Loading Documents for RAG with LangChain

RETRIEVAL AUGMENTED GENERATION (RAG) WITH LANGCHAIN

Meri Nova
Machine Learning Engineer

datacamp

Meet your instructor...

Meri Nova

- Founder at **Break Into Data**
- Machine Learning Engineer
- Content Creator on Linkedin and YouTube

Retrieval Augmented Generation (RAG)

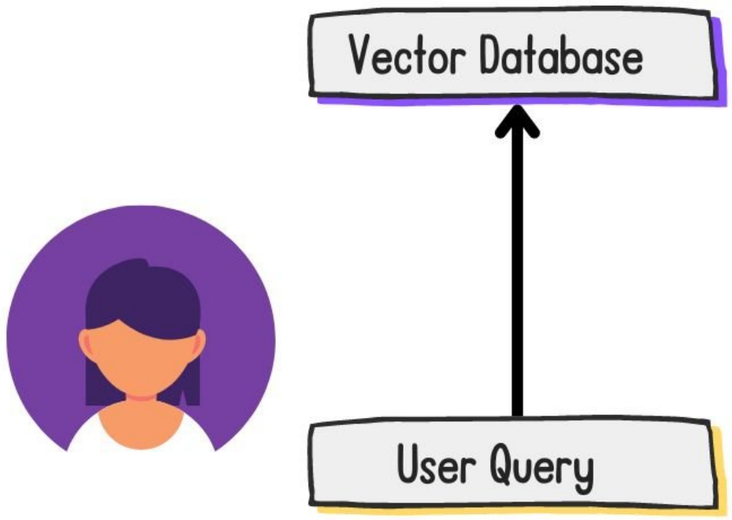
- **LLM Limitation:** knowledge constraints

→ ***RAG:*** *Integrating external data with LLMs*

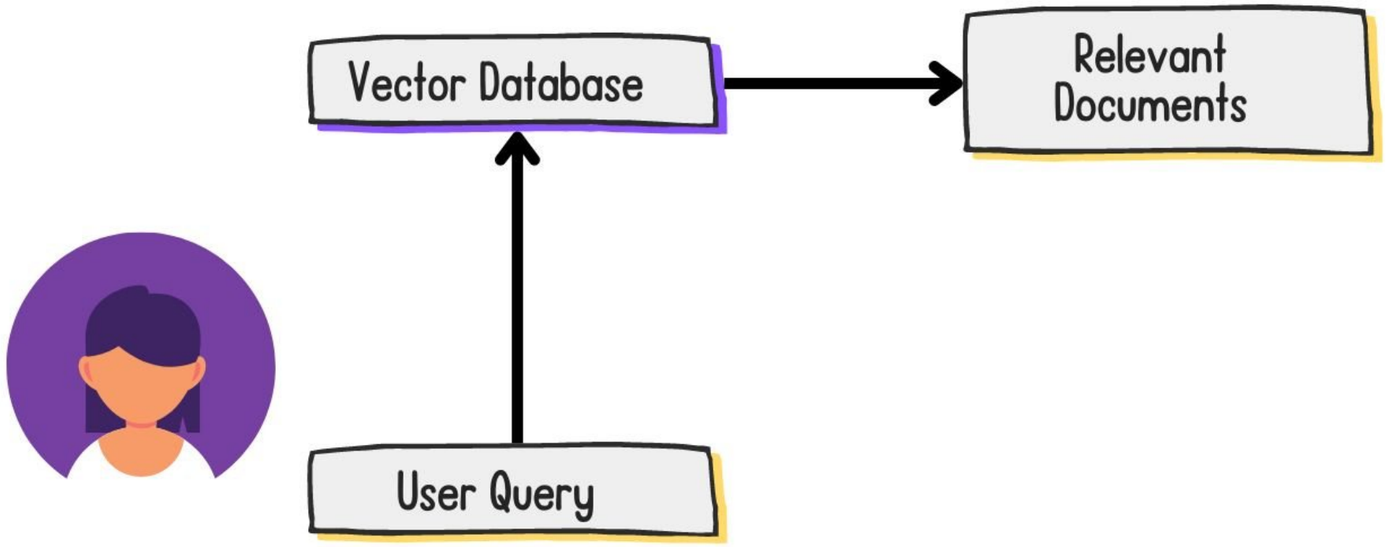
[1] Generated with DALL·E 3

The standard RAG workflow

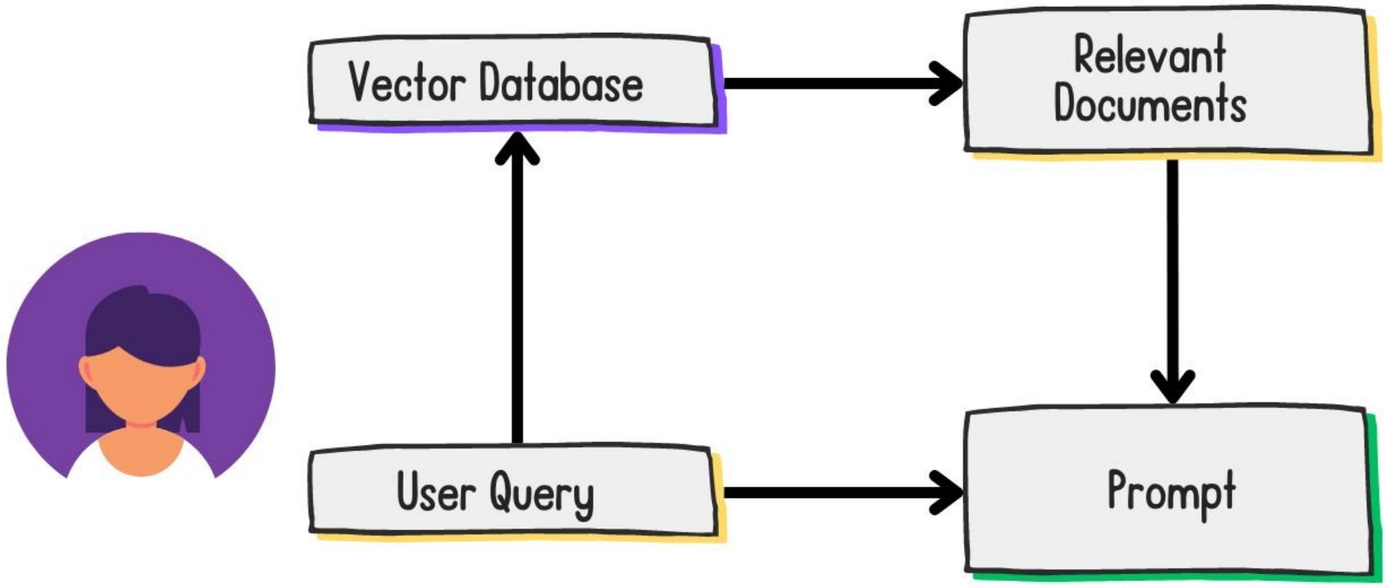
The standard RAG workflow



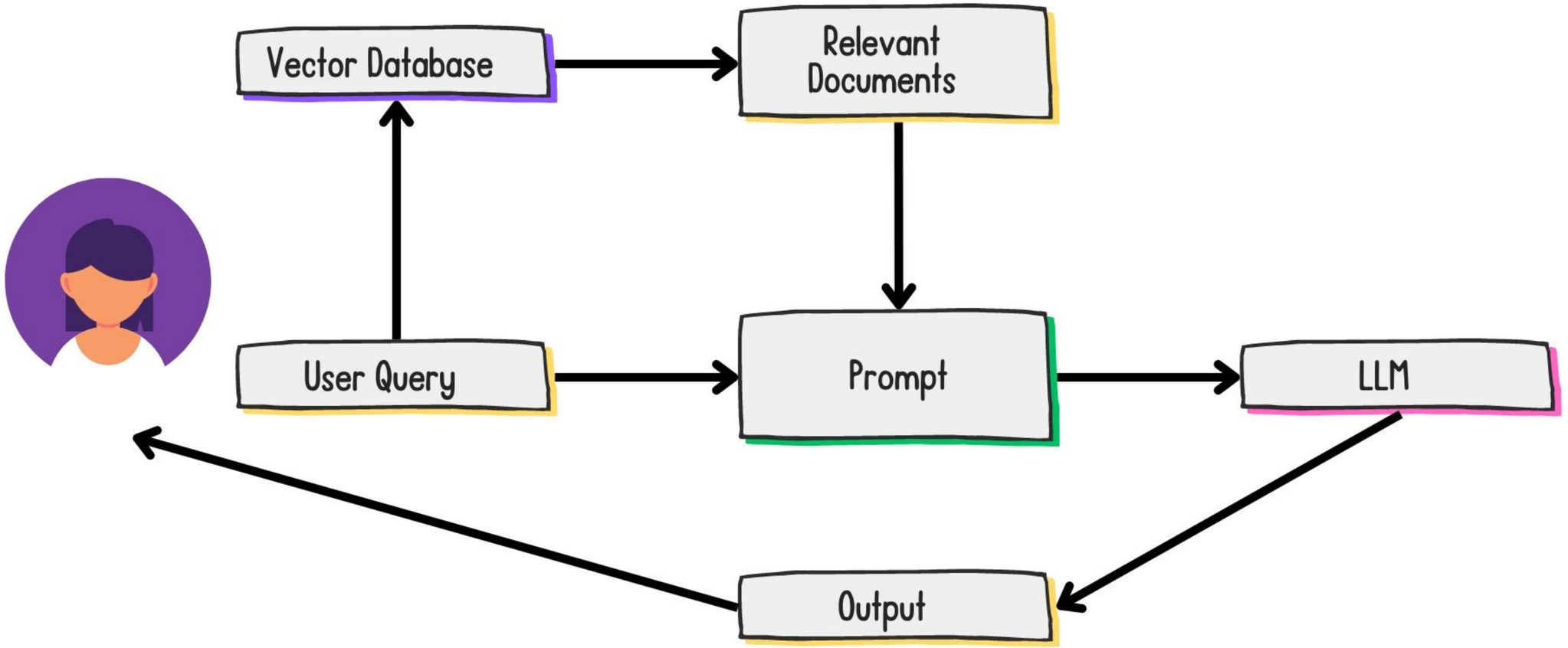
The standard RAG workflow



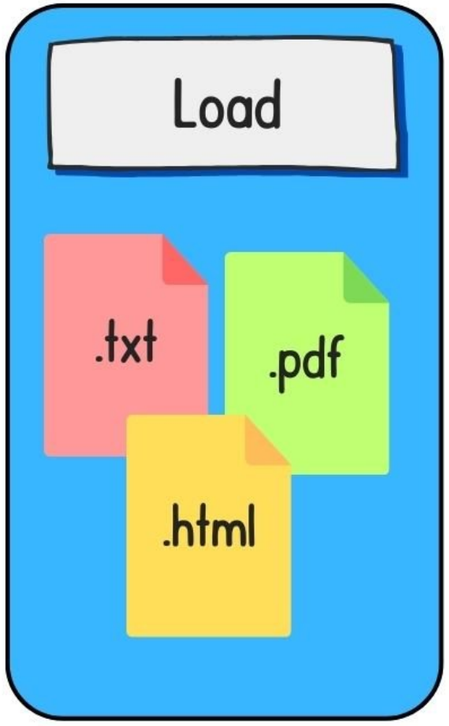
The standard RAG workflow



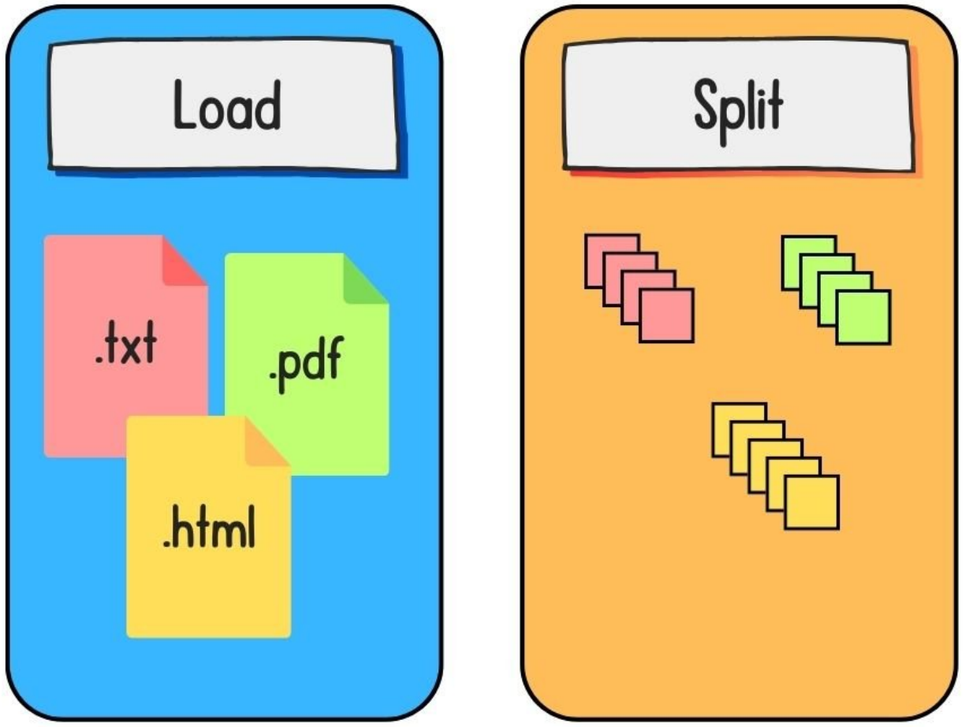
The standard RAG workflow



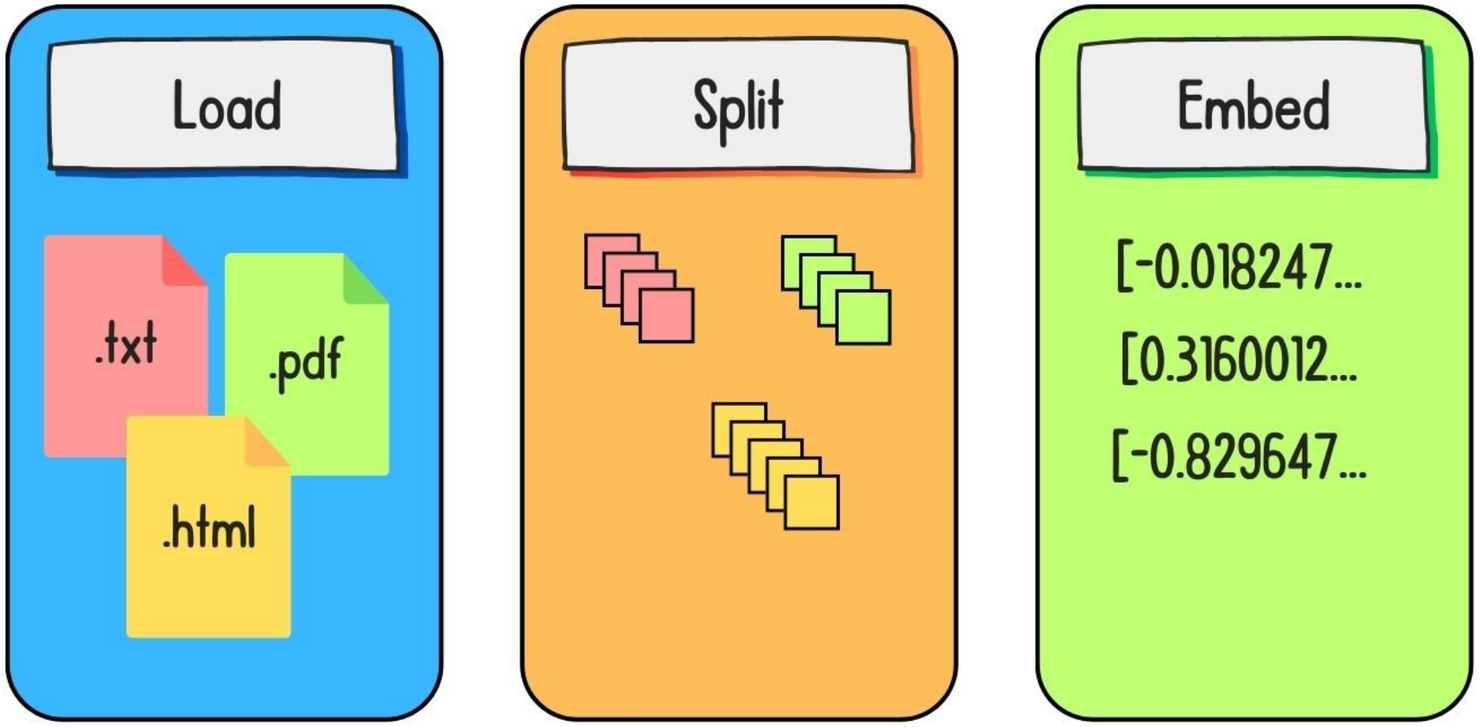
Preparing data for retrieval



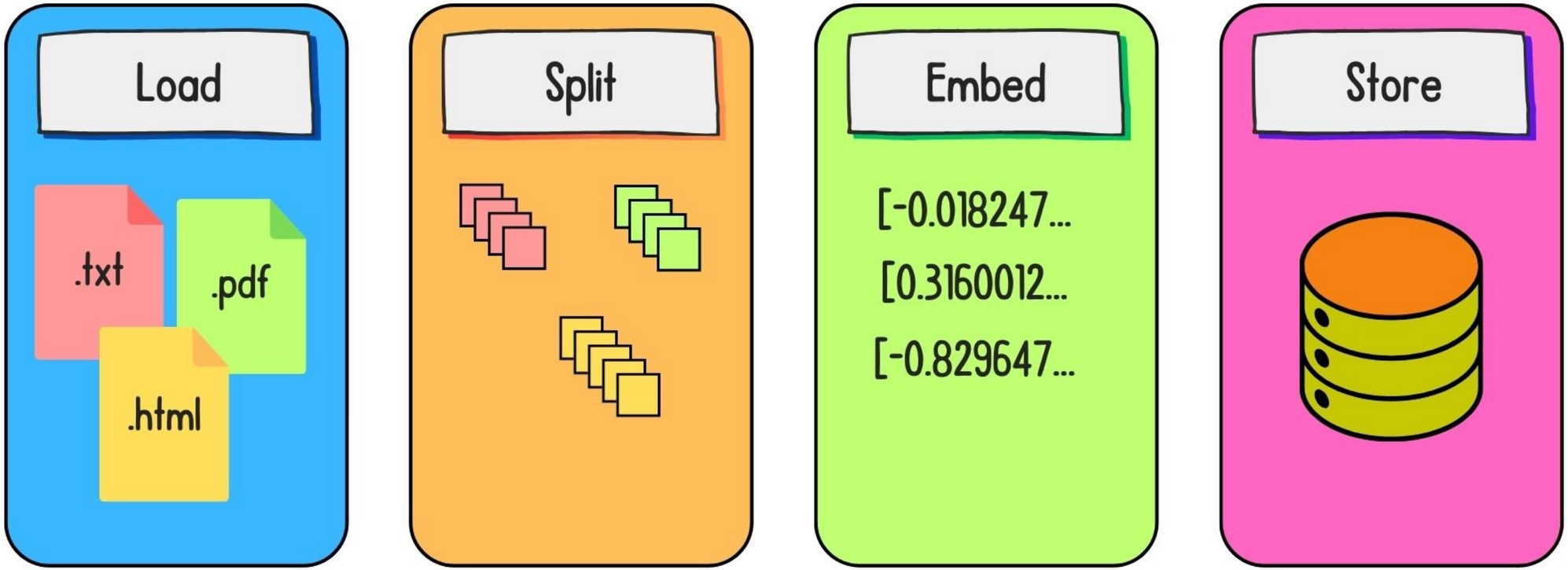
Preparing data for retrieval



Preparing data for retrieval

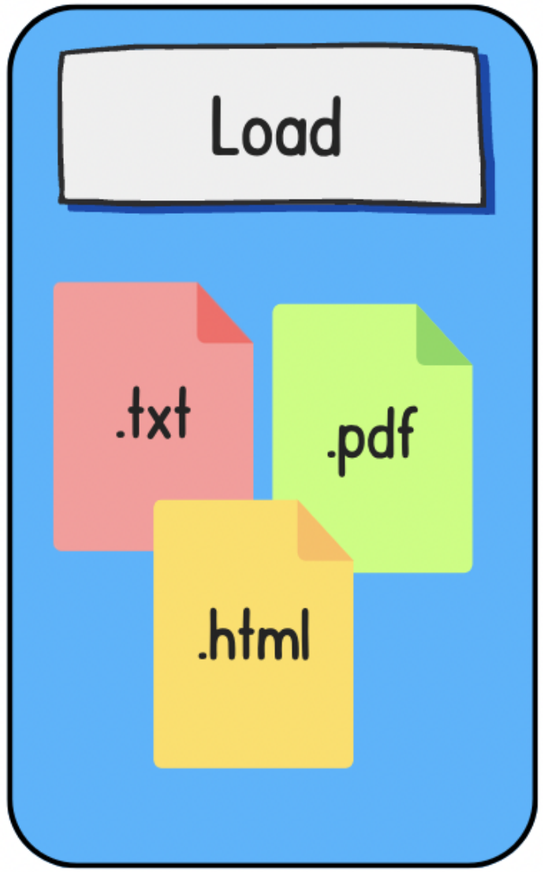


Preparing data for retrieval



Document loaders

- Integrate documents with AI systems
- Support for many common file formats
- Third party document loaders



- `CSVLoader`
- `PyPDFLoader`
- `UnstructuredHTMLLoader`

Loading CSV Files

```
from langchain_community.document_loaders.csv_loader import CSVLoader

csv_loader = CSVLoader(file_path='path/to/your/file.csv')
documents = csv_loader.load()
print(documents)
```

```
[Document(page_content='Team: Nationals\n"Payroll (millions)": 81.34\n"Wins": 98',
          metadata={'source': 'path/to/your/file.csv', 'row': 0}),
 Document(page_content='Team: Reds\n"Payroll (millions)": 82.20\n"Wins": 97',
          metadata={'source': 'path/to/your/file.csv', 'row': 1}),
 Document(page_content='Team: Yankees\n"Payroll (millions)": 197.96\n"Wins": 95',
          metadata={'source': 'path/to/your/file.csv', 'row': 2})]
```

Loading PDF Files

```
from langchain_community.document_loaders import PyPDFLoader

pdf_loader = PyPDFLoader('rag_paper.pdf')
documents = pdf_loader.load()
print(documents)
```

```
[Document(page_content='Retrieval-Augmented Generation for\nKnowledge-Intensive...',
          metadata={'source': 'Rag Paper.pdf', 'page': 0})]
```

Loading HTML Files

```
from langchain_community.document_loaders import UnstructuredHTMLLoader

html_loader = UnstructuredHTMLLoader(file_path='path/to/your/file.html')
documents = html_loader.load()
first_document = documents[0]

print("Content:", first_document.page_content)
print("Metadata:", first_document.metadata)
```

```
Content: Welcome to Our Website
Metadata: {'source': 'path/to/your/file.html', 'section': 0}
```

Let's practice!

RETRIEVAL AUGMENTED GENERATION (RAG) WITH LANGCHAIN

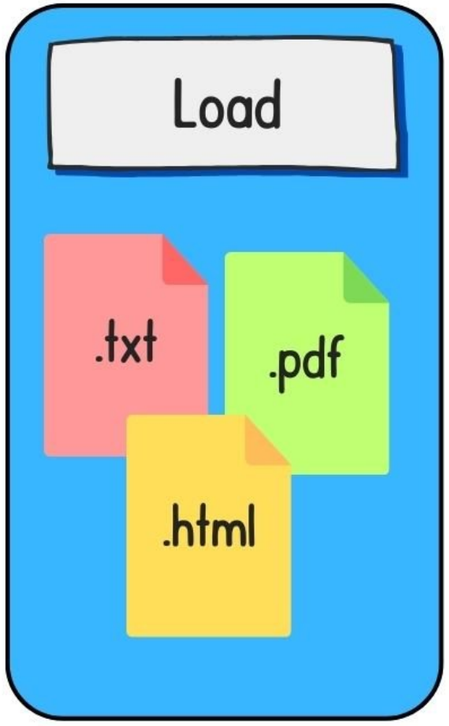
Text splitting, embeddings, and vector storage

RETRIEVAL AUGMENTED GENERATION (RAG) WITH LANGCHAIN

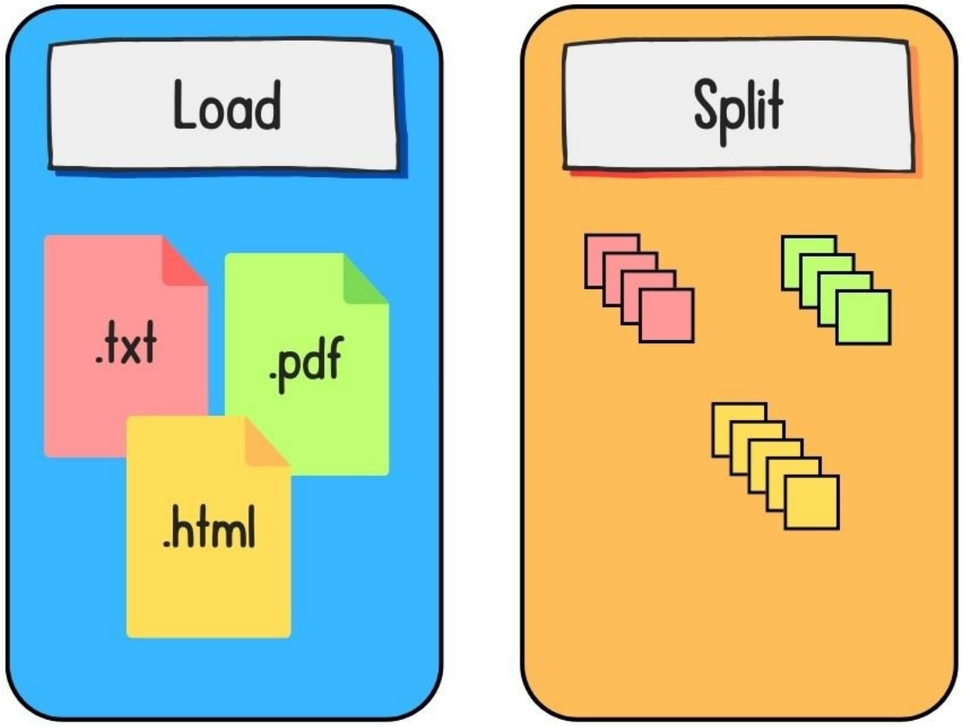
Meri Nova
Machine Learning Engineer

datacamp

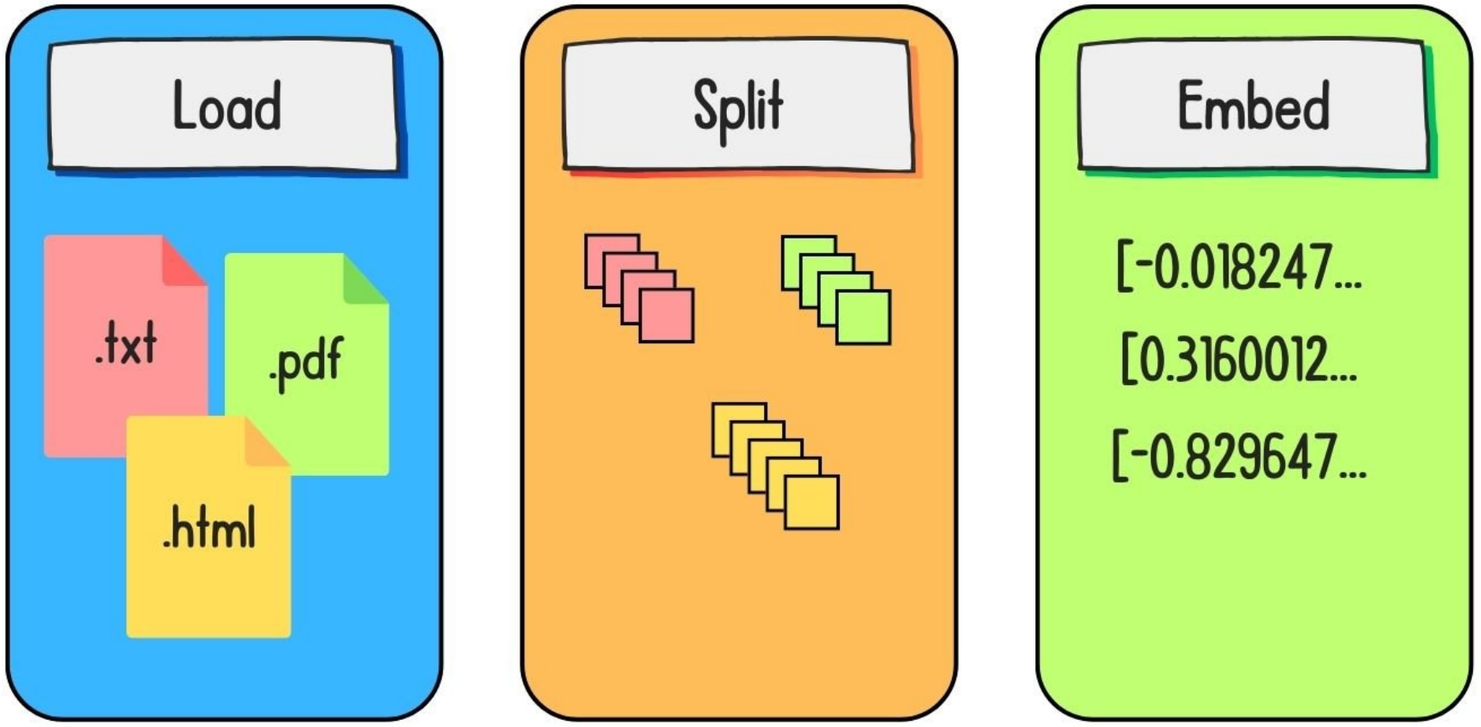
Preparing data for retrieval



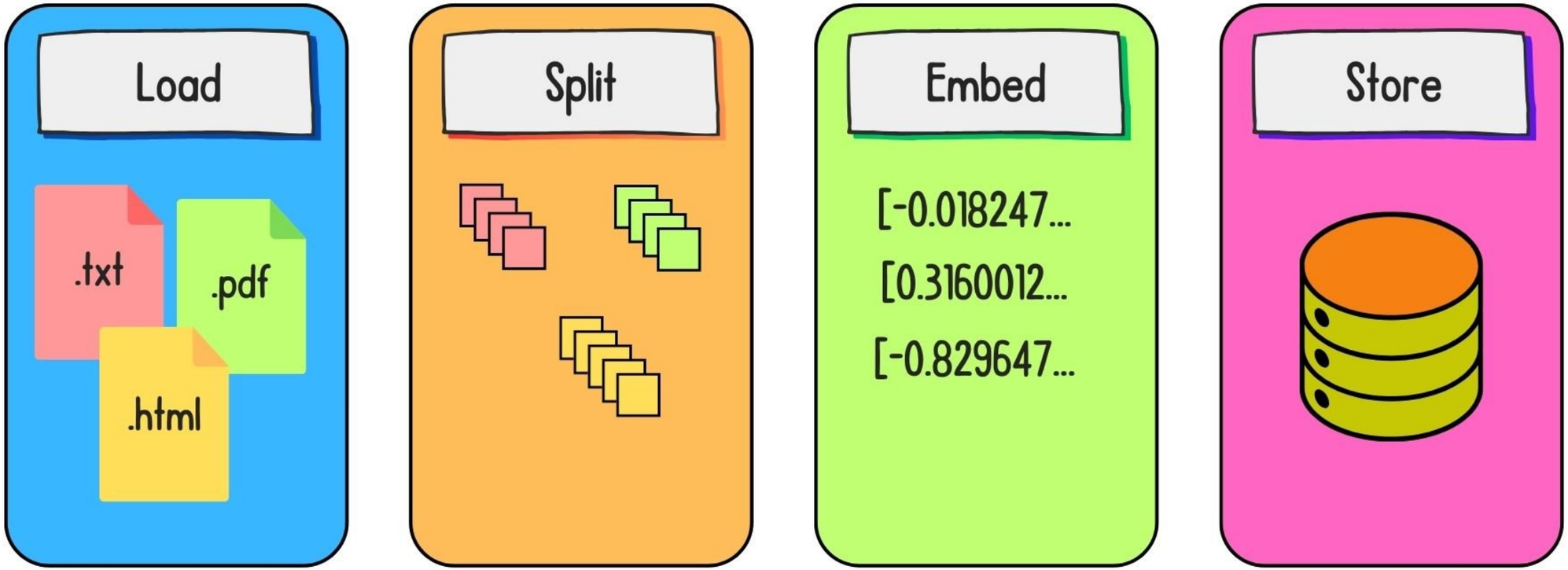
Preparing data for retrieval



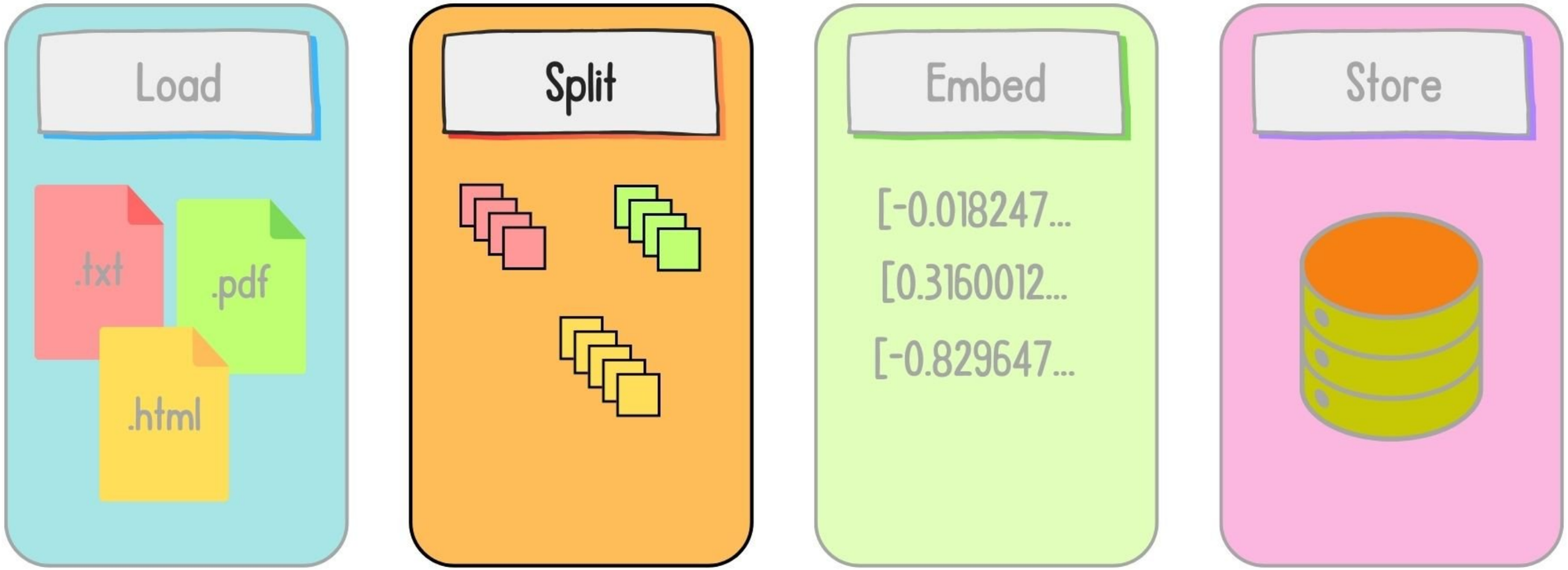
Preparing data for retrieval



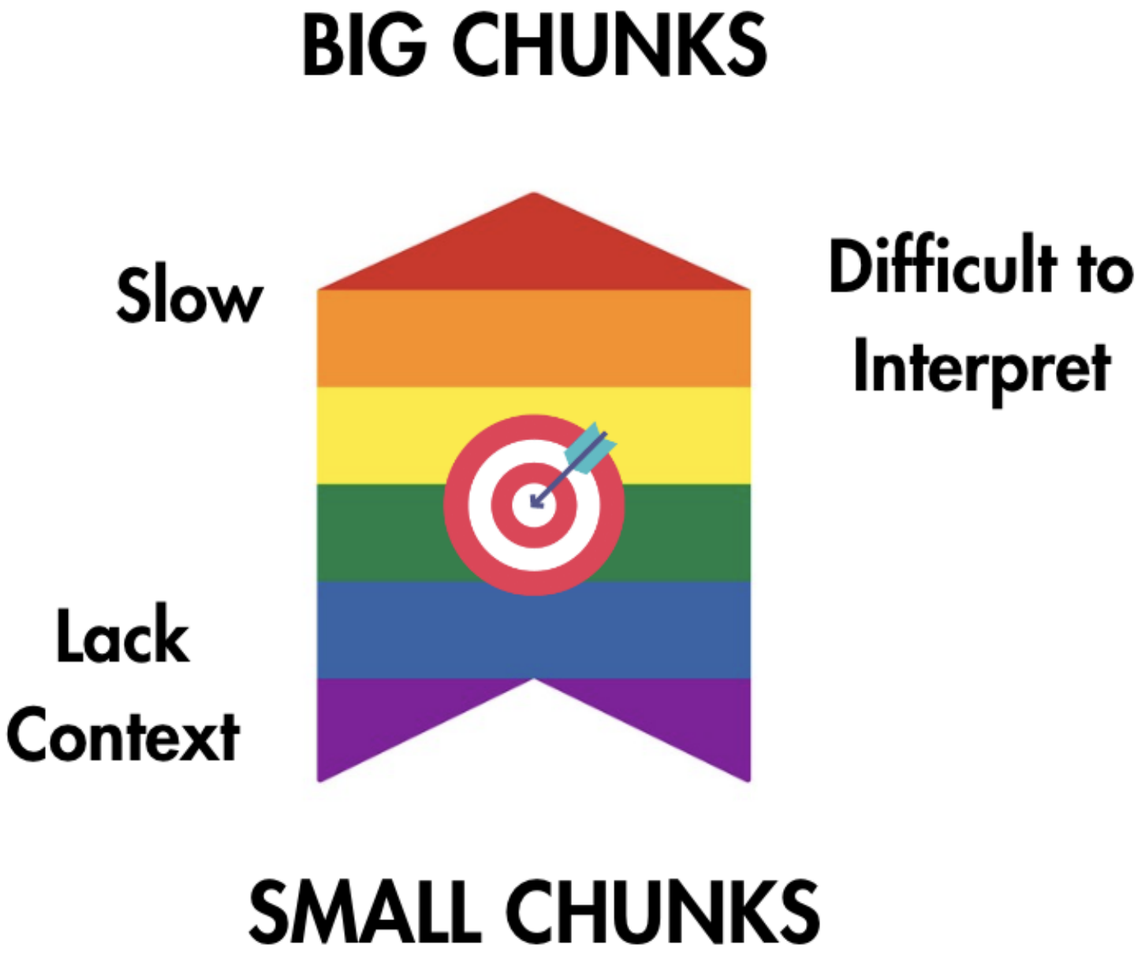
Preparing data for retrieval



Preparing data for retrieval

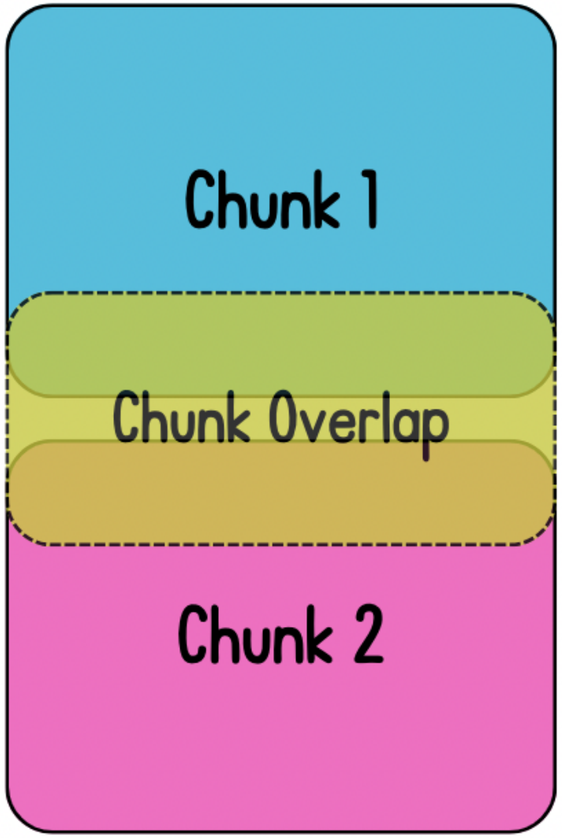


`chunk_size`



`chunk_overlap`

- Include information *beyond* the **boundary**



CharacterTextSplitter

```
from langchain_text_splitters import CharacterTextSplitter

text = """Machine learning is a fascinating field.\n\nIt involves algorithms and models that can learn from data. These models can then make predictions or decisions without being explicitly programmed to perform the task.\nThis capability is increasingly valuable in various industries, from finance to healthcare.\n\nThere are many types of machine learning, including supervised, unsupervised, and reinforcement learning.\nEach type has its own strengths and applications."""

text_splitter = CharacterTextSplitter(
    separator="\n\n",
    chunk_size=100,
    chunk_overlap=10
)
```

CharacterTextSplitter

```
chunks = text_splitter.split_text(text)
print(chunks)
print([len(chunk) for chunk in chunks])
```

```
['Machine learning is a fascinating field.',
 'It involves algorithms and models that can learn from data. These models can...',
 'There are many types of machine learning, including supervised, unsupervised...']
[40, 260, 155]
```

RecursiveCharacterTextSplitter

```
from langchain_text_splitters import RecursiveCharacterTextSplitter


splitter = RecursiveCharacterTextSplitter(
    separators=["\n\n", "\n", " ", ""],
    chunk_size=100,
    chunk_overlap=10
)
```

RecursiveCharacterTextSplitter

```
chunks = splitter.split_text(text)


print(chunks)
print([len(chunk) for chunk in chunks])
```

```
['Machine learning is a fascinating field.',
 'It involves algorithms and models that can learn from data. These models ...',
 'or decisions without being explicitly programmed to perform the task.',
 'This capability is increasingly valuable in various industries, from ...',
 'There are many types of machine learning, including supervised, ...',
 'learning.',
 'Each type has its own strengths and applications.']
[40, 98, 69, 91, 95, 9, 49]
```

Splitting documents

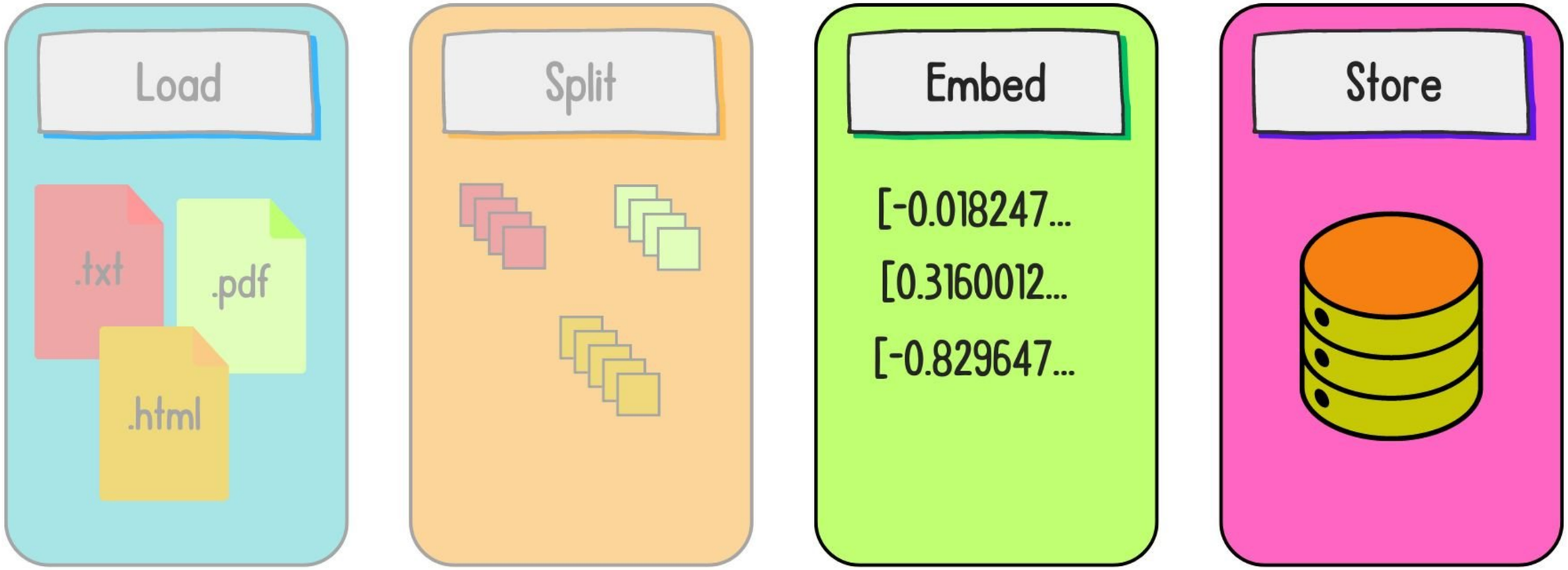
```
from langchain_community.document_loaders import PyPDFLoader

loader = PyPDFLoader("research_paper.pdf")
documents = loader.load()

splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)



chunks = splitter.split_documents(documents)
```

Splitting documents

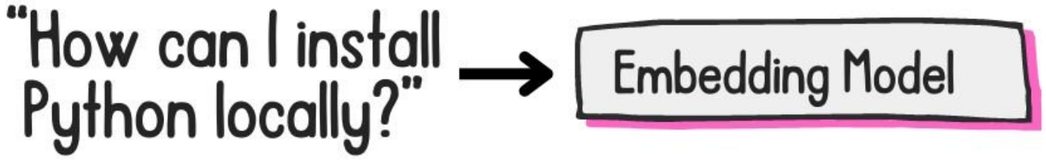
```
print(chunks)
print([len(chunk.page_content) for chunk in chunks])
print(f"Total number of chunks: {len(chunks)}")
```

```
[Document(metadata={'source': 'Rag Paper.pdf', 'page': 0}, page_content='Retrieval-A
 Document(metadata={'source': 'Rag Paper.pdf', 'page': 0}, page_content='decisions a
 Document(metadata={'source': 'Rag Paper.pdf', 'page': 0}, page_content='per token.
[928, 946, 921,...]
Total number of documents: 92
```

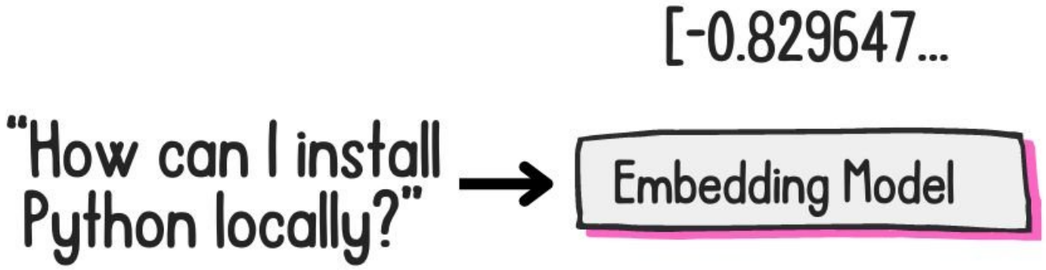
Embedding and storage



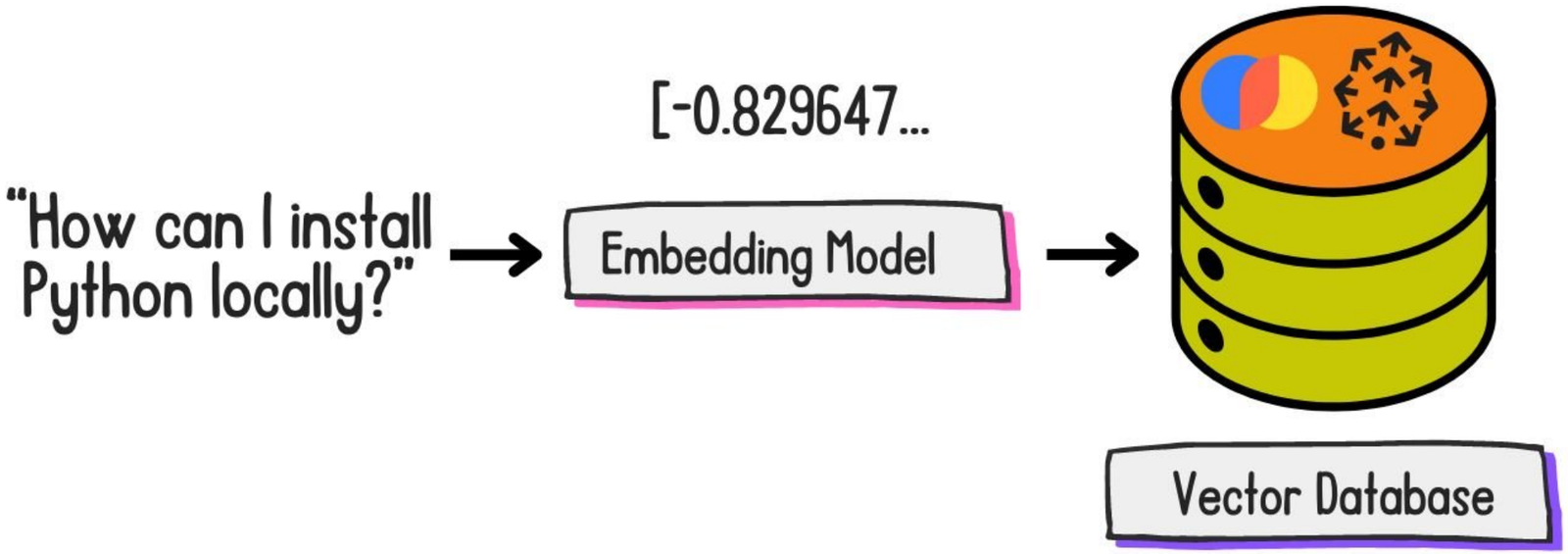
What are embeddings?



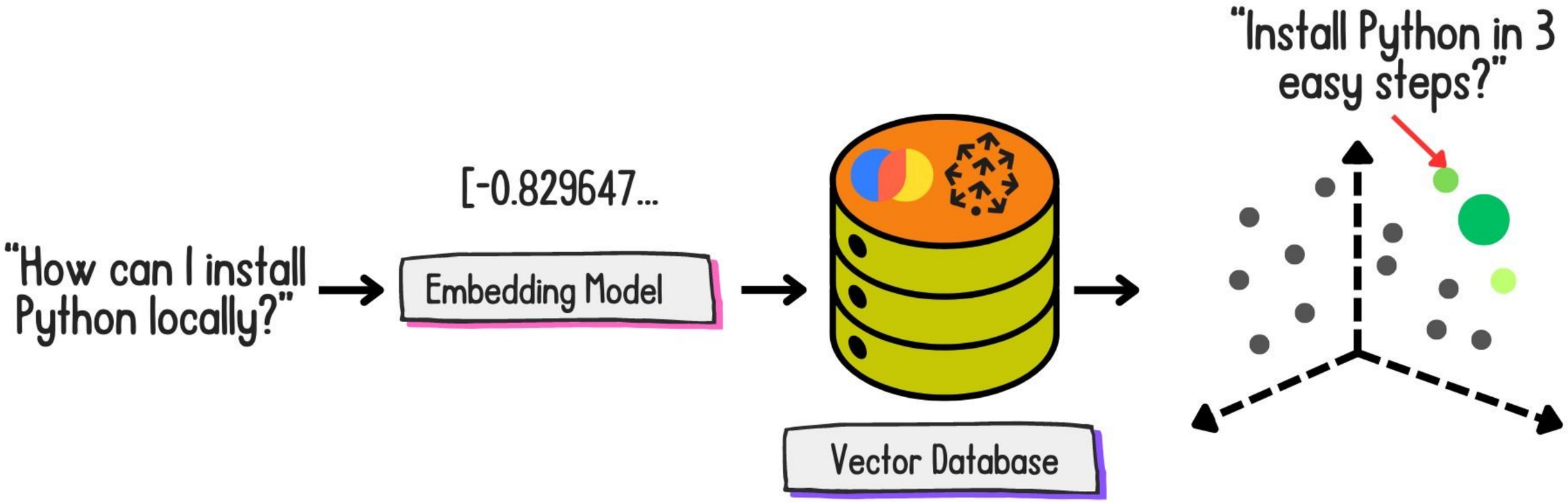
What are embeddings?



What are embeddings?



What are embeddings?



Embedding and storing the chunks

- Embed and store with: **OpenAI** and **ChromaDB**

```
from langchain_openai import OpenAIEmbeddings
from langchain_chroma import Chroma

embedding_model = OpenAIEmbeddings(
    api_key=openai_api_key,
    model="text-embedding-3-small"
)

vector_store = Chroma.from_documents(
    documents=chunks,
    embedding=embedding_model
)
```

Let's practice!

RETRIEVAL AUGMENTED GENERATION (RAG) WITH LANGCHAIN

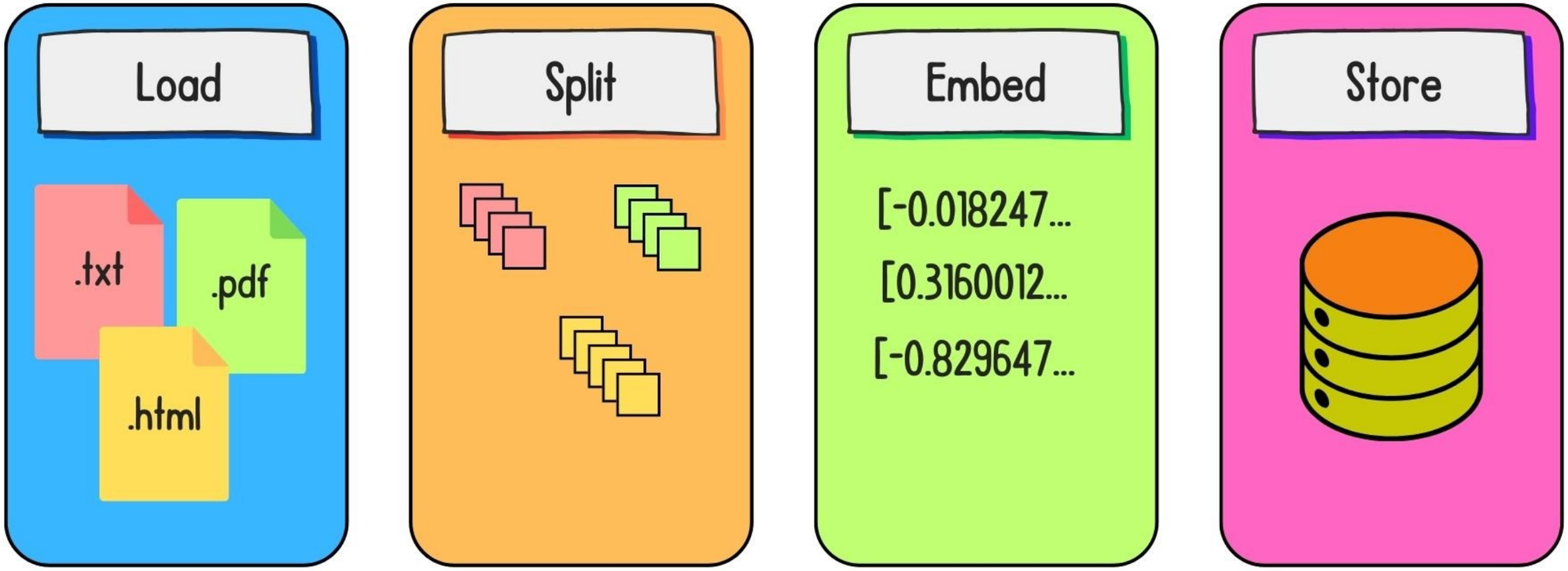
Building an LCEL retrieval chain

RETRIEVAL AUGMENTED GENERATION (RAG) WITH LANGCHAIN

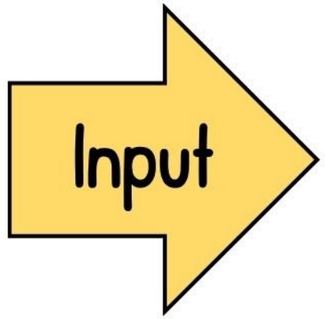
Meri Nova
Machine Learning Engineer

datacamp

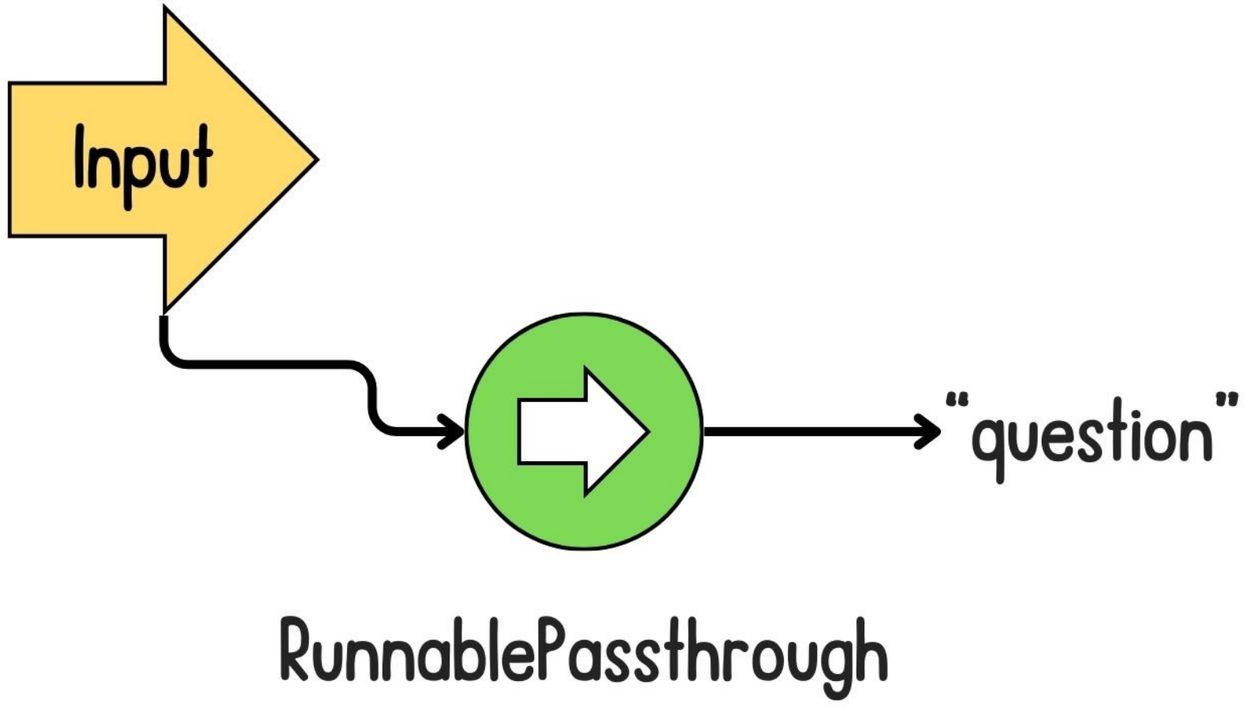
Preparing data for retrieval



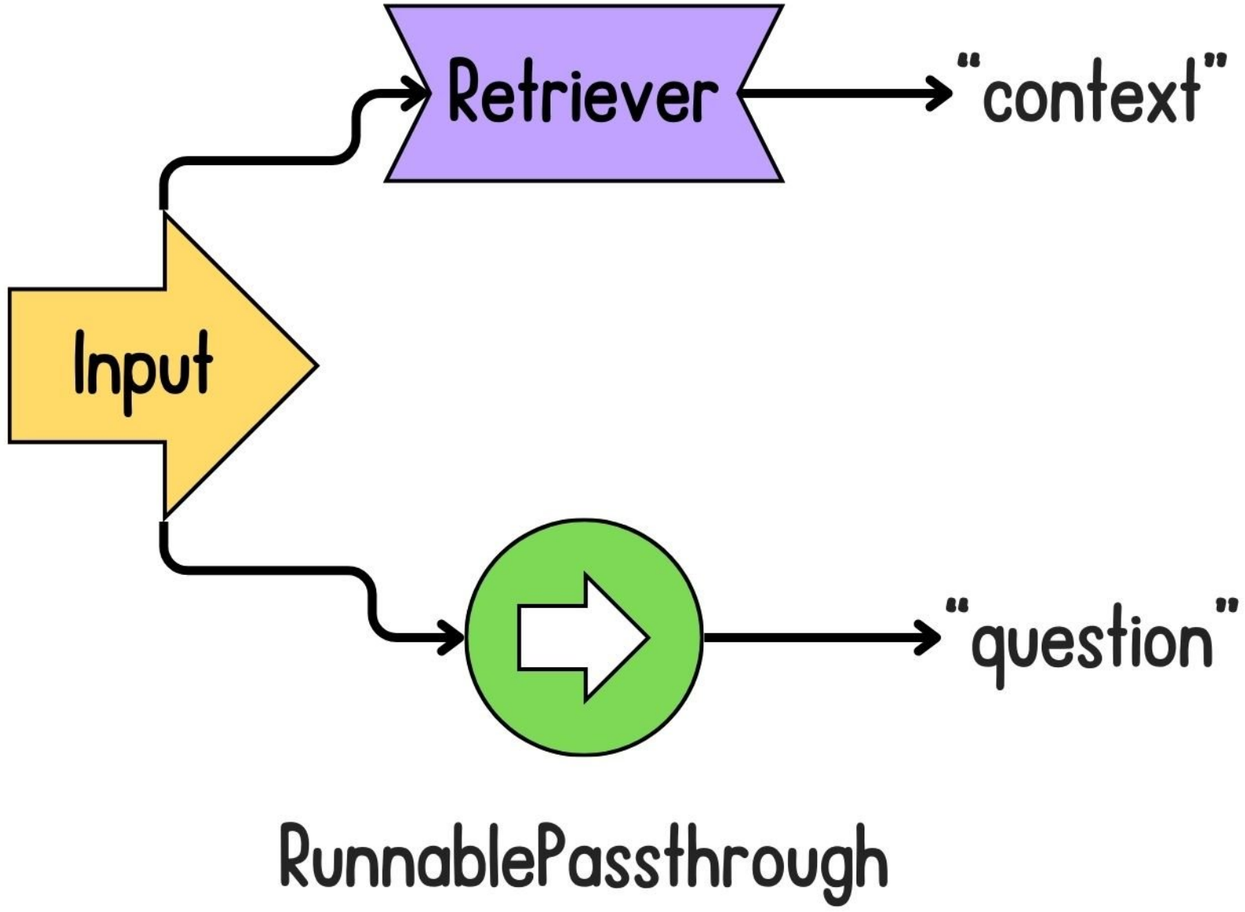
Introduction to LCEL for RAG



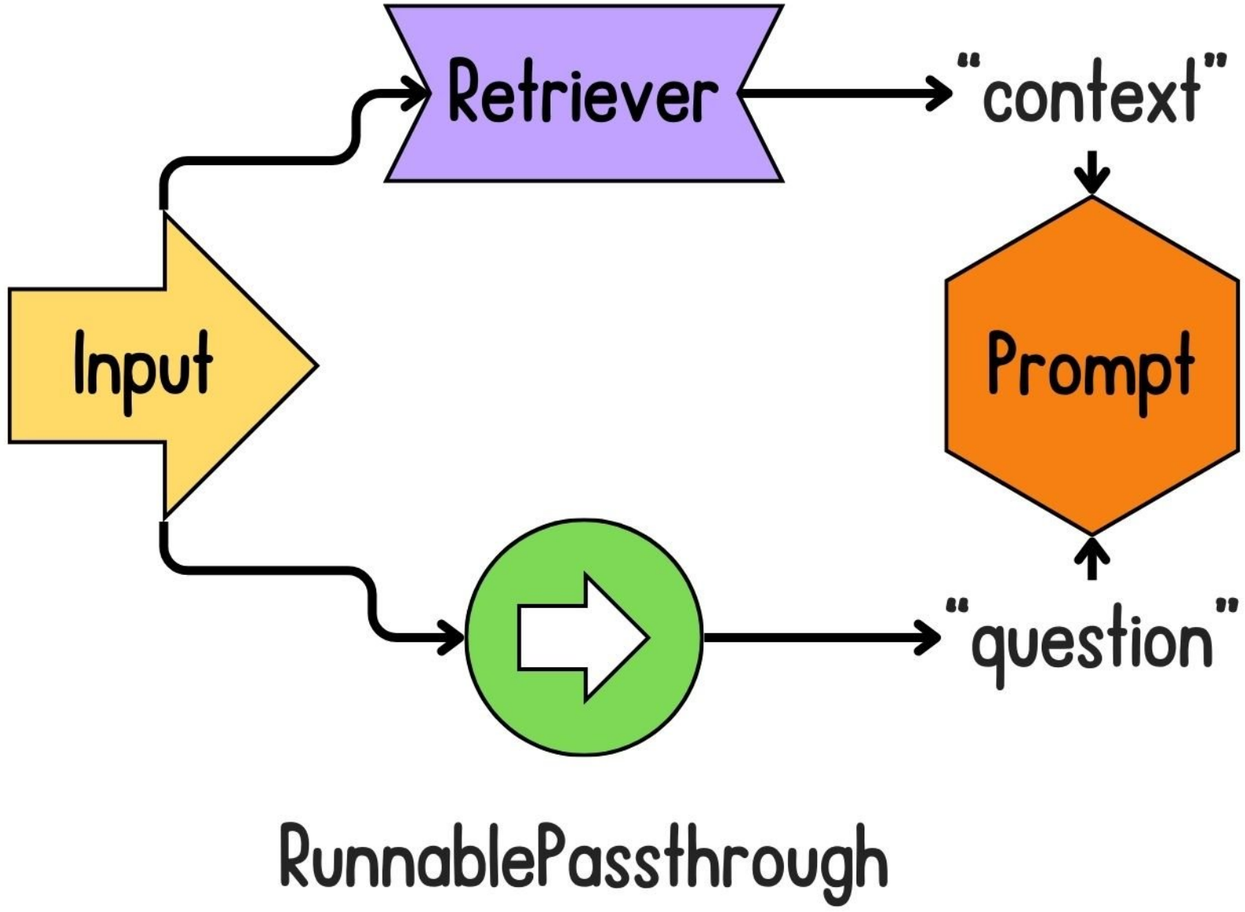
Introduction to LCEL for RAG



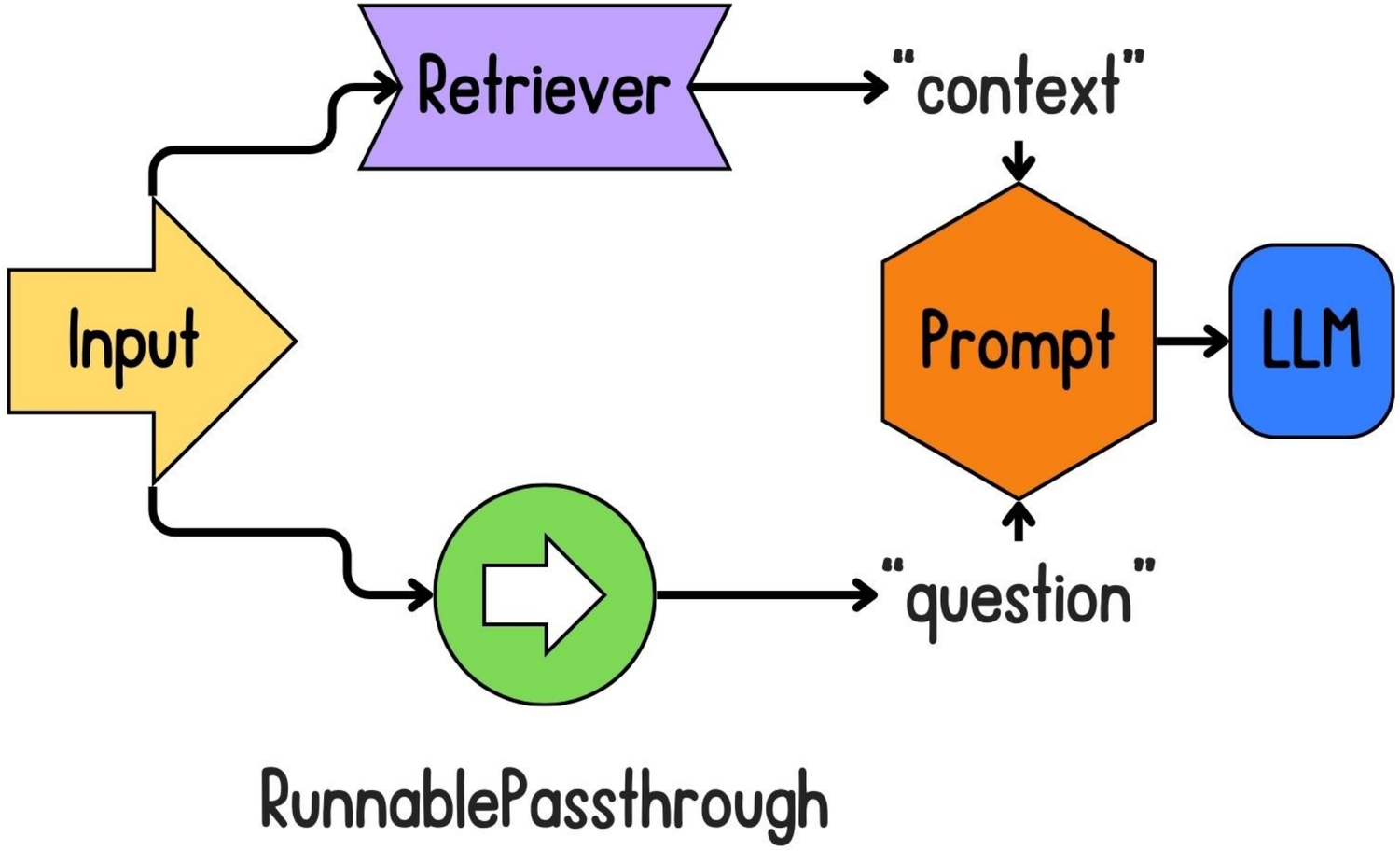
Introduction to LCEL for RAG



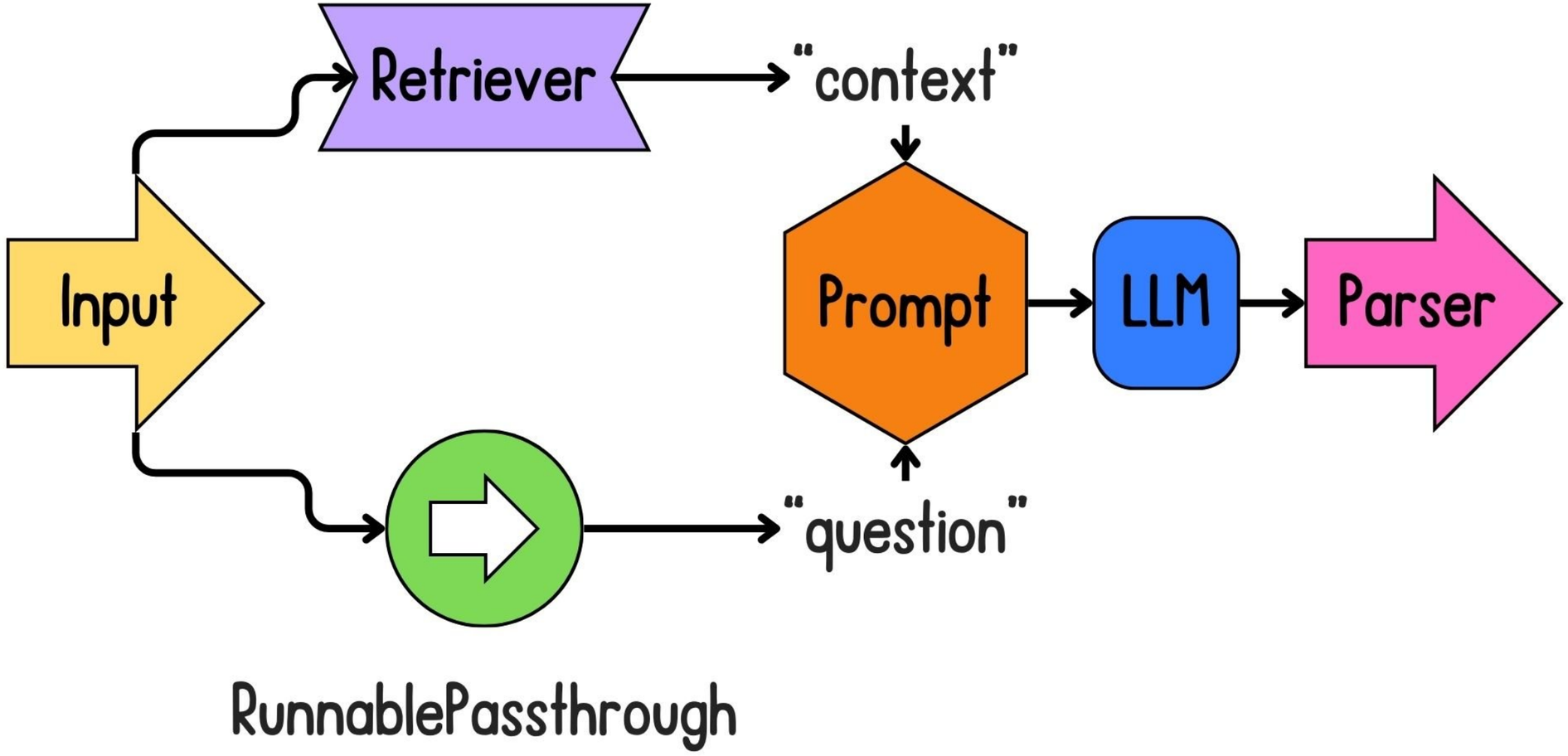
Introduction to LCEL for RAG



Introduction to LCEL for RAG



Introduction to LCEL for RAG



Instantiating a retriever

```
vector_store = Chroma.from_documents(
    documents=chunks,
    embedding=embedding_model
)

retriever = vector_store.as_retriever(
    search_type="similarity",
    search_kwargs={"k": 2}
)
```

Creating a prompt template

```
from langchain_core.prompts import ChatPromptTemplate


prompt = ChatPromptTemplate.from_template("""
Use the following pieces of context to answer the question at the end.
If you don't know the answer, say that you don't know.
Context: {context}
Question: {question}
""")
```

```
llm = ChatOpenAI(model="gpt-4o-mini", api_key="...", temperature=0)
```

Building an LCEL retrieval chain

```
from langchain_core.runnables import RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser

chain = (
    {"context": retriever, "question": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)
```

Invoking the retrieval chain

```
result = chain.invoke({"question": "What are the key findings or results presented in the paper?"})
print(result)
```

```
- Top Performance: RAG models set new records on open-domain question answering tasks...
- Better Generation: RAG models produce more specific, diverse, and factual language...
- Dynamic Knowledge Use: The non-parametric memory allows RAG models to access and ...
```

Let's practice!

RETRIEVAL AUGMENTED GENERATION (RAG) WITH LANGCHAIN