Who Am I?

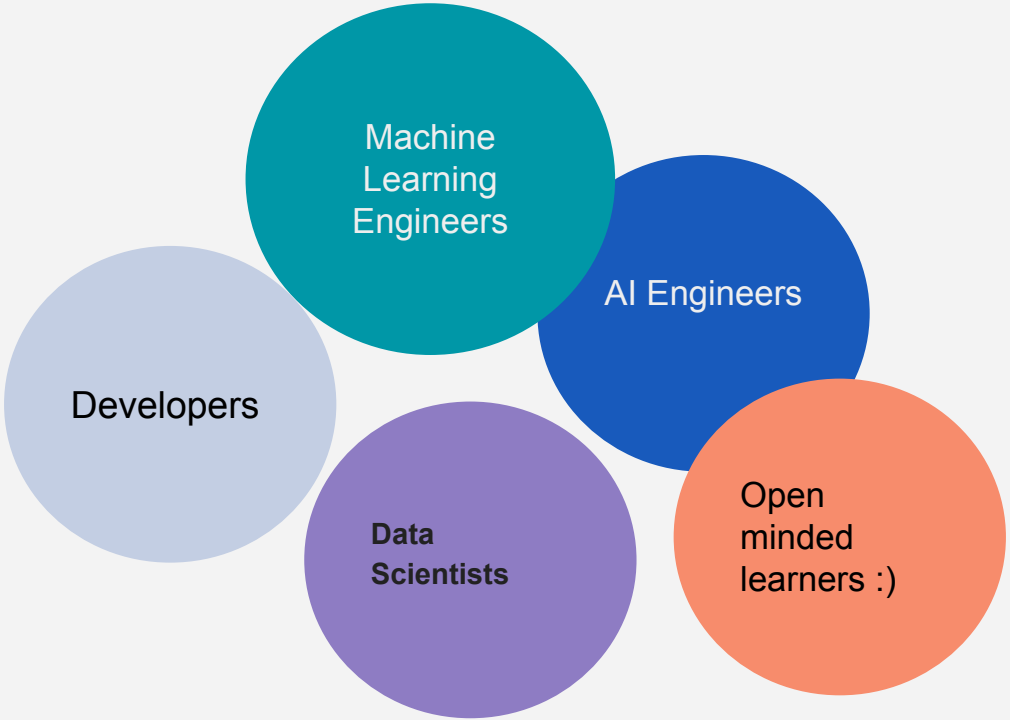
Paulo Dichone

Software, Cloud, AI Engineer
and Instructor

What Is This Course About?

- Ingesting and normalizing content from various data sources and types for LLM applications
  - LLM data preprocessing
    - Unstructured data to structured data
    - Key concepts
    - What problem it solves
    - Normalizing content for LLMs
  - Metadata extraction and chunking
  - Preprocessing PDFs and Images for LLMs
  - Extracting tables from complex document types
  - Best practices and advanced techniques
  - Build a RAG system from normalized content
  - Lots of hands-on

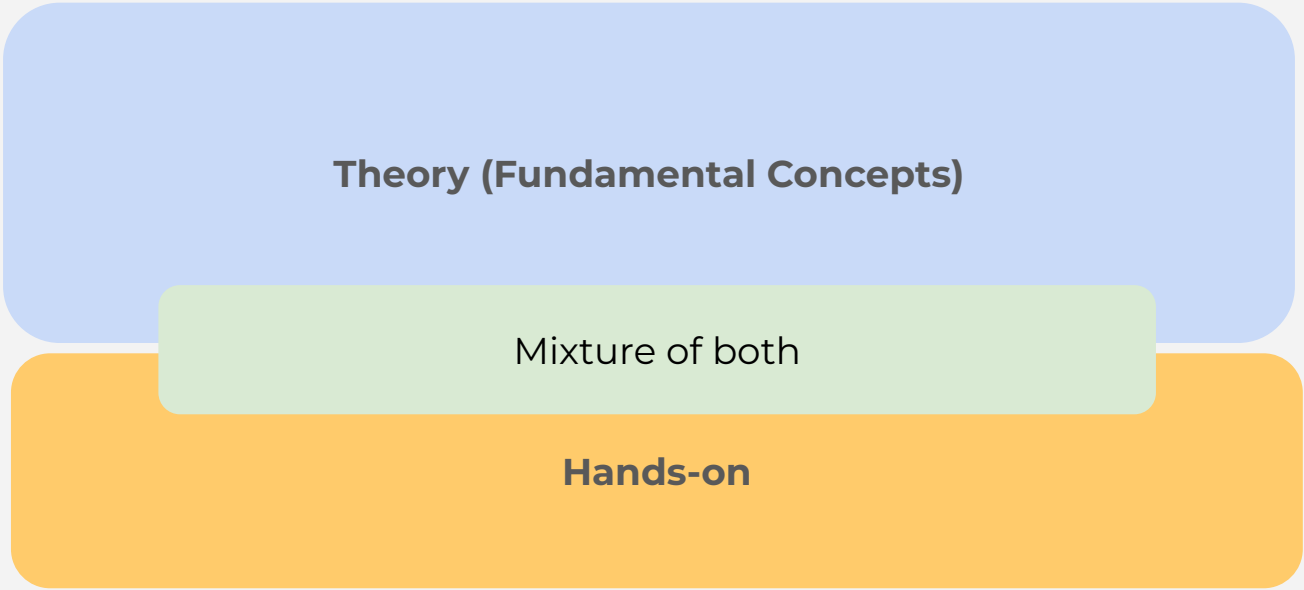
Who Is This Course For



Course Prerequisites

1. Know programming (highly *preferred… at least the basics)*
    a. *We will be using Python*
2. Basics of AI, ML, LLM
3. *This is not a programming course*
4. Willingness to learn :)

Course Structure



Development Environment setup

- Python
- VS Code (or any other code editor)
- OpenAI Account and an OpenAI API Key
- Unstructured Framework account and API Key (Free for the first 1,000 document processing)

Set up OpenAI API Account

***** Please note*** *that you will need an API key to use OpenAI services, and there may be some costs associated with using the API. However, these costs should be minimal.*

Data Preprocessing For LLMs

Overview

- Unstructured data
- Content extraction from unstructured data
  - Why content extraction matters?
  - How it works?
- Key concepts

Unstructured data

***80% of data around is** unstructured...*

What does that mean?

Unstructured data

Information that does not have a pre-defined data model.

Textual data

Emails
Documents
Social media posts...

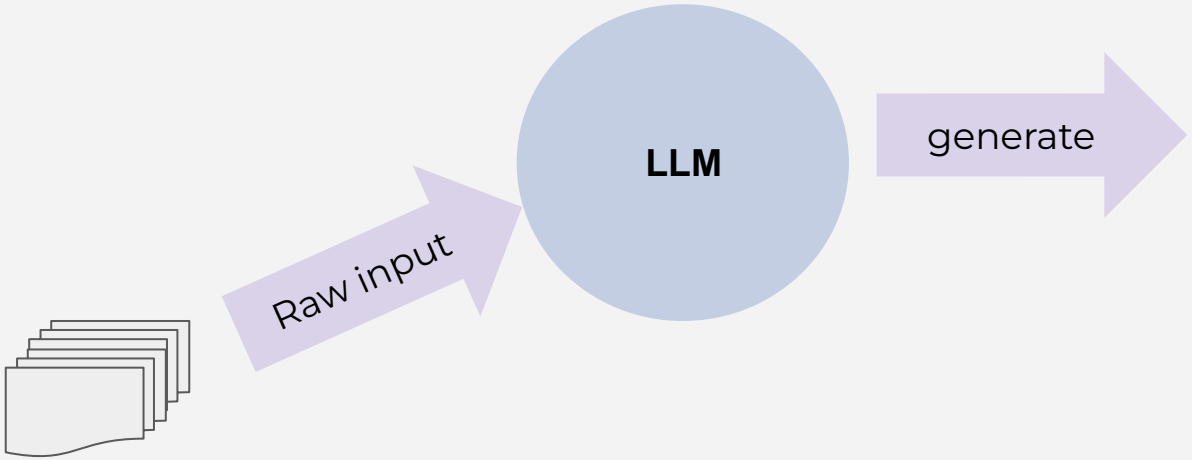
Multimedia content

Images
Videos
Audio files

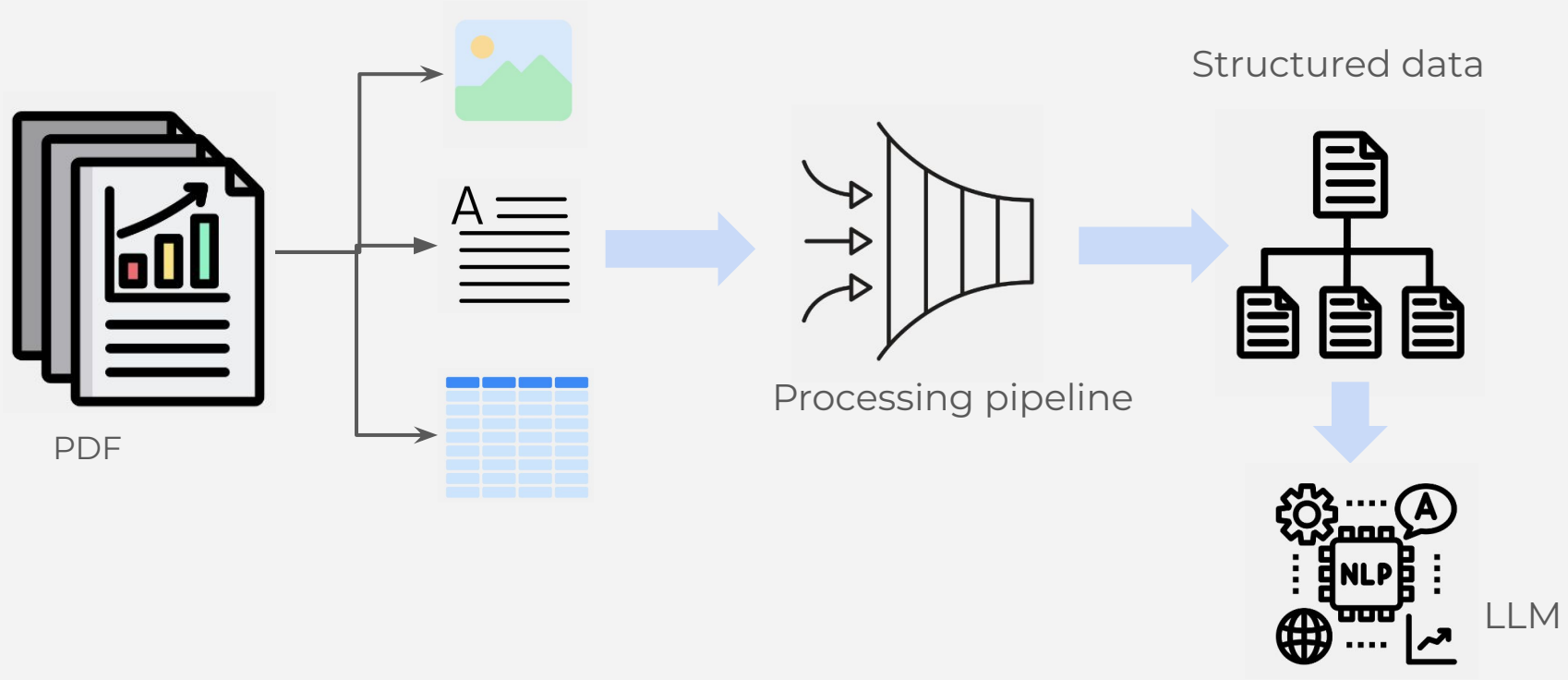
Complex documents

PDFs, PowerPoint presentations
Web pages (HTML)
Scanned forms

In the context of LLMs



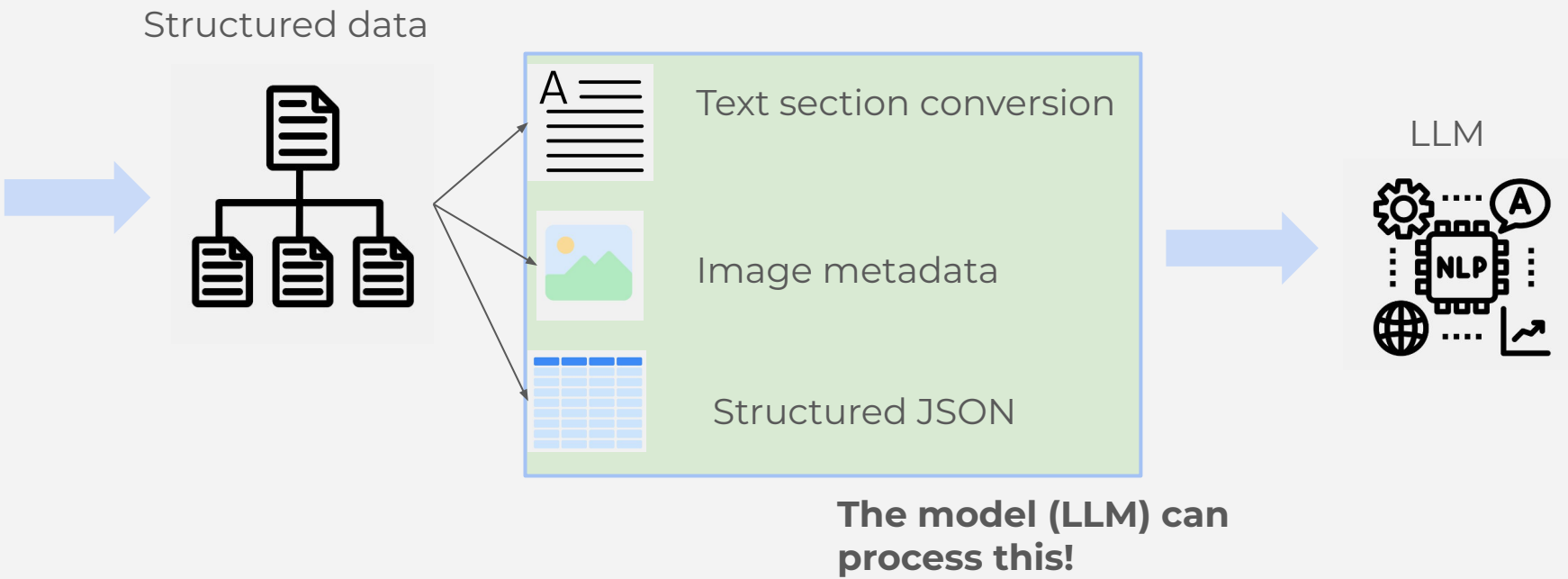
In the context of LLMs - example



Why is data preprocessing hard?

- **Content Cues** - different doc types have unique cues for identifying elements - consistent processing is difficult.
- **Need for standardization**: uniformity in documents is crucial
- **Extraction variability**: different formats require different methods
- **Metadata insight**:relevant metadata extraction demands an in-depth understanding of the document's structure, which adds more complexity when preprocessing.

In the context of LLMs - example



The model (LLM) can process this!

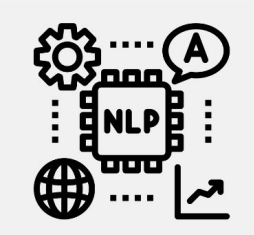
Why it matters - deep dive

(content extraction from Unstructured data)

All relevant data is captured - nothing is lost

- LLMs rely on high-quality input data to generate accurate outputs

LLM



Challenges with unstructured data

Diversity: many forms and formats available

Complexity: mix of content types (e.g., text, images)

Volume: large amount of data (manual processing impractical)

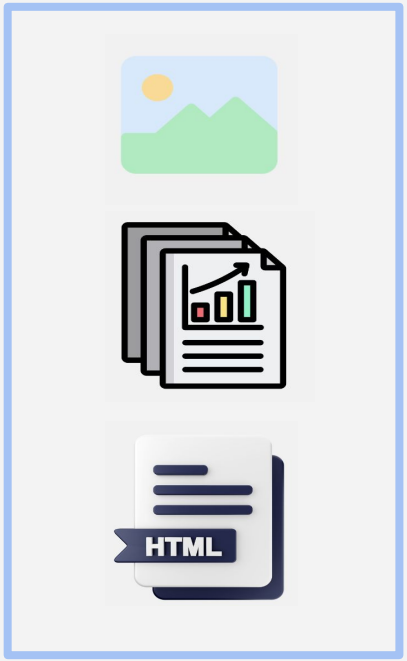
How it works?

(content extraction from Unstructured data)

Text Extraction - tools to extract not only simple docs but complex elements (tables and images

Partitioning - docs are broken down into smaller elements (e.g., paragraphs, sections) to be processed individually downstream

How it works?



Text Extraction

Section 1

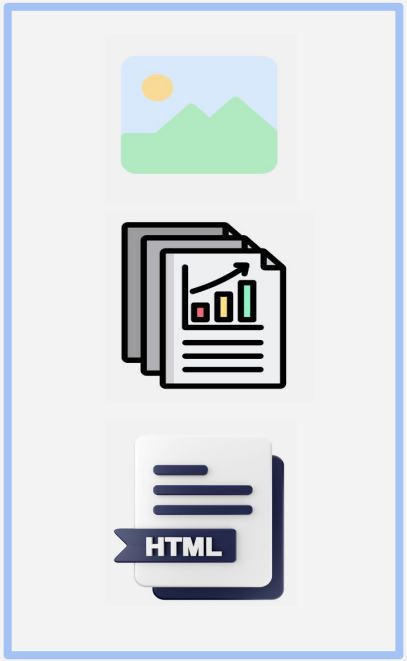
Section 2

Section 3

Section 4

```
[
  JSON Object:
  {
    "section1": "content",
    "section2": "content",
    "section3": "content",
    "section4": "content"
  }
]
```

Cleaning and normalizing data



Noise removal

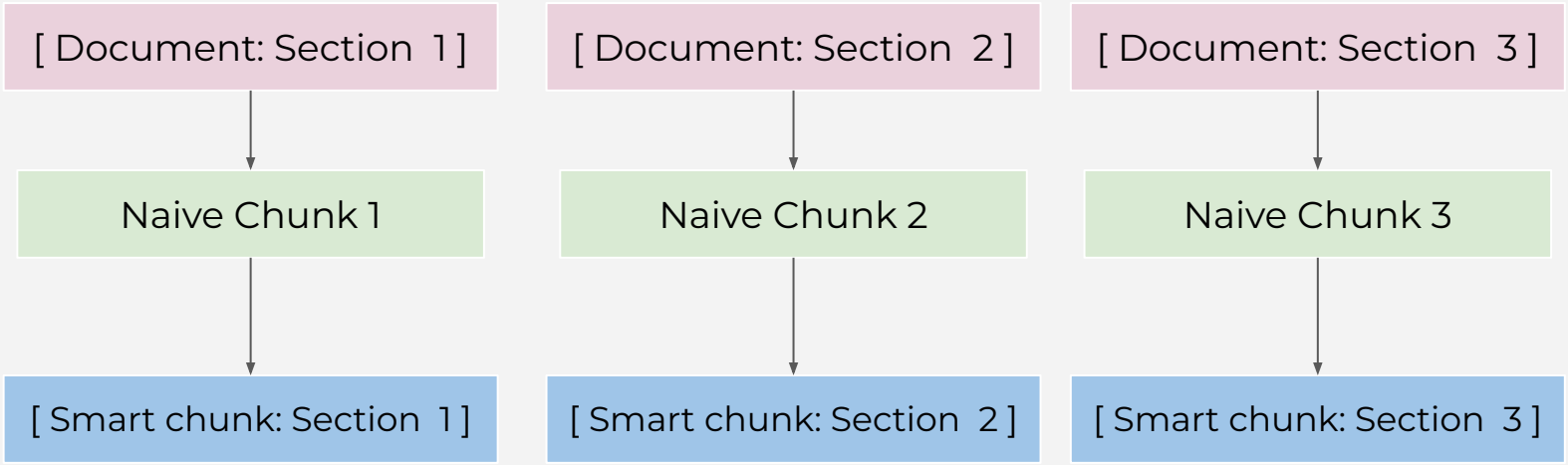
Cleaning & normalizing

Clean content
- Headers and footers removed

```
{
  JSON Object:
  {
    "section1": "content",
    "section2": "content",
    "section3": "content",
    "section4": "content"
  }
}
```

Normalized content - consistent formatting

Smart chunking vs. Naive chunking

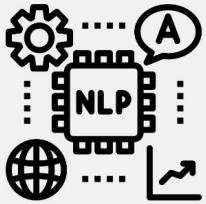


Structuring data

[Smart chunk: Section 1]

```
[
  JSON Object:
  {
    "section1": "content",
    "section2": "content",
    “Metadata”: “info”
  }
]
```

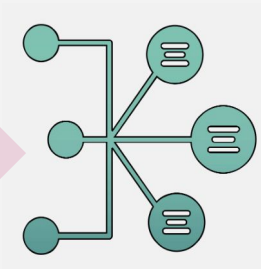
LLM processing



The LLM understands hierarchical and relational aspects of content better - more accurate and contextually aware outputs

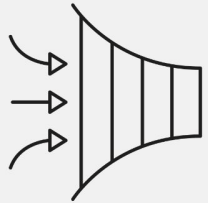
Workflow orchestration

Raw data sources



Source connectors

Processing pipeline



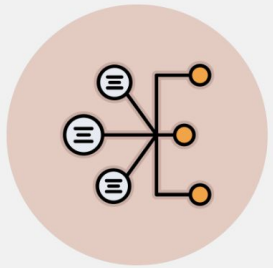
Extraction

Cleaning

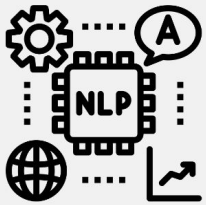
Chunking

Structuring

Destination connectors



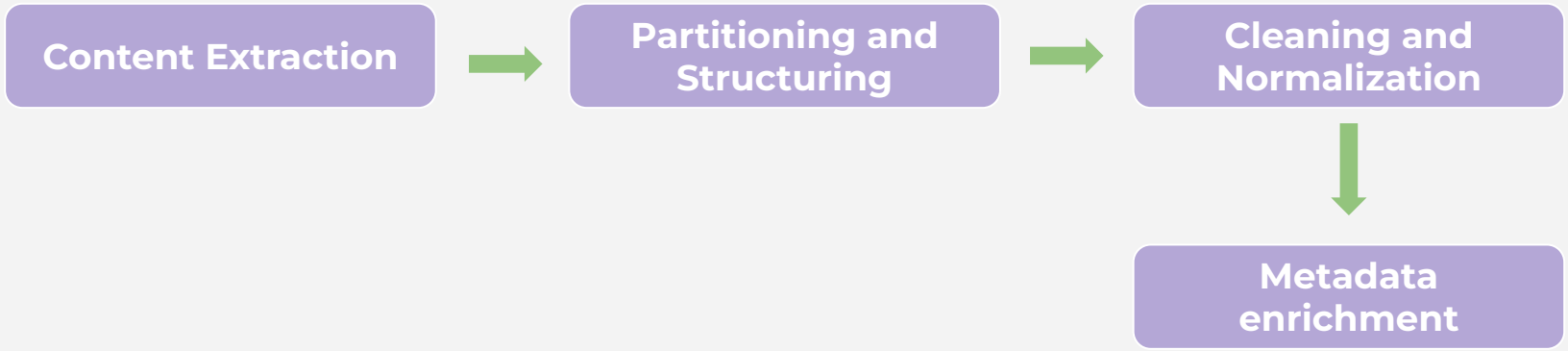
LLM Ingestion



The Unstructured framework

Streamlines *the preprocessing of* ***unstructured*** *data, making it easier to feed into LLMs for RAG systems and other AI applications*

How it works?



How it works?

Raw data sources

Content Extraction
(text, img, tables)

Partition & Structuring
(text, img, tables)

Cleaning & Normalization

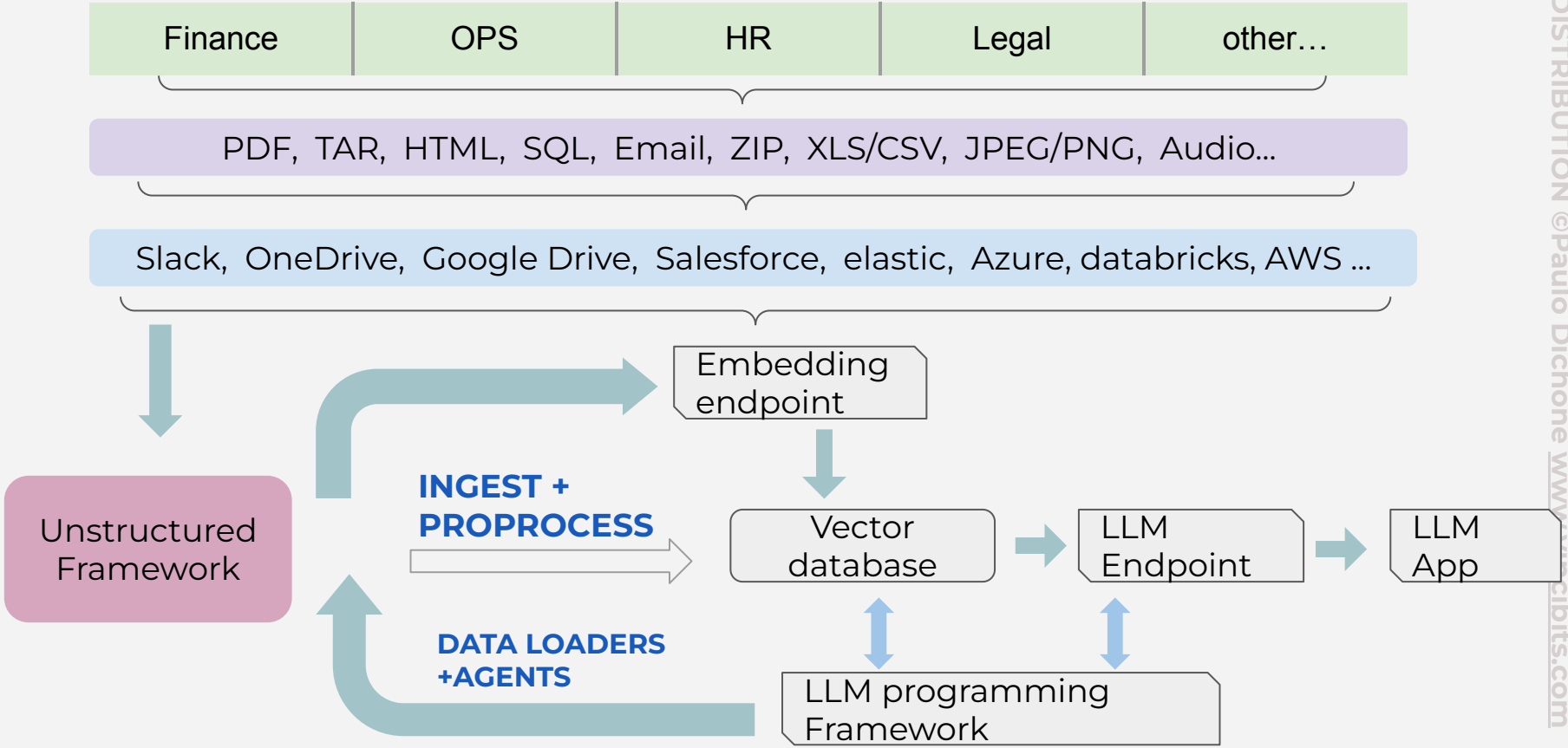
Metadata enrichment

Structured Data (JSON)

Key features

- **Multi-format support**: handles various docs types like PDFs, DOCx, and HTML
- **Smart chunking**: divides content into meaningful sections, preserving context
- **Automation-ready**: can be integrated into large-scale preprocessing pipelines

Data preprocessing and LLMs - Unstructured



Hands-on - set up unstructured

- Set up unstructured framework (API & SDK)
- Create a project - extract and normalize different types of documents using the Unstructured framework tools

Hands-on - why normalize content

Documents come in a ***variety*** of formats (PDF, Word, EPUB, Markdown, etc)

Raw data sources (various formats)

Transform into a common format

Titles

Narrative Text

Why normalize content

Benefits of normalization:

- **Uniform Processing:** Allows all documents to be processed consistently, regardless of their original format.
- **Filtering**: Removes unwanted elements like headers and footers.
- **Chunking**: Breaks down documents into manageable sections.
- **Cost Efficiency**: Initial document processing is the most resource-intensive step; normalized documents reduce costs for subsequent tasks.
- **Flexibility**: Enables experimentation with various chunking techniques without reprocessing the entire document.

Data Serialization

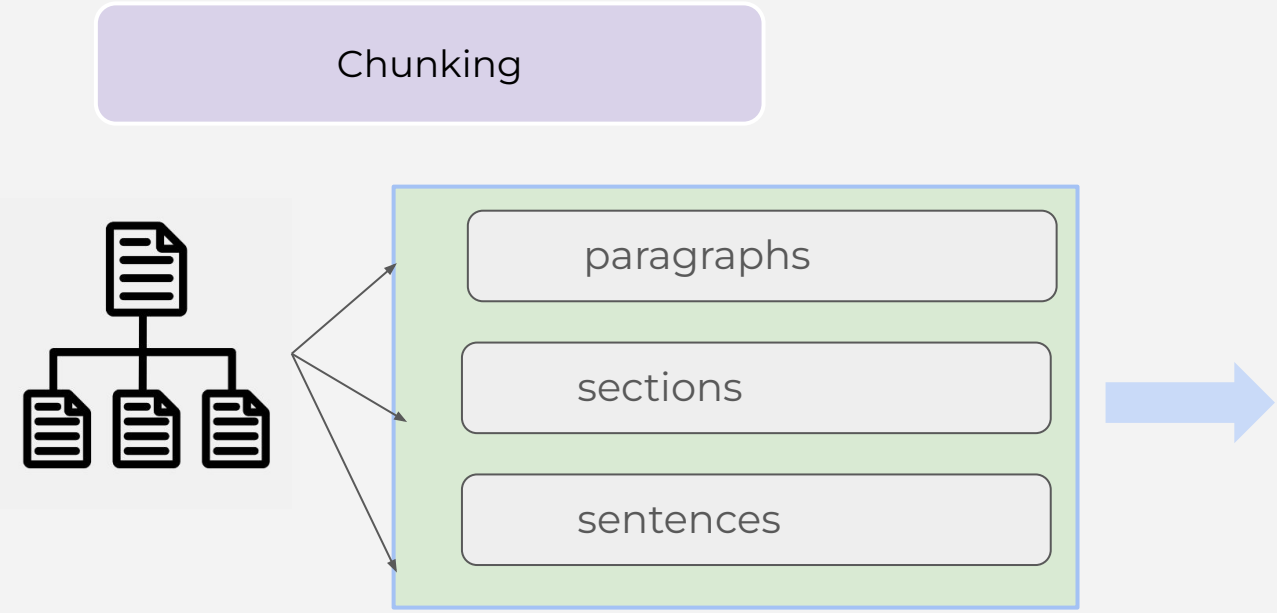
Serialization - allows you to save the results of documents preprocessing so they can be reused later, which enhances performance

Data Serialization - advantages

Advantages of Using JSON:

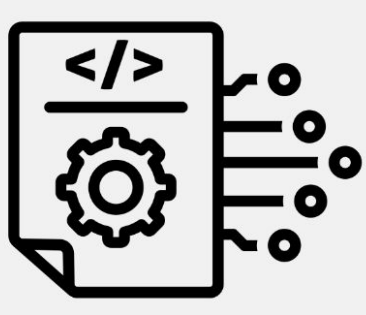
- Common and well-understood structure: JSON is widely recognized and easy to work with
- **Standard HTTP response**: JSON is the default format for web APIs
- **Versatile across programming languages**: JSON can be easily used in various programming environments
- **Streaming capabilities**: JSON can be converted to JSONL (JSON Lines) for streaming data scenarios

Content chunking and Metadata Extraction



The model (LLM) can process this!

What is metadata?



Document

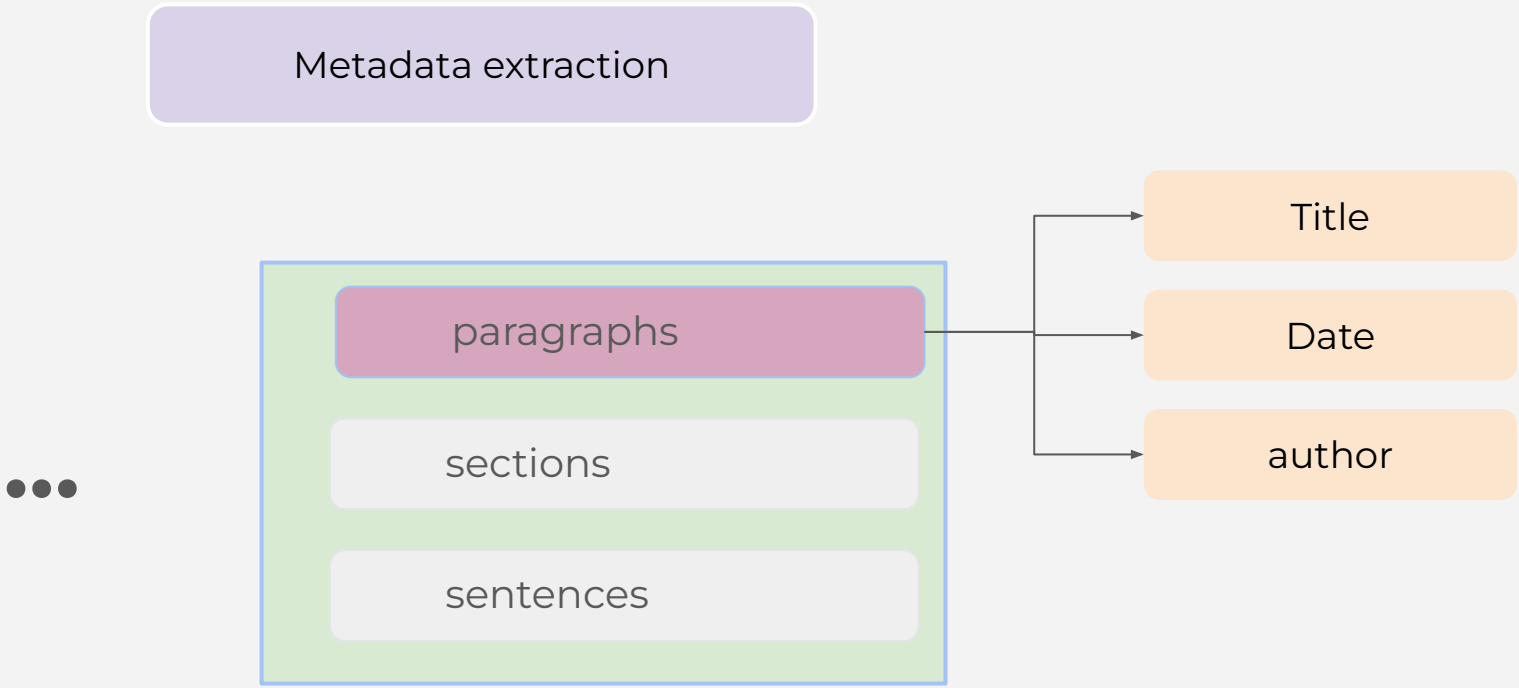
Metadata

Source Info
- Filename: “report.pdf”
- URL: “http://example.com/report
- Filetype: “PDF”

Structural Metadata
- Section: "Introduction"
- Element: "Header"
- Level: "1"
- Content: "This report covers..."

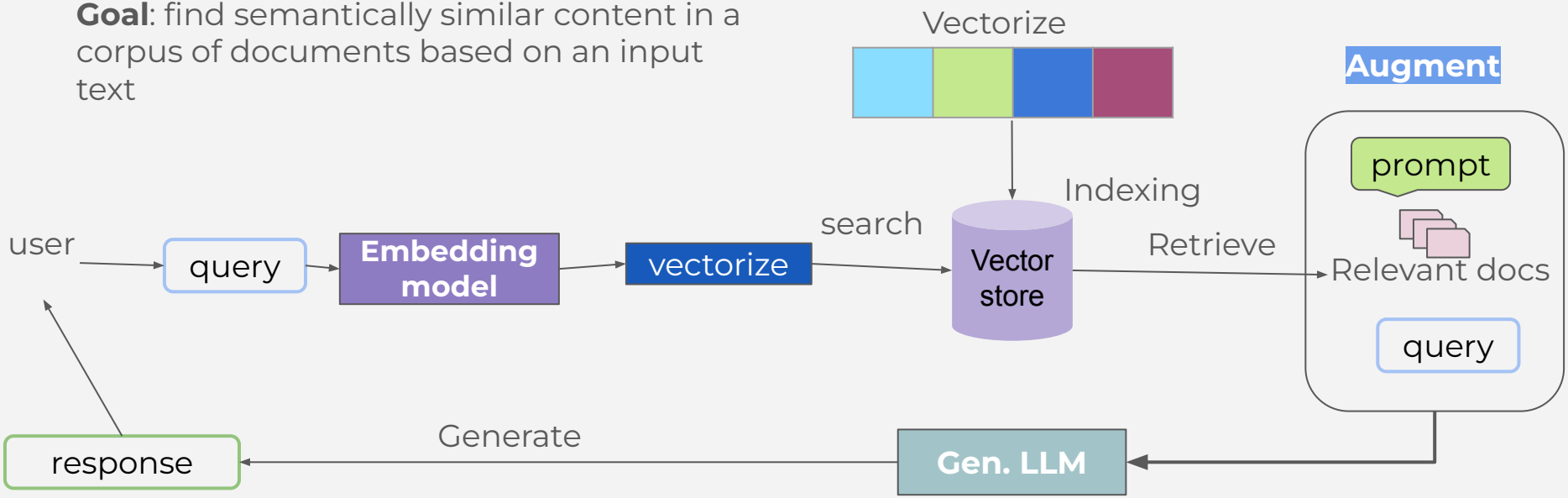
Search Metadata
- Keywords: ["report", "analysis", "data"]
- Tags: ["finance", "Q2"]

Content chunking and Metadata Extraction



Semantic similarity search for LLMs

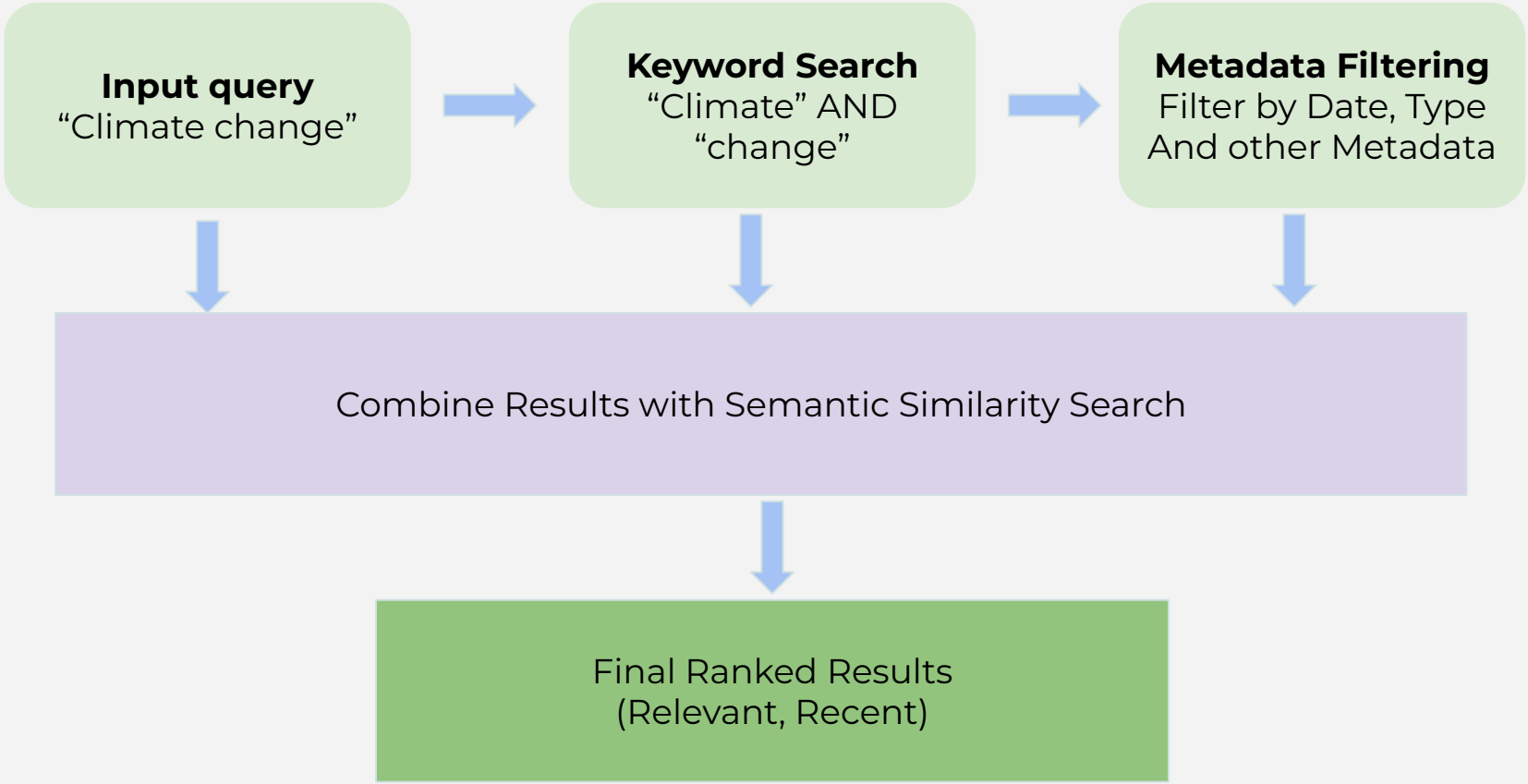
Goal: find semantically similar content in a corpus of documents based on an input text



Semantic similarity search challenges

- **Overabundance of matches-** the search may return too many semantically similar results, overwhelming the user
- **Recency preference -** user my prioritize the most up-to-date information rather than just the most semantically similar content
- **Information Loss** - Important details, like section-specific information, may be lost during the search process.

Solution: Hybrid search strategy



Hybrid search example - information recency

Without recency filter

With recency filter

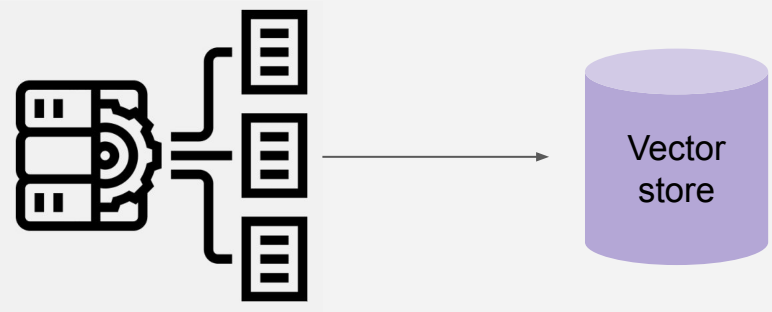
Article 1:

- **Title:** "The Impact of Climate Change on Polar Ice Caps"
- **Date:** Published in **2010**
- **Summary:** Discusses climate models from the early 2000s predicting ice cap melting.

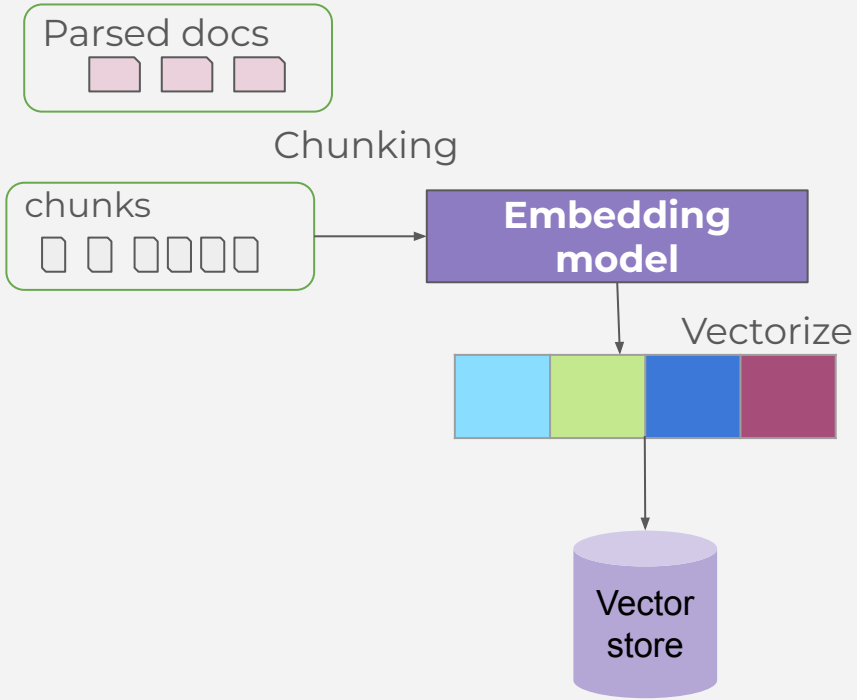
Article 2:

- **Title:** "Recent Data on Polar Ice Cap Melting"
- **Date:** Published in **2023**
- **Summary:** Presents the latest satellite data showing accelerated melting rates due to current environmental policies.

Hands-on -Load elements into a vector database

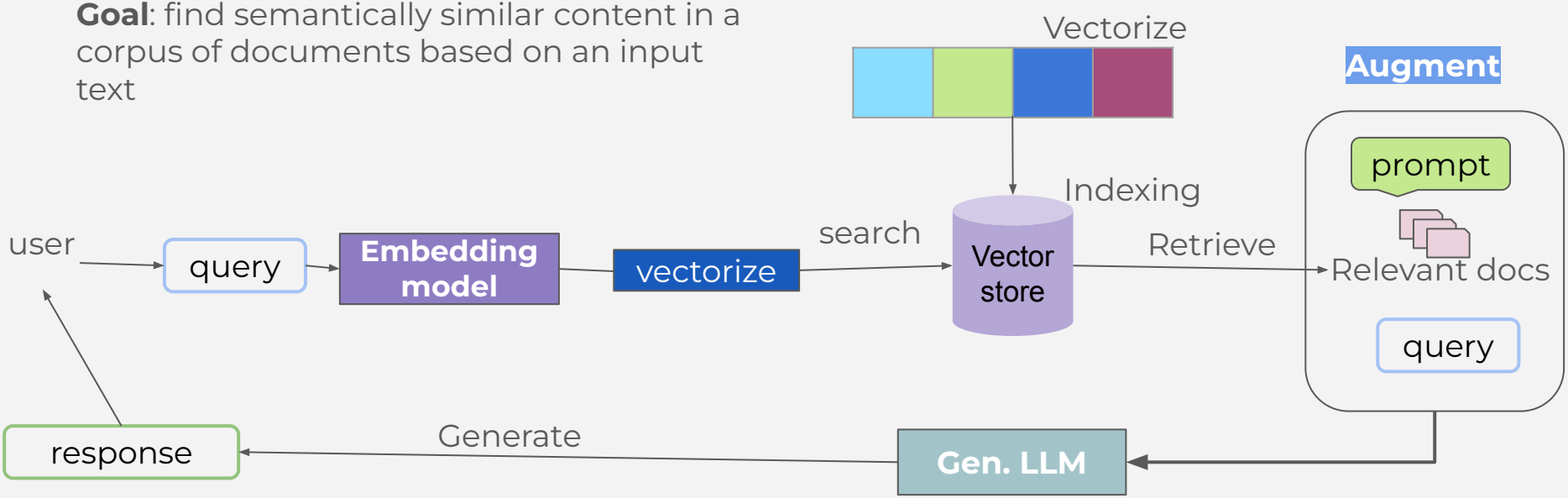


Hands-on: chunking



Hands-on: chunking

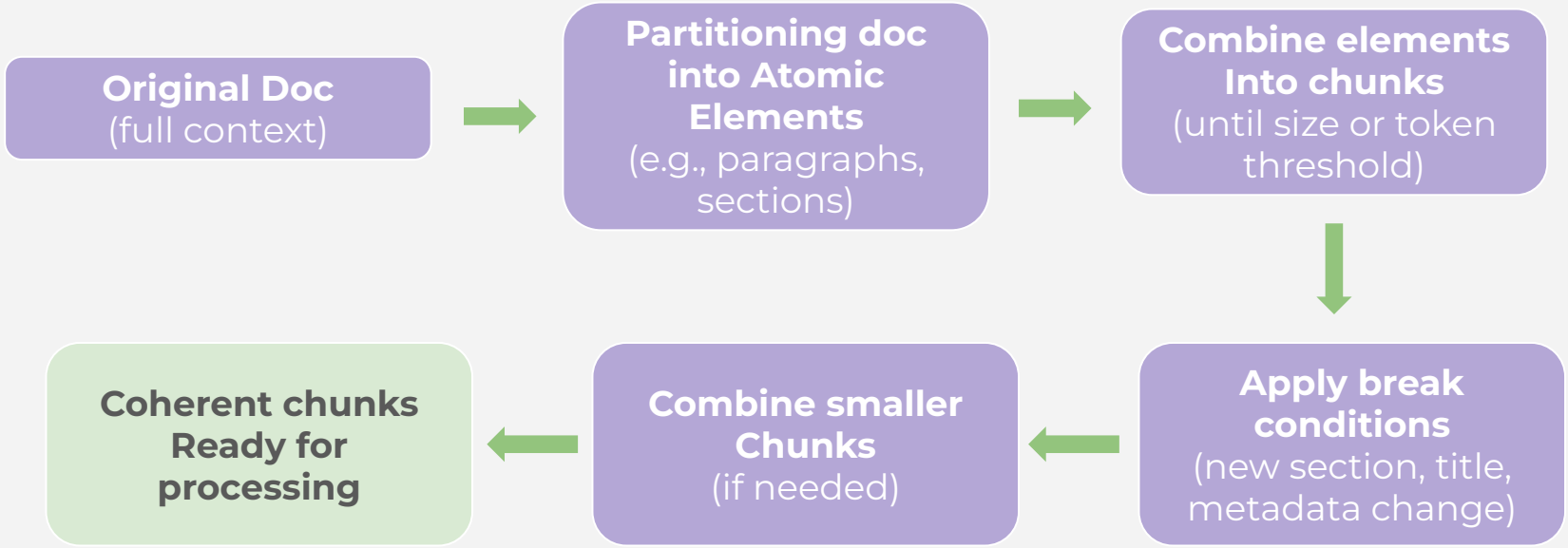
Goal: find semantically similar content in a corpus of documents based on an input text



Benefits of chunking

- **LLMs context window (limited tokens)**
  - LLMs have so much context window you can pass into an LLM
    - Chunking - allows you to pass pieces of the document, not the entire, large document the LLM can’t accept

Chunking from elements



Benefits: Chunking with Document Elements vs. Character Splitting

Character splitting

Divides text into chunks based on character counter - ***without considering the content's structure***

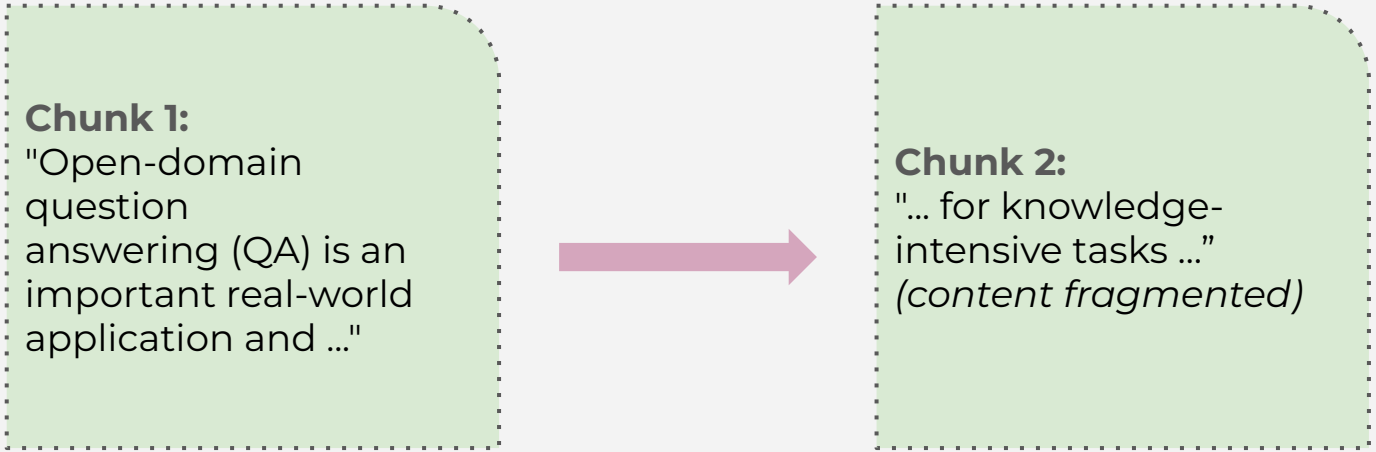
- **Outcome**: may result in fragmented chunks that split related information - incoherent, less meaningful sections

Chunking with Document Elements

Chunks are based on logical elements (paragraphs, sections, headings...)

- **Outcome**: related content stays together - more coherent and contextually accurate chunks

Character Splitting



Chunking with document elements

Chunk 1:
"3.1 Open-domain Question Answering"

"Open-domain answering (QA) is an important real-world important real-world application and ..."

Chunk 2:
"3.2 Abstractive Question Answering"

"RAG models can go beyond simple extractive QA ..."

Benefits:

- **Coherence**: logical flow of content is preserved
- **Contextual Accuracy**: chunks contain complete ideas - more meaningful during retrieval and processing

Summary

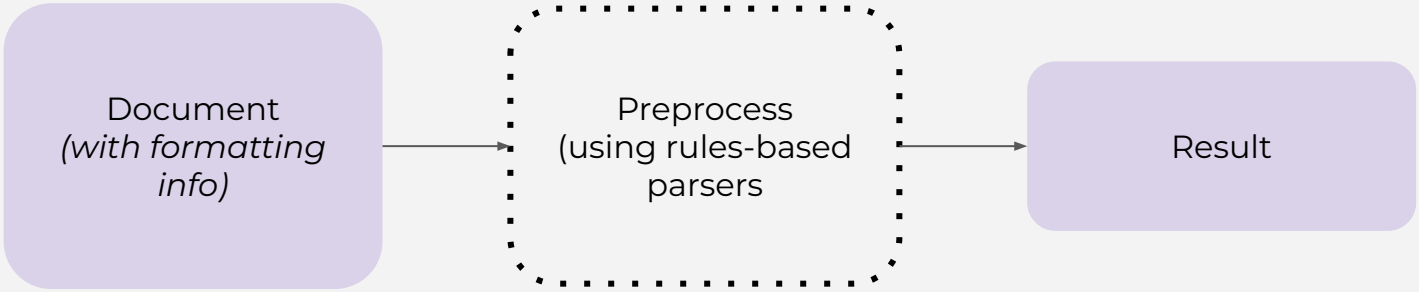
- Content chunking and Metadata extraction
  - Finding elements associated with chapters
  - Semantic similarity search & Hybrid Search advantages
  - Saving documents to a vector database
  - Semantic similarity search challenges - solution: Hybrid Search
  - Chunking for elements and benefits
  - Hands-on

Preprocessing Complex Documents

PDFs & Images

- Preprocessing PDFs and Images
- Document Image Analysis Techniques (DIA)
  - How it works?
- Key concepts
  - Document Layout Detection
  - Vision Transformers
- How to use these techniques to preprocess PDFs and Images

Preprocessing with rules-based parsers



- HTML, Word Docs and Markdown include formatting information
  - Can be preprocessed with **rules-based parsers**

What about documents with Visual Information?

Preprocessing complex documents

Visual formatting information

DIA (Document Image Analysis)

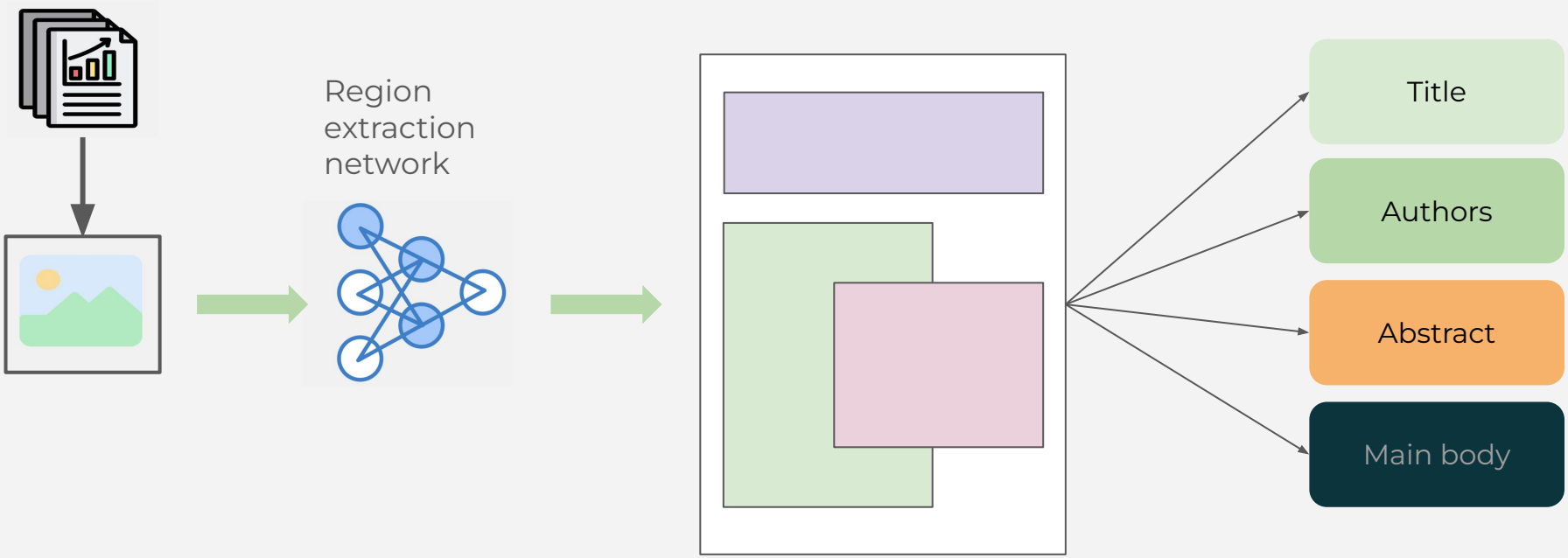
Extract formatting information and text from raw images of documents

DIA (Document Image Analysis) methods

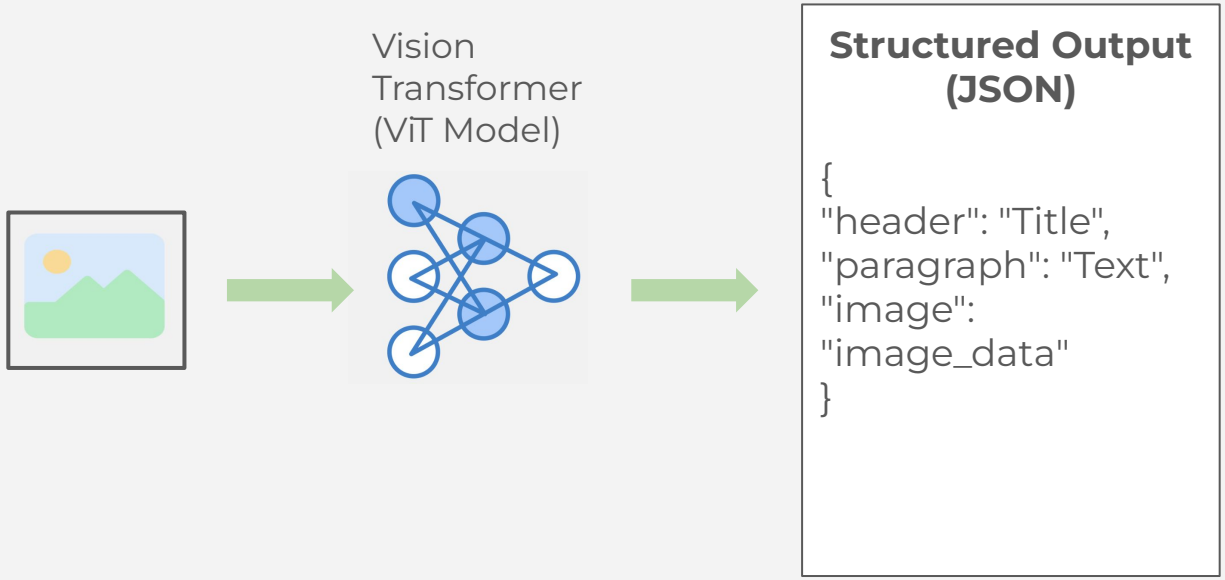
We'll focus on two DIA techniques:

- Document Layout Detection
- Vision Transformers

DLD (Document Layout Detection)



Vision Transformer (ViT)



Advantages and disadvantages

Document Layout Models

Advantages:

- Fixed set of element types for consistent processing
- Provides bounding box information for precise element positioning

Disadvantages:

- Requires two separate model calls (object detection and OCR -**Optical Character Recognition** - increases complexity
- Less flexible when dealing with non-standard document layouts

Advantages and disadvantages

Vision Transformers (ViT)

Advantages:

- Highly flexible - great for documents like forms
- Easily adaptable to new ontologies and document structures

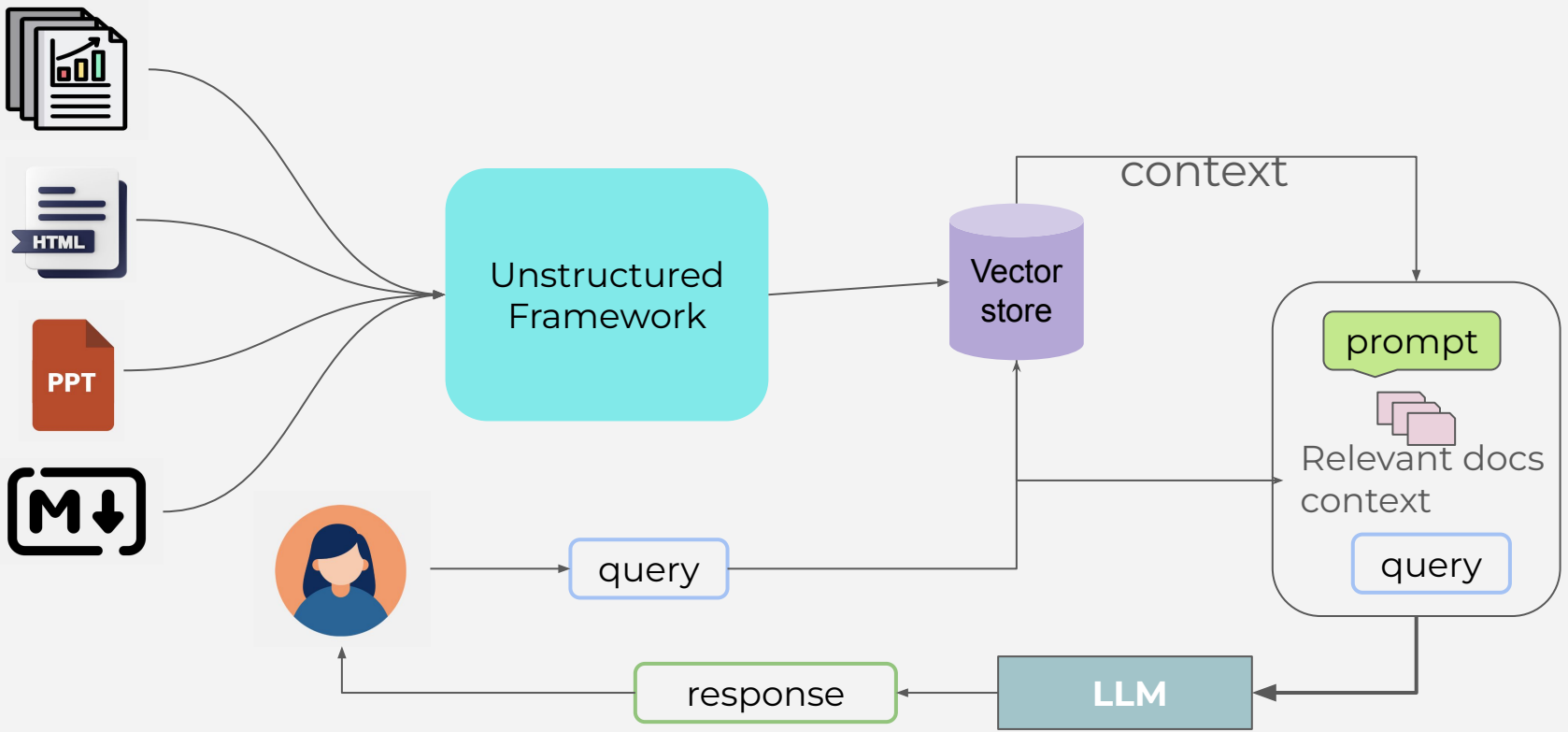
Disadvantages:

- Generative in nature - may lead to hallucination or repetition errors
- Computationally expensive - requires significant processing power

Preprocessing complex documents - Tables

Put it all together: Build a RAG Application

- Build a RAG Bot system to converse with your own data



Congratulations!

You made it to the end!

- Next steps...

Course Summary

- Ingesting and normalizing content from various data sources and types for LLM applications
  - LLM data preprocessing
    - Unstructured data to structured data
    - Key concepts
    - What problem it solves
    - Normalizing content for LLMs
  - Metadata extraction and chunking
  - Preprocessing PDFs and Images for LLMs
  - Extracting tables from complex document types
  - Best practices and advanced techniques
  - Build a RAG system from normalized content
  - Lots of hands-on (Build a Full RAG System - Chatbot)

Wrap up - Where to Go From Here?

- Keep learning
  - Extend the projects we worked on in this course
  - Design and implement your own RAG Systems and Applications using the Unstructured Framework
- https://docs.unstructured.io/welcome

Thank you!