

# The LangChain ecosystem

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN



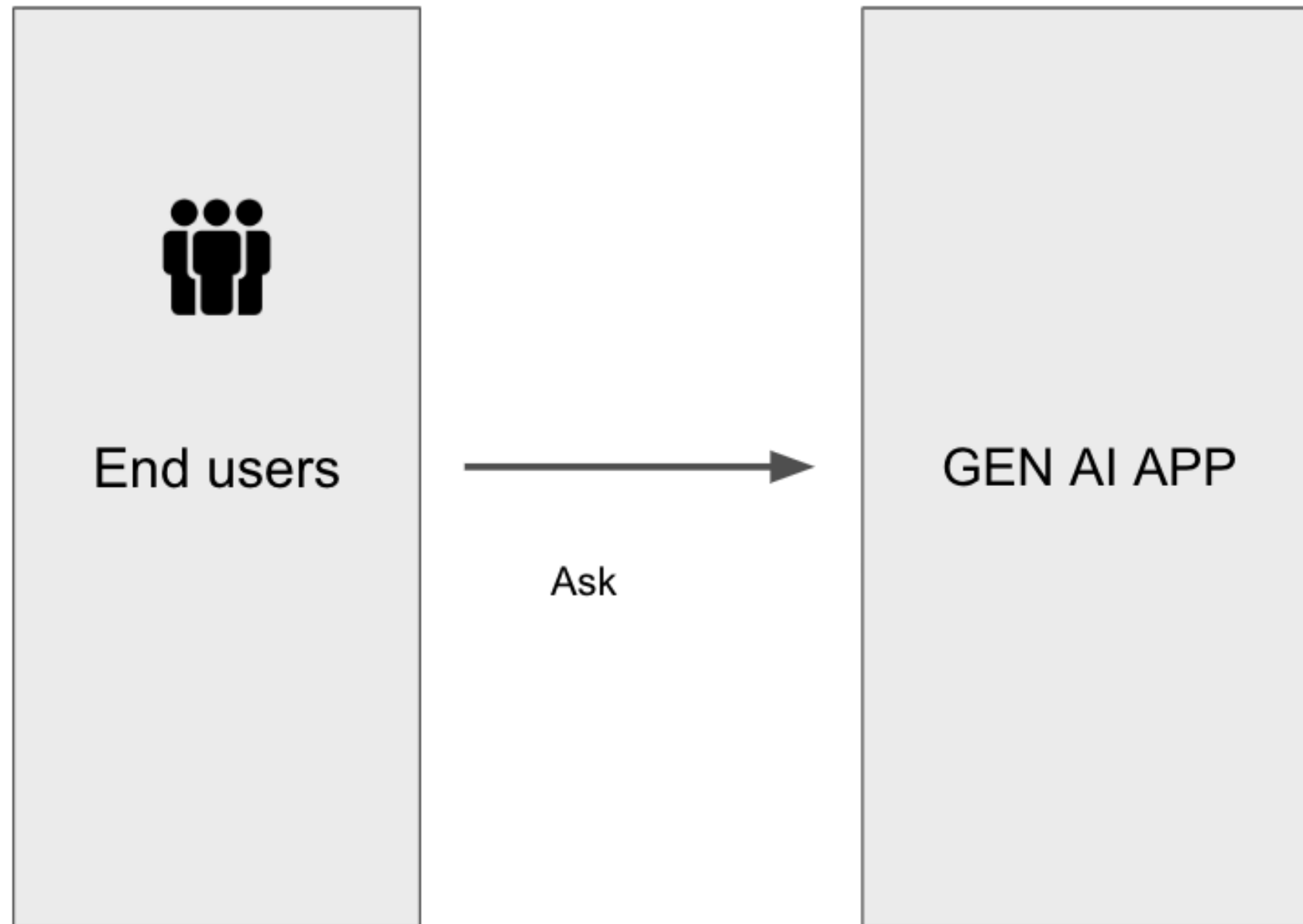
**Jonathan Bennion**

AI Engineer & LangChain Contributor

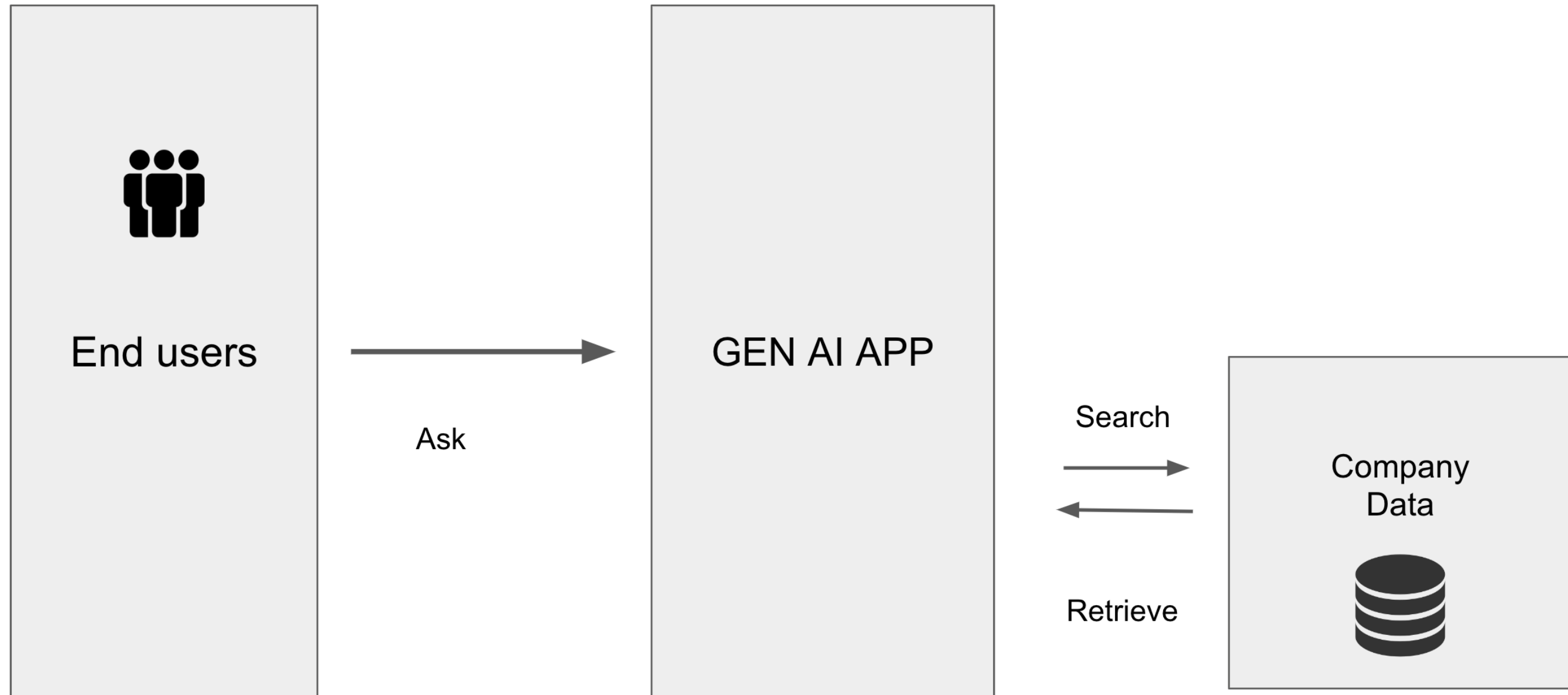
# Retrieval Augmented Generation (RAG)



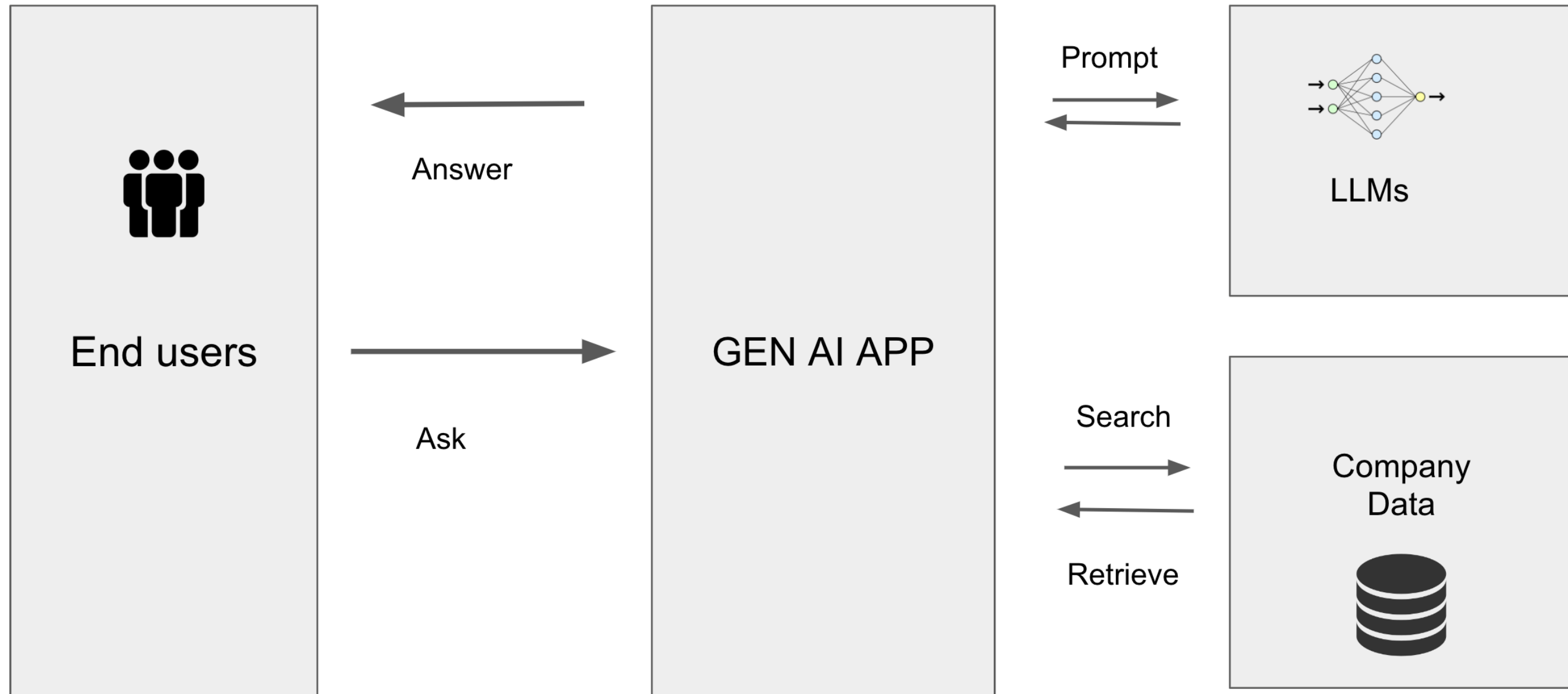
# Retrieval Augmented Generation (RAG)



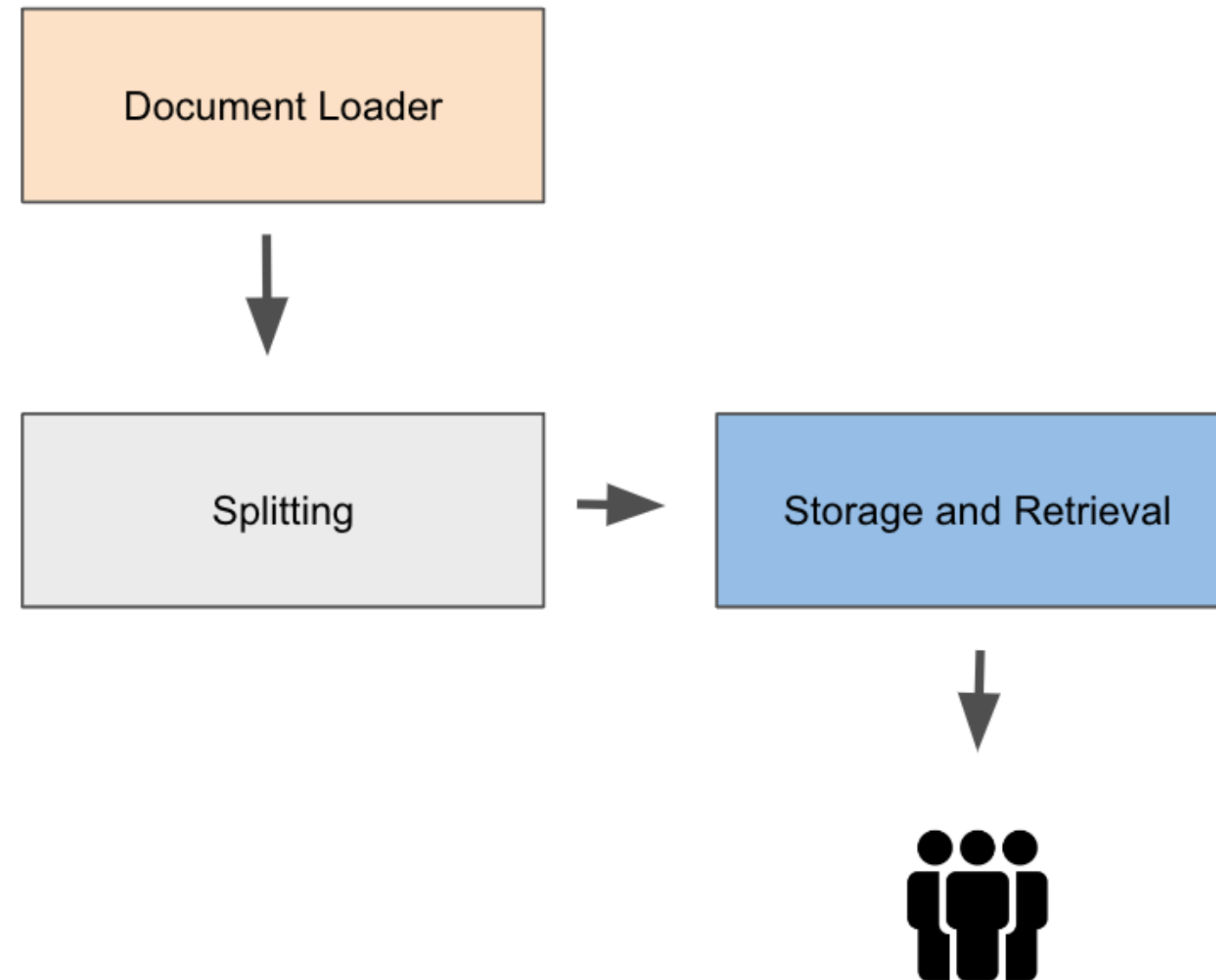
# Retrieval Augmented Generation (RAG)



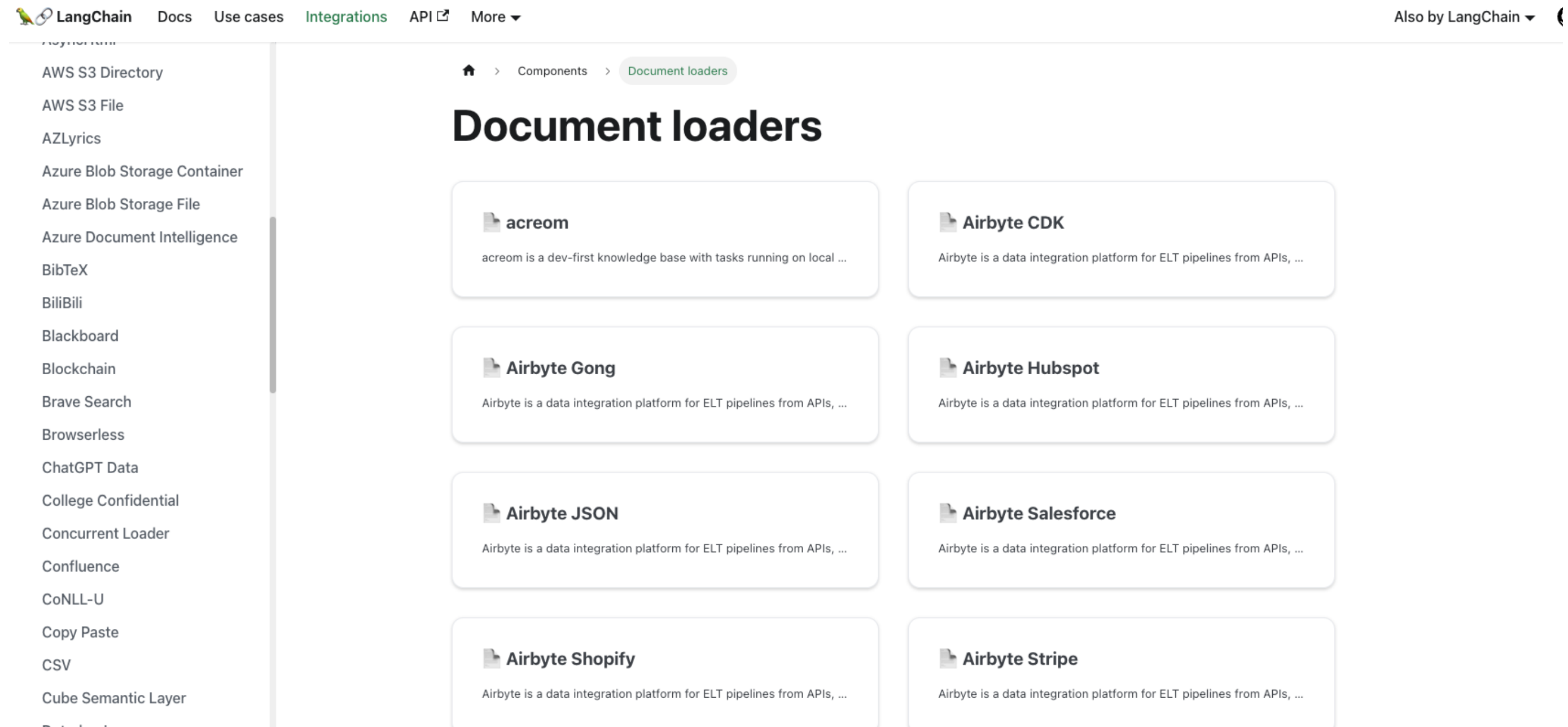
# Retrieval Augmented Generation (RAG)



# RAG development steps are threefold



# Document loaders in LangChain




The screenshot shows the LangChain website's 'Integrations' page, specifically the 'Document loaders' section. The page features a sidebar with a list of integrations and a main content area with a grid of loader cards. The top navigation bar includes links for 'LangChain', 'Docs', 'Use cases', 'Integrations' (highlighted), 'API', and 'More'. The sidebar lists various integrations such as 'AWS S3 Directory', 'AWS S3 File', 'AZLyrics', 'Azure Blob Storage Container', 'Azure Blob Storage File', 'Azure Document Intelligence', 'BibTeX', 'Bilibili', 'Blackboard', 'Blockchain', 'Brave Search', 'Browserless', 'ChatGPT Data', 'College Confidential', 'Concurrent Loader', 'Confluence', 'CoNLL-U', 'Copy Paste', 'CSV', and 'Cube Semantic Layer'. The main content area has a breadcrumb trail: 'Home > Components > Document loaders'. The title 'Document loaders' is prominently displayed. Below it, there is a grid of eight loader cards, each with an icon, a title, and a brief description. The loaders are: 'acreom' (a dev-first knowledge base), 'Airbyte CDK' (a data integration platform for ELT pipelines), 'Airbyte Gong' (a data integration platform for ELT pipelines), 'Airbyte Hubspot' (a data integration platform for ELT pipelines), 'Airbyte JSON' (a data integration platform for ELT pipelines), 'Airbyte Salesforce' (a data integration platform for ELT pipelines), 'Airbyte Shopify' (a data integration platform for ELT pipelines), and 'Airbyte Stripe' (a data integration platform for ELT pipelines).


LangChain Docs Use cases Integrations API More


Also by LangChain


Home > Components > Document loaders


## Document loaders


**acreom**  
acreom is a dev-first knowledge base with tasks running on local ...


**Airbyte CDK**  
Airbyte is a data integration platform for ELT pipelines from APIs, ...


**Airbyte Gong**  
Airbyte is a data integration platform for ELT pipelines from APIs, ...

**Airbyte Hubspot**  
Airbyte is a data integration platform for ELT pipelines from APIs, ...

**Airbyte JSON**  
Airbyte is a data integration platform for ELT pipelines from APIs, ...

**Airbyte Salesforce**  
Airbyte is a data integration platform for ELT pipelines from APIs, ...

**Airbyte Shopify**  
Airbyte is a data integration platform for ELT pipelines from APIs, ...

**Airbyte Stripe**  
Airbyte is a data integration platform for ELT pipelines from APIs, ...

<sup>1</sup> [https://python.langchain.com/docs/integrations/document\\_loaders](https://python.langchain.com/docs/integrations/document_loaders)

# PDF document loader

```
from langchain_community.document_loaders import PyPDFLoader
```

- Requires installation of the `pypdf` package

```
loader = PyPDFLoader("attention_is_all_you_need.pdf")  
  
data = loader.load()  
print(data[0])
```

```
Document(page_content='Provided proper attribution is provided, Google hereby grants  
permission to\nreproduce the tables and figures in this paper solely for use in [...]
```



# CSV document loader

```
from langchain_community.document_loaders.csv_loader import CSVLoader

loader = CSVLoader(file_path='fifa_countries_audience.csv')

data = loader.load()
print(data[0])
```

```
Document(page_content='country: United States\nconfederation: CONCACAF\npopulation_share: [...]
```

# Third-party document loader

```
from langchain_community.document_loaders import HNLoader

loader = HNLoader("https://news.ycombinator.com")
data = loader.load()

print(data[0])
print(data[0].metadata)
```

```
Document(page_content='Nrsc5: Receive NRSC-5 digital radio stations [...]\n{'source': 'https://news.ycombinator.com',\n 'title': 'Nrsc5: Receive NRSC-5 digital
```

# Let's practice!

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN

# Splitting external data for retrieval

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN



**Jonathan Bennion**

AI Engineer & LangChain Contributor

# What is document splitting and why is it needed?

## 1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Line 1:

```
Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks
```

Line 2:

```
in particular, have been firmly established as state of the art approaches in sequence modeling and
```

# What is the best document splitting strategy?

---



Methods include:

1. `CharacterTextSplitter`
2. `RecursiveCharacterTextSplitter`
3. Many others

<sup>1</sup> Wikipedia Commons

# Concept of the chunk overlap

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

# Example text for chunk size comparison

```
quote = 'One machine can do the work of fifty ordinary humans.  
No machine can do the work of one extraordinary human.'
```

```
len(quote)
```

```
103
```

```
chunk_size = 24  
chunk_overlap = 3
```

<sup>1</sup> Elbert Hubbard



# CharacterTextSplitter to split documents

```
from langchain.text_splitter import CharacterTextSplitter

ct_splitter = CharacterTextSplitter(
    separator='.',
    chunk_size=chunk_size,
    chunk_overlap=chunk_overlap)

docs = ct_splitter.split_text(text)
print(docs)
```

```
['One machine can do the work of fifty ordinary humans.'
 'No machine can do the work of one extraordinary human.']
```

# RecursiveCharacterTextSplitter

```
from langchain.text_splitter import RecursiveCharacterTextSplitter

rc_splitter = RecursiveCharacterTextSplitter(
    chunk_size=chunk_size,
    chunk_overlap=chunk_overlap)

docs = rc_splitter.split_text(quote)
print(docs)
```

```
['One machine can do the',
 'work of fifty ordinary',
 'humans. No machine can do',
 'do the work of one',
 'extraordinary human.']
```

# RecursiveCharacterTextSplitter with HTML

```
from langchain_community.document_loaders import UnstructuredHTMLLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter

loader = UnstructuredHTMLLoader("white_house_executive_order_nov_2023.html")
data = loader.load()

rc_splitter = RecursiveCharacterTextSplitter(
    chunk_size=chunk_size,
    chunk_overlap=chunk_overlap,
    separators=['.'])

docs = rc_splitter.split_documents(data)
print(docs[0])
```

```
Document(page_content="To search this site, enter a search term [...]
```

# Let's practice!

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN

# RAG storage and retrieval using vector databases

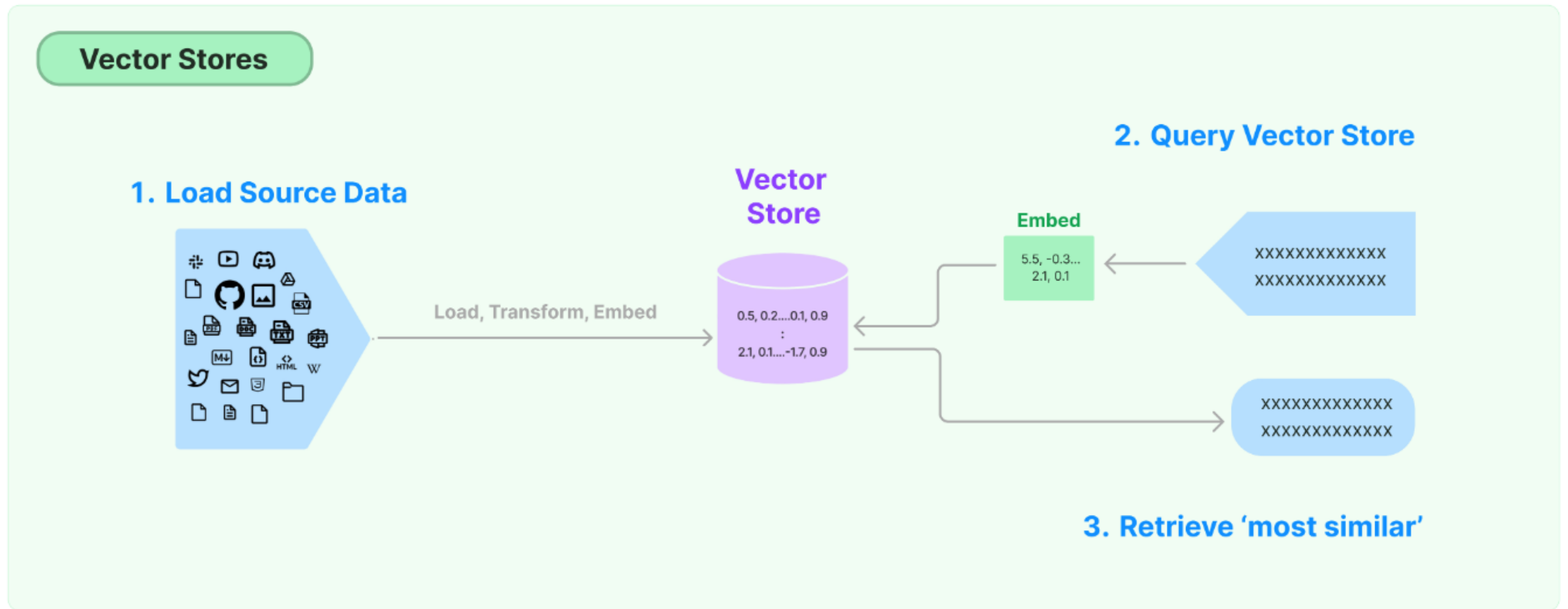
DEVELOPING LLM APPLICATIONS WITH LANGCHAIN



**Jonathan Bennion**

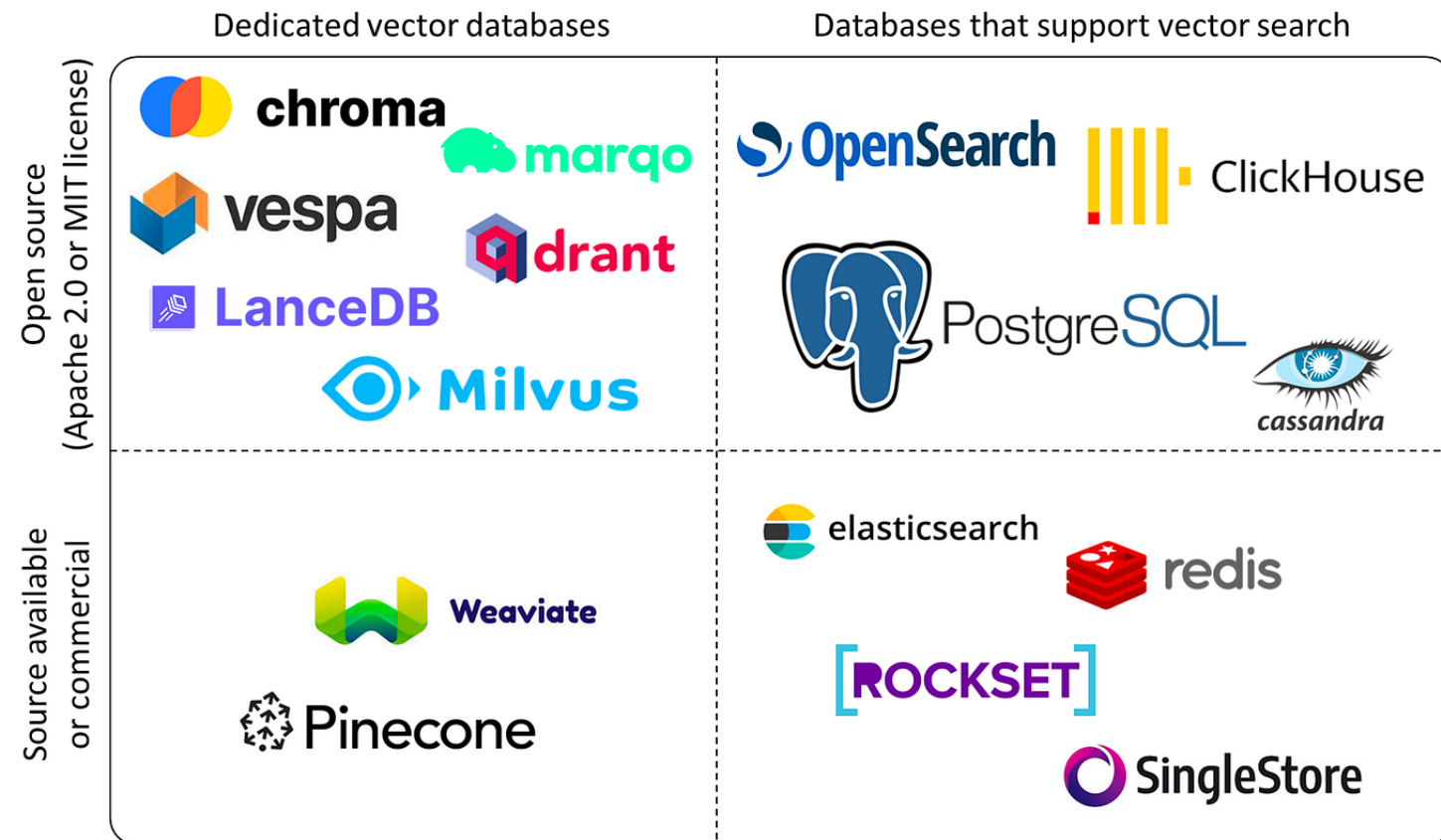
AI Engineer & LangChain Contributor

# What is a vector database and why do I need it?



<sup>1</sup> LangChain

# Which vector database should I use?



Need to consider:

- Open source vs. closed source (license)
- Cloud vs. on-premises
- Lightweight vs. powerful for large filestores

<sup>1</sup> Image Credit: Yingjun Wu

# Preparing the data for storage

```
quote = 'There is a kingdom of lychee fruit that are alive and thriving in Iceland, but they feel taken advantage of and are not fast enough for you.'
```

```
from langchain.text_splitter import RecursiveCharacterTextSplitter
```

```
chunk_size = 40
```

```
chunk_overlap = 10
```

```
splitter = RecursiveCharacterTextSplitter(  
    chunk_size=chunk_size,  
    chunk_overlap=chunk_overlap)
```

```
docs = splitter.split_text(quote)
```



# Choosing an embeddings model

- Hugging Face embeddings models requires `transformers` library

The screenshot shows the Hugging Face website interface. At the top, there's a navigation bar with the Hugging Face logo, a search bar, and links to Models, Datasets, Spaces, Docs, Solutions, and Pricing. A yellow banner below the navigation bar promotes joining an organization. The main content area is divided into a left sidebar and a main grid of model cards.

**Left Sidebar:**

- Tasks:** Libraries, Datasets, Languages, Licenses, Other. A search bar "Filter Tasks by name" is present.
- Multimodal:** Feature Extraction, Text-to-Image, Image-to-Text, Text-to-Video, Visual Question Answering, Document Question Answering, Graph Machine Learning.
- Computer Vision:** Depth Estimation, Image Classification, Object Detection, Image Segmentation, Image-to-Image, Unconditional Image Generation, Video Classification, Zero-Shot Image Classification.
- Natural Language Processing:** Text Classification, Token Classification.

**Main Grid (Models 476, embeddings filter):**

- jinaai/jina-embeddings-v2-base-en:** Feature Extraction • Updated 4 days ago • 61.5k • 436
- jinaai/jina-embeddings-v2-small-en:** Feature Extraction • Updated 4 days ago • 43.1k • 65
- dranzerstar/SD-textual-inversion-embeddings-repo:** Updated about 7 hours ago • 110
- NeuML/pubmedbert-base-embeddings:** Sentence Similarity • Updated 26 days ago • 934 • 17
- consciousAI/cai-lunaris-text-embeddings:** Sentence Similarity • Updated Jun 22 • 3.68k • 1
- Xenova/jina-embeddings-v2-base-en:** Feature Extraction • Updated 17 days ago • 6 • 5
- BogdanKuloren/continual-learning-paper-embeddings-m...:** Feature Extraction • Updated Aug 1, 2021 • 5 • 1
- DataikuNLP/average\_word\_embeddings\_glove.6B.300d:** Sentence Similarity • Updated Sep 1, 2021 • 1
- GroNLP/gpt2-medium-dutch-embeddings:** Text Generation • Updated Sep 11 • 66 • 1
- GroNLP/gpt2-medium-italian-embeddings:** Text Generation • Updated Sep 11 • 401 • 1
- GroNLP/gpt2-small-dutch-embeddings:** Text Generation • Updated Sep 11 • 40
- GroNLP/gpt2-small-italian-embeddings:** Text Generation • Updated Jun 2 • 76

# Setting up a Chroma vector database

```
from langchain_openai import OpenAIEmbeddings
from langchain_community.vectorstores import Chroma

embedding_function = OpenAIEmbeddings(openai_api_key=openai_api_key)

vectordb = Chroma(
    persist_directory=persist_directory,
    embedding_function=embedding_function)

vectordb.persist()

docstorage = Chroma.from_texts(docs, embedding_function)
```

# RAG generation using a vector database

```
from langchain.chains import RetrievalQA

qa = RetrievalQA.from_chain_type(llm=OpenAI(model_name="gpt-3.5-turbo-instruct", openai_api_key=openai_ap
chain_type="stuff", retriever=docstorage.as_retriever())

query = "Where do lychee fruit live?"
print(qa.run(query))
```

```
['Lychee fruit live in kingdoms.']
```

```
quote = 'There is a kingdom of lychee fruit that are alive and thriving in Iceland, but they feel
taken advantage of and are not fast enough for you.'
```

# Returning references to RAG data sources

```
from langchain.chains import RetrievalQAWithSourcesChain

qa = RetrievalQAWithSourcesChain.from_chain_type(
    OpenAI(temperature=0, openai_api_key=openai_api_key), chain_type="stuff",
    retriever=docstorage.as_retriever())

results = qa({"question": "what is the primary architecture presented in the document?"},
    return_only_outputs=True)
print(results)
```

```
{'answer': ' The primary architecture presented in the document is the Transformer model architecture.
\n', 'sources': 'attention_is_all_you_need.pdf'}
```

# Let's practice!

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN