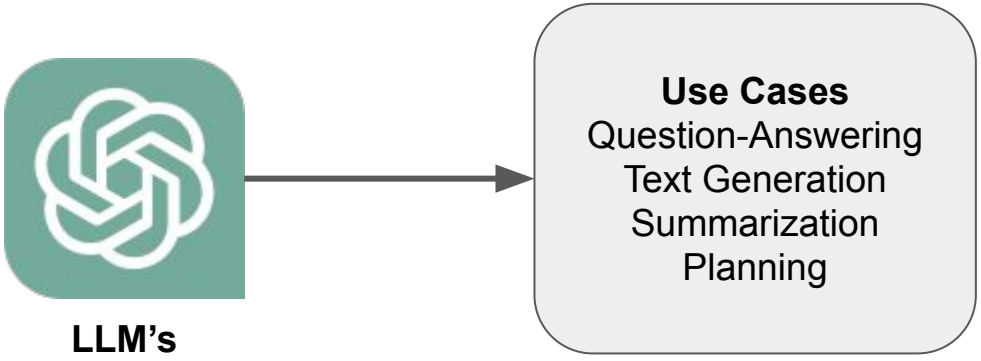
Evaluating and Optimizing your RAG App

Jerry Liu, LlamaIndex co-founder/CEO

RAG

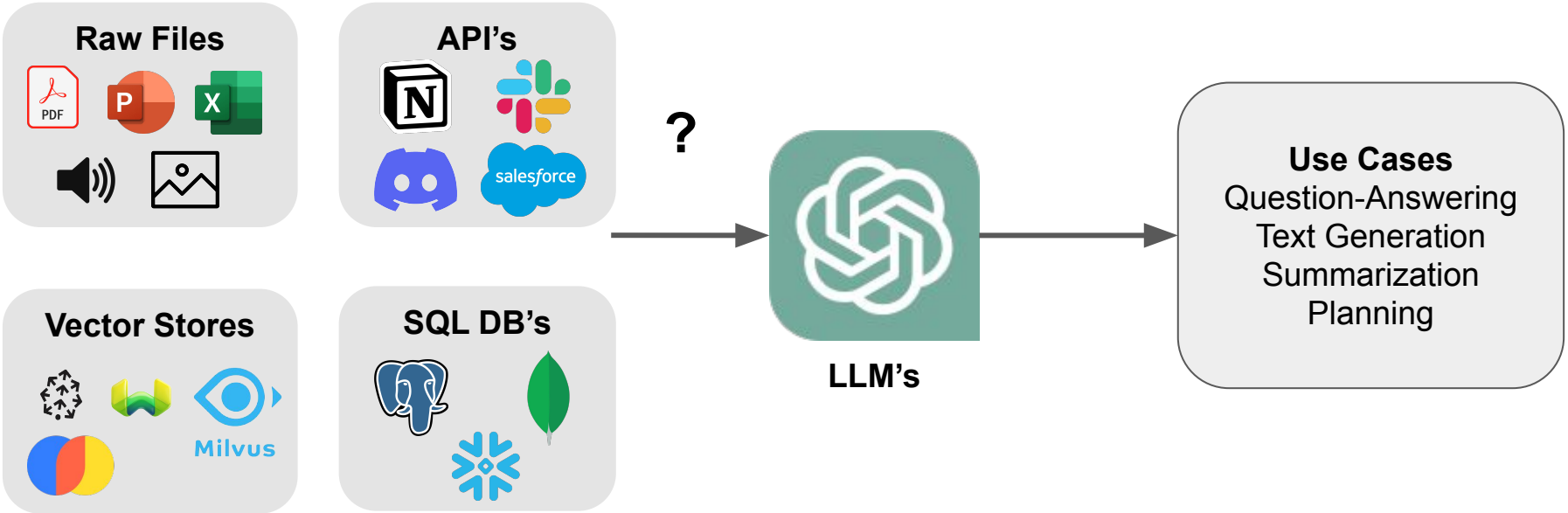
Context

- LLMs are a phenomenal piece of technology for knowledge generation and reasoning. They are pre-trained on large amounts of **publicly available data**.



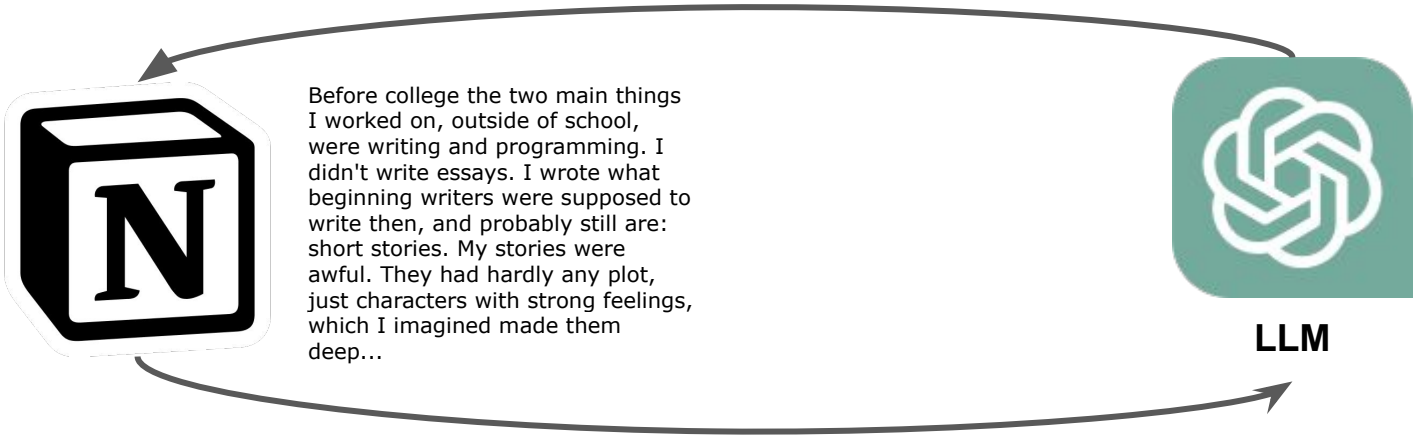
Context

- How do we best augment LLMs with our own **private data**?



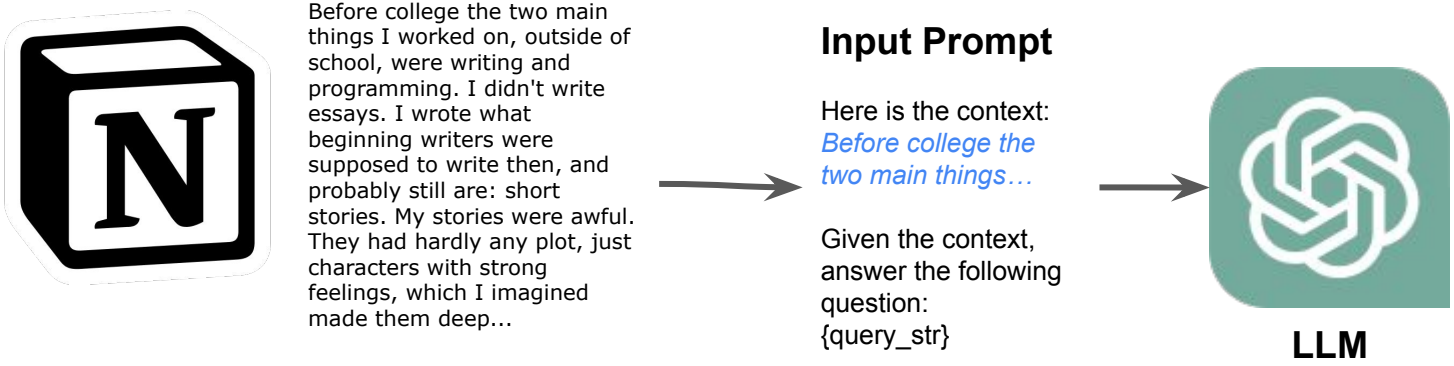
Paradigms for inserting knowledge

Fine-tuning - baking knowledge into the weights of the network



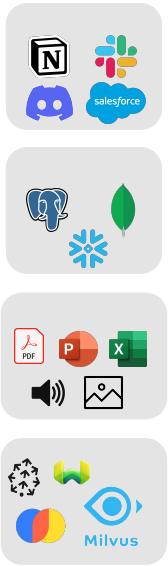
Paradigms for inserting knowledge

Retrieval Augmentation - Fix the model, put context into the prompt



LlamaIndex: A data framework for LLM applications

- Data Management and Query Engine for your LLM application
- Offers components across the data lifecycle: ingest, index, and query over data



Data Ingestion
(LlamaHub 🦙)

- Connect your existing data sources and data formats (API's, PDF's, docs, SQL, etc.)

Data Structures

- Store and index your data for different use cases. Integrate with different db's (vector db, graph db, kv db)

Queries

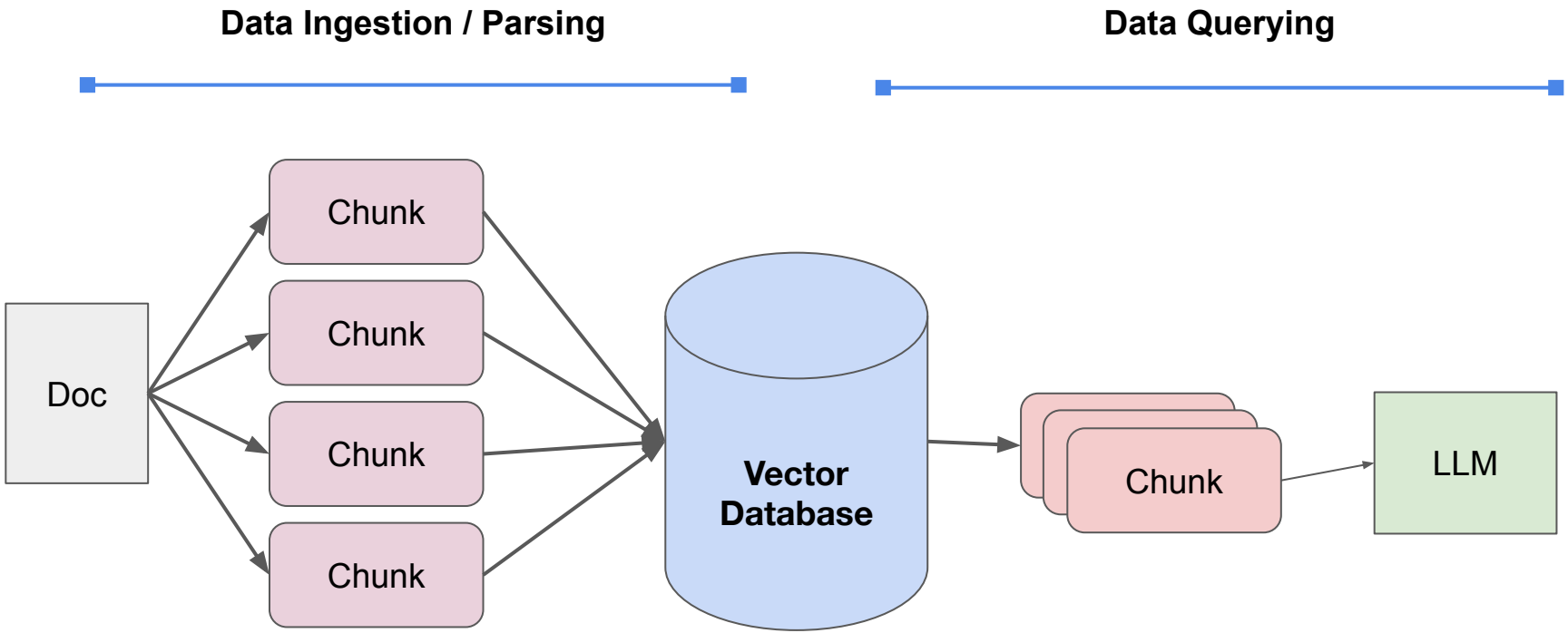
- Retrieve and query over data
- Includes: QA, Summarization, Agents, and more

```
from llama_index import VectorStoreIndex, SimpleDirectoryReader

documents = SimpleDirectoryReader('data').load_data()
index = VectorStoreIndex.from_documents(documents)
query_engine = index.as_query_engine()
response = query_engine.query("What did the author do growing
print(response)
```

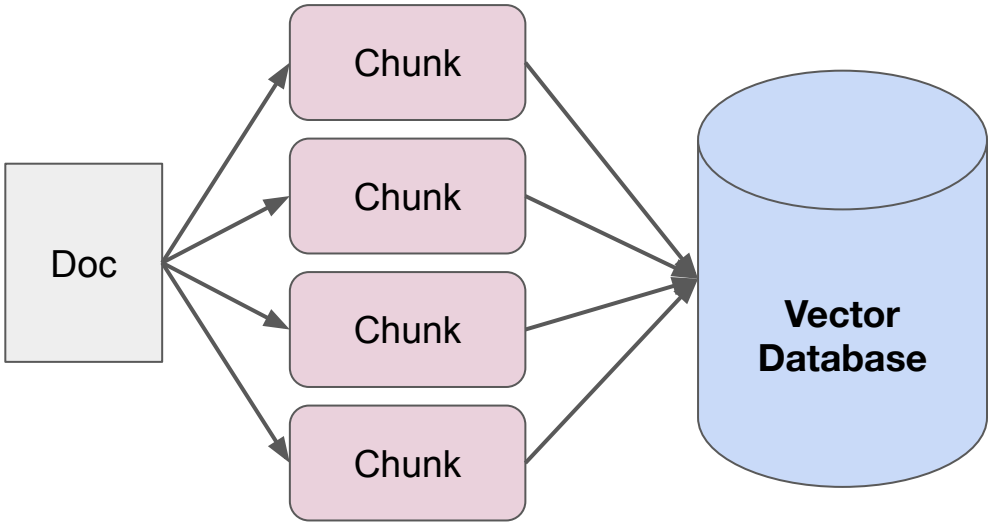
RAG Stack

Current RAG Stack for building a QA System



5 Lines of Code in LlamaIndex!

Current RAG Stack (Data Ingestion/Parsing)



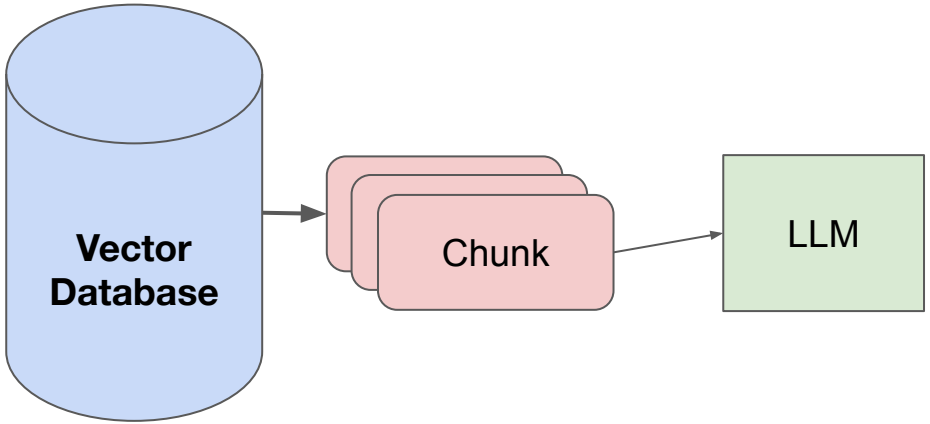
Process:

- Split up document(s) into even chunks.
- Each chunk is a piece of raw text.
- Generate embedding for each chunk (e.g. OpenAI embeddings, sentence_transformer)
- Store each chunk into a vector database

Current RAG Stack (Querying)

Process:

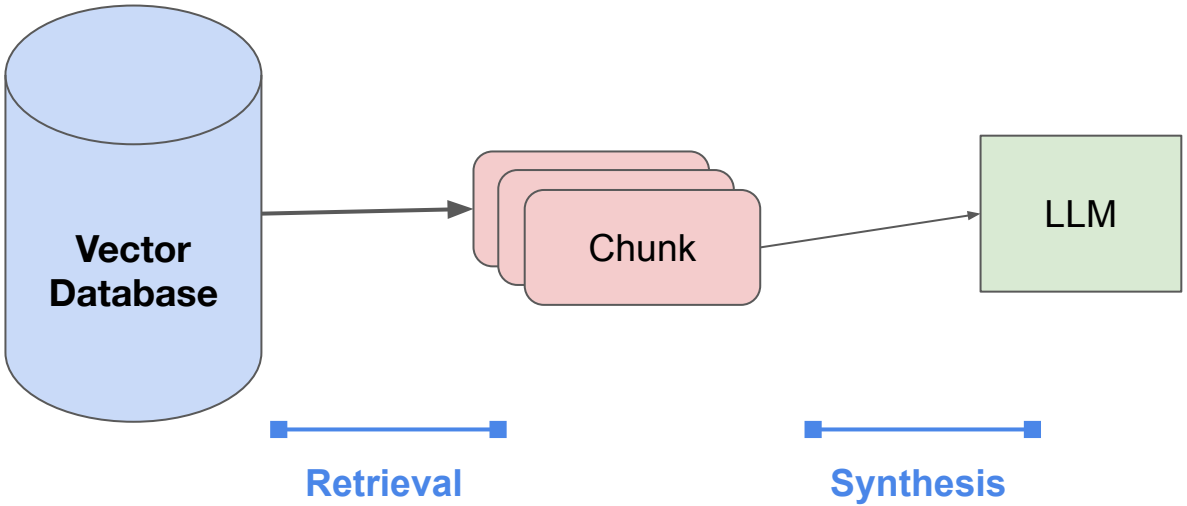
- Find top-k most similar chunks from vector database collection
- Plug into LLM response synthesis module



Current RAG Stack (Querying)

Process:

- Find top-k most similar chunks from vector database collection
- Plug into LLM response synthesis module



Challenges with “Naive” RAG

Challenges with Naive RAG

- Failure Modes
  - Quality-Related (Hallucination, Accuracy)
  - Non-Quality-Related (Latency, Cost, Syncing)

Challenges with Naive RAG (Response Quality)

- Bad Retrieval
  - **Low Precision:** Not all chunks in retrieved set are relevant
    - Hallucination + Lost in the Middle Problems
  - **Low Recall:** Now all relevant chunks are retrieved.
    - Lacks enough context for LLM to synthesize an answer
  - **Outdated information:** The data is redundant or out of date.

Challenges with Naive RAG (Response Quality)

- Bad Retrieval
  - **Low Precision:** Not all chunks in retrieved set are relevant
    - Hallucination + Lost in the Middle Problems
  - **Low Recall:** Now all relevant chunks are retrieved.
    - Lacks enough context for LLM to synthesize an answer
  - **Outdated information:** The data is redundant or out of date.
- Bad Response Generation
  - **Hallucination:** Model makes up an answer that isn't in the context.
  - **Irrelevance:** Model makes up an answer that doesn't answer the question.
  - **Toxicity/Bias:** Model makes up an answer that's harmful/offensive.

What do we do?

- **Data:** Can we store additional information beyond raw text chunks?
- **Embeddings:** Can we optimize our embedding representations?
- **Retrieval:** Can we do better than top-k embedding lookup?
- **Synthesis:** Can we use LLMs for more than generation?

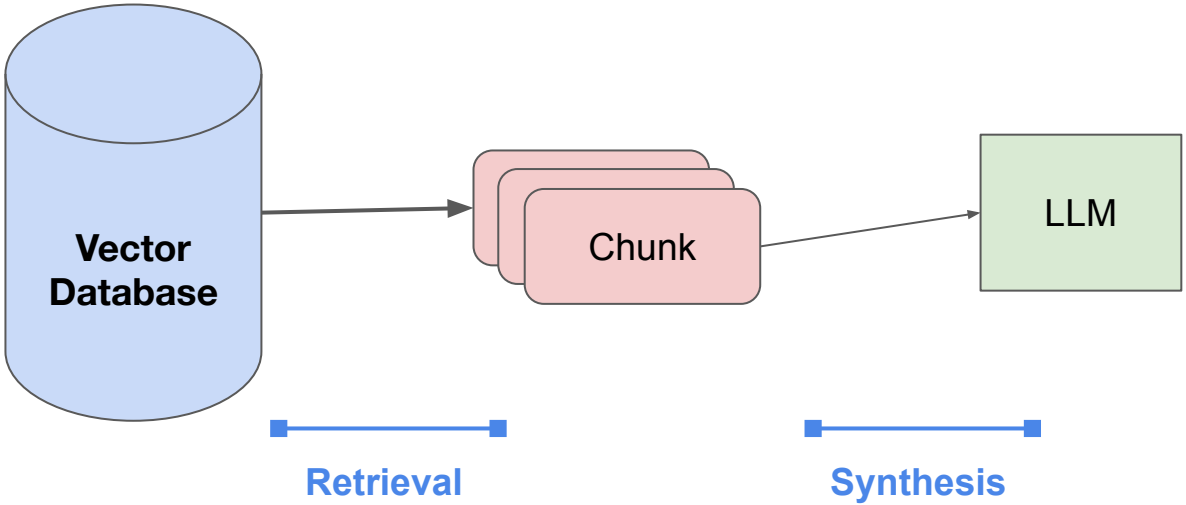
But before all this…

We need a way to measure performance

Evaluation

Evaluation

- How do we properly evaluate a RAG system?
  - Evaluate in isolation (retrieval, synthesis)
  - Evaluate e2e
- Open question: which one should we do first?

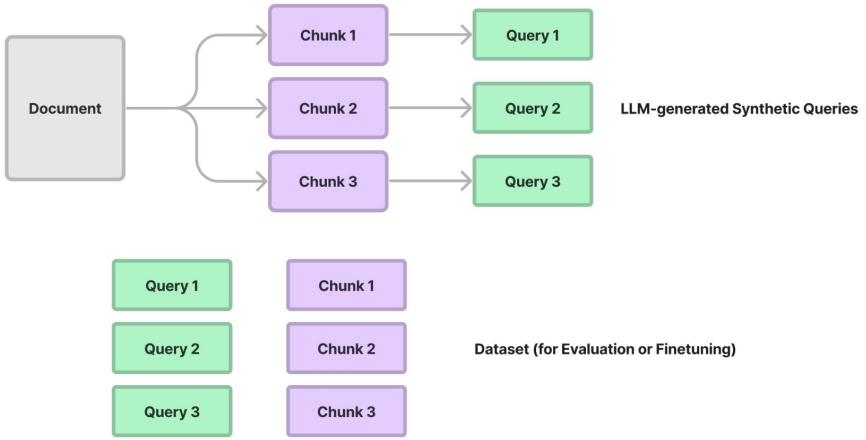


Evaluation in Isolation (Retrieval)

- **Details:** Evaluate quality of retrieved chunks given user query
- Collect dataset
  - Input: query
  - Output: the “ground-truth” documents relevant to the query
- Run retriever over dataset
- Measure **ranking metrics**
  - Success rate / hit-rate
  - MRR
  - Hit-rate

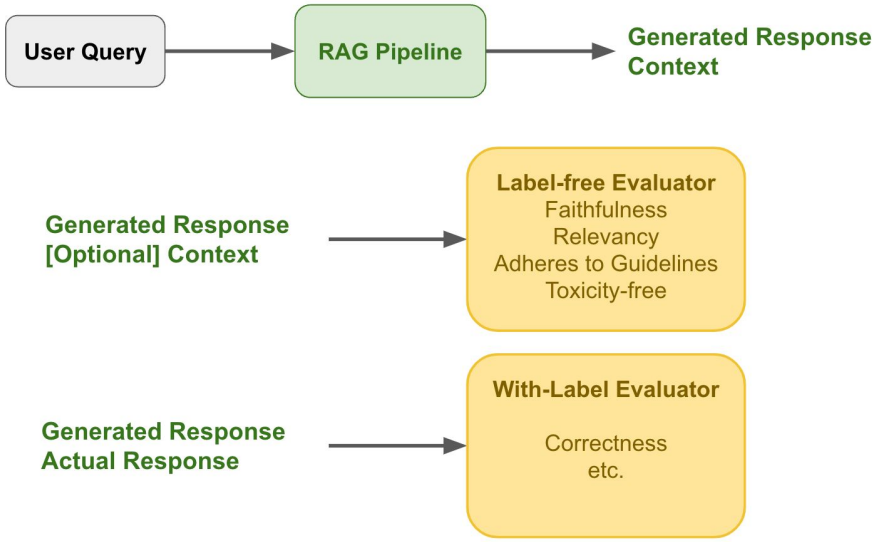
Synthetic Dataset Generation for Retrieval Evals

1. Parse / chunk up text corpus
2. Prompt GPT-4 to generate questions from each chunk (or subset of chunks)
3. Each (question, chunk) is now your evaluation pair!



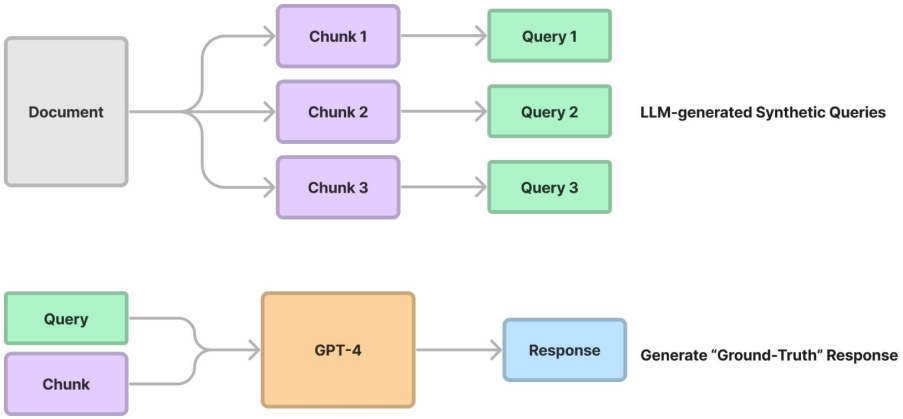
Evaluation E2E

- **Details:** Evaluation of final generated response given input
- Collect dataset
  - Input: query
  - [Optional] Output: the “ground-truth” answer
- Run through full RAG pipeline
- Collect evaluation metrics:
  - **If no labels:** label-free evals
  - **If labels:** with-label evals



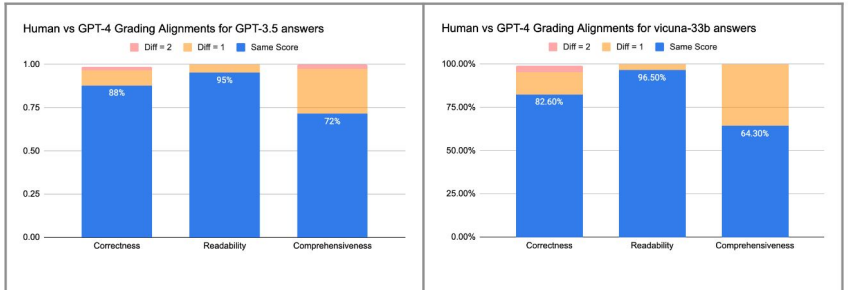
Synthetic Dataset Generation for E2E Evals

1. Parse / chunk up text corpus
2. Prompt GPT-4 to generate questions from each chunk
3. Run (question, context) through GPT-4 → Get a “ground-truth” response
4. Each (question, response) is now your evaluation pair!

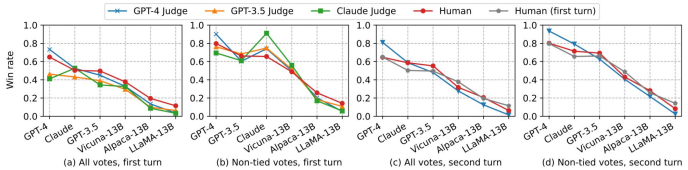


LLM-based Evaluation Modules

- GPT-4 is a good human grader
- **Label-free Modules**
  - **Faithfulness**: whether response matches retrieved context
  - **Relevancy**: whether response matches query
  - **Guidelines**: whether response matches guidelines
- **With-Labels**
  - **Correctness**: whether response matches “golden” answer.



https://www.databricks.com/blog/LLM-auto-eval-best-practices-RAG

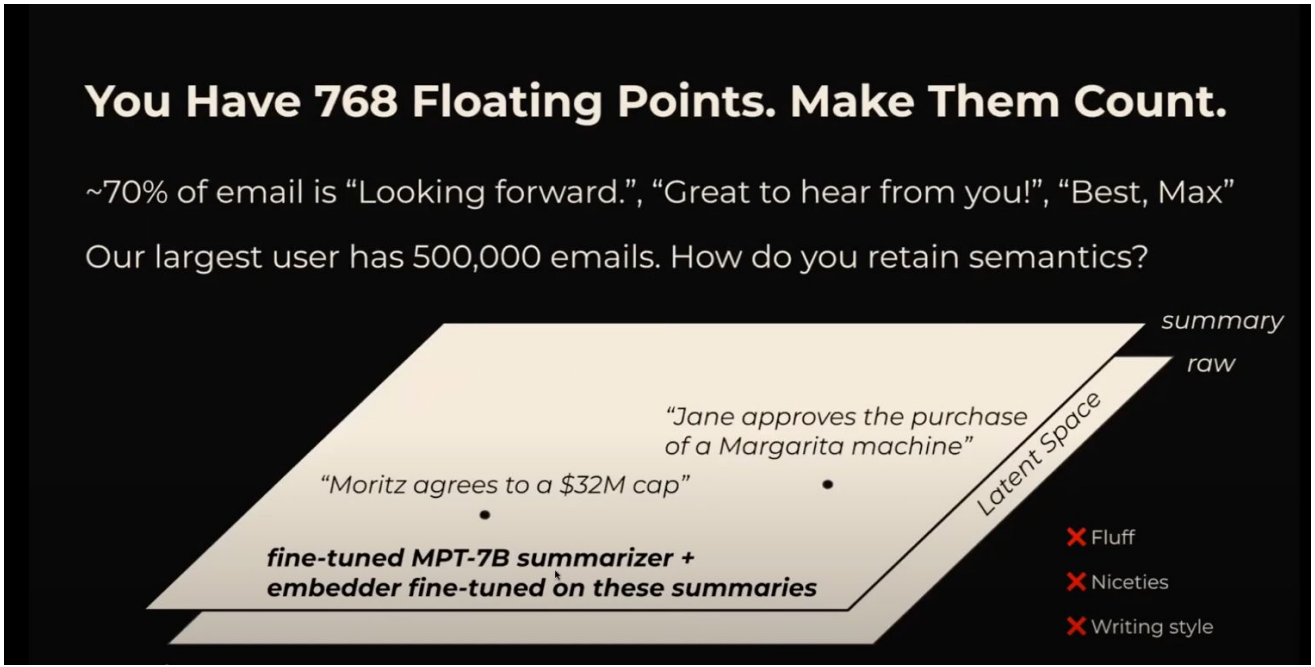


https://arxiv.org/pdf/2306.05685.pdf

Techniques for Better Performing RAG

Decouple Embeddings from Raw Text Chunks

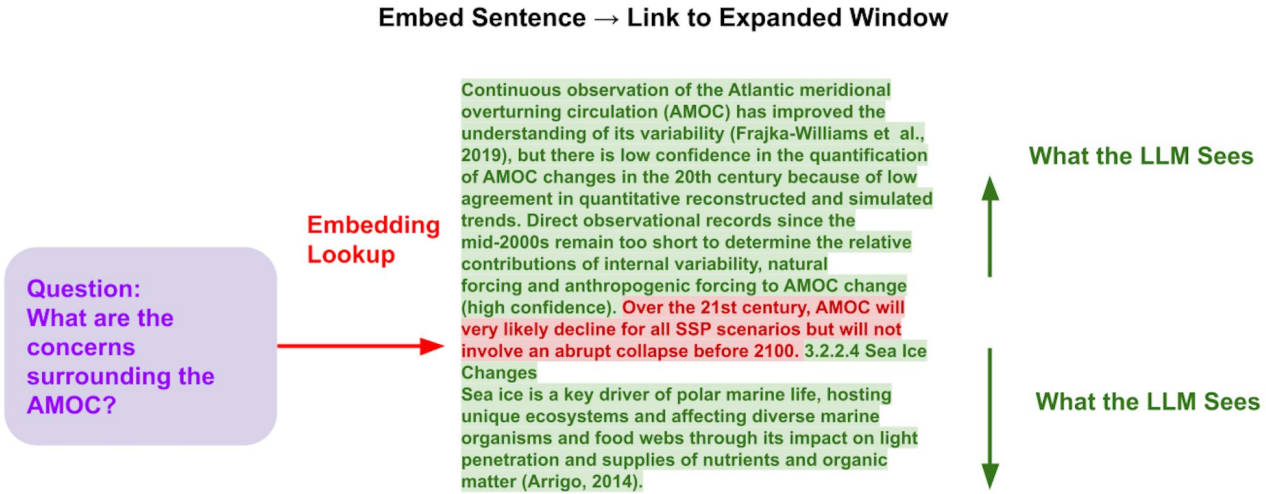
Raw text chunks can bias your embedding representation with filler content (Max Rumpf, sid.ai)



Small-to-Big Retrieval

Solutions:

- Embed text at the sentence-level - then **expand** that window during LLM synthesis



Small-to-Big Retrieval

Solutions:

- Embed text at the sentence-level - then **expand** that window during LLM synthesis

```
There is low confidence in the quantification of AMOC changes in the 20th ce
ntury due to low agreement in quantitative reconstructed and simulated trend
s. Additionally, direct observational records since the mid-2000s remain too
short to determine the relative contributions of internal variability, natur
al forcing, and anthropogenic forcing to AMOC change. However, it is very li
kely that AMOC will decline over the 21st century for all SSP scenarios, but
there will not be an abrupt collapse before 2100.
```

Sentence Window Retrieval (k=2)

```
I'm sorry, but the concerns surrounding the AMOC (Atlantic Meridional Overtu
rning Circulation) are not mentioned in the provided context.
```

Naive Retrieval (k=5)

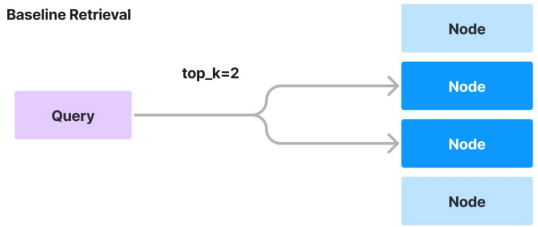
Only one out of the 5 chunks is relevant - “lost in the middle” problem

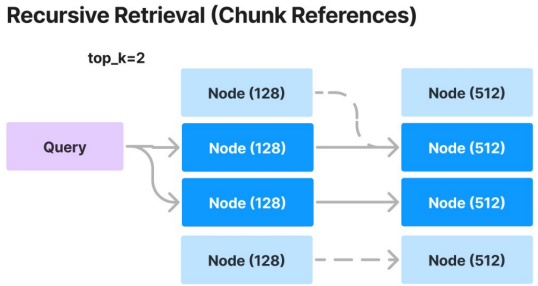
Embed References to Text Chunks

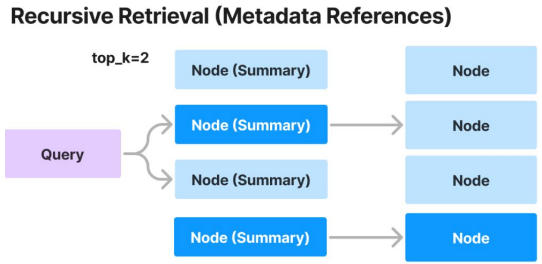
Solutions:

- Embed “**references**” to text chunks instead of the text chunks directly.
- Examples
  - Smaller Chunks
  - Metadata
  - Summaries
- Retrieve those references first, then the text chunks.

	retrievers	hit_rate	mrr
0	Base Retriever	0.269155	0.191413
1	Retriever (Chunk References)	0.292731	0.254551
2	Retriever (Metadata References)	0.286837	0.240858



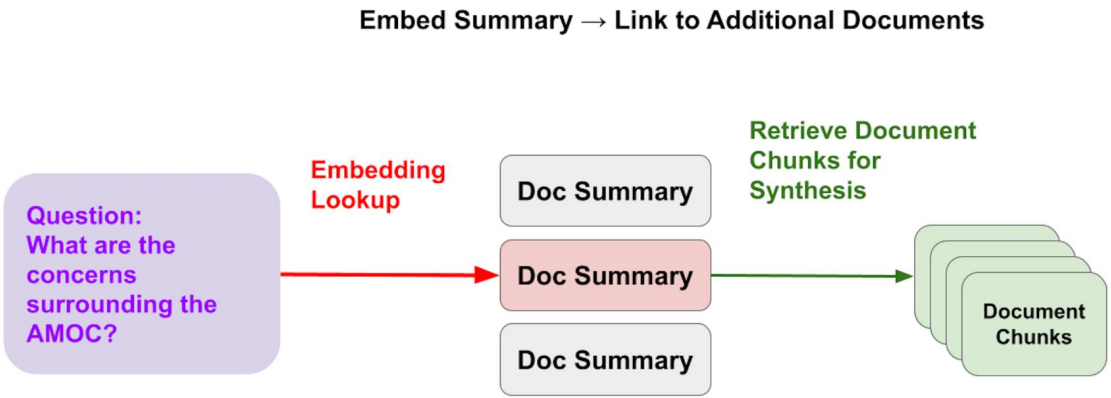




Organize your data for more structured retrieval (Recursive Retrieval)

Summaries → documents

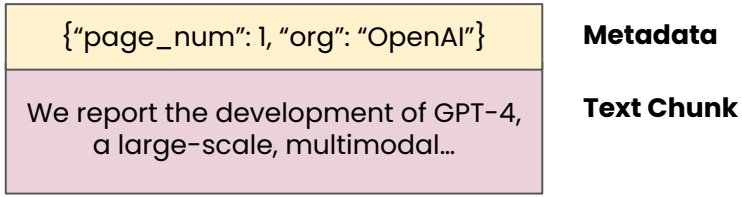
- Embed larger documents via **summaries.** First retrieve documents by summaries, then retrieve chunks within those documents



Organize your data for more structured retrieval (Metadata)

- **Metadata:** context you can inject into each text chunk
- Examples
  - Page number
  - Document title
  - Summary of adjacent chunks
  - Questions that chunk can answer (reverse HyDE)
- **Benefits**
  - **Can Help Retrieval**
  - **Can Augment Response Quality**
  - **Integrates with Vector DB Metadata Filters**

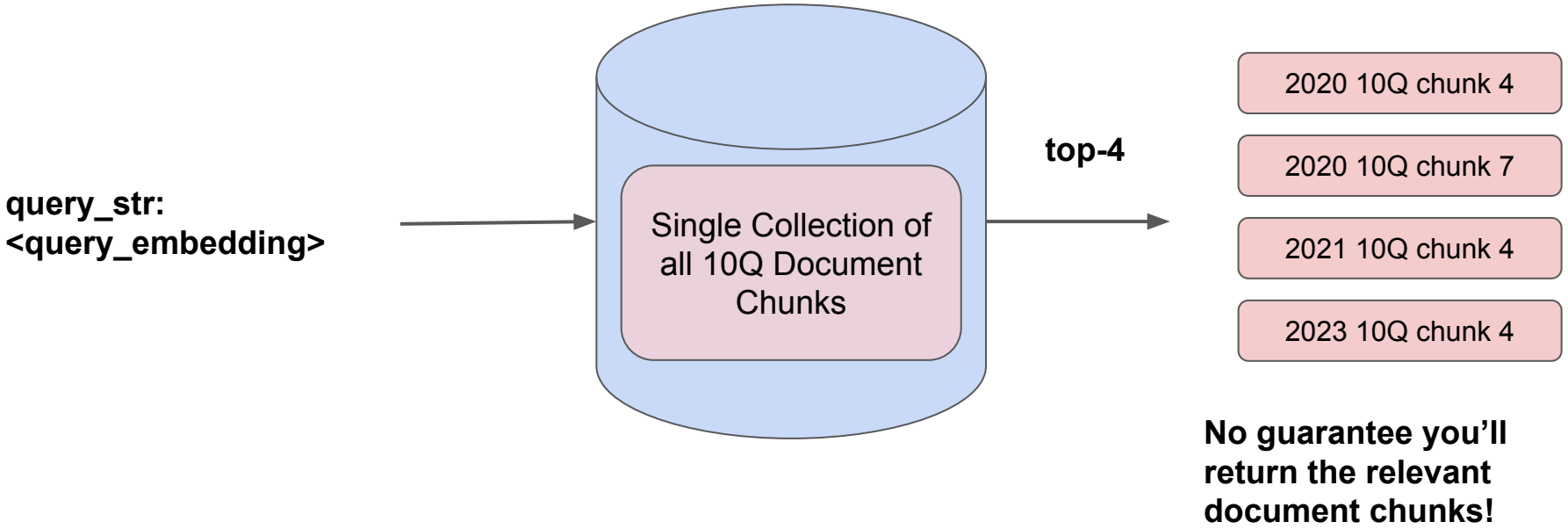
Example of Metadata



Organize your data for more structured retrieval (Metadata Filters)

Question: “Can you tell me about Google’s R&D initiatives from 2020 to 2023?”

- Dumping chunks to a single collection doesn’t work.



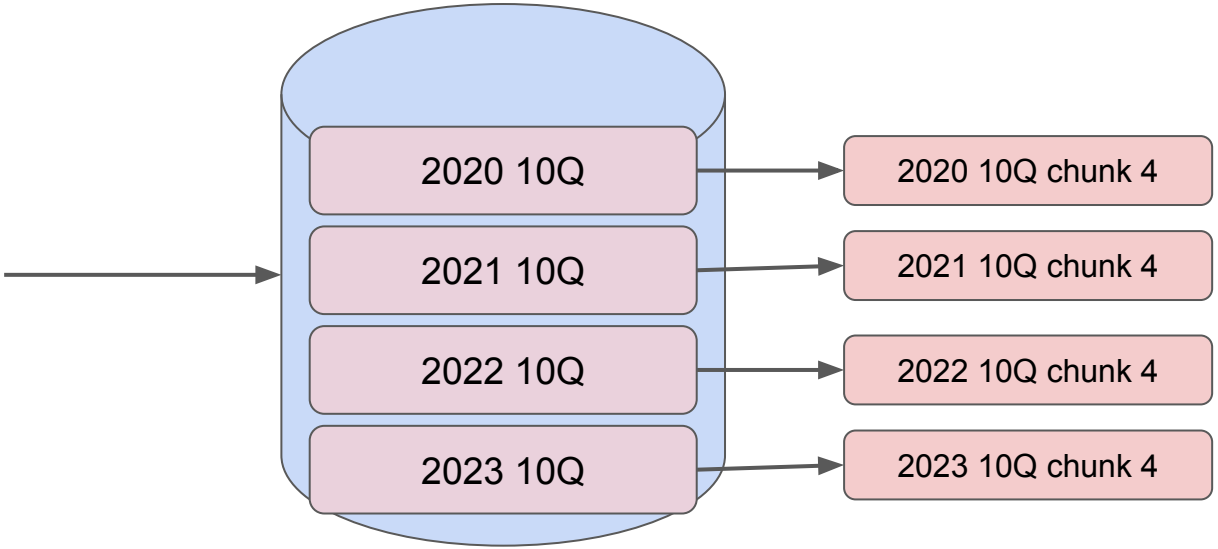
Organize your data for more structured retrieval (Metadata Filters)

Question: “Can you tell me about Google’s R&D initiatives from 2020 to 2023?”

- Here, we separate and tag the documents with **metadata filters**.
- During query-time, we can *infer* these metadata filters in addition to semantic query.

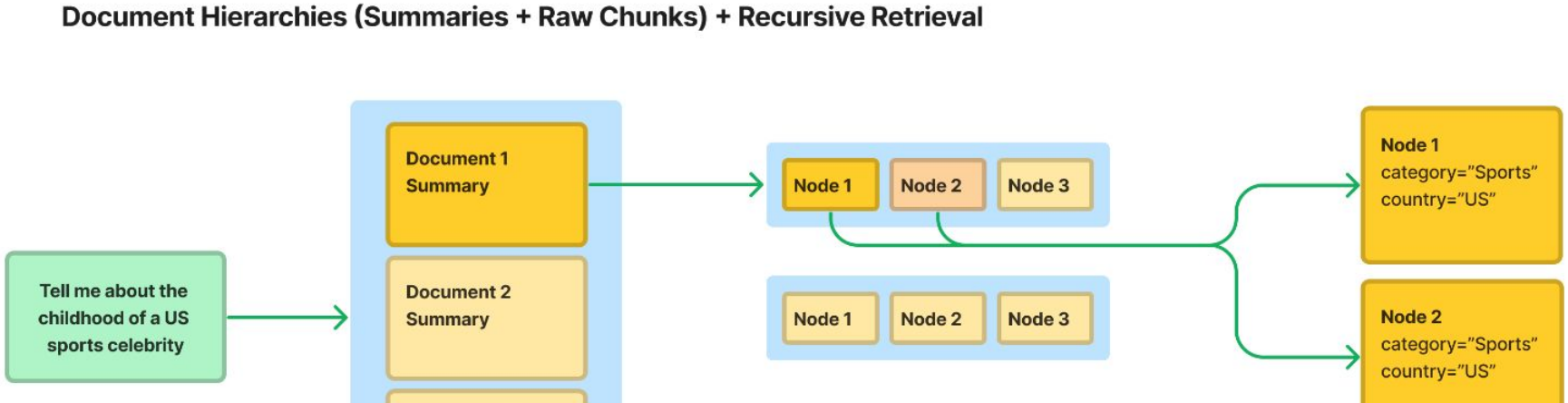
query_str:
<query_embedding>

Metadata tags:
<metadata_tags>



Organize your data for more structured retrieval (Recursive Retrieval)

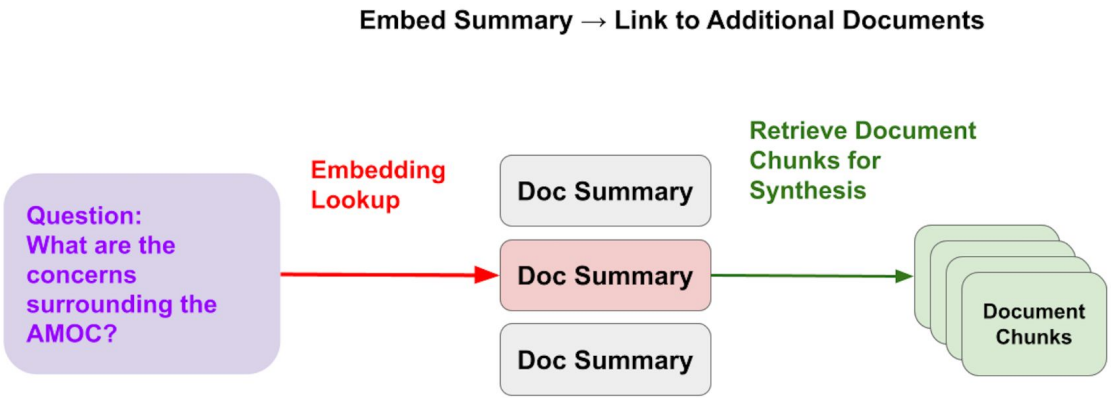
- Organize your data **hierarchically**
  - Summaries → documents
  - Documents → embedded objects (Tables/Graphs)



Organize your data for more structured retrieval (Recursive Retrieval)

Summaries → documents

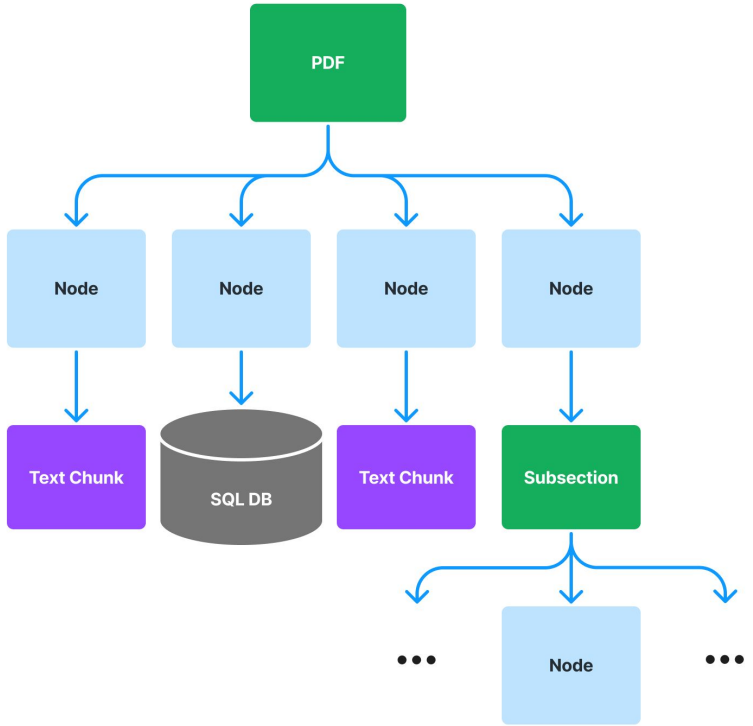
- Embed larger documents via **summaries.** First retrieve documents by summaries, then retrieve chunks within those documents



Organize your data for more structured retrieval (Recursive Retrieval)

Documents → Embedded Objects

- If you have embedded objects in your PDF documents (tables, graphs), first retrieve entities by a **reference object,** then query the underlying object.



Production RAG Guide

https://gpt-index.readthedocs.io/en/latest/end_to_end_tutorials/dev_practices/production_rag.html

Fine-Tuning

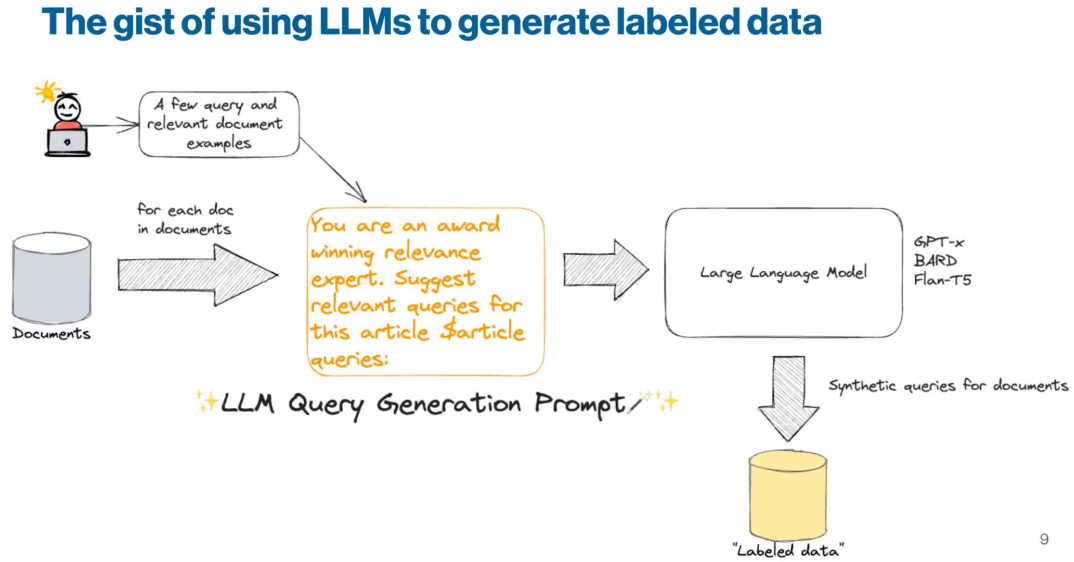
Fine-tuning

You can choose to fine-tune the **embeddings** or the **LLM**

Fine-tuning (Embeddings)

Generate a synthetic query dataset from raw text chunks using LLMs

NOTE: Similar process to generating an evaluation dataset!



Credits: Jo Bergum, vespa.ai

Fine-tuning (Embeddings)

Use this synthetic dataset to finetune an embedding model.

- Directly finetune sentence_transformers model
- Finetune a black-box adapter (linear, NN, any neural network)

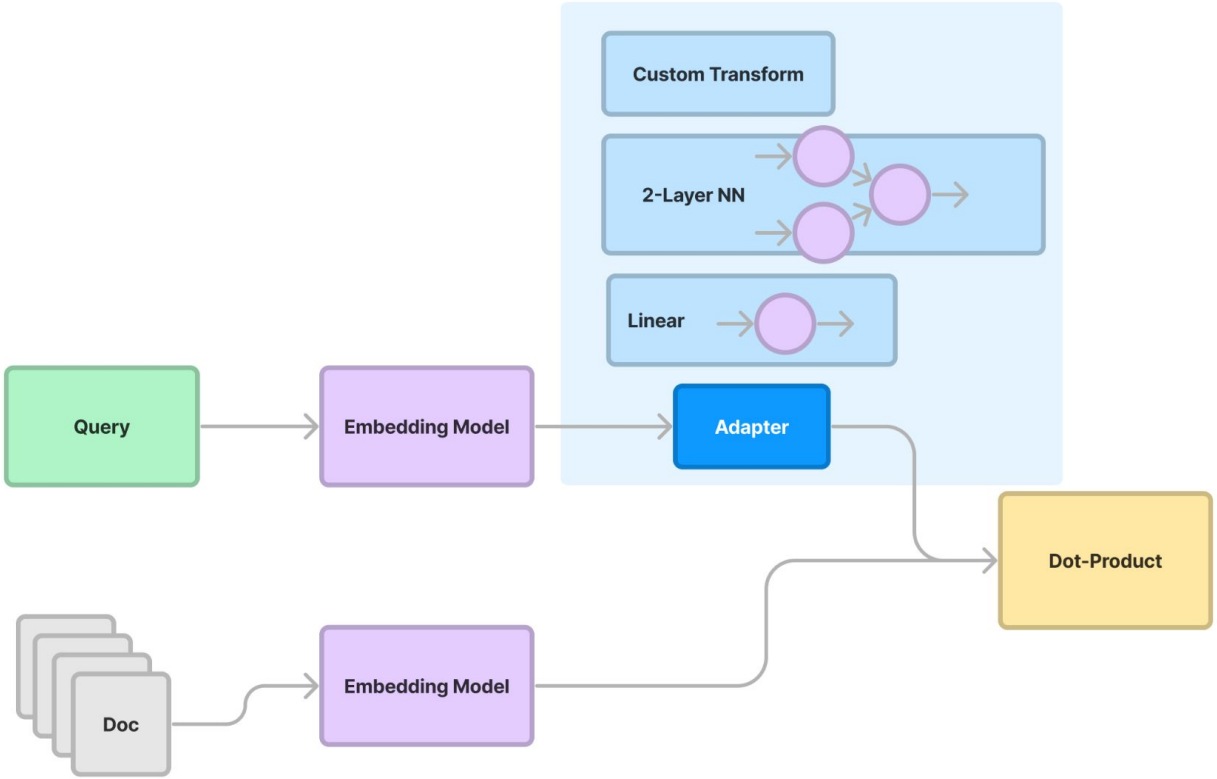
Run Embedding Finetuning

[9]:
```
from llama_index.finetuning import SentenceTransformersFinetuneEngine
```

[10]:
```
finetune_engine = SentenceTransformersFinetuneEngine(
    train_dataset,
    model_id="BAAI/bge-small-en",
    model_output_path="test_model",
    val_dataset=val_dataset,
)
```

[11]:
```
finetune_engine.finetune()
```

Fine-tuning a Black-box Adapter



Fine-tuning (LLMs)

Use OpenAI to distill GPT-4 to gpt-3.5-turbo

- Final response generation
- Agent intermediate reasoning

Original

In [9]:
```
from llama_index.response.notebook_utils import display_response
from llama_index import ServiceContext
from llama_index.llms import OpenAI


gpt_35_context = ServiceContext.from_defaults(
    llm=OpenAI(model="gpt-3.5-turbo", temperature=0.3),
    context_window=2048,  # limit the context window artifically to test refine process
)
```

In [10]:
```
query_engine = index.as_query_engine(service_context=gpt_35_context)

response = query_engine.query(questions[12])

display_response(response)
```

Final Response: According to the report, a key barrier globally for ocean health, governance, and adaptation to climate change is the availability of technology, knowledge, and financial support, as well as existing governance structures.

Fine-Tuned

In [12]:
```
from llama_index import ServiceContext
from llama_index.llms import OpenAI


ft_context = ServiceContext.from_defaults(
    llm=OpenAI(model=ft_model_name, temperature=0.3),
    context_window=2048,  # limit the context window artifically to test refine process
)
```

In [13]:
```
query_engine = index.as_query_engine(service_context=ft_context)

response = query_engine.query(questions[12])

display_response(response)
```

Final Response: The report identifies a broad range of barriers and limits for adaptation to climate change in ecosystems and human systems. These limitations include the availability of technology, knowledge, and financial support, as well as existing governance structures. Existing ocean-governance structures are already facing multi-dimensional, scale-related challenges because of climate change.

Finetuning Abstractions in LlamaIndex

https://gpt-index.readthedocs.io/en/latest/end_to_end_tutorials/finetuning.html