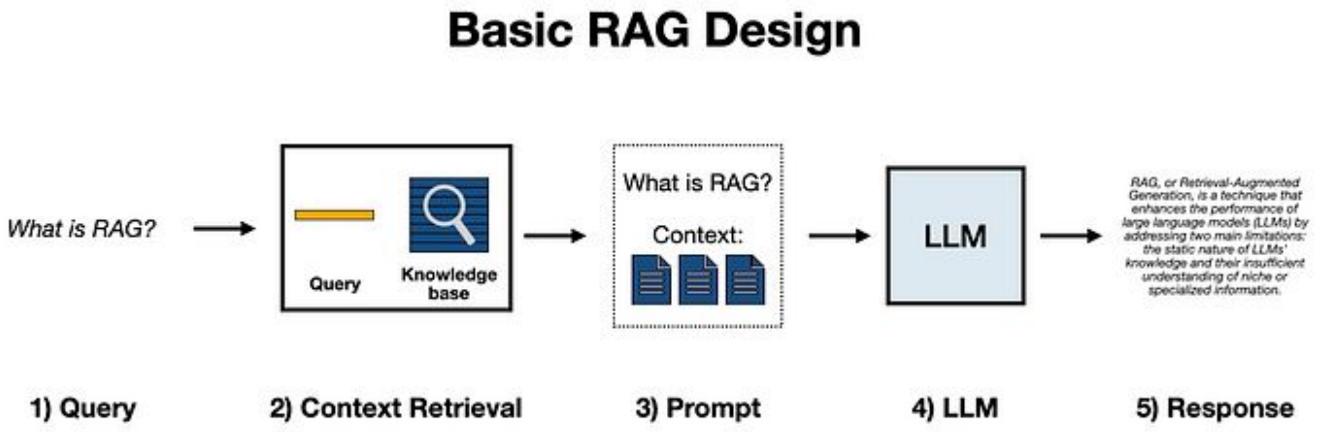
Multimodal RAG

December 19, 2024

Himanshu Singh Rao
Math 608
012629202

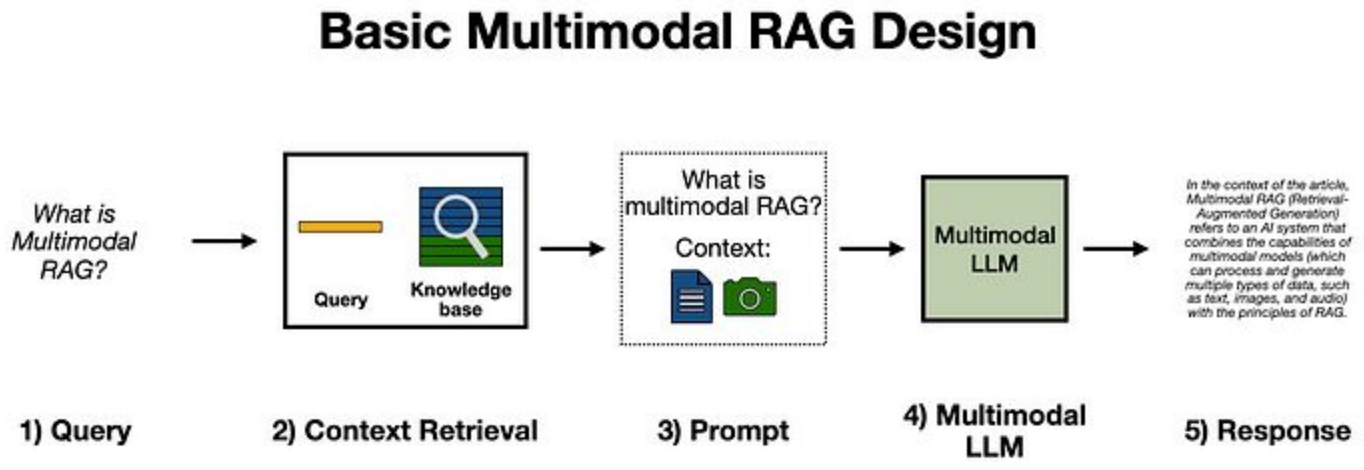
What is Retrieval Augmentation Generation(RAG)?

- LLMs dont have the context of our data, if we give LLMs context from the knowledge base, they become really powerful
- Add your PDFs, add your database, add your knowledge base.
- API call to LLMs = query + my_context + prompt
- Each time you make a call just pass your context in it



What is Multimodal-RAG?

- What if I don't want to make my llm only text based?
- What if I want my LLM to understand, photo, video, audio and text?
- I want to add images too. How can I do it??
- Multimodal RAG understands all modalities (Image, text, audio, video)



Why it is Important?

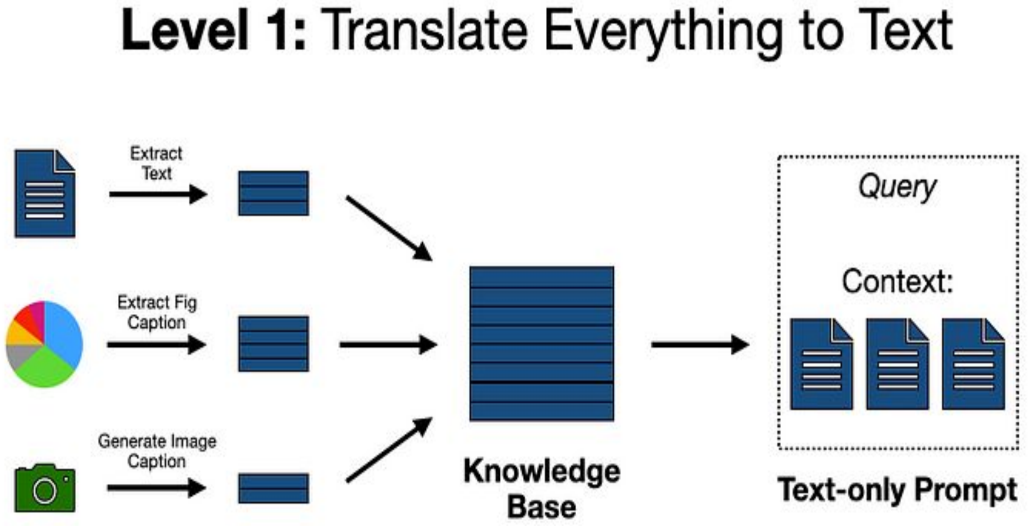
- Chat with Videos
- Chat with Photos
- Doing RAG with both Structured/Unstructured data
- Get full context of pdfs
- All modalities considered
- Make really cool AI Applications

3 Levels of MRAG

Level 1	Level 2	Level 3
Translate modalities to text.	Text-only retrieval + MLLM	Multimodal retrieval + MLLM

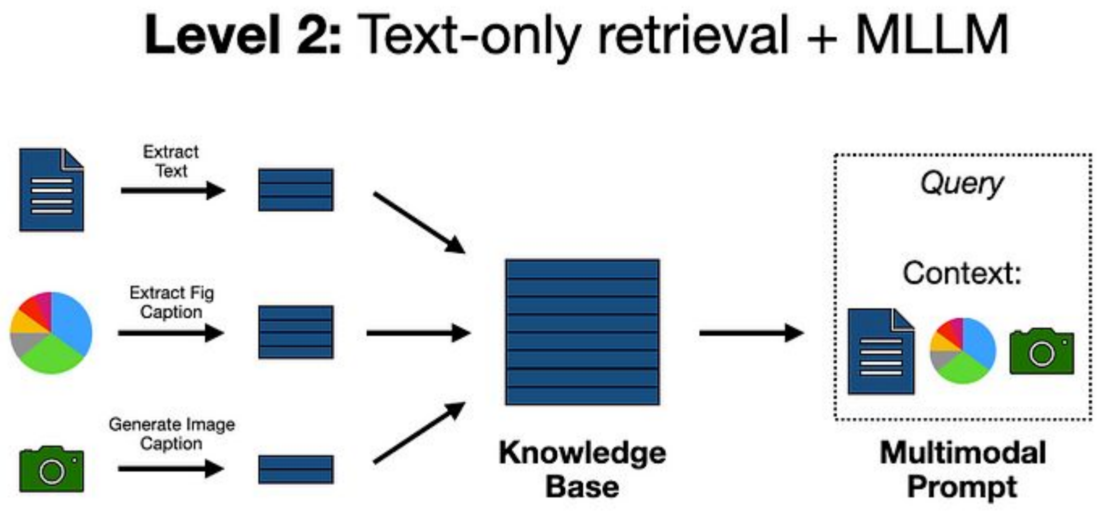
Level 1 - Translate modalities to text

- A simple way to make a RAG system multimodal is by translating new modalities to text before storing them in the knowledge base
- This includes tasks like transcribing meeting recordings, generating image captions with MLLMs, or converting tables into readable formats like .csv or .json.).



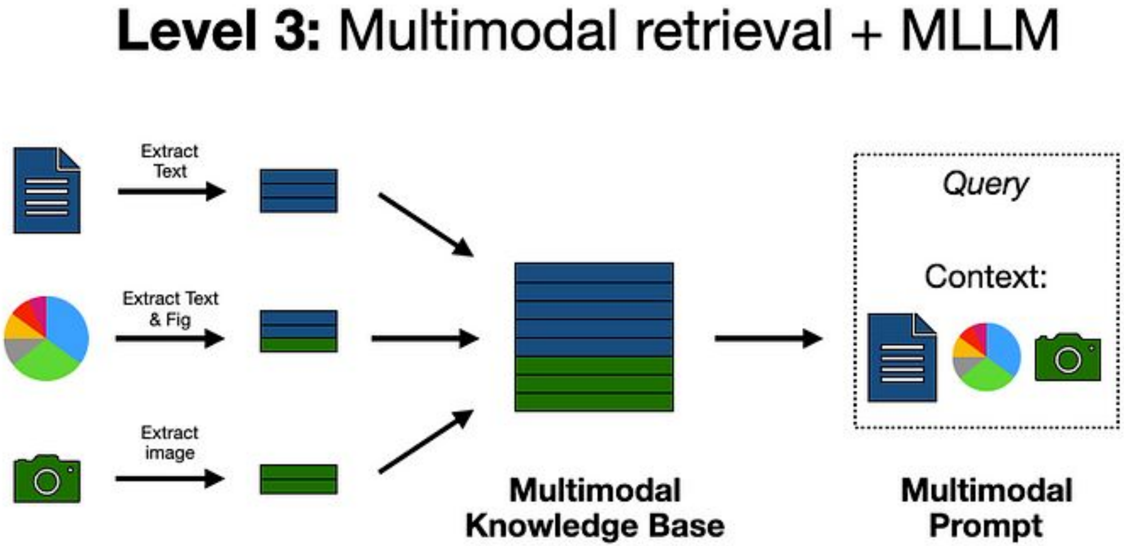
Level 2 - Text-only retrieval + MLLM

- Another approach involves generating text representations, like descriptions and meta-tags, for knowledge base items to aid retrieval, while passing the original modality to a multimodal LLM (MLLM). For instance, image metadata is used for retrieval, and the image itself is sent to the model for inference.



Level 3 - Multimodal retrieval + MLLM

- Use multimodal embeddings to perform multimodal retrieval. This works the same way as text-based vector search, but now the embedding space co-locates similar concepts independent of its original modality. The results of such a retrieval strategy can then be passed directly to a MLLM.



Implementation/Code walkthrough

- Import data
- Multi-vector retriever
- Building a RAG chain
- Querying
- Getting relevant images from the context

The project focuses on searching through a PDF that includes images and tables, with the LLM being able to understand both the content of the images and the text, and tables.

```python
import logging
import zipfile
import requests

logging.basicConfig(level=logging.INFO)

data_url = "https://storage.googleapis.com/benchmarks-artifacts/langchain-docs-benchmarking/cj.zip"
result = requests.get(data_url)
filename = "cj.zip"
with open(filename, "wb") as file:
    file.write(result.content)

with zipfile.ZipFile(filename, "r") as zip_ref:
    zip_ref.extractall()
```

```python
from langchain_community.document_loaders import PyPDFLoader

loader = PyPDFLoader("./cj/cj.pdf")
docs = loader.load()
tables = []
texts = [d.page_content for d in docs]
```

Importing Data

Loading a pdf with images and tables into the google colab

```python
def create_multi_vector_retriever(
    vectorstore, text_summaries, texts, table_summaries, tables, image_summaries, images
):
    """
    Create retriever that indexes summaries, but returns raw images or texts
    """

    # Initialize the storage layer
    store = InMemoryStore()
    id_key = "doc_id"

    # Create the multi-vector retriever
    retriever = MultiVectorRetriever(
        vectorstore=vectorstore,
        docstore=store,
        id_key=id_key,
    )

    # Helper function to add documents to the vectorstore and docstore
    def add_documents(retriever, doc_summaries, doc_contents):
        doc_ids = [str(uuid.uuid4()) for _ in doc_contents]
        summary_docs = [
            Document(page_content=s, metadata={id_key: doc_ids[i]})
            for i, s in enumerate(doc_summaries)
        ]
        retriever.vectorstore.add_documents(summary_docs)
        retriever.docstore.mset(list(zip(doc_ids, doc_contents)))

    # Add texts, tables, and images
    # Check that text_summaries is not empty before adding
    if text_summaries:
        add_documents(retriever, text_summaries, texts)
    # Check that table_summaries is not empty before adding
    if table_summaries:
        add_documents(retriever, table_summaries, tables)
    # Check that image_summaries is not empty before adding
    if image_summaries:
        add_documents(retriever, image_summaries, images)

    return retriever
```

Creating a retriever

Creating a multivector retriever that stores images, text and table data.

```python
# The vectorstore to use to index the summaries
vectorstore = Chroma(
    collection_name="mm_rag_cj_blog",
    embedding_function=VertexAIEmbeddings(model_name="textembedding-gecko@latest"),
)

# Create retriever
retriever_multi_vector_img = create_multi_vector_retriever(
    vectorstore,
    text_summaries,
    texts,
    table_summaries,
    tables,
    image_summaries,
    img_base64_list,
)
```

Chroma Db Vectorstore

Using chroma db Vectorstore to make a multi-vector vector retriever, which is an in-memory store.

```python
def multi_modal_rag_chain(retriever):
    """
    Multi-modal RAG chain
    """

    # Multi-modal LLM
    model = ChatVertexAI(
        temperature=0, model_name="gemini-pro-vision", max_output_tokens=1024
    )

    # RAG pipeline
    chain = (
        {
            "context": retriever | RunnableLambda(split_image_text_types),
            "question": RunnablePassthrough(),
        }
        | RunnableLambda(img_prompt_func)
        | model
        | StrOutputParser()
    )

    return chain


# Create RAG chain
chain_multimodal_rag = multi_modal_rag_chain(retriever_multi_vector_img)
```

Creating RAG Chain

Create RAG chain with Google Vertex Api, Langchain and Chroma db.

```python
query = "What are the EV / NTM and NTM rev growth for MongoDB, Cloudflare, and Datadog?"
docs = retriever_multi_vector_img.get_relevant_documents(query, limit=1)

# We get relevant docs
len(docs)
```

```
2
```

```python
plt_img_base64(docs[0])
```

Company		EV / NTM Rev	EV / 2024 Rev	EV / NTM FCF	NTM Rev Growth	Gross Margin
1	Snowflake	15.5x	13.4x	55x	27%	66%
2	MongoDB	14.6x	12.9x	133x	17%	74%
3	Palantir	14.5x	13.9x	58x	19%	80%
4	Cloudflare	13.4x	12.6x	153x	28%	76%
5	Datadog	13.1x	12.4x	52x	19%	80%
6	CrowdStrike	12.5x	11.1x	37x	31%	74%
7	Adobe	12.3x	11.9x	30x	12%	88%
8	ServiceNow	12.2x	11.6x	38x	21%	79%
9	Samsara	11.8x	10.5x	393x	31%	72%
10	Zscaler	11.8x	10.5x	48x	27%	78%
Average		13.2x	12.1x	100x	23%	77%
Median		12.8x	12.2x	54x	24%	77%
Overall Median		5.0x	4.8x	33.7x	15%	75%

Clouded Judgement @jaminball

Querying

We can query the entire PDF, including images, text, and tables.

Result

The retrieved image based on the query.

```python
chain_multimodal_rag.invoke("what is the document about?")
```

```
'The document is about the valuation of SaaS businesses. It discusses the use of revenue multiples as a short
```

```python
result = chain_multimodal_rag.invoke(query)
print(result)
```

```
| Company | EV/NTM Rev | NTM Rev Growth |
|---|---|---|
| MongoDB | 14.6x | 17% |
| Cloudflare | 13.4x | 28% |
| Datadog | 13.1x | 19% |
```

```python
from IPython.display import Markdown as md
md(result)
```

Company	EV/NTM Rev	NTM Rev Growth
MongoDB	14.6x	17%
Cloudflare	13.4x	28%
Datadog	13.1x	19%

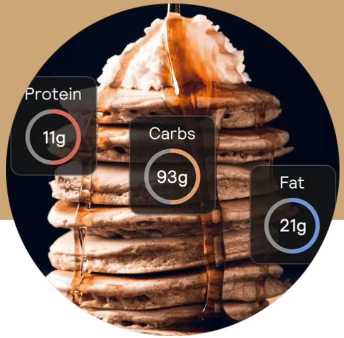
Result

We asked LLM a very specific question about the revenue which was present only in the image, and LLM was able to answer it properly.

Applications that you can build

Virtual Interviewer

An AI virtual interviewer, that you can practice interviews with.



Calorie Tracker

A calorie tracker app that give the calories of the food based on the image provided

Image Search Engine

A search engine that can understand images, and videos.

Stock Analyzer

A stock analyzer that can understands complex tables and charts

Conclusion

We understood what Multimodal RAG is, its importance, applications, and also examined the implementation of a Multimodal RAG chain.