21.11.23

OPENCAMPUS.sh

Practical Engineering with LLMs

Introduction to Retrieval Augmented Generation

TODAY’S SCHEDULE

- Quiz
- Homework presentation
- Short Recap
- Projects
- Homework for next week

TODAY’S SCHEDULE

- Quiz
- Homework presentation
- Short Recap
- Projects
- Homework for next week

QUIZ

https://forms.office.com/r/3KqHTA2CPj

TODAY’S SCHEDULE

- Quiz
- Homework presentation
- Short Recap
- Projects
- Homework for next week

HOMEWORK PRESENTATION

Tasks

1. Replace the OpenAI model with a chat model of your choice (e.g. gpt-3.5-turbo, Cohere Chat, etc.).
2. Replace the OpenAIEmbeddings with an open-source text embedding model for example from HuggingFace. A list of available embeddings integrations can be found here. A leaderboard of the best (open-source) embedding models can be found here.
3. Replace the vector store with a different vector store that either works locally (e.g. Chroma, etc.) or a hosted cloud vector store (e.g. Weaviate, Pinecone, etc.)
4. Do further modifications like changing the text splitter or changing the PDF document.

TODAY’S SCHEDULE

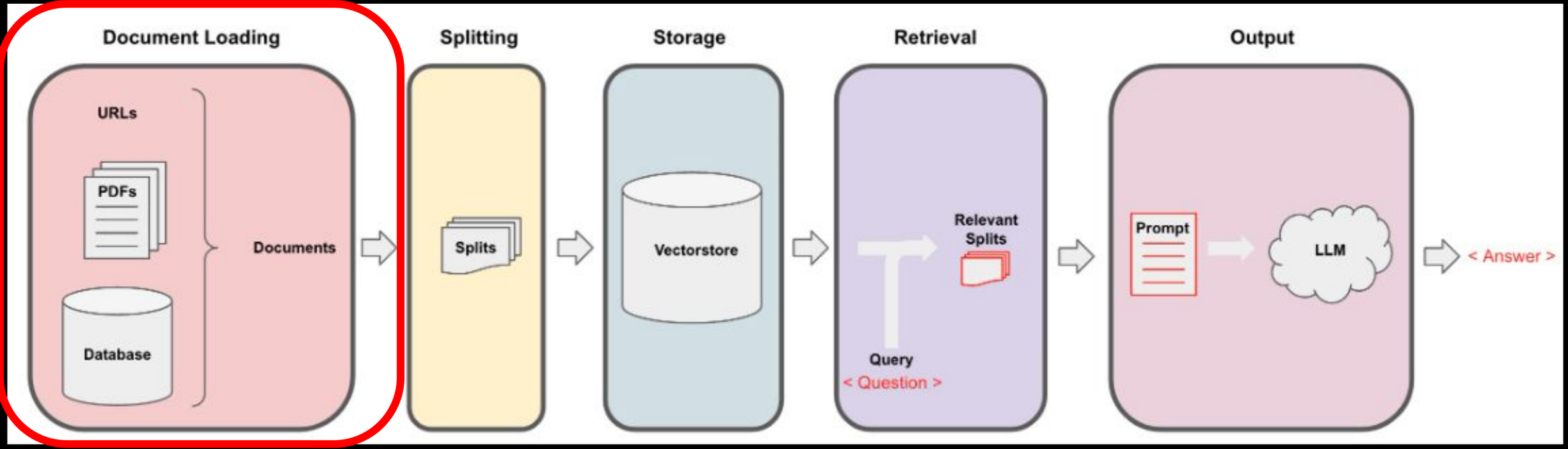
- Quiz
- Homework presentation
- Short Recap
- Projects
- Homework for next week

QUICK RECAP: LangChain

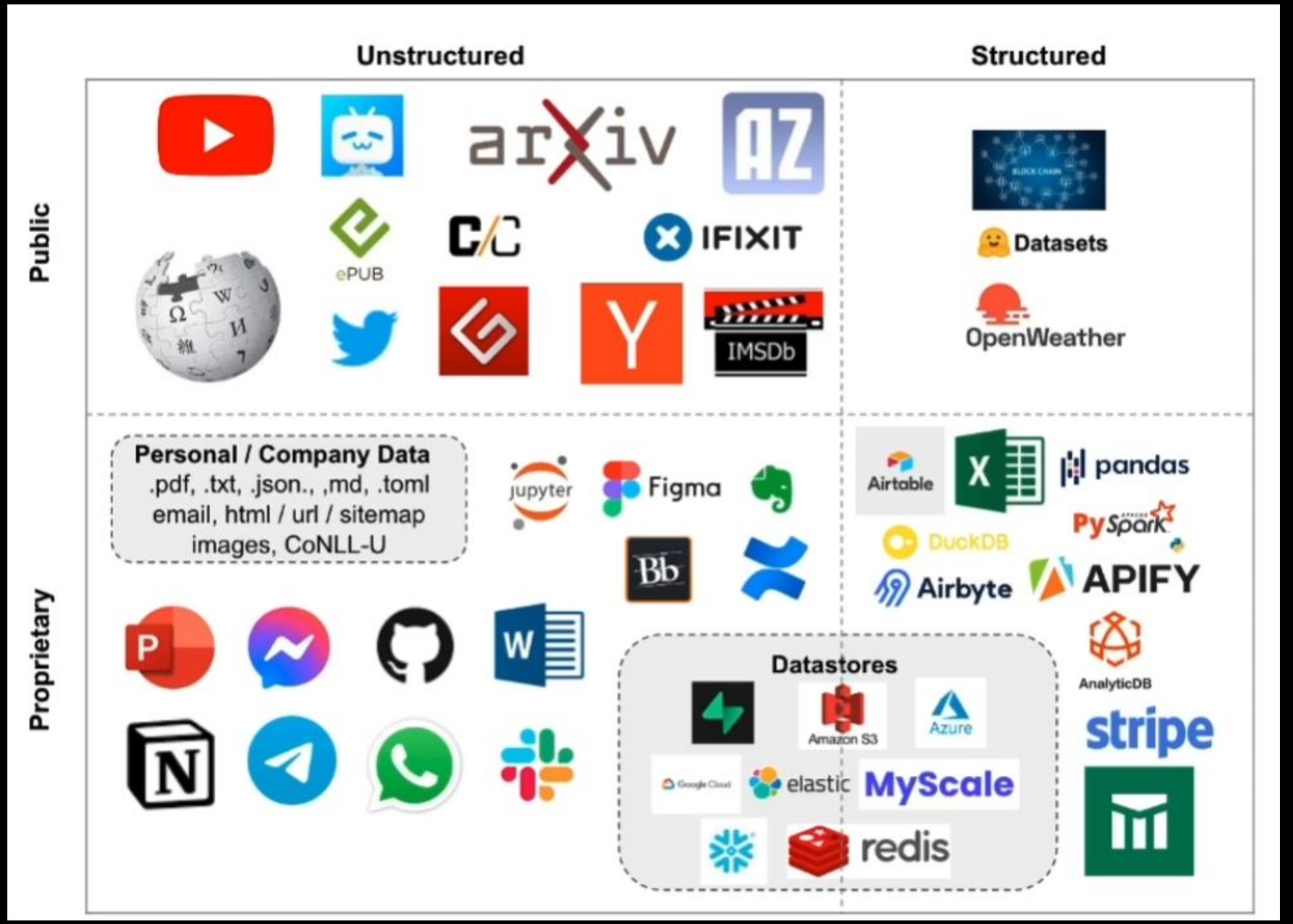
Components

- Prompts
  - Prompt Templates
  - Output Parsers: 5+ implementations
    - Retry/fixing logic
  - Example Selectors: 5+ implementations
- Models
  - LLM's: 20+ integrations
  - Chat Models
  - Text Embedding Models: 10+ integrations
- Indexes
  - Document Loaders: 50+ implementations
  - Text Splitters: 10+ implementations
  - Vector stores: 10+ integrations
  - Retrievers: 5+ integrations/implementations
- Chains
  - Can be used as building blocks for other chains
  - More application specific chains: 20+ different types
- Agents
  - Agent Types: 5+ types
    - Algorithms for getting LLMs to use tools
  - Agent Toolkits: 10+ implementations
    - Agents armed with specific tools for a specific application

QUICK RECAP: From Docs to Answers



QUICK RECAP: Document Loaders



QUICK RECAP: Document Loaders

- Loaders deal with the specifics of accessing and converting data
  - Accessing
    - Web Sites
    - Data Bases
    - YouTube
    - arXiv
    - ...
  - Data Types
    - PDF
    - HTML
    - JSON
    - Word, PowerPoint...
- Returns a list of `Document` objects:

standard document format

- **data**
- **metadata**

https://python.langchain.com/docs/modules/data_connection/document_loaders/

QUICK RECAP: Document Loaders

syntax template

```
from langchain.document_loaders import [name of loader]
loader = [name of loader]("[file or url]")
pages = loader.load
```

QUICK RECAP: Document Loaders

example: pdfs

```
from langchain.document_loaders import PyPDFLoader
loader = PyPDFLoader("docs/cs229_lectures/MachineLearning-Lecture01.pdf")
pages = loader.load()
```

QUICK RECAP: Document Loaders

loader.load() -> list of langchain.Document objects

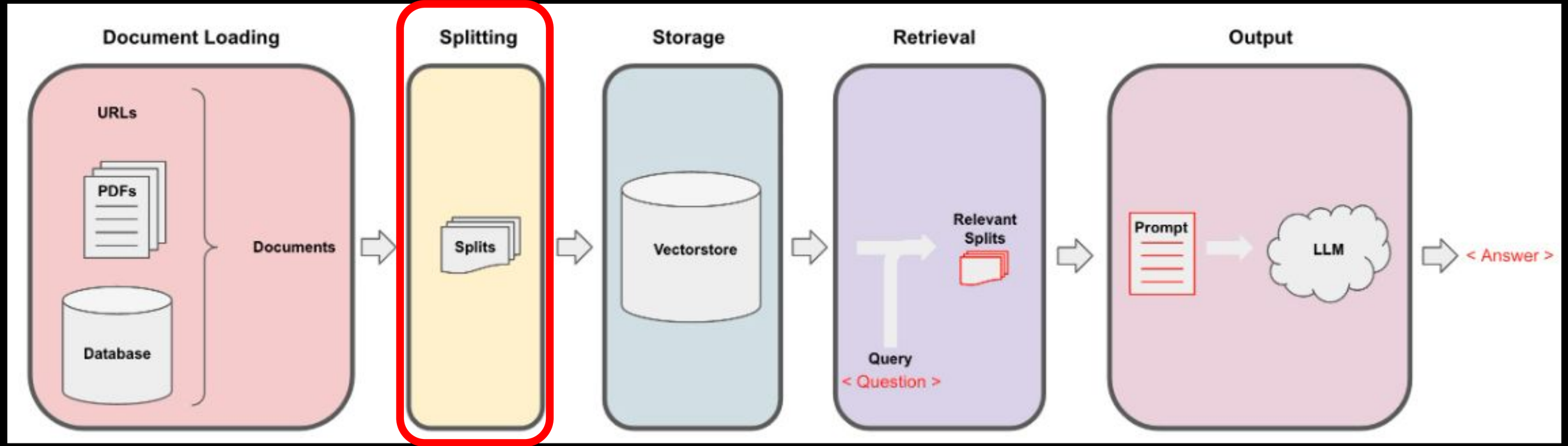
- **metadata**: A dictionary containing metadata information about the document.
- **text**: The text content of the document.
- **pages**: A list of langchain.Page objects, where each page represents a page of the document.
- **images**: A list of langchain.Image objects, where each image represents an image extracted from the document.

QUICK RECAP: Document Loaders

Data Source	Loader
PDF	PyPDFLoader
Websites	WebBaseLoader
Notion	NotionDirectoryLoader
Youtube Videos	GenericLoader + YoutubeAudioLoader + OpenAIWhisperParser
and many more	…

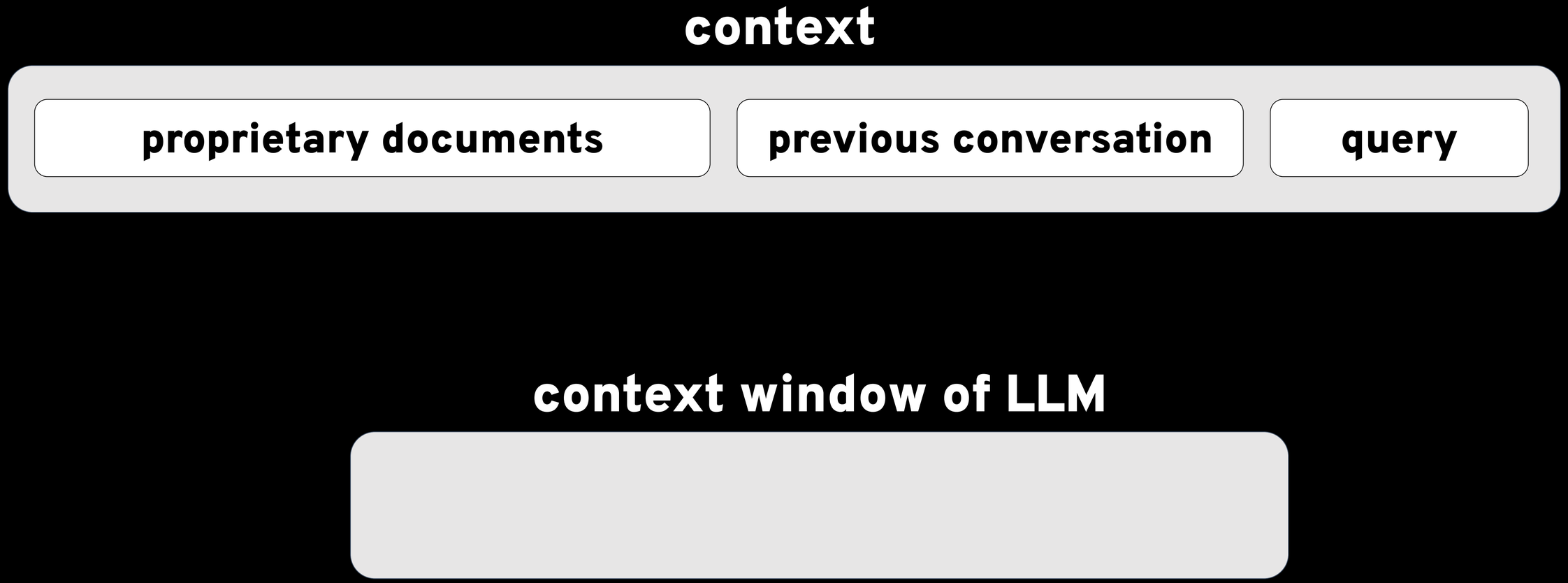
List of all Loaders:
https://python.langchain.com/docs/integrations/document_loaders

QUICK RECAP: From Docs to Answers



QUICK RECAP: Splitting

Why split?



QUICK RECAP: Splitting

How can we split without messing up the content?

...
on this model. The Toyota Camry has a head-snapping
80 HP and an eight-speed automatic transmission that will
...

Chunk 1: on this model. The Toyota Camry has a head-snapping

Chunk 2: 80 HP and an eight-speed automatic transmission that will

Question: What are the specifications on the Camry?

QUICK RECAP: Splitting

Use appropriate chunk sizes and overlap!

QUICK RECAP: Splitting

example: CharacterTextSplitter

```
langchain.text_splitter.CharacterTextSplitter(
    separator: str = "\n\n"
    chunk_size=4000,
    chunk_overlap=200,
    length_function=<builtin function len>,
)
```

Methods:

create_documents() - Create documents from a list of texts.

split_documents() - Split documents.

QUICK RECAP: Splitting

What is a good breakpoint?

- characters
- tokens
- special characters “.”
- line breaks \n
- markup headers ###
- …

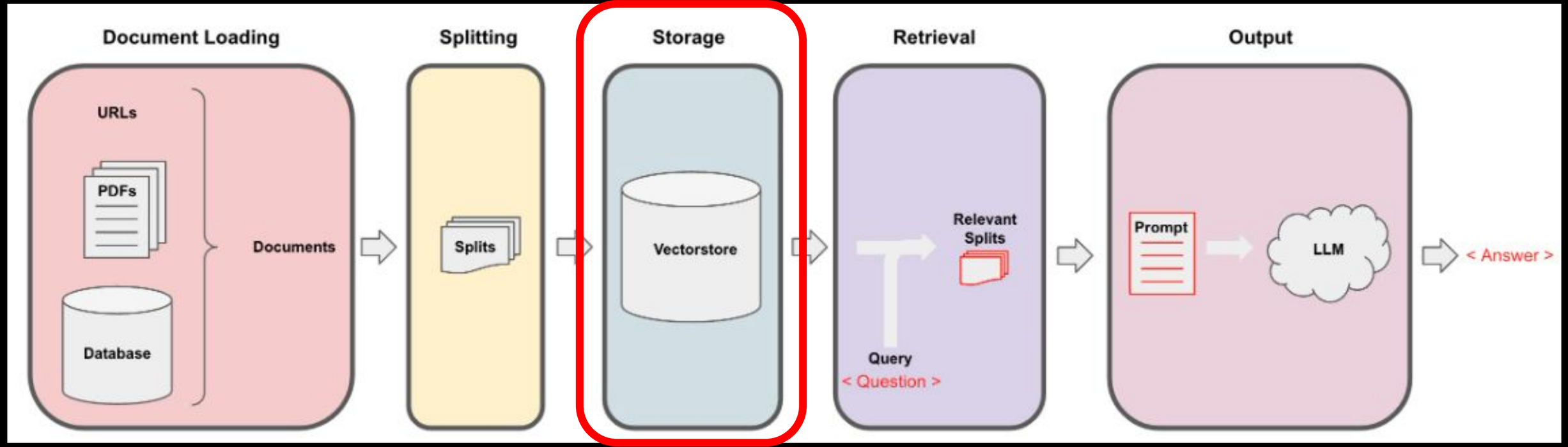
QUICK RECAP: Splitting

langchain.text_splitter.

- **CharacterTextSplitter()**- Implementation of splitting text that looks at characters.
- **MarkdownHeaderTextSplitter**() - Implementation of splitting markdown files based on specified headers.
- **TokenTextSplitter**() - Implementation of splitting text that looks at tokens.
- **SentenceTransformersTokenTextSplitter()** - Implementation of splitting text that looks at tokens.
- ***RecursiveCharacterTextSplitter()*** **- Implementation of splitting text that looks at characters. Recursively tries to split by different characters to find one that works.**
- **Language()** – for CPP, Python, Ruby, Markdown etc
- **NLTKTextSplitter**() - Implementation of splitting text that looks at sentences using NLTK (Natural Language Tool Kit)
- **SpacyTextSplitter() -** Implementation of splitting text that looks at sentences using Spacy

https://python.langchain.com/docs/modules/data_connection/document_transformers

QUICK RECAP: From Docs to Answers



QUICK RECAP: Embeddings

- Embeddings are high-dimensional vectors that store semantic information of the embedded data
- Embeddings can be used to map words, texts, images, audio, etc. to a vector space
  - Similar embedding vectors are in a similar position in the vector space
- Similarity of embeddings can be measured by using a similarity function (e.g. dot product, cosine similarity, etc.)

QUICK RECAP: Embeddings

- Dot product: Calculate the dot product of two vectors
  - Large dot product means high similarity
  - Small or negative dot product means low similarity

- Cosine similarity: Similar to the dot product but only in the range [-1, 1].
  - Cosine similarity close to 1 → very high similarity
  - Cosine similarity close to -1 → very low similarity

QUICK RECAP: Vector Stores

- Database specilized to store and retrieve embeddings
  - Documents are stored along with a vector (embedding) and metadata
  - Usually a much faster retrieval than traditional databases
- Most important part of a Retrieval Augmented Generation pipeline
- Different possibilities of data store:
  - In-memory
  - Persistent storage in a file
  - Persistent storage in a hosted database
- A vector store can have multiple indices to separate documents

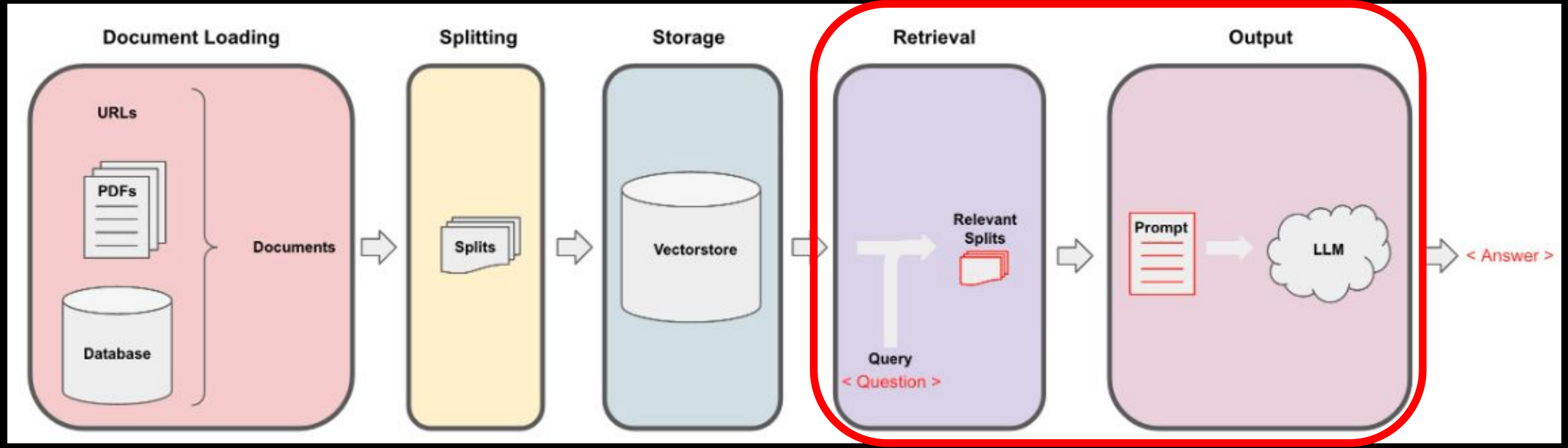
QUICK RECAP: Vector Stores

- Qdrant
  - Open-source vector database by a startup from Berlin
  - Many additional features like a recommendation API
  - Used by X to retrieve similar posts and current information inside Grok
- Weaviate
  - Open-source vector database from the Netherlands
  - Offer a free cloud hosted version (has to be renewed every two weeks)
- Pinecone
  - Proprietary vector database
  - Offer a free cloud hosted plan
  - Used in a lot of tutorials

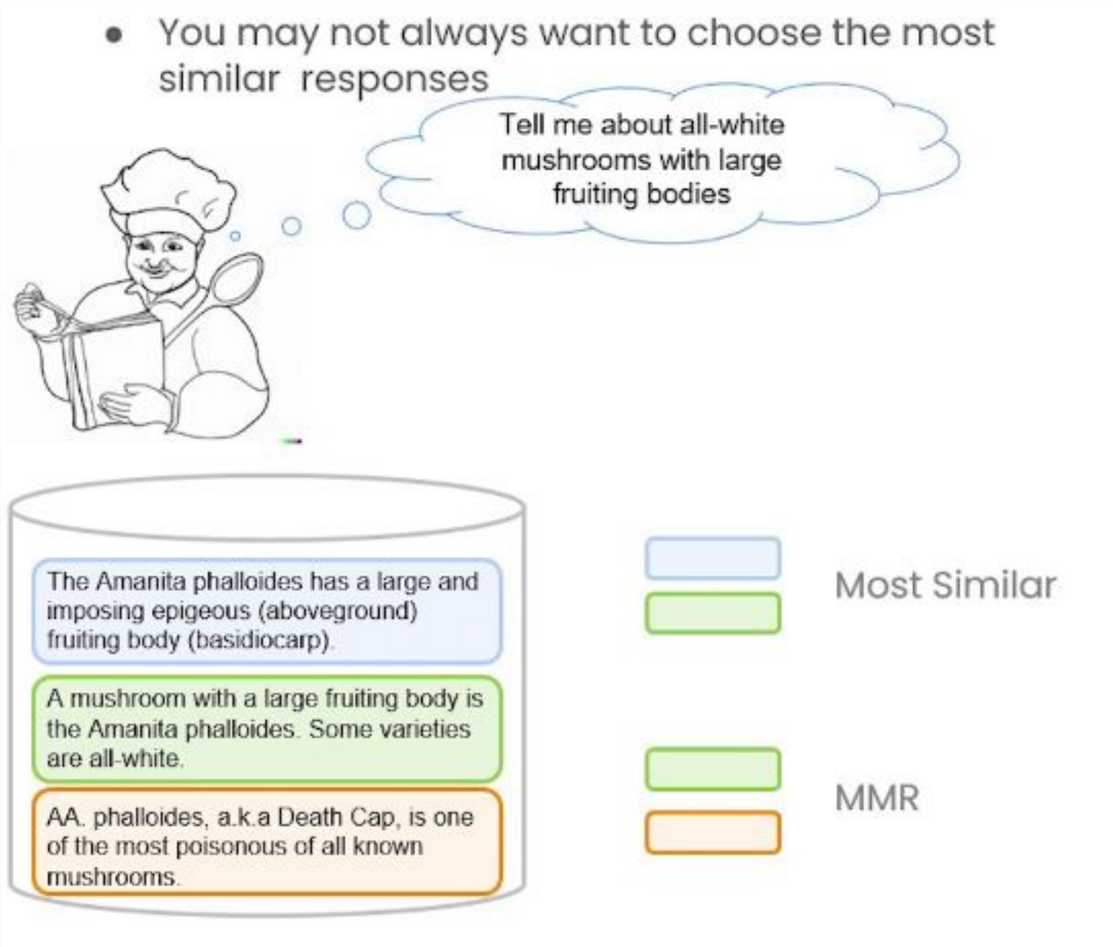
QUICK RECAP: Vector Stores

- Other dedicated vector databases
  - Milvus (open-source)
  - ChromaDB (open-source)
  - and many more
- Other databases with vector store capabilities
  - Elasticsearch
  - PostgreSQL
  - MongoDB
  - SupaBase
  - and many more

QUICK RECAP: From Docs to Answers



QUICK RECAP: Retrieval SimSearch vs. MMR




```
texts = [
    """The Amanita phalloides has a large and imposing epigeous (aboveground)
    """A mushroom with a large fruiting body is the Amanita phalloides. Some va
    """A. phalloides, a.k.a Death Cap, is one of the most poisonous of all know
]
```

```
smalldb = Chroma.from_texts(texts, embedding=embedding)
```

```
question = "Tell me about all-white mushrooms with large fruiting bodies"
```

```
smalldb.similarity_search(question, k=2)
```

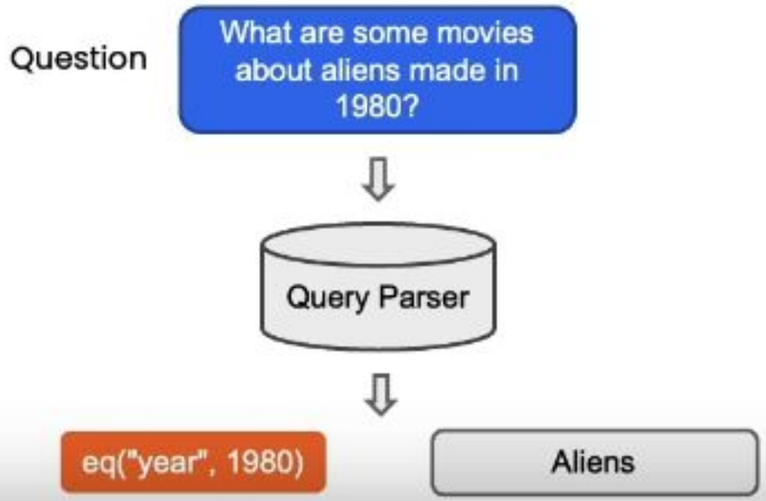
```
[Document(page_content='A mushroom with a large fruiting body is the Amanita p
halloides. Some varieties are all-white.', metadata={}),
 Document(page_content='The Amanita phalloides has a large and imposing epigeo
us (aboveground) fruiting body (basidiocarp).', metadata={})]
```

```
smalldb.max_marginal_relevance_search(question,k=2, fetch_k=3)
```

```
[Document(page_content='A mushroom with a large fruiting body is the Amanita p
halloides. Some varieties are all-white.', metadata={}),
 Document(page_content='A. phalloides, a.k.a Death Cap, is one of the most poi
sonous of all known mushrooms.', metadata={})]
```

QUICK RECAP: meta data

- There are several situations where the Query applied to the DB is more than just the Question asked.
- One is SelfQuery, where we use an LLM to convert the user question into a query




```
question = "what did they say about regression in the third lecture?"
```

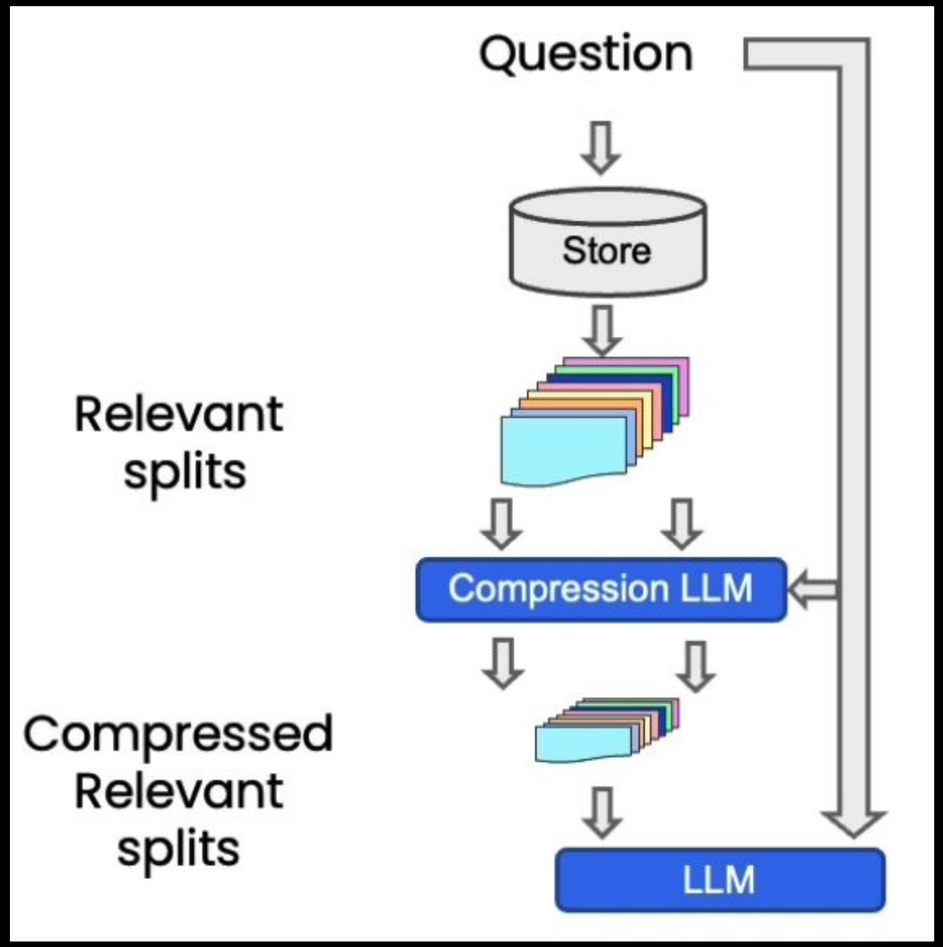
```
docs = vectordb.similarity_search(
    question,
    k=3,
    filter={"source":"docs/cs229_lectures/MachineLearning-Lecture03.pdf"})
```

```
for d in docs:
    print(d.metadata)
```

```
{'source': 'docs/cs229_lectures/MachineLearning-Lecture03.pdf', 'page': 0}
{'source': 'docs/cs229_lectures/MachineLearning-Lecture03.pdf', 'page': 14}
{'source': 'docs/cs229_lectures/MachineLearning-Lecture03.pdf', 'page': 4}
```

```
from langchain.llms import OpenAI
from langchain.retrievers.self_query.base import SelfQueryRetriever
from langchain.chains.query_constructor.base import AttributeInfo
```

QUICK RECAP: Compression (with MMR)




```python
from langchain.retrievers import ContextualCompressionRetriever
from langchain.retrievers.document_compressors import LLMChainExtractor
```

```python
def pretty_print_docs(docs):
    print(f"\n{'-' * 100}\n".join([f"Document {i+1}:\n\n" + d.page_content for
```

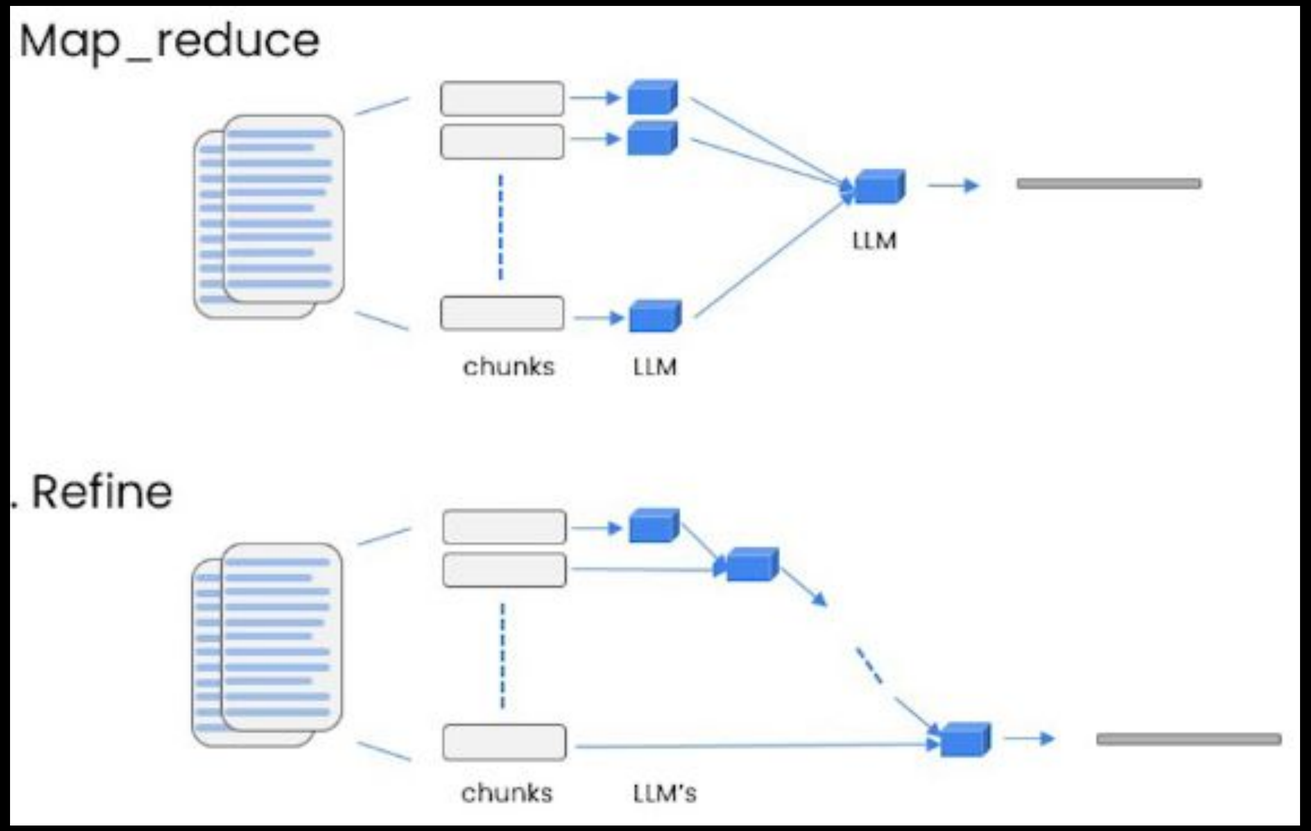
```python
llm = OpenAI(temperature=0)
compressor = LLMChainExtractor.from_llm(llm)
```

```python
compression_retriever = ContextualCompressionRetriever(
    base_compressor=compressor,
    base_retriever=vectordb.as_retriever()
)
```

```python
compression_retriever = ContextualCompressionRetriever(
    base_compressor=compressor,
    base_retriever=vectordb.as_retriever(search_type="mmr")
)
```

QUICK RECAP -

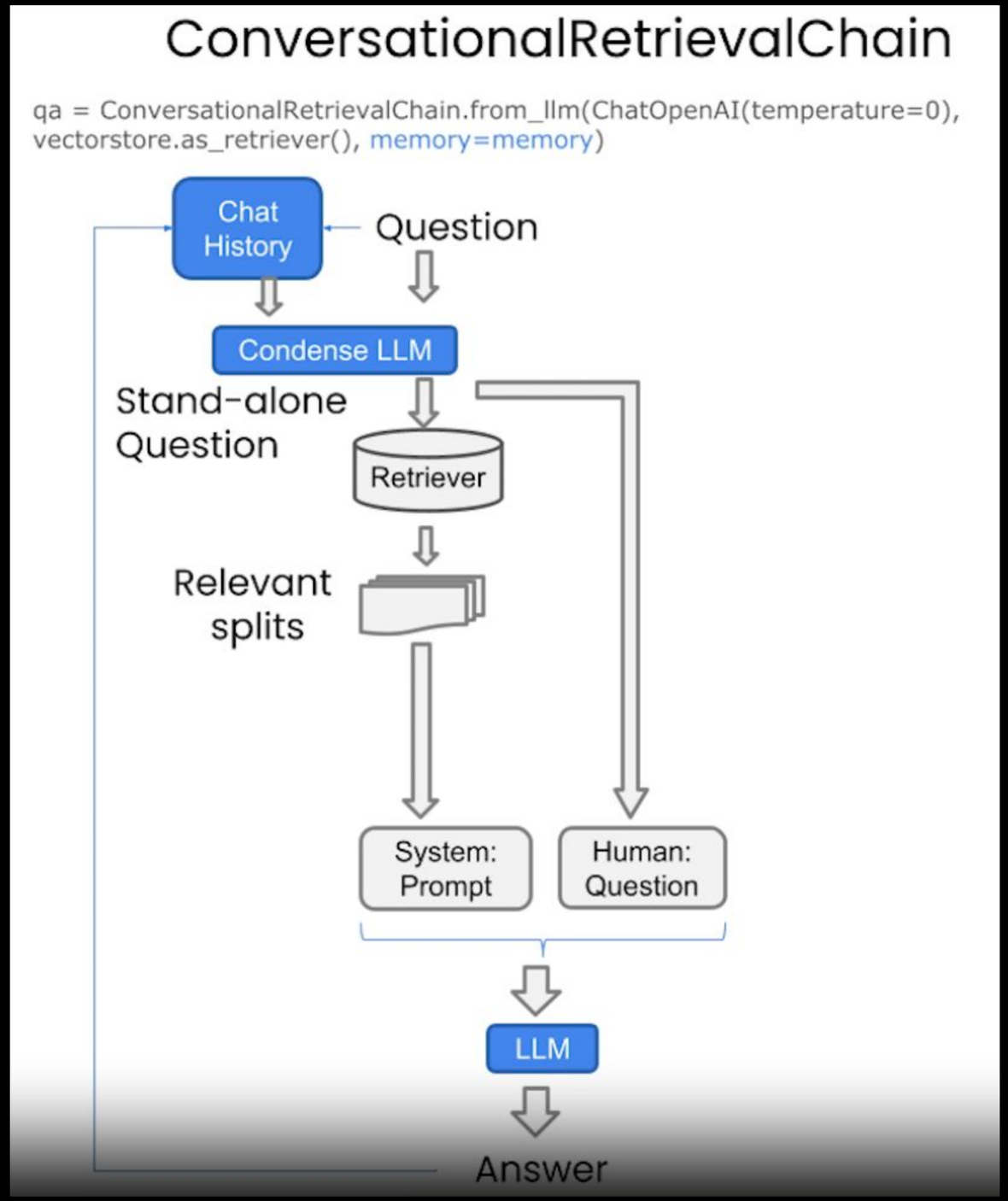
```python
from langchain.chains import RetrievalQA
```



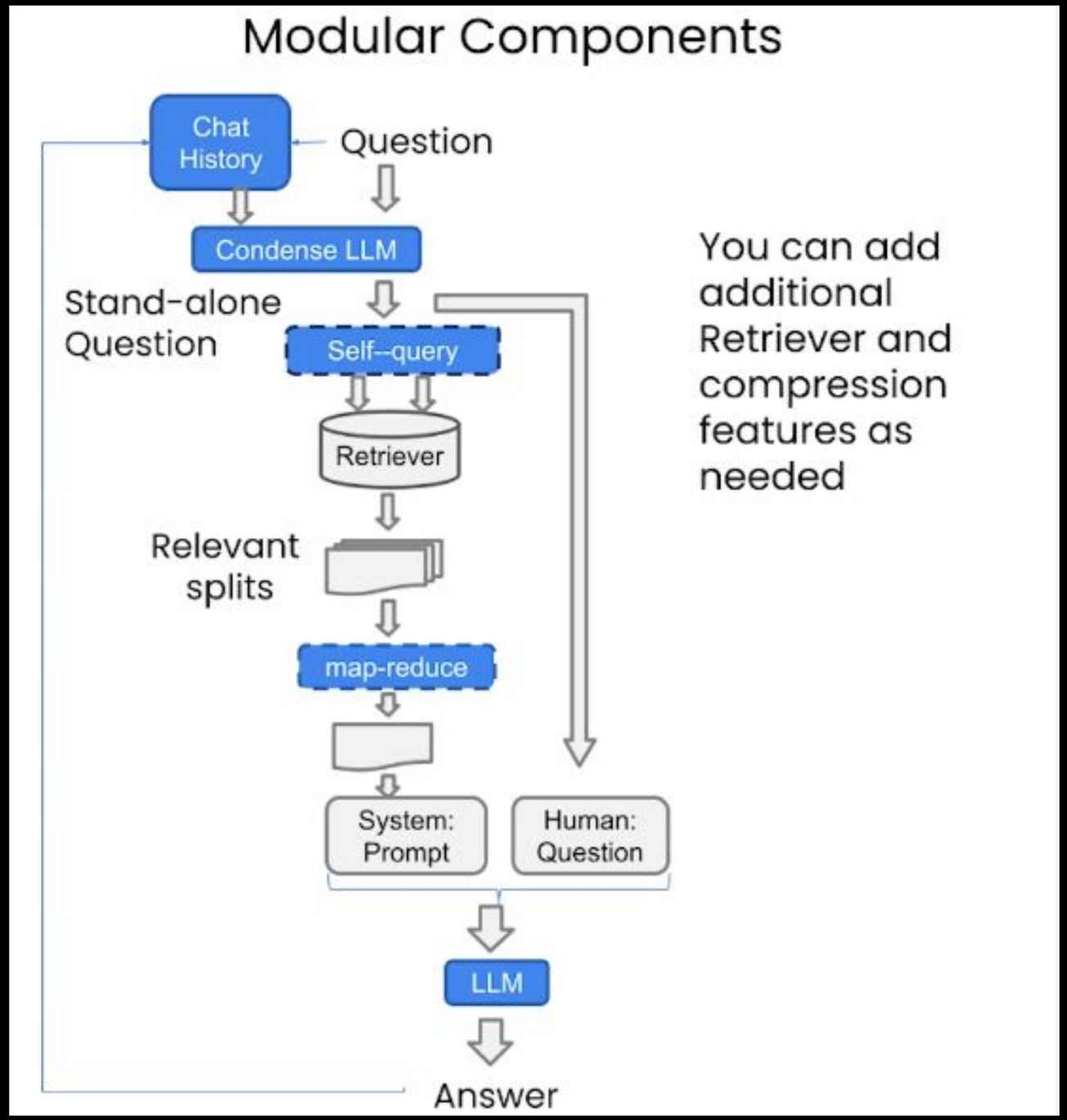
LangSmith

```python
import os
os.environ["LANGCHAIN_TRACING_V2"] = "true"
os.environ["LANGCHAIN_ENDPOINT"] = "https://api.langchain.plus"
os.environ["LANGCHAIN_API_KEY"] = LCP_API_KEY
```

QUICK RECAP - Memory



QUICK RECAP - Memory



TODAY’S SCHEDULE

- Quiz
- Homework presentation
- Short Recap
- Projects
- Homework for next week

Projects

Private and local chatbot
- Christiane Laging
- Dirk Brockhausen
- **Kristian Boroz**
- Khanh Ho
- **Joanne Loader**
- **Luca Palmieri**

Interpret financial data
- Frederik Brinkmann
- Patrick Wagner
- **Kristian Boroz**

Stoic Companion
- **Joanne Loader**
- Jonas Gutzke
- **Benjamin Müller**
- Sandra Schulze
- Adrian Klabunde

LISA
- Anna Dahlhaus
- Anna-Lena Hansen
- Yinghan Zhao

Automation helper for literature review
- Kaan Apaydin
- **Dikshyant Acharya**
- **Luca Palmieri**
- Stefan David Horak
- Yorck Zisgen

Extract chain of events from documents
- Abdullah Al Amin
- Mithun Das
- Sarker Miraz Mahfuz

Language Learning App
- **Benjamin Müller**

Private GPT
- **Dikshyant Acharya**

Breakout rooms

1. Discuss in your project group **what exactly the project should look like**! Use the template for this!
2. Develop a battle plan in your project group on **how you can develop a very rough prototype with LangFlow** by next week. Who does what?

Breakout rooms

Discuss in your project group what exactly the project should look like!
Use the template for this!

1. Working Title: For example, "Chat with Excel"
2. Data Sources: For example, Excel tables or text-based tables.
3. Functions: For example, users can pose questions and receive graphical and statistical reports in response.
4. Use Cases: For instance, a layperson can quickly analyze large datasets.
5. Challenges: For example, generating correctly formatted Python code for graphics and statistical analysis.
6. People interested: Horst & Elvira

Breakout rooms

Develop a battle plan in your project group on how you can develop a very rough prototype with LangFlow by next week. Who does what?

LangFlow:

https://github.com/logspace-ai/langflow

https://www.youtube.com/watch?v=KJ-ux3hre4s

Alternative 1: Flowise

https://github.com/FlowiseAI/Flowise

Alternative 2: GPTs (ChatGPT Plus required)

https://chat.openai.com/

TODAY’S SCHEDULE

- Quiz
- Homework presentation
- Short Recap
- Projects
- Homework for next week

Homework Until Next Week

1. **Watch the course “Large Language Models with Semantic Search”**
2. **Prepare a short presentation (max 5 min per group) with 2 parts:**
    - Present your project outline using the template (or improvise)
    - Present a rough prototype (or mockup) using LangFlow

Homework Until Next Week

Project Template

1. Working Title: For example, "Chat with Excel"
2. Data Sources: For example, Excel tables or text-based tables.
3. Functions: For example, users can pose questions and receive graphical and statistical reports in response.
4. Use Cases: For instance, a layperson can quickly analyze large datasets.
5. Challenges: For example, generating correctly formatted Python code for graphics and statistical analysis.
6. People interested: Horst & Elvira

Homework Until Next Week

Use one of the available no-code tools (or, if you prefer, LangChain) to test the feasibility of your idea by creating a first small prototype.

LangFlow: https://github.com/logspace-ai/langflow
Flowise: https://github.com/FlowiseAI/Flowise
GPTs: https://openai.com/blog/introducing-gpts