Introduction to RAG (Retrieval-Augmented Generation)

Mami Hayashida and Vikram Gazula

February 20, 2025

Center for Computational Sciences

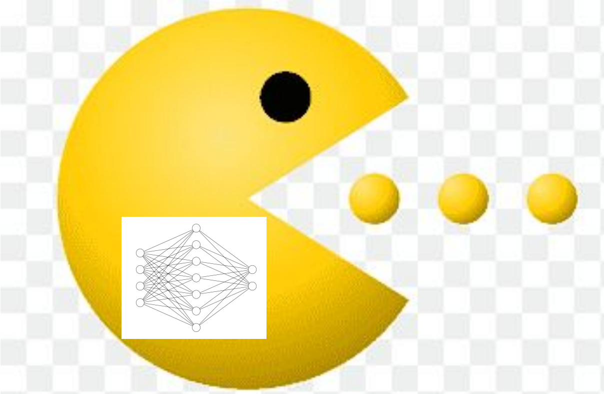
Outline

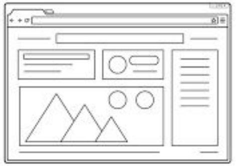
- What is LLM?
- What is RAG?
- Try running LLM + RAG pipeline locally (demo)
- Use cases
  - ACCESS-CI
  - FABRIC
- Q & A

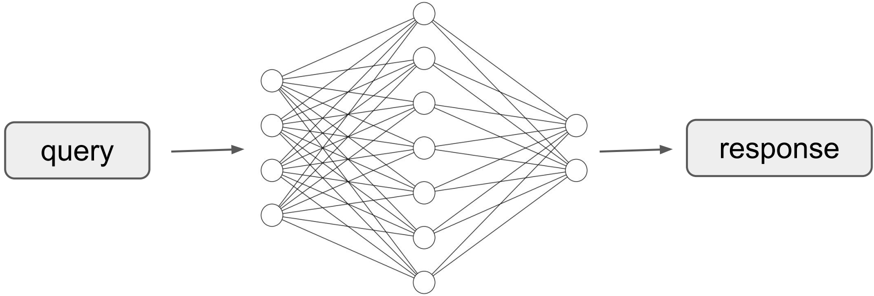
LLM

A large language model (LLM) is a type of machine learning model designed for natural language processing tasks such as language generation. LLMs are language models with many parameters, and are trained with self-supervised learning on a vast amount of text.

(https://en.wikipedia.org/wiki/Large_language_model)







Growing number of GenAI options

source:
https://www.pbs.org/newshour/science/what-is-deepseek-heres-a-quick-guide-to-the-chinese-ai-company

source: https://www.techradar.com/computing/artificial-intelligence/best-llms

GenAI shortcomings

1. **Lack of domain-specific/custom knowledge**

GenAI shortcomings

1. **Lack of domain-specific/custom knowledge**

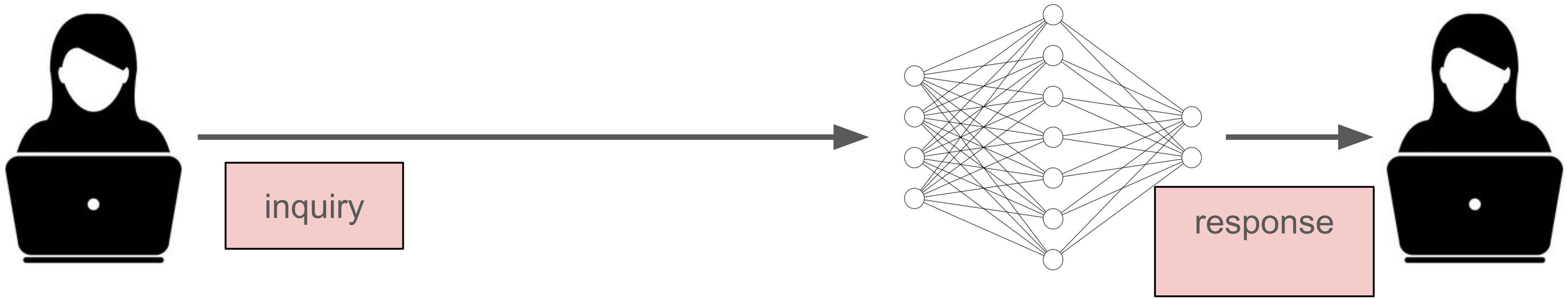
- Train a new model 😱
- Fine-tune an existing model 😕
- Use RAG 🙂

RAG

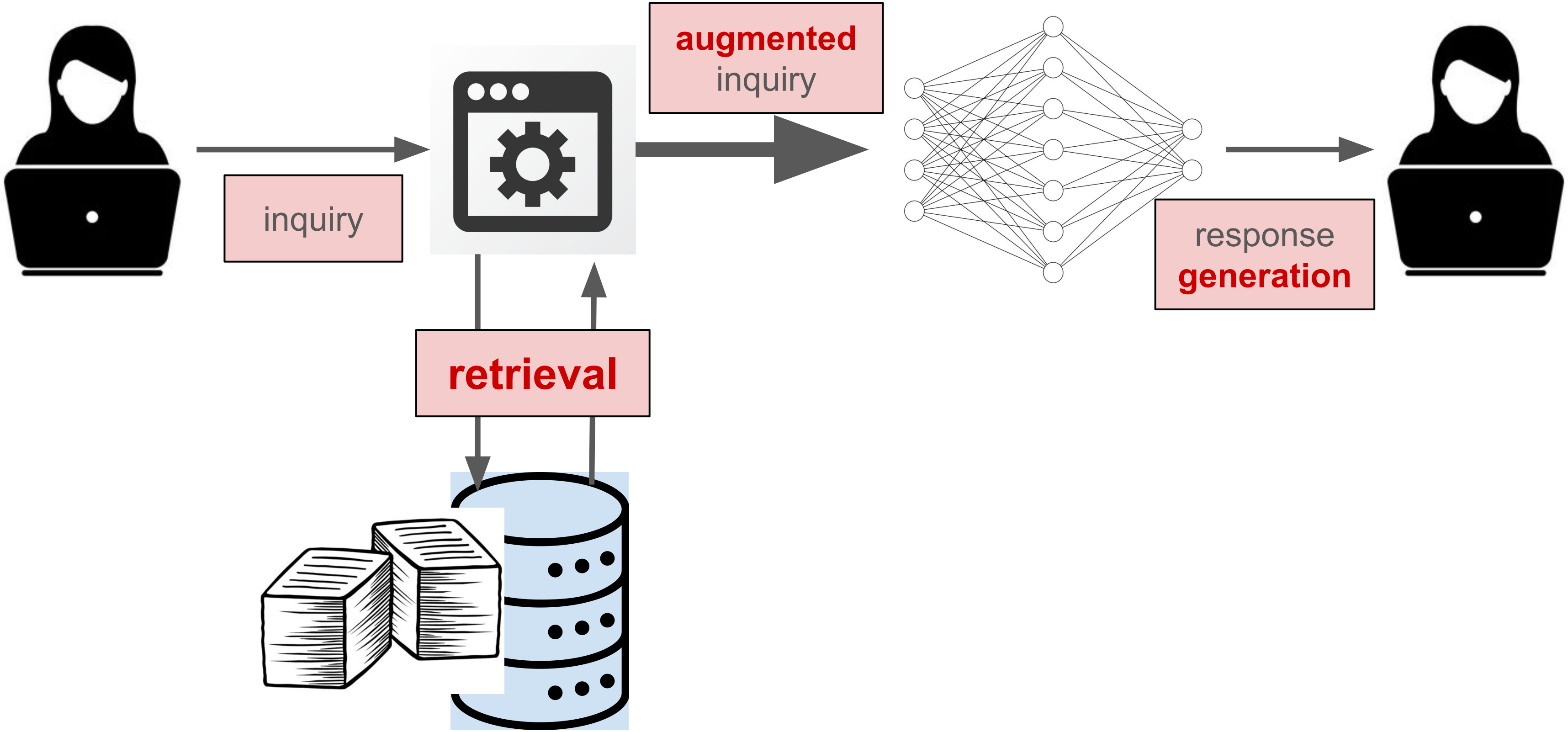
(Retrieval Augmented Generation)

Center for
Computational Sciences

Without RAG

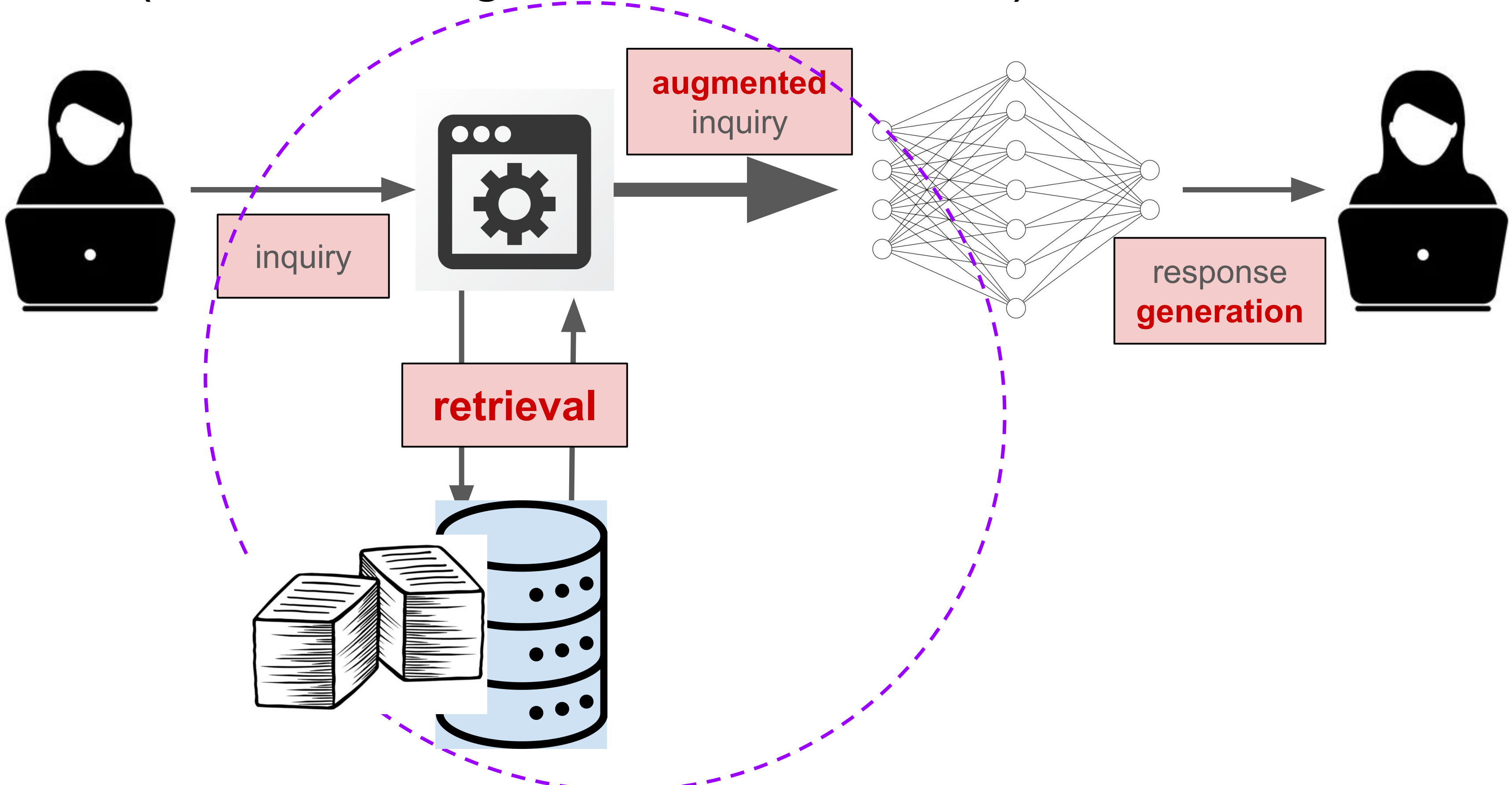


RAG (Retrieval Augmented Generation)

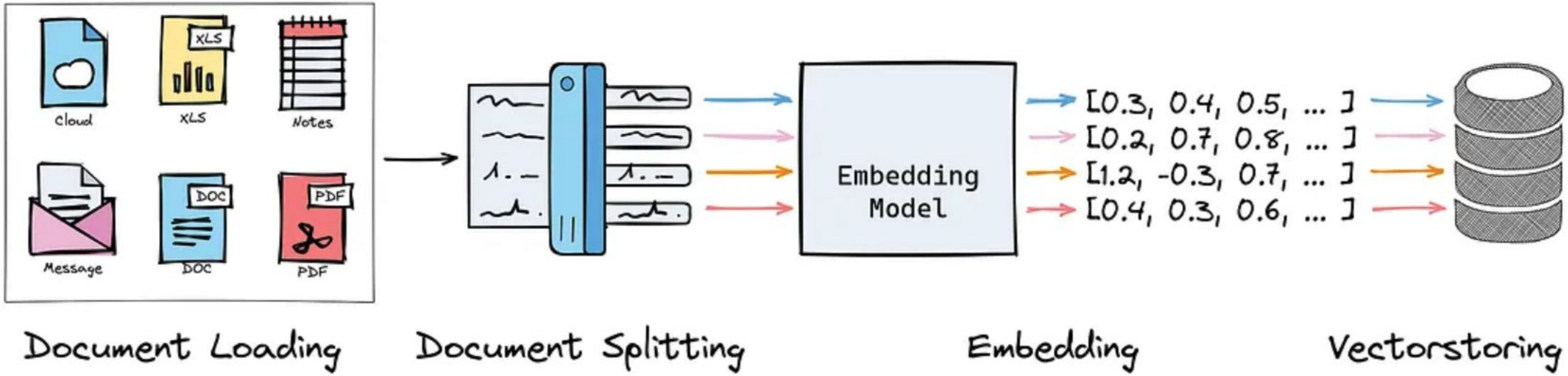


Center for
Computational Sciences

RAG (Retrieval Augmented Generation)



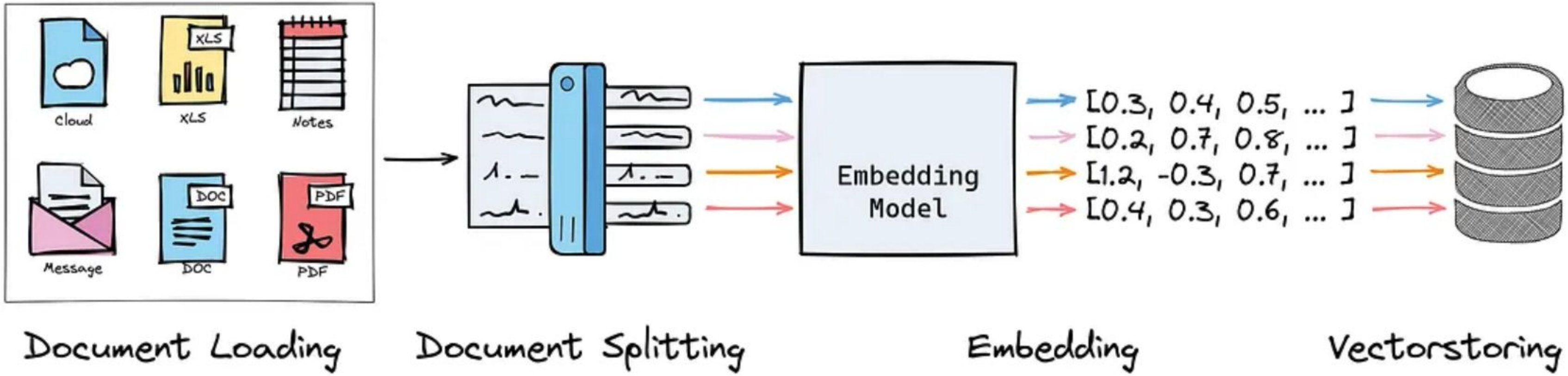
Storing your data on Vector Database: 1. Document Loading



https://pub.towardsai.net/langchain-101-part-3a-talking-to-documents-load-split-and-simple-rag-with-lcel-26b005ccb30a

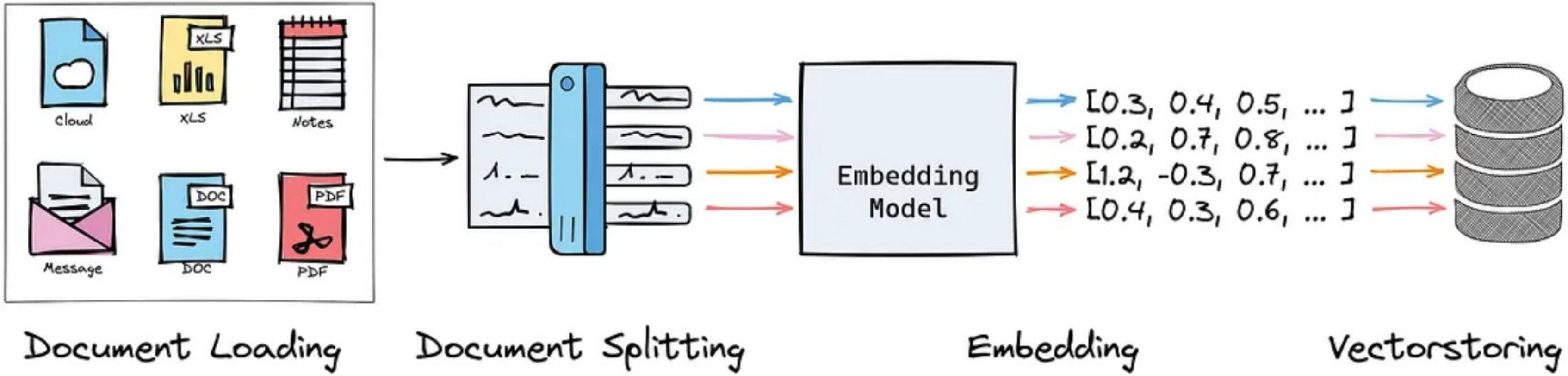
Center for Computational Sciences

Storing your data on Vector Database: 2. Document Splitting



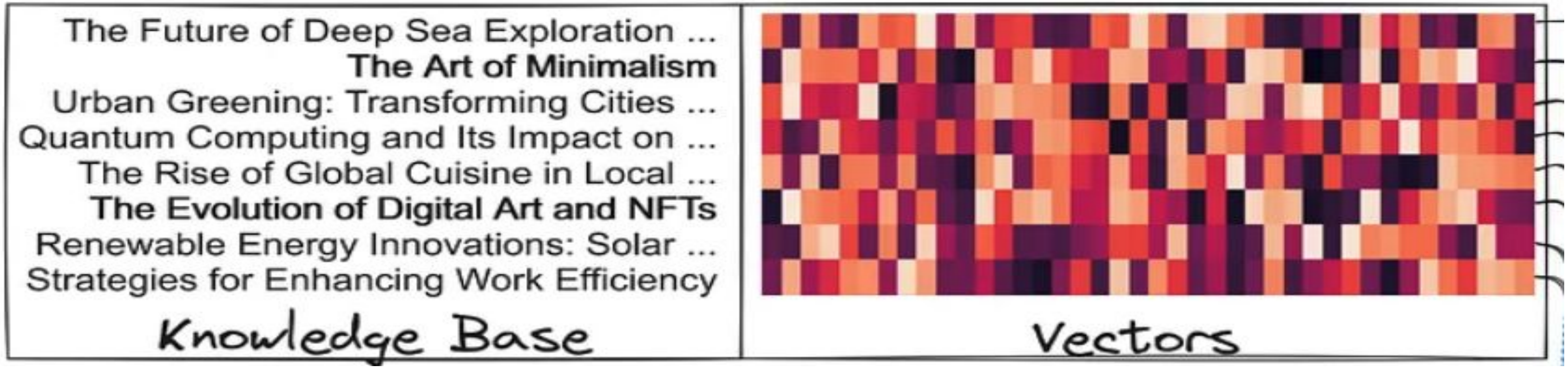
https://pub.towardsai.net/langchain-101-part-3a-talking-to-documents-load-split-and-simple-rag-with-lcel-26b005ccb30a

Storing your data on Vector Database: 3. Embedding



https://pub.towardsai.net/langchain-101-part-3a-talking-to-documents-load-split-and-simple-rag-with-lcel-26b005ccb30a

Storing your data on Vector Database: 3. Embedding

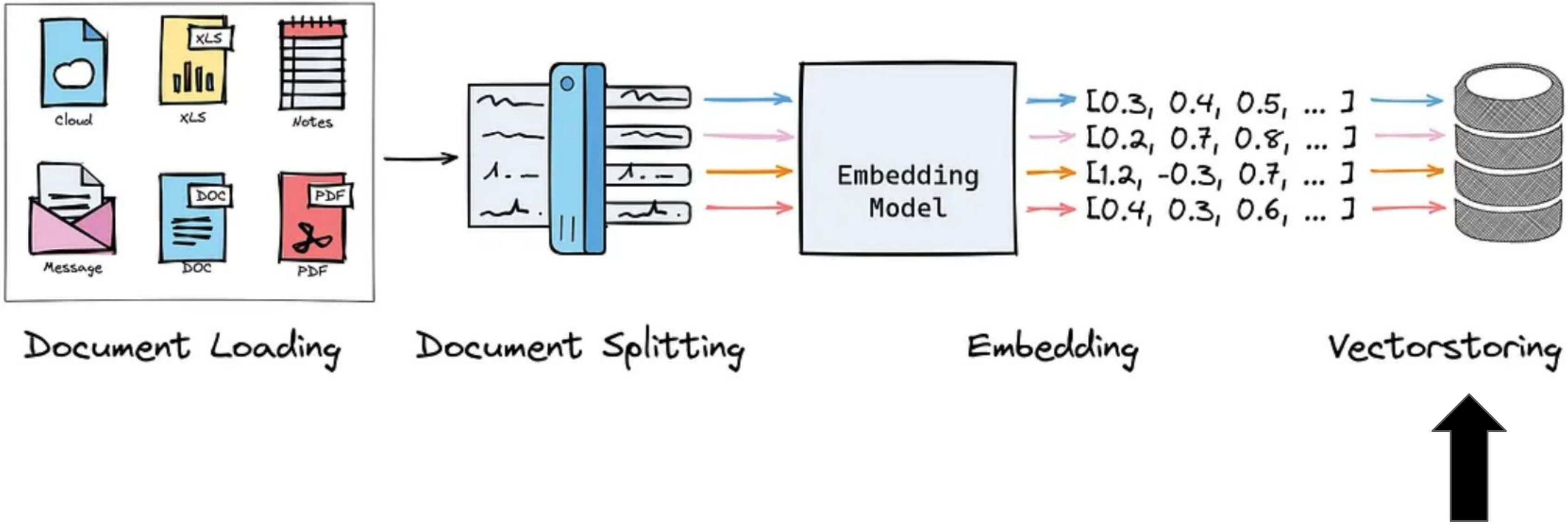


Example: embed query

- Embedding translates each of the split text “chunks” into a high-dimensional numerical representation.
- Goal is to preserve the semantic and contextual similarities and differences.

https://pub.towardsai.net/langchain-101-part-3b-talking-to-documents-embeddings-and-vectorstores-c37d460f1519

Storing your data on Vector Database: 4. Vectorstoring



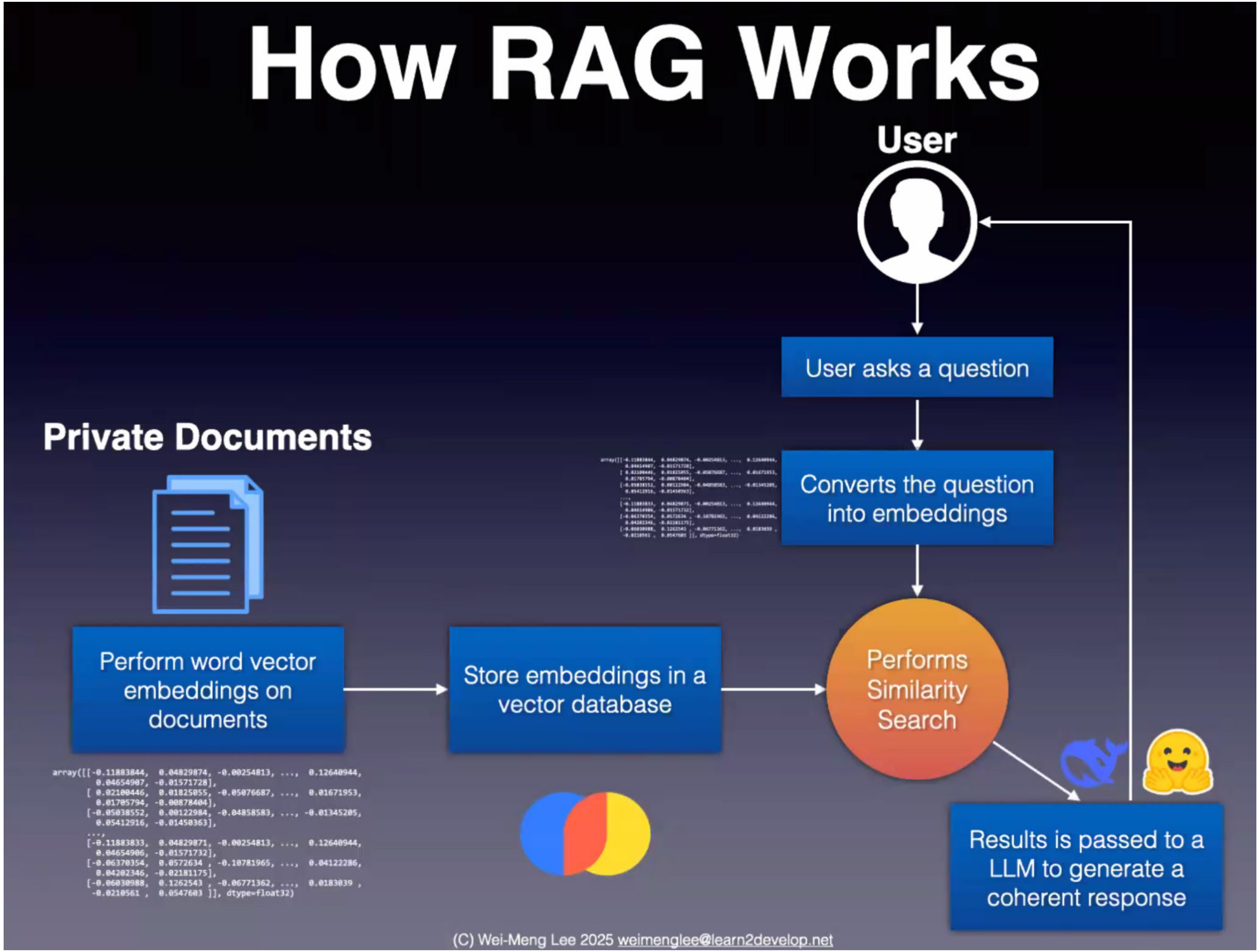
https://pub.towardsai.net/langchain-101-part-3a-talking-to-documents-load-split-and-simple-rag-with-lcel-26b005ccb30a

Storing your data on Vector Database: 4. Vectorstoring

In short, Vectorstore store vectors where chunks are organized by their similarities vs. differences.

(Embedding and vectorstoring are still relatively slow.)

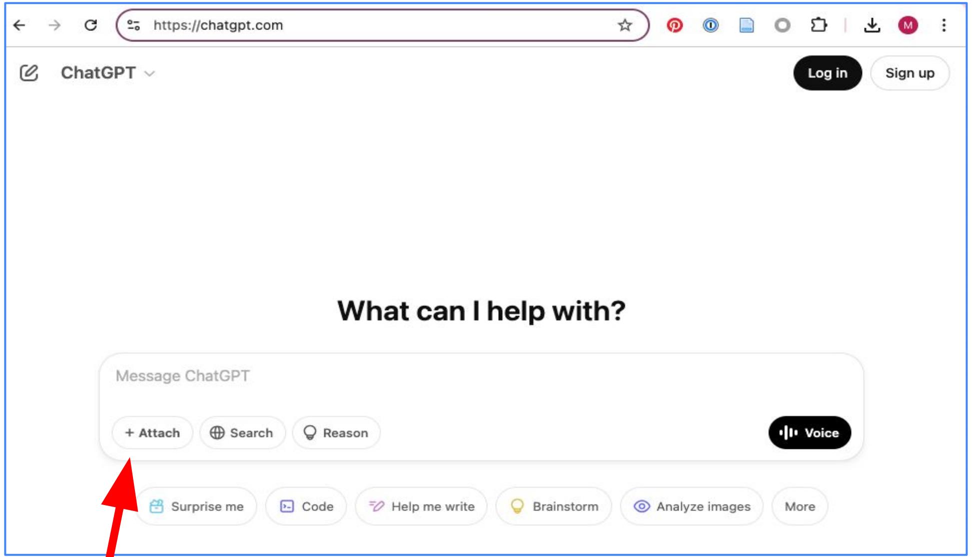
https://pub.towardsai.net/langchain-101-part-3b-talking-to-documents-embeddings-and-vectorstores-c37d460f1519

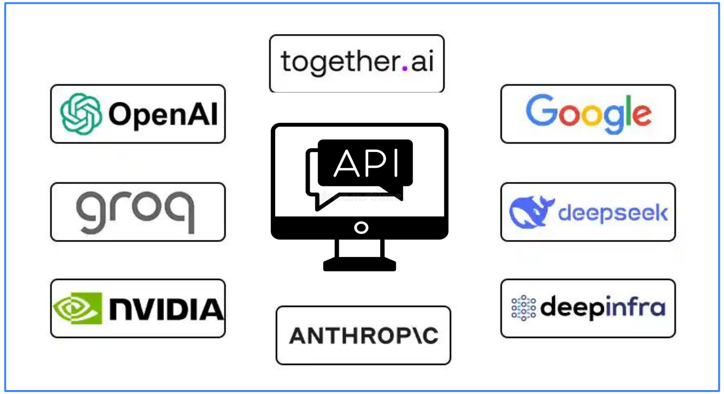


Center for Computational Sciences

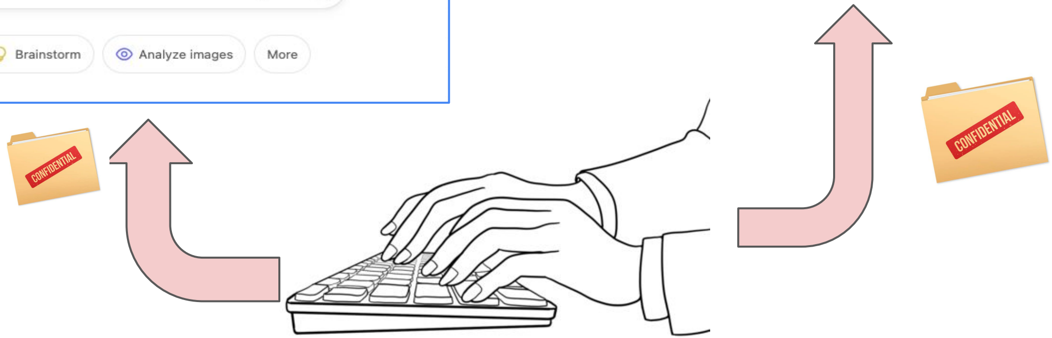
credit: Wei-Meng Lee

Uploading your data to …. 😱





source: https://www.analyticsvidhya.com/blog/2024/10/free-and-paid-apis/



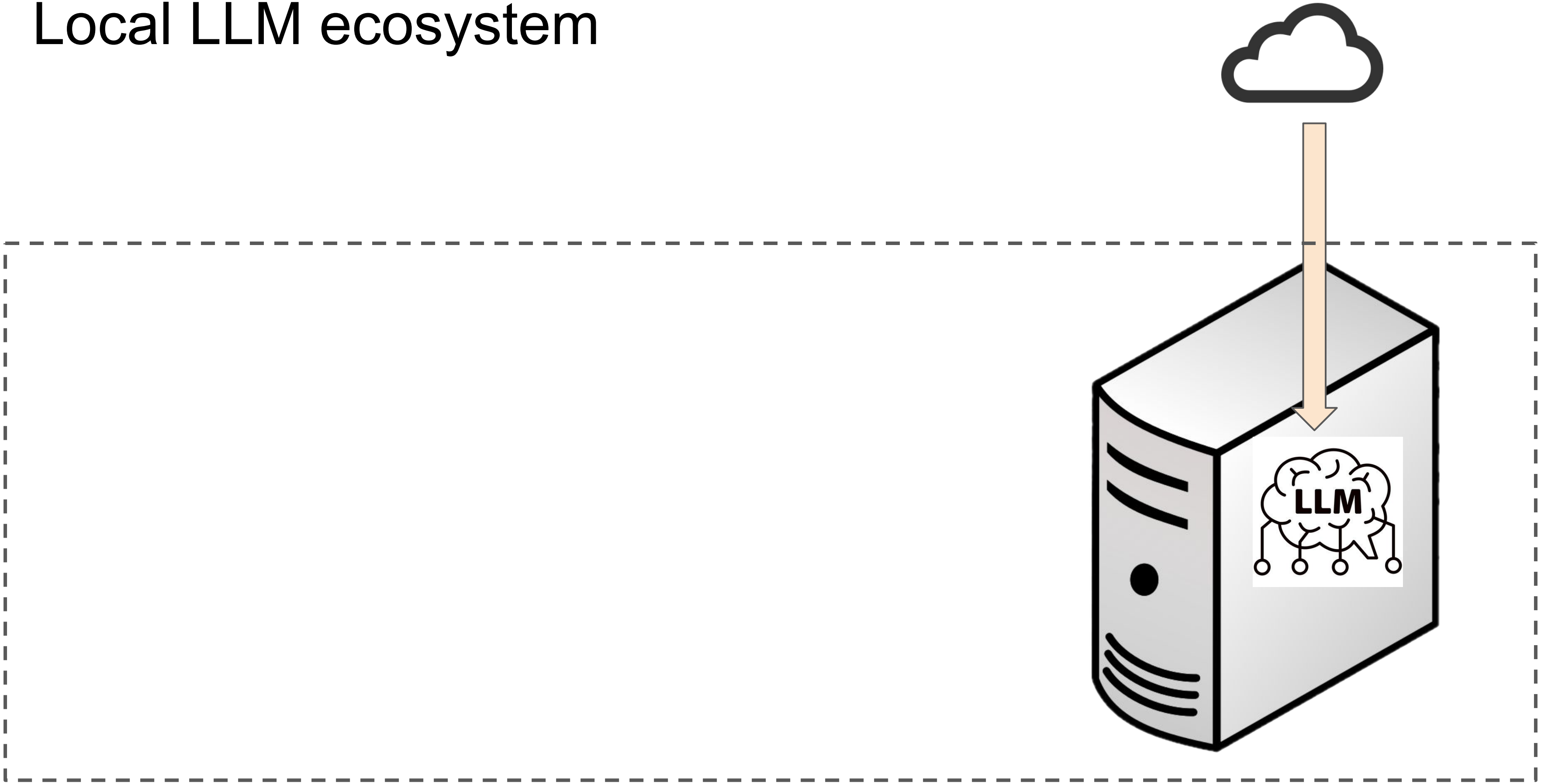
GenAI shortcomings

1. **Lack of domain/specific knowledge**
2. **Privacy/confidentiality concerns**

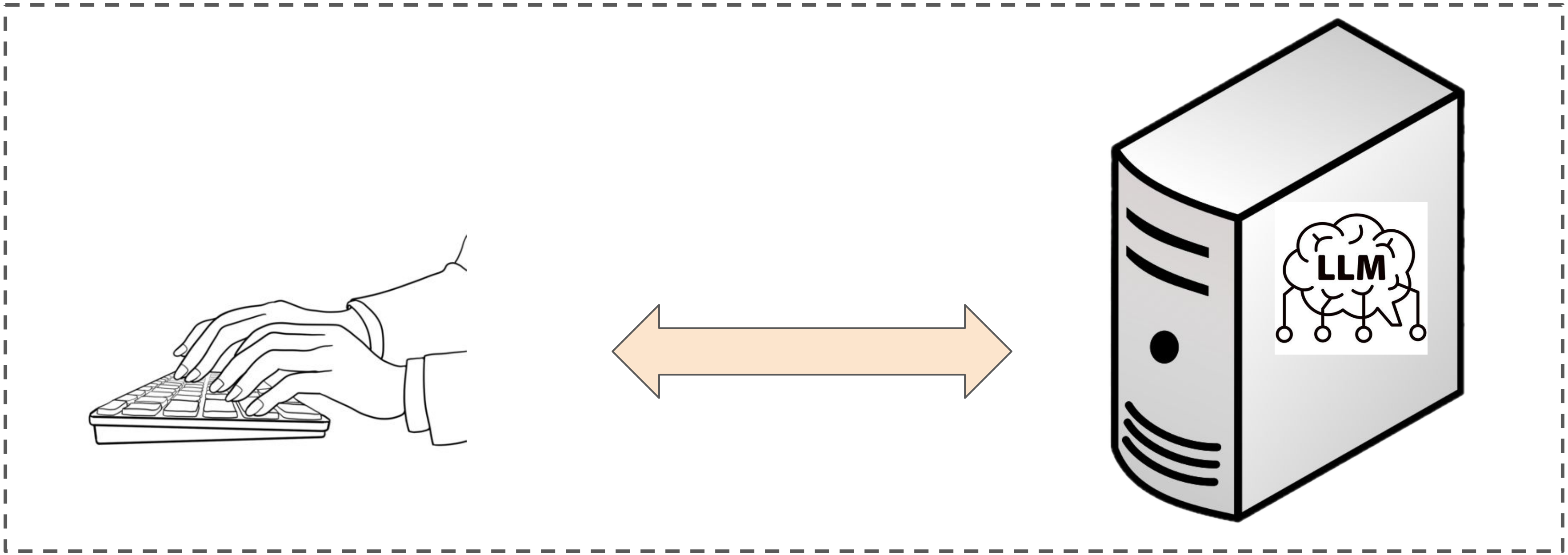
Running LLM + RAG Locally

Center for
Computational Sciences

Local LLM ecosystem



Local LLM ecosystem



Running your LLM Locally

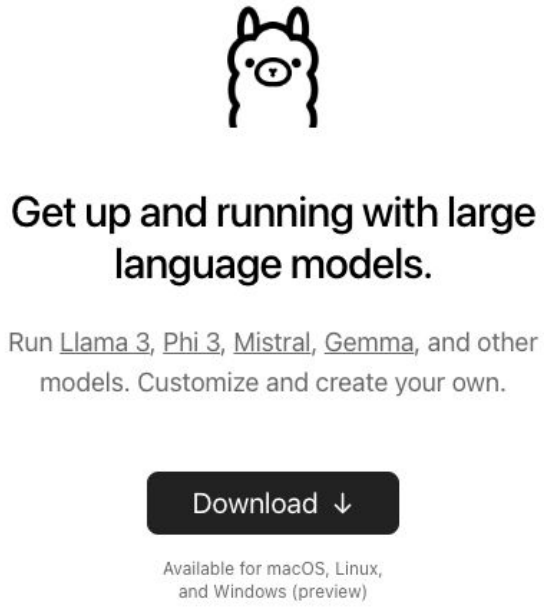
- Ollama
- LM Studio
- llama.cpp

All three support a REST API, allowing you to integrate LLMs into your apps.

credit: Wei-Meng Lee

Ollama

https://ollama.com/



Ollama allows you to run open-source large language models, such as Llama 3, deepseek-r1 locally.

Amazingly simple!

Runs on CPU/GPU

⭐⭐⭐⭐⭐

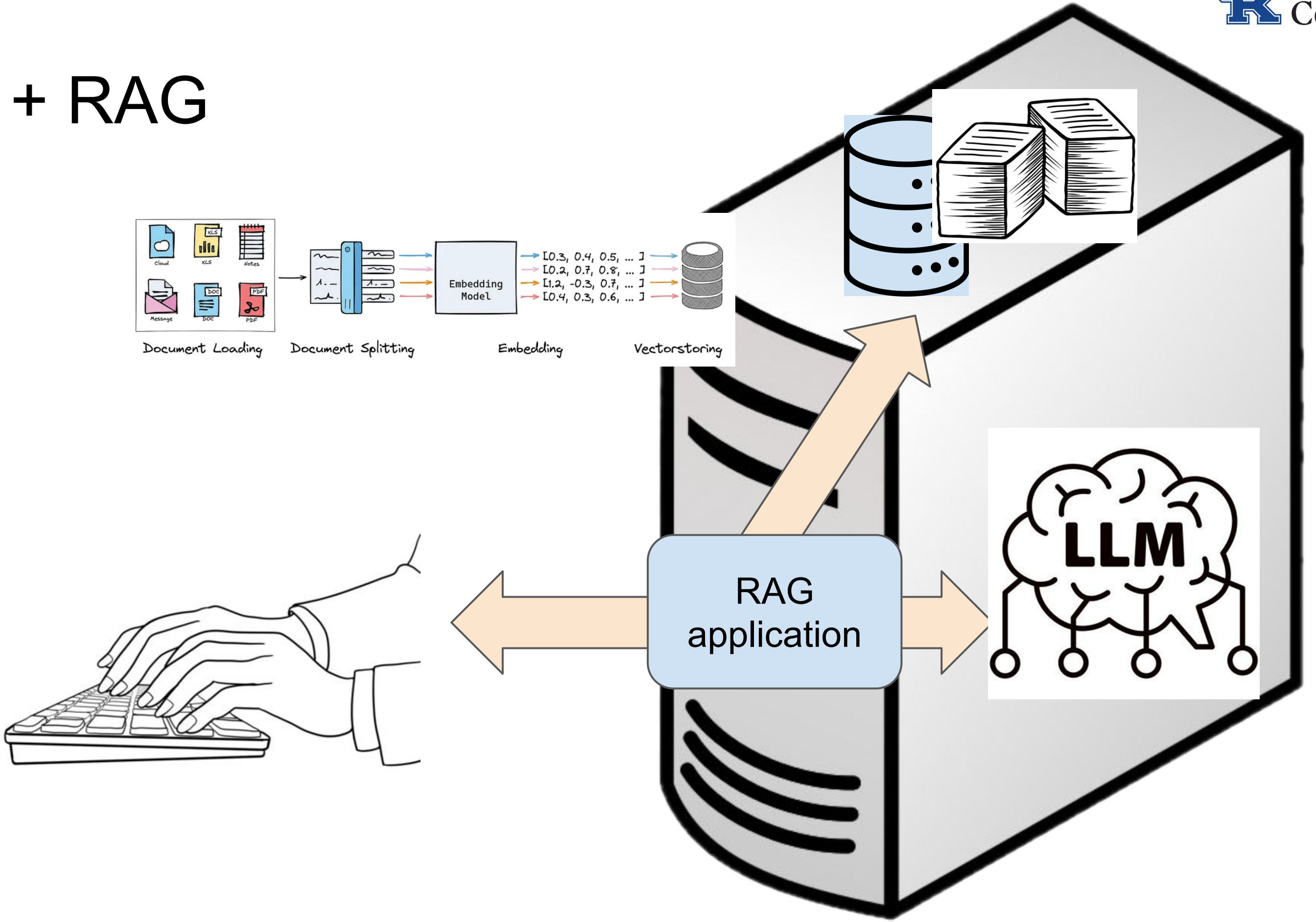
```
systemctl start ollama.service
ollama run <LLM model>
```

Running Ollama

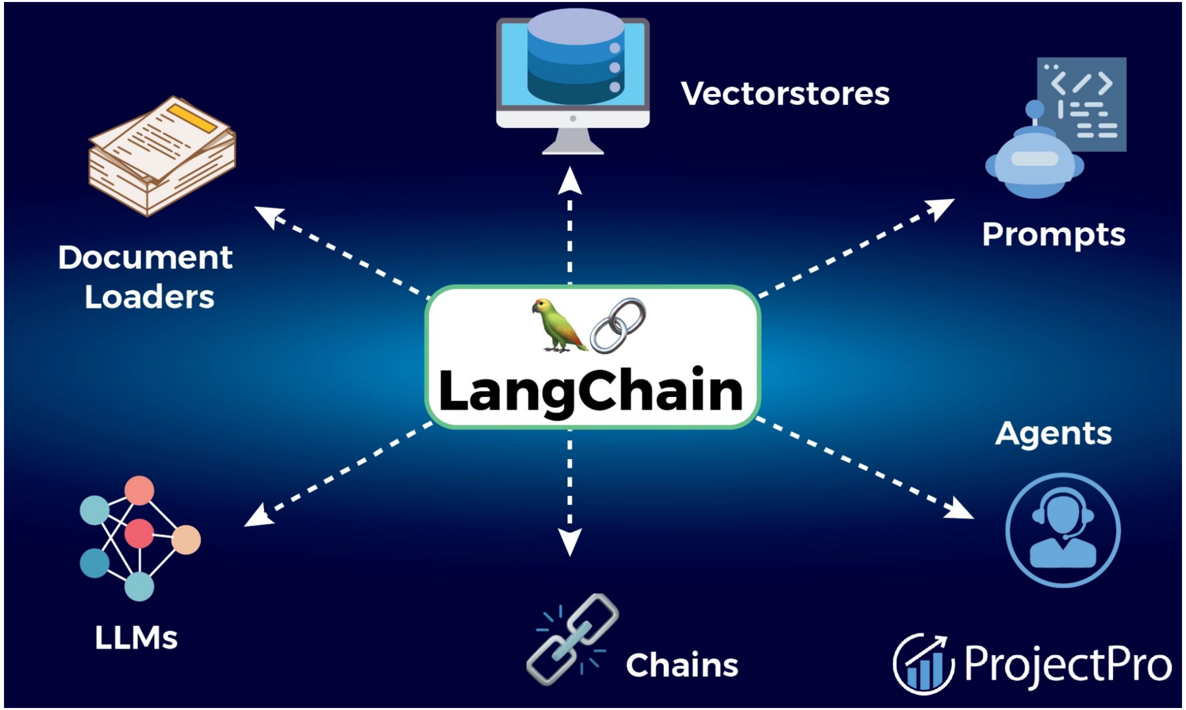
```
[mhaya2@gh3-internal ~]$ ollama list
NAME                    ID              SIZE      MODIFIED
mistral-large:latest    bbcf36dc47ad    73 GB     2 weeks ago
codestral:latest        0898a8b286d5    12 GB     2 months ago
codegemma:7b            0c96700aaada    5.0 GB    2 months ago
llama3.1:8b             91ab477bec9d    4.7 GB    5 months ago
codellama:13b           9f438cb9cd58    7.4 GB    6 months ago
llama3.1:70b            073e22d7e65d    39 GB     6 months ago
codellama:latest        8fdf8f752f6e    3.8 GB    6 months ago
codellama:70b           e59b580dfce7    38 GB     6 months ago
[mhaya2@gh3-internal ~]$
[mhaya2@gh3-internal ~]$
[mhaya2@gh3-internal ~]$ ollama run llama3.1:8b
>>> Send a message (/? for help)
```

Center for
Computational Sciences

LLM + RAG


Embedding Model
[0.3, 0.4, 0.5, ...]
[0.2, 0.7, 0.8, ...]
[1.2, -0.3, 0.7, ...]
[0.4, 0.3, 0.6, ...]
Document Loading
Document Splitting
Embedding
Vectorstoring
RAG application
LLM

Langchain



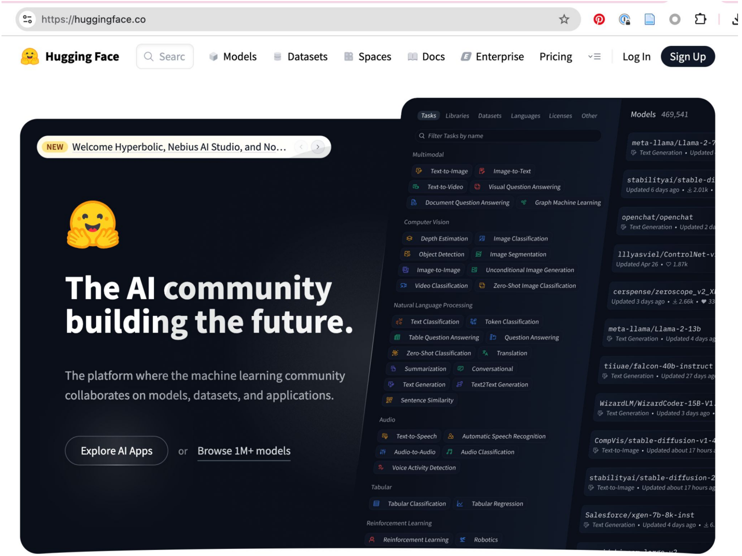
LangChain is a framework designed to simplify the creation of applications using large language models

https://www.projectpro.io/article/langchain/894

Hugging Face

Hugging Face

- “GitHub for AI”
- Platform for sharing datasets, models etc.
  - Example: https://huggingface.co/models
- Keeping up with the latest/hottest
  - https://huggingface.co/spaces/mteb/leaderboard



https://huggingface.co/

Center for Computational Sciences

(Very simple) local RAG demo/tutorial

- Interact with Ollama using python-ollama
- Interact with Ollama using Langchain
- Create a Chroma vector database from a PDF file
- Run a query on RAG pipeline using Langchain, Ollama, and Chroma DB

LangChain

Chroma

https://lafayette.fcps.net/

Demo notebooks

https://github.com/UKY-CCS-ITS-RCI/Workshops/tree/main/2025-02-20_ai_seminar

RAG Application Examples

Center for Computational Sciences

1. ACCESS-CI

https://access-ci.org/

Introducing ACCESS-CI

- What is ACCESS-CI?
  - Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (https://access-ci.org/)
  - Connects researchers to computational resources in the U.S.

- Key Features:
  - Diverse computing resources, Data and storage services, Scientific applications, workflow management, science gateways

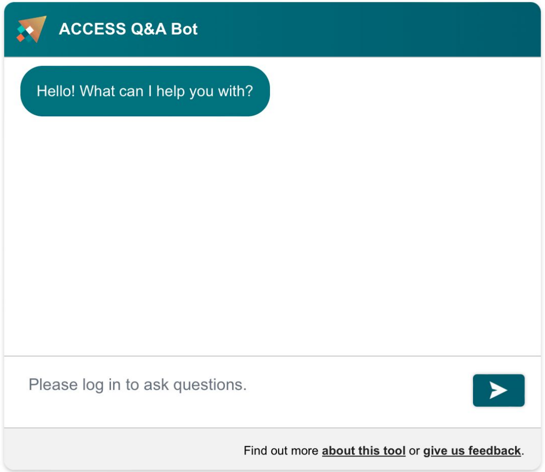
  - Comprehensive user support, resources for your class, training, and workshops, as well as connecting with communities that share your interests and learn from one another(affinity groups).

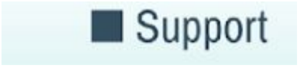
Support

Q&A Tool

Providing information about the ACCESS ecosystem

- AI-driven information retrieval system
- Retrieval-Augmented Generation (RAG) and LLMs
- Added to Support portal in Sept 2024 (https://support.access-ci.org)
- Not a chatbot—each query is independent
- Reviewing questions, answers, and feedback to assess performance
- Developing a process to manage assessment, data sources, and cadence for data updates



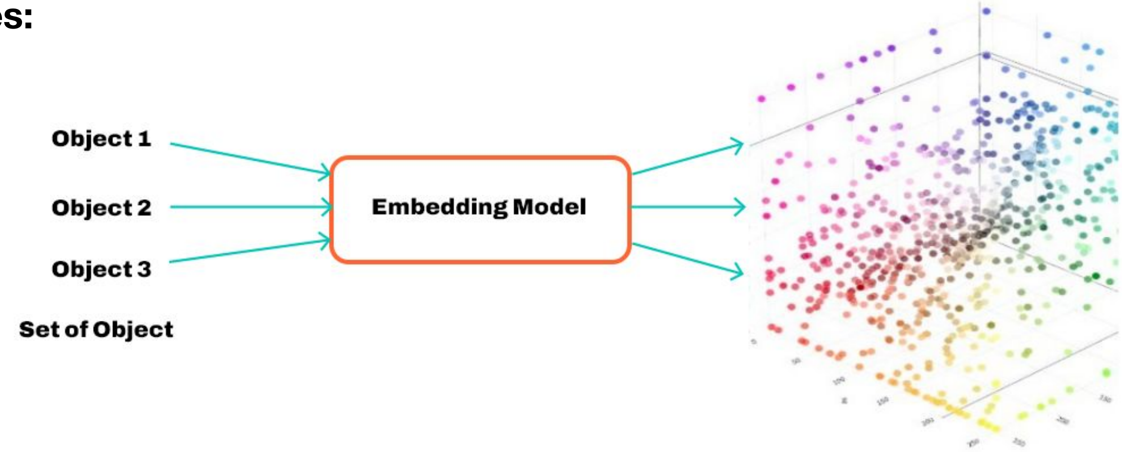


Data Ingestion

141 data sources indexed from ACCESS and RP sources

Database of embeddings sources:

- Data source url
- Maintaining team/track
- Description
- Keywords
- Embedding date



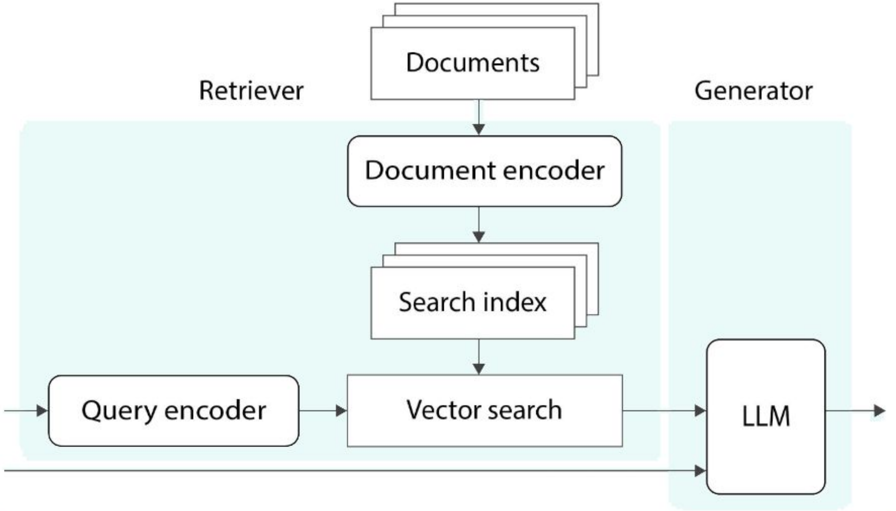
Support

RAG LLM

Technologies used

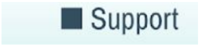
- **OpenAI for LLM inferencing**
  - GPT-4o-mini in production
- **Document embeddings use OpenAI models**
  - text-embedding-3-small in production
  - testing out text-embedding-3-large
- **Vector stores database**
  - ChromaDB in production
  - Testing Qdrant and FAISS for expansion/scaling purposes
- llamaindex framework to integrate components

These technologies are used with extensive data curation and tailored metadata to enhance and improve the tool's responses.



AI Agentic Capabilities:

- Enhances structured retrieval processes.
- Filters and categorizes responses based on content metadata.
- Ensures dynamic adaptability to new data sources without compromising accuracy.



Key Considerations for Effective RAG Implementation (1):

Vector Databases, Chunking and Metadata in RAG

Vector Databases store document embeddings for efficient similarity search. Selecting the right vector database and indexing method affects retrieval accuracy and speed.

Chunking breaks long documents into smaller sections (256-512 tokens per chunk) before embedding.

Each chunk is embedded into a high-dimensional vector (e.g., 1536D for OpenAI embeddings).

Why Chunking? Smaller chunks improve search relevance by focusing on specific context.

Metadata storage: Store **document titles, keywords, and section headers** for better filtering.

Center for Computational Sciences

Key Considerations for Effective RAG Implementation (2):

Vector Quantization vs. LLM Quantization

Vector Quantization (Used in Vector Databases & RAG)

Purpose: Reduce storage and speed up similarity search for embeddings.

Method: Compresses document embeddings (e.g., FP32 → INT8).

Impact: Enables fast retrieval in **ChromaDB, FAISS, Pinecone**.

LLM Quantization (Used for Model Inference & Deployment)

Purpose:Reduce LLM size to run efficiently on CPUs/GPUs.

Method:Compresses model weights (e.g., FP32 → INT4).

Impact: Enables Llama 3, GPT-4, Mistral to run on edge devices.

Key Difference:

Vector quantization helps RAG search faster.

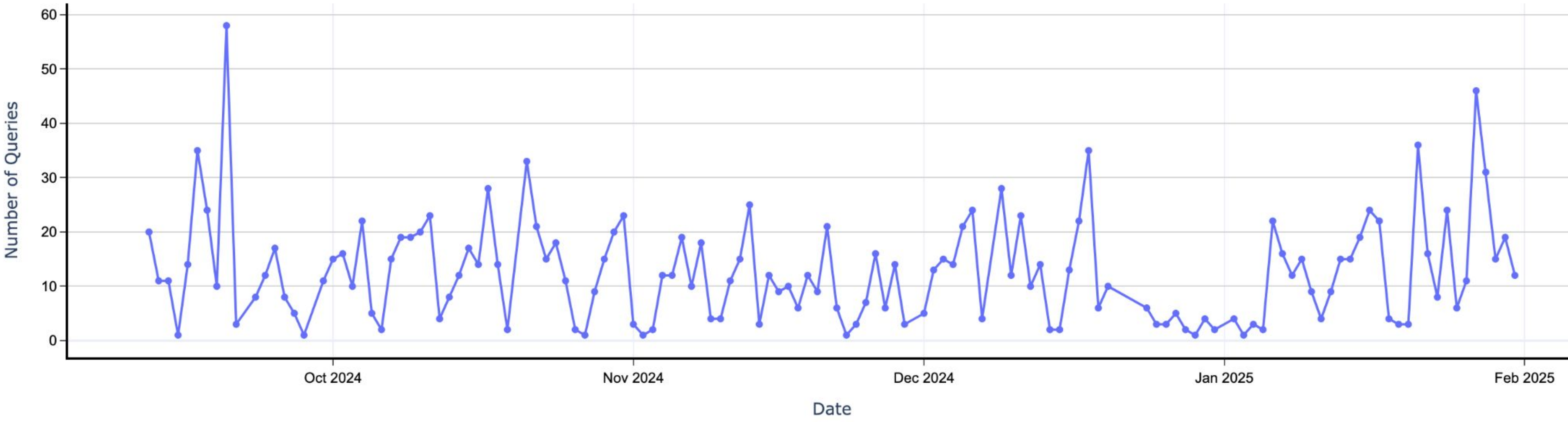
LLM quantization speeds up model inference.

Quantization is a technique that reduces the precision of numerical data (e.g., embeddings or model weights) to optimize storage and speed while maintaining accuracy.

Support

Q&A Tool

1822 queries since September 2024



Support

Center for Computational Sciences

2. FABRIC Testbed

https://portal.fabric-testbed.net/

FABRIC project

Setting up and running experiments require writing a python code (notebooks encouraged)

Setup the Experiment

Import the FABlib Library

[ ]:
```
from fabrictestbed_extensions.fablib.fablib import FablibManager as fablib_manager

fablib = fablib_manager()

fablib.show_config();
```

Create the Experiment Slices

Create the slice and set the specific node attributes. Note that the cores, ram, and disk are only *hints*. The actual values will be the closest instance type that is larger than the chosen values.

Amounts of cores, ram, and disk will be rounded up to the closest instance type. These amounts should be considered minimums rather than specific requirements. You may find this article useful in explaining how VM size allocation works in FABRIC.

Note that your project must have `VM.NoLimit` permission set in order to allocate VMs larger than 2 cores, 10G of RAM and 10G of disk. Otherwise you will receive an error of the type `PDP Authorization check failed – Policy Violation: Your project is lacking VM.NoLimitCPU or VM.NoLimit tag to provision VM with more than 2 cores.`

[ ]:
```
slice_name = 'MySlice1'

#Create Slice
slice = fablib.new_slice(slice_name)

# Add node
node = slice.add_node(name='Node1',
                      site='MAX',
                      cores=4,
                      ram=16,
                      disk=100,
                      image='default_ubuntu_20')

#Submit Slice Request
slice.submit()
```

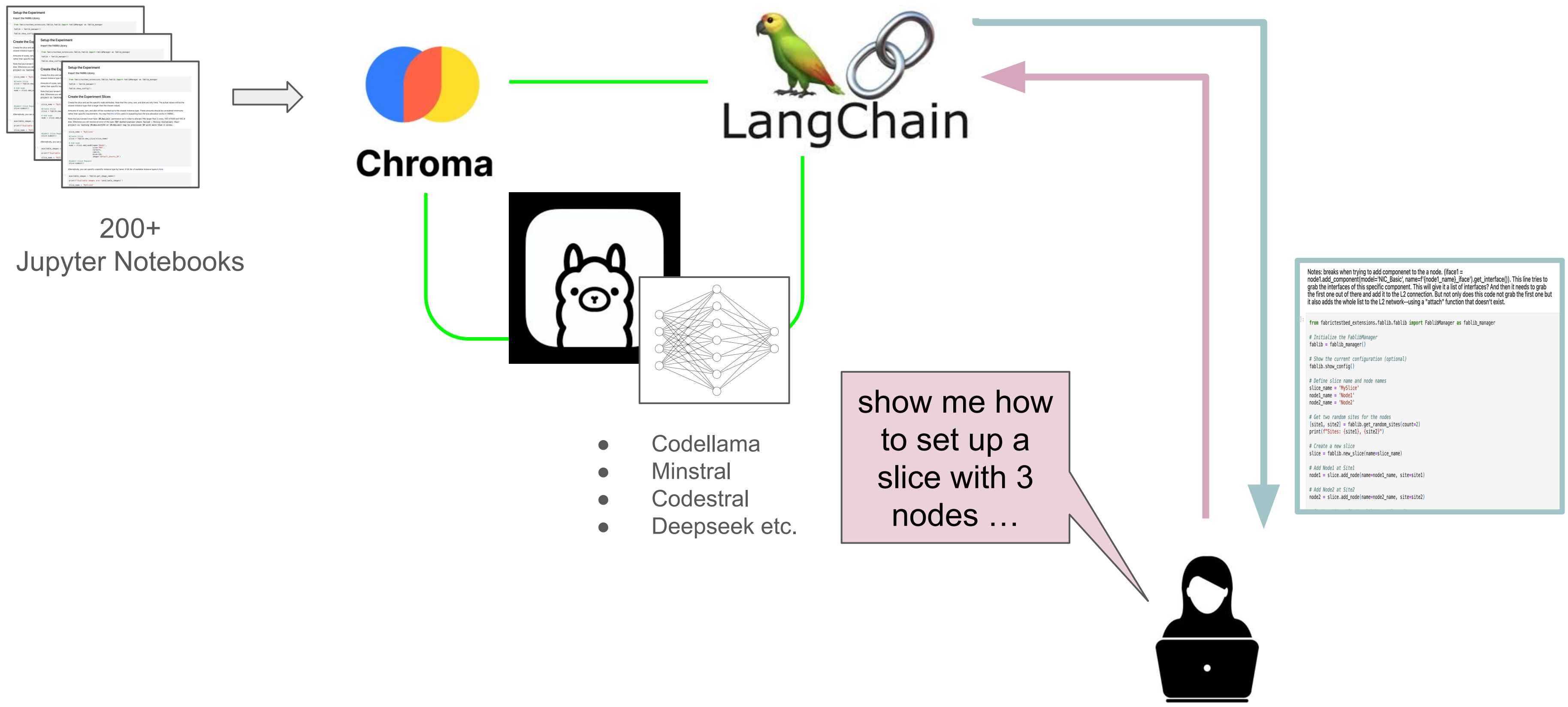
Alternatively, you can specify a specific instance type by name. A full list of available instance types is here.

[ ]:
```
available_images = fablib.get_image_names()

print(f'Available images are: {available_images}')

slice_name = 'MySlice2'
```

FABRIC code auto-generation using LLM + RAG



Questions?

Mami Hayashida (mami.hayashida@uky.edu)
Vikram Gazula (gazula@uky.edu)