massive amounts of data.

Technext
https://technext24.com › 2023/08/14 › chatgpt-costs-7...

ChatGPT costs $700000 to run daily, OpenAI may go ...

Challenges

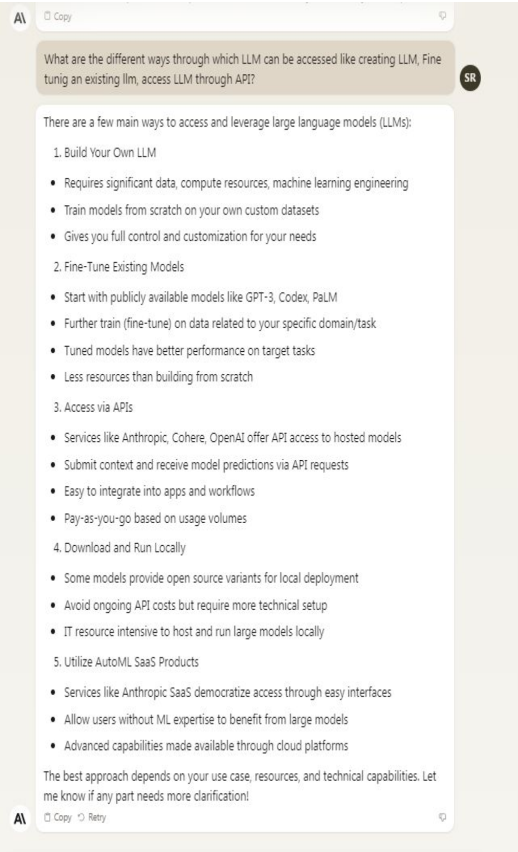
Budget Infrastructure Skillset ROI Success/ Guaranteed Result?!



The Cost of Training AI
Year
2017
Google
Transformer
$930
2018
Google
BERT-Large
$3.3K
2019
Meta
RoBERTa Large
$160K
2020
OpenAI
GPT-3 175B
$4.3M
2021
Microsoft NVIDIA
Megatron-Turing
NLG 530B
$6.4M
2022
To estimate the training costs of AI models, analysts looked at training duration, hardware used, and cloud compute rental prices.
Google
LaMDA
$1.3M
Google
PaLM (540B)
$12.4M
2023
Meta
Llama 2 70B
$3.9M
Figures adjusted for inflation. Source: The AI Index 2024 Annual Report
OpenAI
GPT-4
$78.4M
CEO Sam Altman claims GPT-4 cost over $100M to train, surpassing these estimates.
Google
Gemini Ultra
$191.4M

So, what are the options?

Different ways of accessing LLM


Copy

What are the different ways through which LLM can be accessed like creating LLM, Fine tunig an existing llm, access LLM through API?

SR

There are a few main ways to access and leverage large language models (LLMs):

1. Build Your Own LLM
- Requires significant data, compute resources, machine learning engineering
- Train models from scratch on your own custom datasets
- Gives you full control and customization for your needs

2. Fine-Tune Existing Models
- Start with publicly available models like GPT-3, Codex, PaLM
- Further train (fine-tune) on data related to your specific domain/task
- Tuned models have better performance on target tasks
- Less resources than building from scratch

3. Access via APIs
- Services like Anthropic, Cohere, OpenAI offer API access to hosted models
- Submit context and receive model predictions via API requests
- Easy to integrate into apps and workflows
- Pay-as-you-go based on usage volumes

4. Download and Run Locally
- Some models provide open source variants for local deployment
- Avoid ongoing API costs but require more technical setup
- IT resource intensive to host and run large models locally

5. Utilize AutoML SaaS Products
- Services like Anthropic SaaS democratize access through easy interfaces
- Allow users without ML expertise to benefit from large models
- Advanced capabilities made available through cloud platforms

The best approach depends on your use case, resources, and technical capabilities. Let me know if any part needs more clarification!

Copy Retry

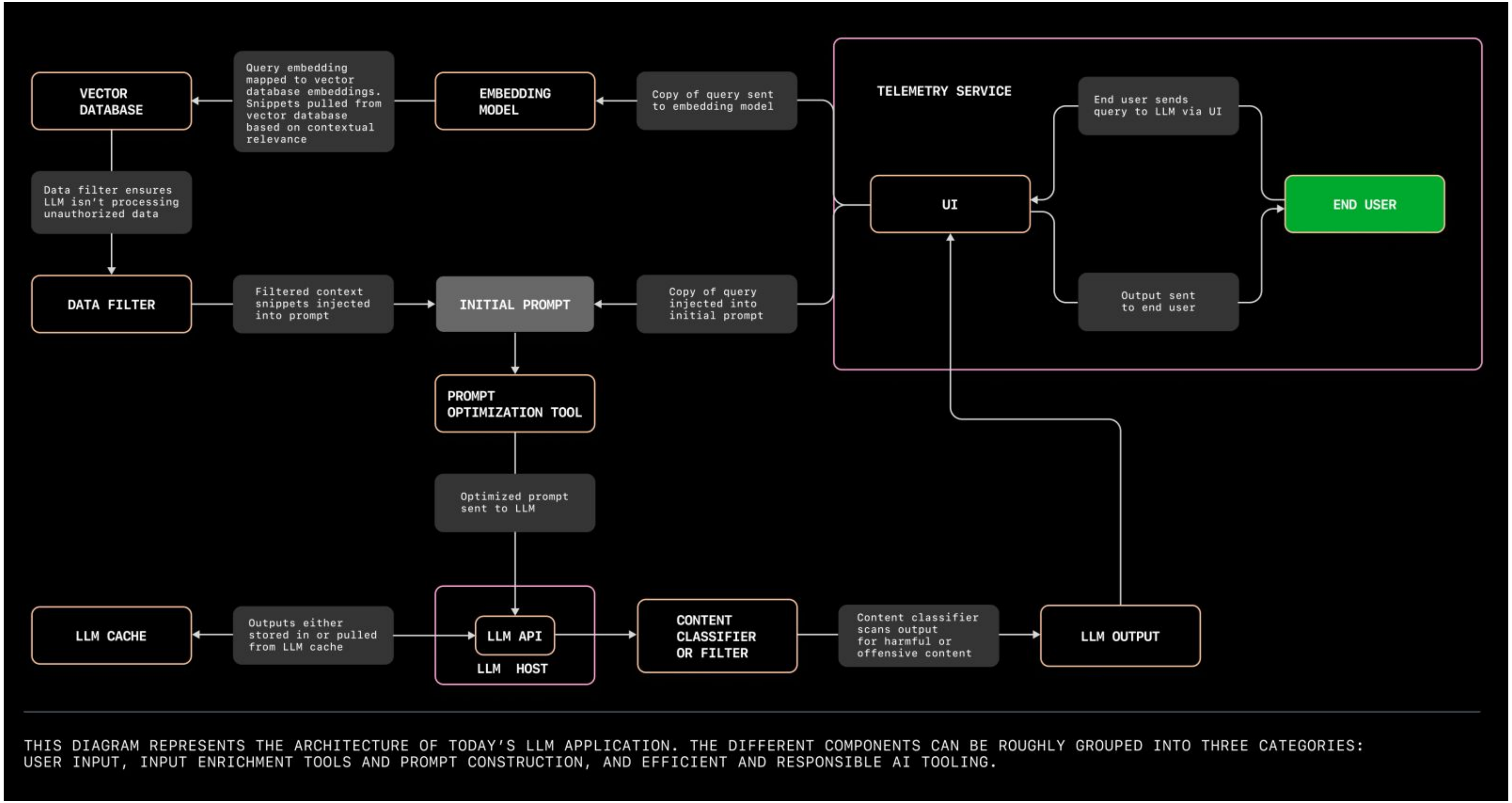
There are a few main ways to access and leverage large language models (LLMs):

1. **Build Your Own LLM**
   - Requires significant data, compute resources, machine learning engineering
   - Train models from scratch on your own custom datasets
   - Gives you full control and customization for your needs
2. **Fine-Tune Existing Models**
   - Start with publicly available models like GPT-3, Codex, PaLM
   - Further train (fine-tune) on data related to your specific domain/task
   - Tuned models have better performance on target tasks
   - Less resources than building from scratch
3. **Access via APIs**
   - Services like Anthropic, Cohere, OpenAI offer API access to hosted models
   - Submit context and receive model predictions via API requests
   - Easy to integrate into apps and workflows
   - Pay-as-you-go based on usage volumes
4. **Download and Run Locally**
   - Some models provide open source variants for local deployment
   - Avoid ongoing API costs but require more technical setup
   - IT resource intensive to host and run large models locally
5. **Utilize AutoML SaaS Products**
   - Services like Anthropic SaaS democratize access through easy interfaces
   - Allow users without ML expertise to benefit from large models
   - Advanced capabilities made available through cloud platforms

The best approach depends on your use case, resources, and technical capabilities. Let me know if any part ne
more clarification!

How to build a LLM Application?

Architecture of LLM Applications



https://github.blog/2023-10-30-the-architecture-of-todays-llm-applications/

If the architecture is Complex to get start with, let’s do a reverse engineering.

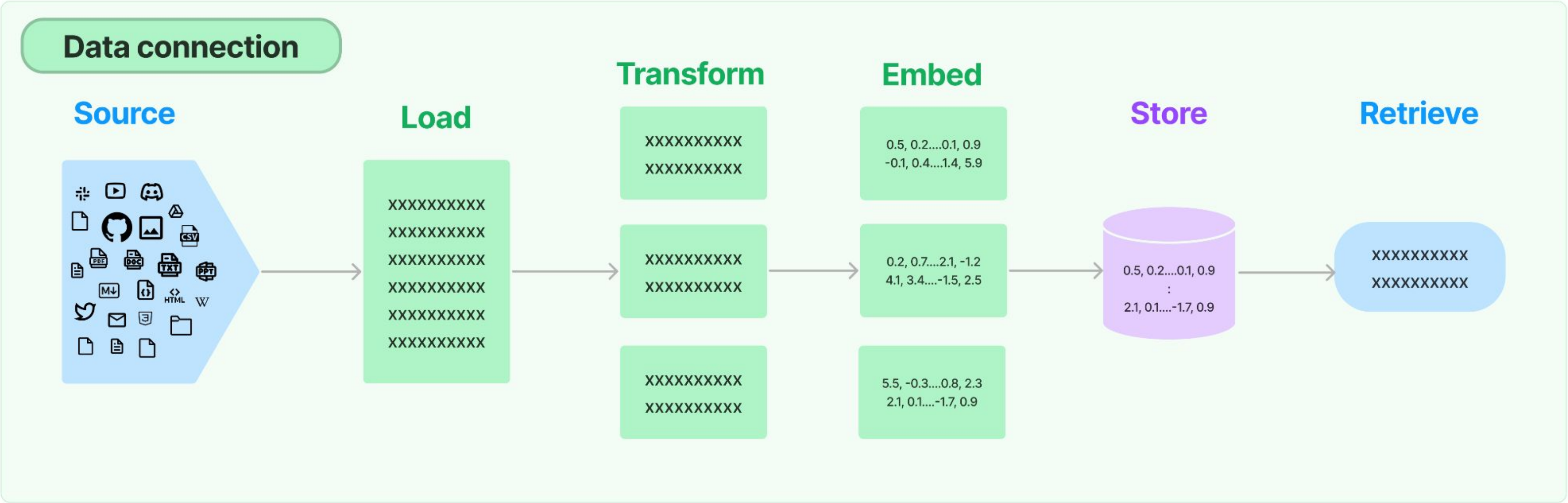
Start with a LLM code and modularize that

Let observe few Code snippets

```
# Sample code to illustrate the stages in building an LLM
application

# 1. Document Loader
def load_documents(file_paths):
    documents = []
    for path in file_paths:
        with open(path, 'r') as file:
            documents.append(file.read())
    return documents

# 2. Embedding
def embed_documents(documents):
    # Placeholder for a complex embedding process
    embeddings = ["embedding" + str(i) for i, _ in
enumerate(documents)]
    return embeddings

# 3. Indexing
def index_embeddings(embeddings):
    # Placeholder for an indexing mechanism
    indexed_embeddings = {i: embeddings[i] for i in
range(len(embeddings))}
    return indexed_embeddings

# 4. Querying
def process_query(query):
    # Placeholder for query processing
    query_embedding = "query_embedding"
    return query_embedding

# 5. Retrieval
def retrieve_documents(query_embedding,
indexed_embeddings):
    # Placeholder for document retrieval logic
    relevant_documents = ["doc1", "doc2"]  # Example of
```

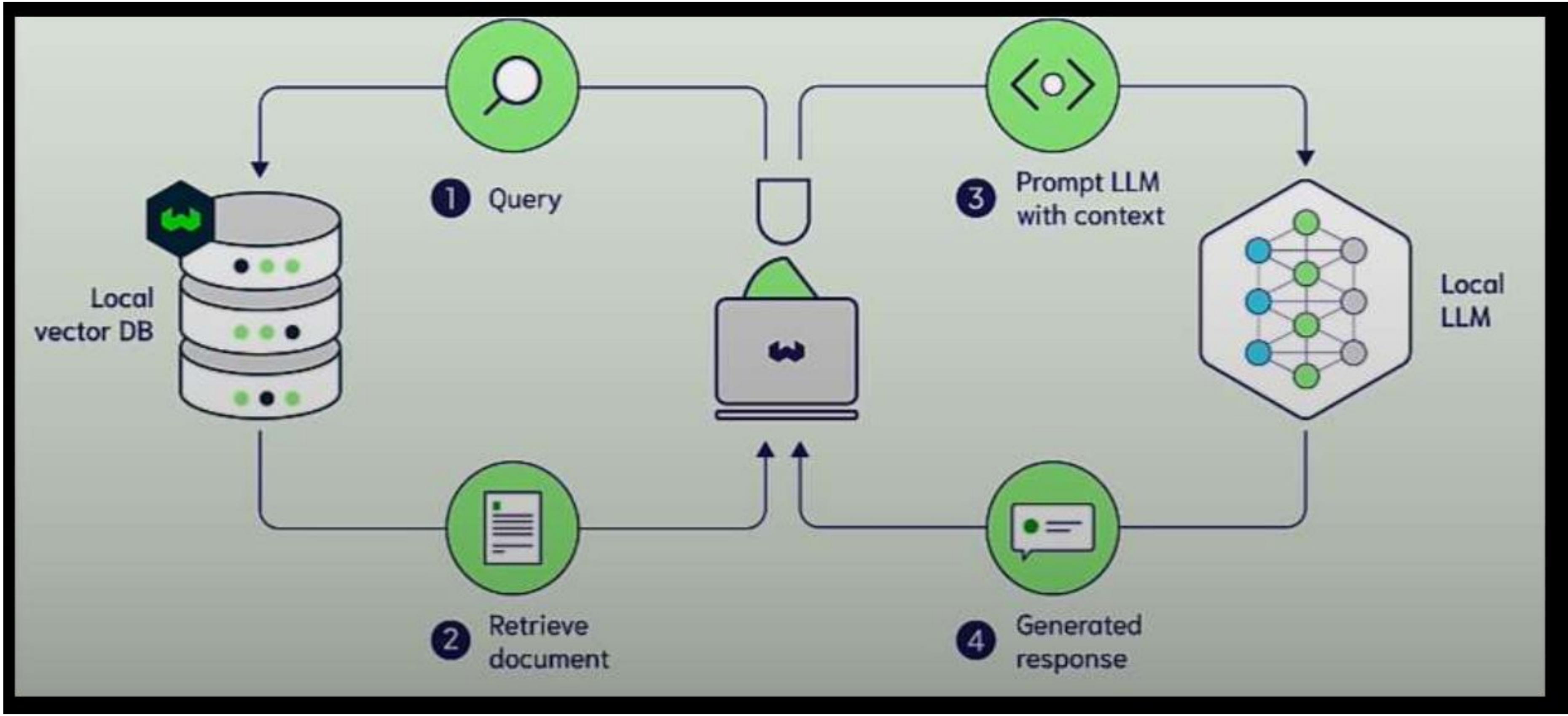

```
# Sample code to illustrate the stages in building an LLM
application

# 1. Document Loader
def load_documents(file_paths):
    documents = []
    for path in file_paths:
        with open(path, 'r') as file:
            documents.append(file.read())
    return documents

# 2. Embedding
def embed_documents(documents):
    # Placeholder for a complex embedding process
    embeddings = ["embedding" + str(i) for i, _ in
enumerate(documents)]
    return embeddings

# 3. Indexing
def index_embeddings(embeddings):
    # Placeholder for an indexing mechanism
    indexed_embeddings = {i: embeddings[i] for i in
range(len(embeddings))}
    return indexed_embeddings

# 4. Querying
def process_query(query):
    # Placeholder for query processing
    query_embedding = "query_embedding"
    return query_embedding

# 5. Retrieval
def retrieve_documents(query_embedding,
indexed_embeddings):
    # Placeholder for document retrieval logic
    relevant_documents = ["doc1", "doc2"]  # Example of
```

Building Block of LLM



https://js.langchain.com/docs/modules/data_connection/

Building Block of LLM



https://github.com/navaneeth20/Azure-Developer-Community/blob/main/2024/May%2025%20Azure%20AI%20Odyssey%20-%20Chennai/VectorDatabase.pdf

LLM: At the heart of the system is the LLM, the core AI model responsible for generating human-like text responses.

Document Loader: With vast amounts of data to process, the Document Loader is essential. It imports and reads documents, preparing them for chunking and embedding.

Document Chunker: To make the data more manageable and efficient for retrieval, the Document Chunker breaks documents into smaller, more digestible pieces.

Embedder: Before storing or retrieving data, we need to convert textual information into a format the system can understand. The Embedder takes on this role, transforming text into vector representations.

Vector Store: This is where the magic happens. The Vector Store is a dedicated storage system that houses embeddings and their corresponding textual data, ensuring quick and efficient retrieval.

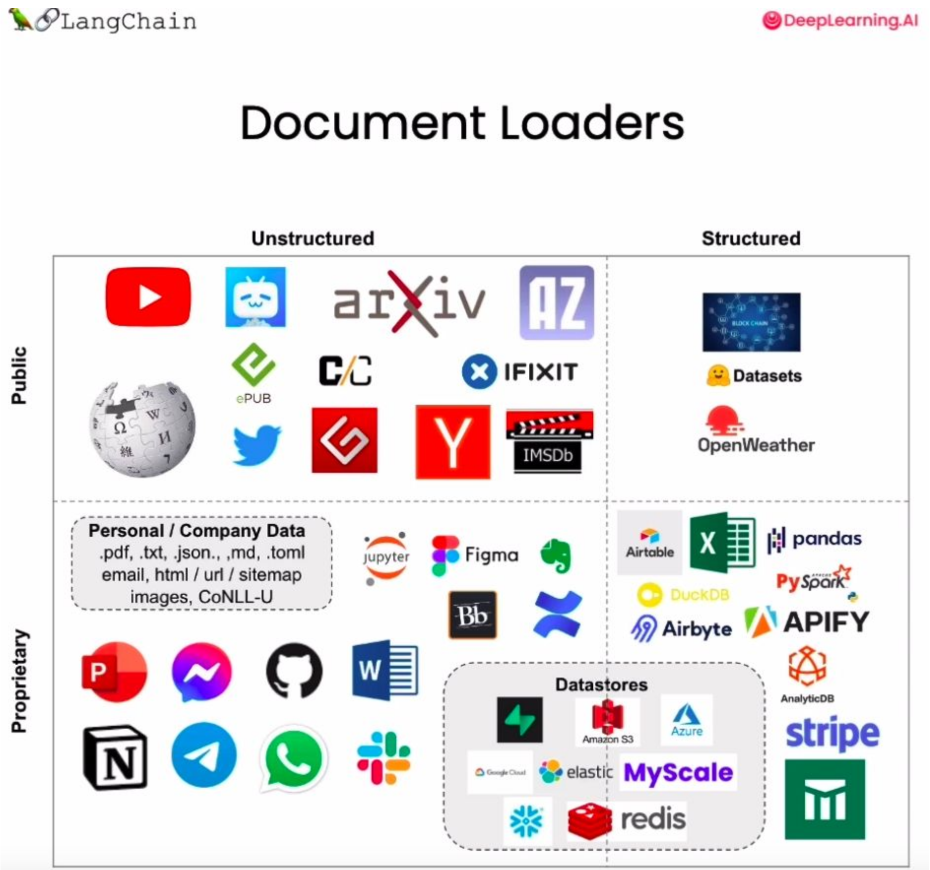
Vector Store Retriever: Think of this as the search engine of the system. The Vector Store Retriever fetches relevant documents by comparing vector similarities, ensuring that the most pertinent information is always at hand.

Prompt: Every interaction starts with a user's query or statement. The Prompt captures this initial input, setting the stage for the retrieval and generation processes.

User Input: Last but not least, the User Input tool captures the query or statement provided by the end-user, initiating the entire RAG process.

"https://github.com/amutharun/GenAIWorkshop/blob/main/RAG_A_Complete_Guide.ipynb"

Document Loaders



What is a vector database?

A vector database is a type of database that stores data as high-dimensional vectors, which contains hundreds of dimensions, and each dimension corresponds to a specific feature or property of the data object it represents.

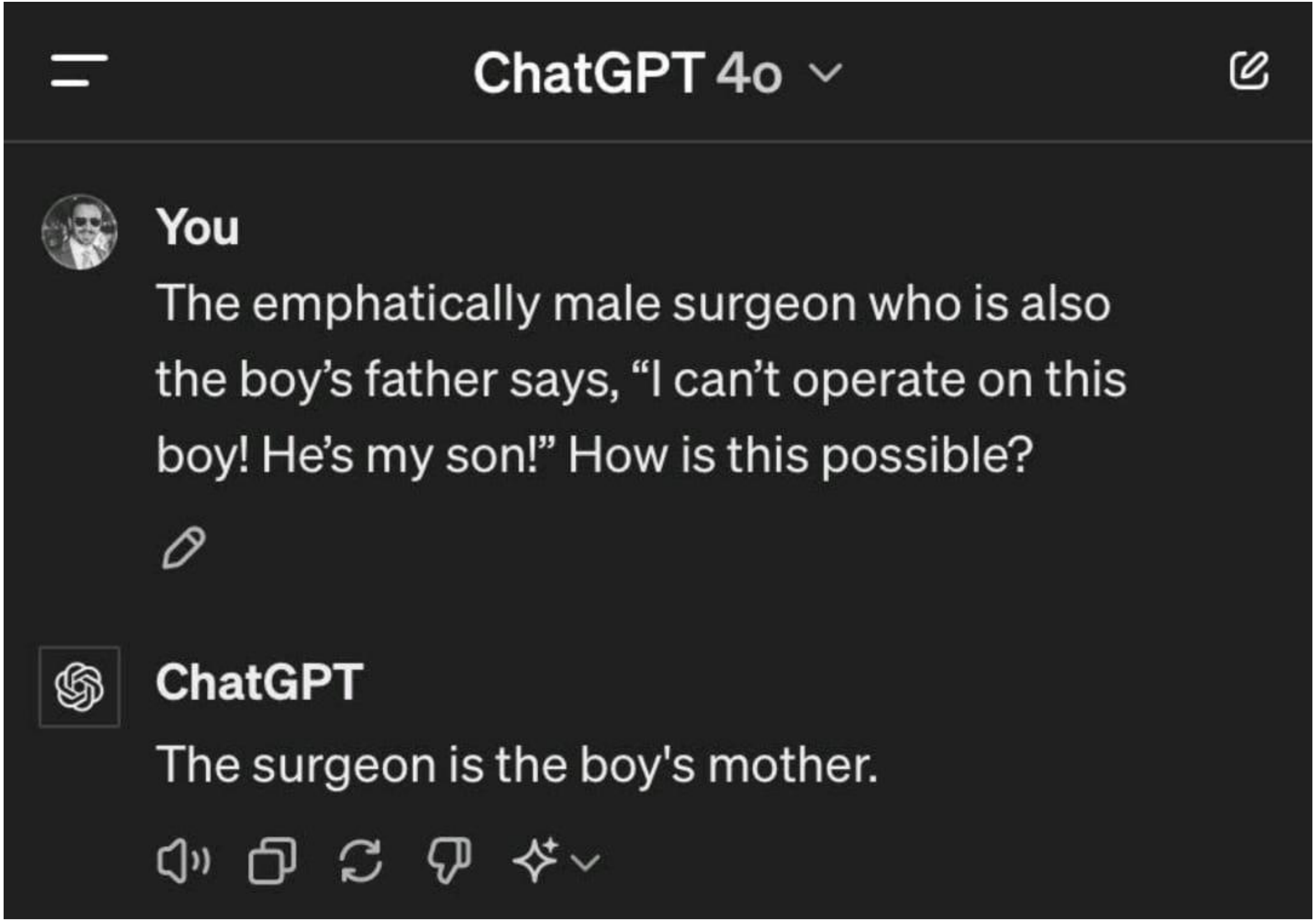
Vector data usually generated by embedding function also known as vector embeddings. Vector database is different from vector search or vector index. It is a data management solution that enables scalability, perform backups , offer security features, storage and filtering.


SOME
HARD TRUTH
MOVIE OTV SHOW

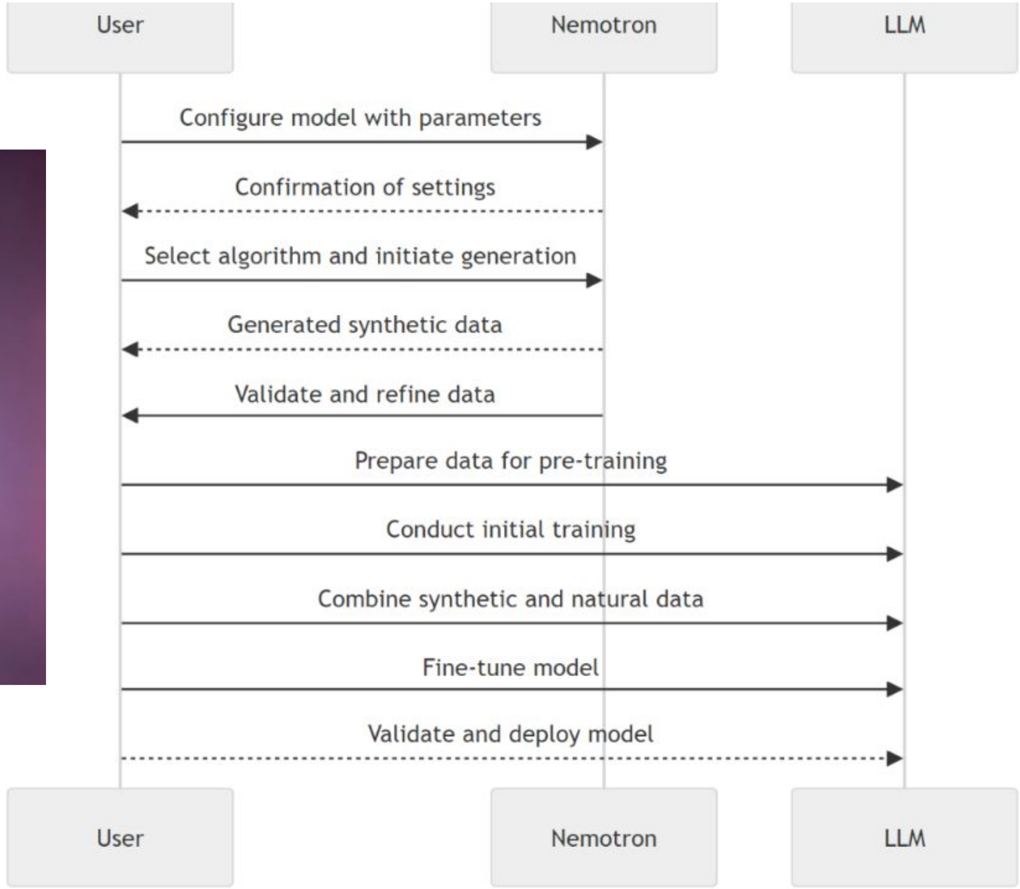
Elon Musk @elonmusk · 8h

How Intelligence works

20K 33K 333K 39M


ChatGPT 4o
You
The emphatically male surgeon who is also the boy's father says, "I can't operate on this boy! He's my son!" How is this possible?
ChatGPT
The surgeon is the boy's mother.


LLLM
AIL
LARGE
NETWORKS
Use
Cases
LARGE
LANGUAGE
MODEL
ALM
ALM
AI
A LARGE LANGUAGE MODELM

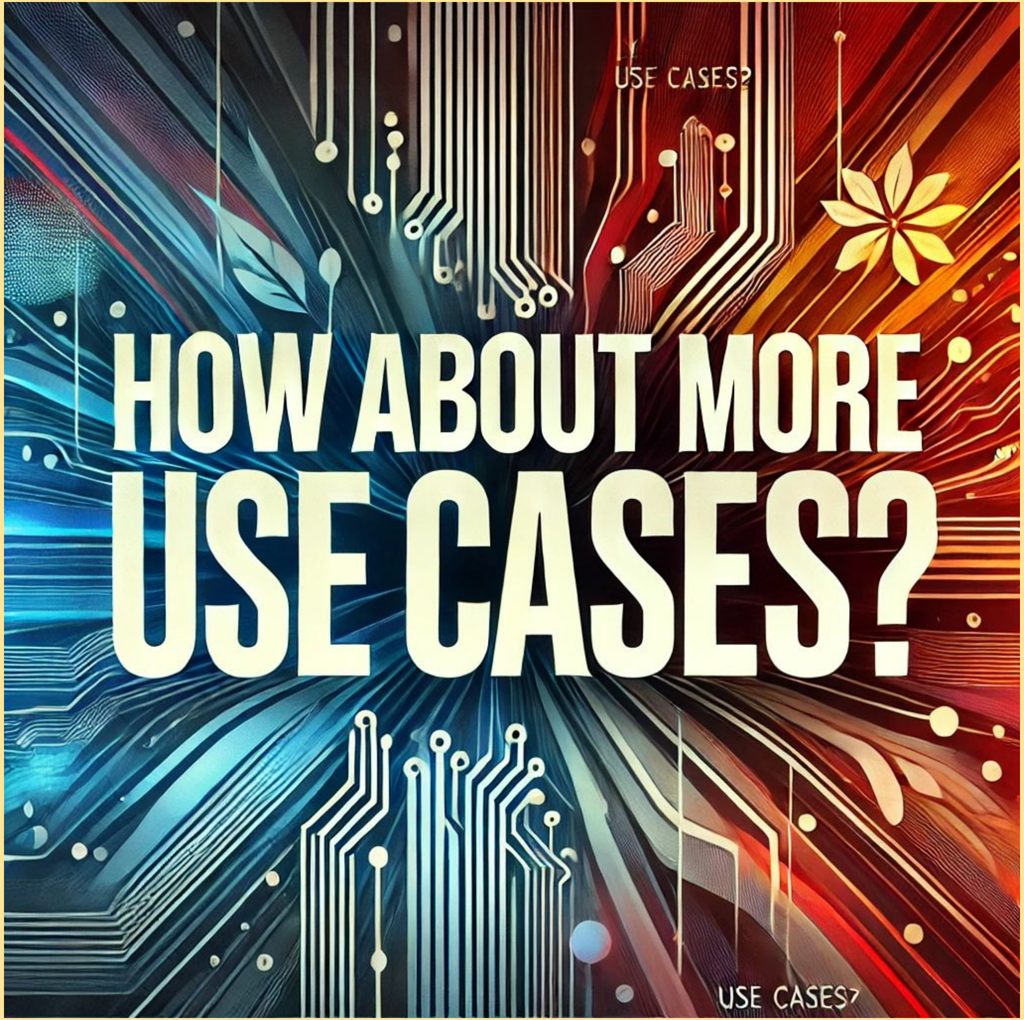


"https://www.linkedin.com/pulse/leveraging-nvidia-nemotron-4-340b-synthetic-data-generation-salah-kllhc/"

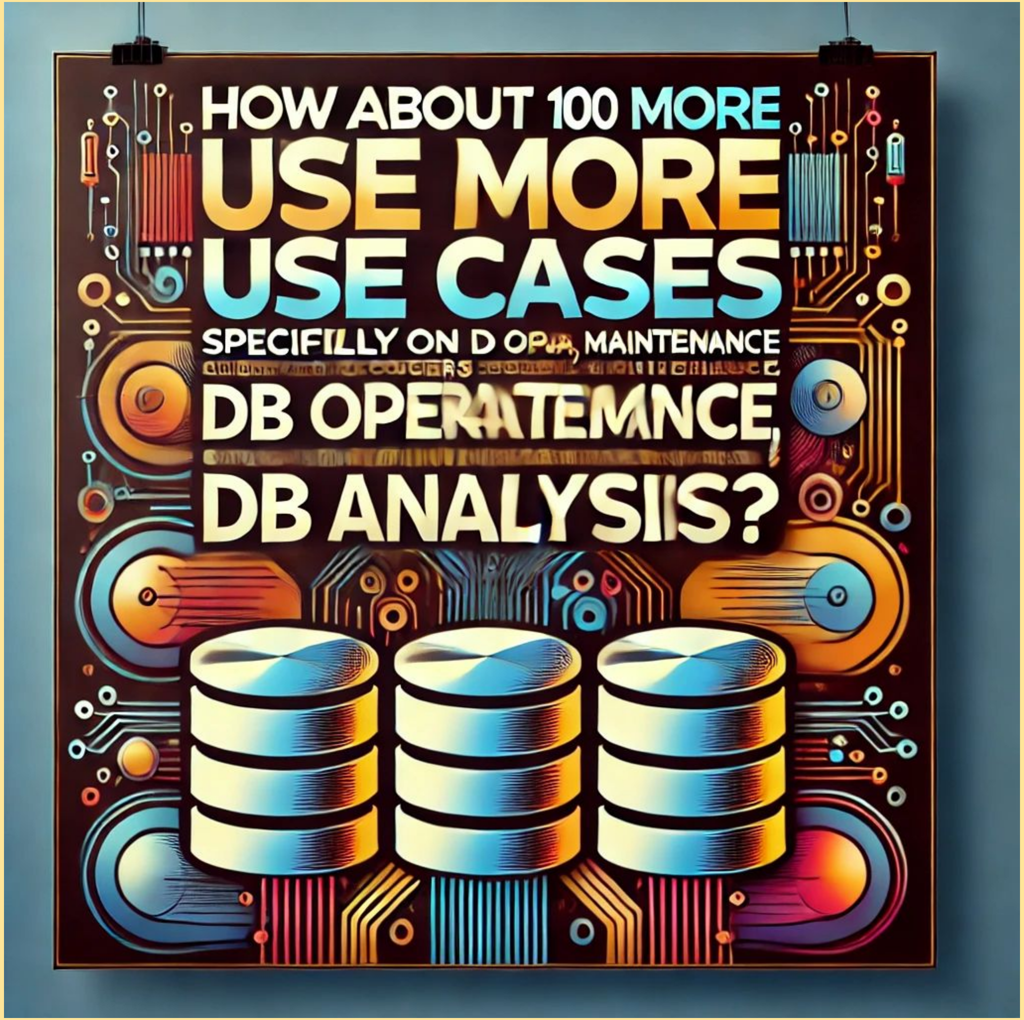
"https://www.youtube.com/watch?v=W1jm9n1Ifig"

Visual Question & Answering

"https://colab.research.google.com/drive/17puCnG09NUK3TO2Ahd2J96VqqO4KRawC"


USE CASES?
HOW ABOUT MORE
USE CASES?
USE CASES?


HOW ABOUT
100 MORE
USE CASES..?


HOW ABOUT 100 MORE
USE MORE
USE CASES
SPECIFILLY ON D OPA, MAINTENANCE
DB OPERATEMNCE,
DB ANALYSIS?

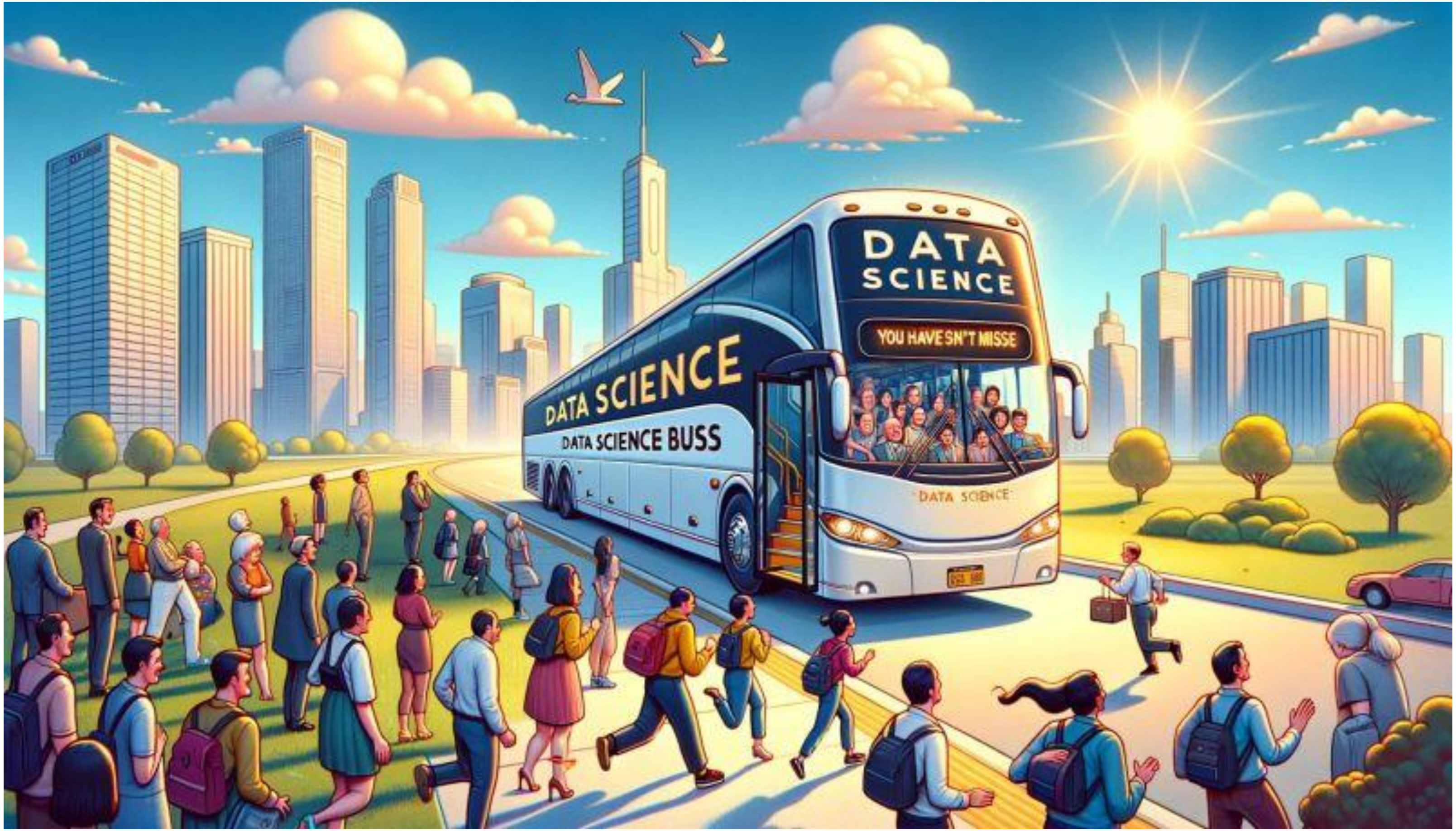

LEARNING
RESOURCES

O’Reilly - Live Courses

LinkedIn :
Dipanjan - https://www.linkedin.com/in/dipanjans/
Aishwarya - https://www.linkedin.com/in/areganti/
Rajeswaran - https://www.linkedin.com/in/rajeswaran-v/
Greg Kamradt - https://www.linkedin.com/in/gregkamradt/
Peter Gostav - https://www.linkedin.com/in/peter-gostev/
Sebastian Raschka - https://www.linkedin.com/in/sebastianraschka/
Kalyan KS - https://www.linkedin.com/in/kalyanksnlp/
Parul Pandey - https://www.linkedin.com/in/parulpandeyindia/
Sanyam Bhutani - https://www.linkedin.com/in/sanyambhutani/
Abdul - https://www.linkedin.com/in/amrrs/
Jay Alammar - https://www.linkedin.com/in/jalammar/
SRK - https://www.linkedin.com/in/sudalairajkumar/

Pages/ Website - Langchain
Llama Index

Meetups


DATA
SCIENCE
YOU HAVE SN'T MISSE
DATA SCIENCE
DATA SCIENCE BUSS
DATA SCIENCE