

From Developer lens

# Let understand Interacting with LLM through UI vs Code

The screenshot displays the ChatGPT web interface. On the left is a dark sidebar with navigation options: ChatGPT, ChatGPT Classic, DALL-E, Learn Anything, and Explore. Below these are sections for 'Today' (containing 'Age of Under 40'), 'Yesterday' (containing 'LLM Workflow Example'), and 'Previous 7 Days' (containing 'GPT Multimodal Title Slides', 'Extract and Export to Excel', and 'DALL-E Proud Picture'). At the bottom of the sidebar is the user's profile, 'Sivakumar Rajendran'.

The main chat area is titled 'ChatGPT 4'. It shows a conversation where the user provides a context of ages for Rachel (30), Bob (45), and Kevin (65), and asks 'Who is under 40 years old?'. ChatGPT responds with a bulleted list of the ages and then states, 'The person who is under 40 years old is Rachel, as she is 30 years old.' Below the response are icons for thumbs up, thumbs down, and a refresh button. At the bottom of the chat area is a feedback prompt: 'Is this conversation helpful so far?' with thumbs up/down icons and a close button.

At the very bottom of the interface is a text input field with the placeholder 'Message ChatGPT...' and a send button (an upward arrow icon).

## 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)

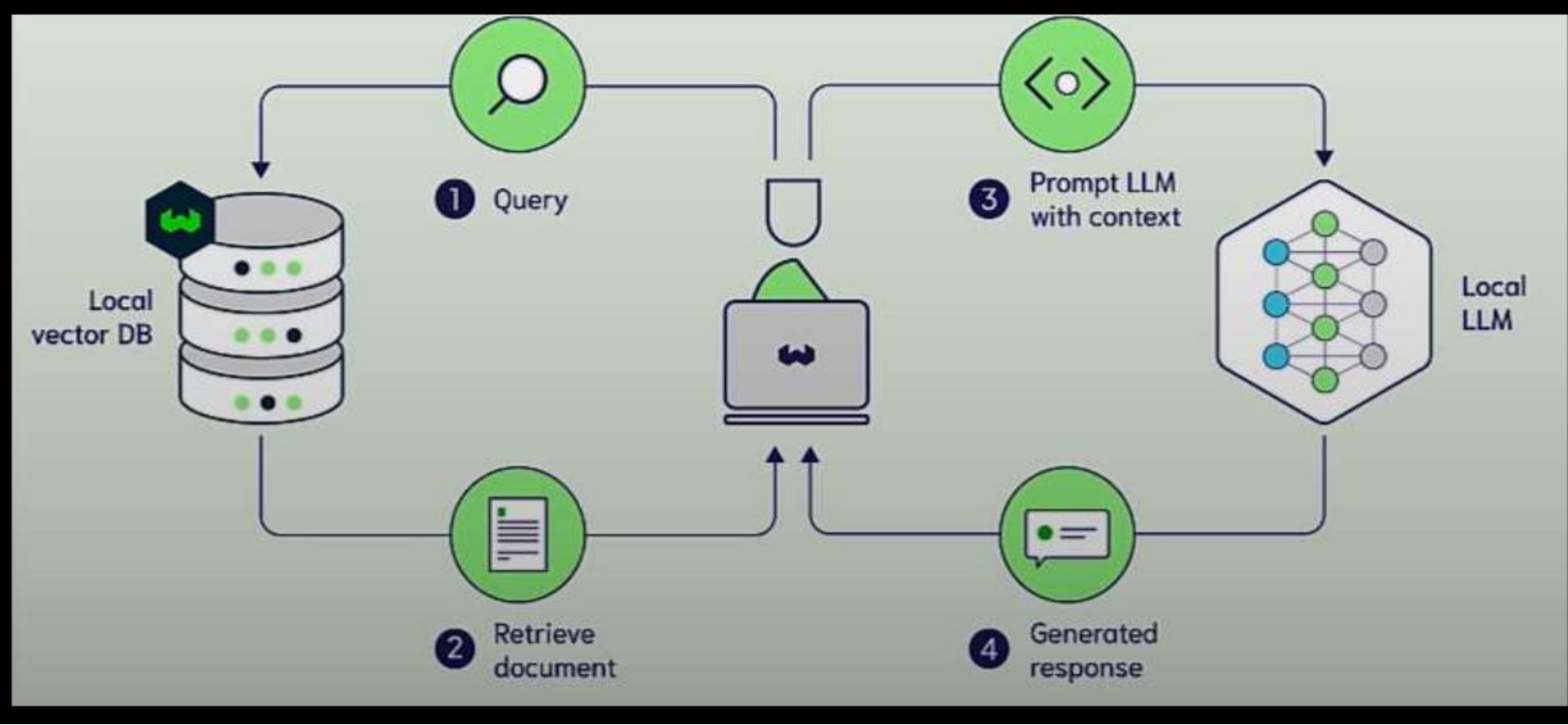
# I strip the text to remove the leading and trailing whitespace
print (output.strip())
```

Rachel is under 40 years old.

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.

# Building Block of LLM



# 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
```

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

Start with a LLM code and modularize that

**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”](https://github.com/amutharun/GenAIWorkshop/blob/main/RAG_A_Complete_Guide.ipynb)



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

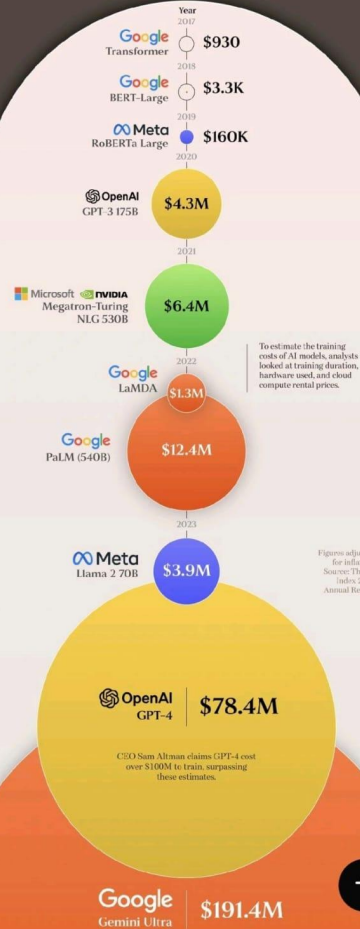
<https://technext24.com> › 2023/08/14 › chatgpt-costs-7... ⋮

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

## Challenges

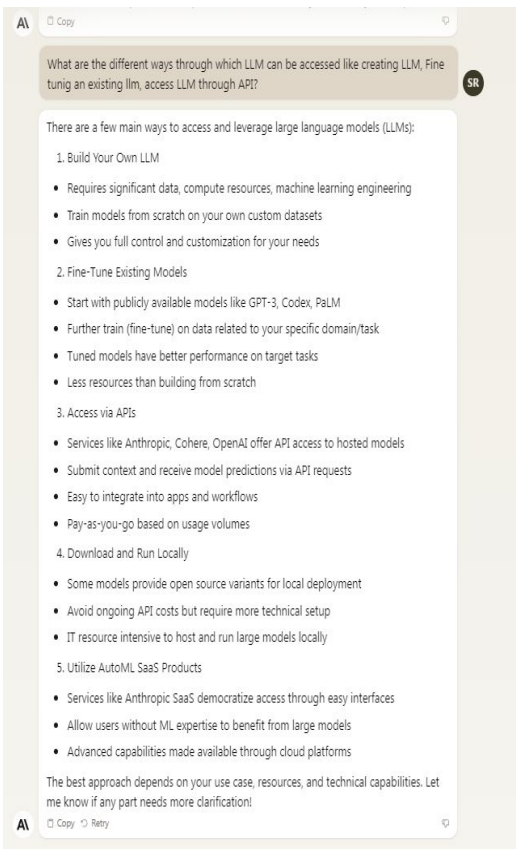
Budget      Infrastructure      Skillset      ROI      Success/ Guaranteed Result?!

# The Cost of Training AI



So, what are the options?

# Different ways of accessing LLM



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!



Why Organizations still find it hard to embrace and implement GenAI?

DATA PRIVACY



Elon Musk   @elonmusk · 8h

...

How  Intelligence works



 20K

 33K

 333K

 39M



HALLUCINATION

[“https://www.youtube.com/watch?v=cn5QSQNEr4c](https://www.youtube.com/watch?v=cn5QSQNEr4c)

[“https://www.youtube.com/watch?v=4WDIMhaxF\\_g”](https://www.youtube.com/watch?v=4WDIMhaxF_g)

COMPLIANCE

[“https://www.youtube.com/watch?v=22gobhdavRw””](https://www.youtube.com/watch?v=22gobhdavRw)

- The above video links are used only for Educational purpose



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.









# LEARNING RESOURCES

## O'Reilly - Live Courses

### LinkedIn :

Dipanjani - <https://www.linkedin.com/in/dipanjani/>

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 Alammari - <https://www.linkedin.com/in/jayammari/>

SRK - <https://www.linkedin.com/in/sudalairajkumar/>

**Pages/ Website** - Langchain  
Llama Index

### Meetups



