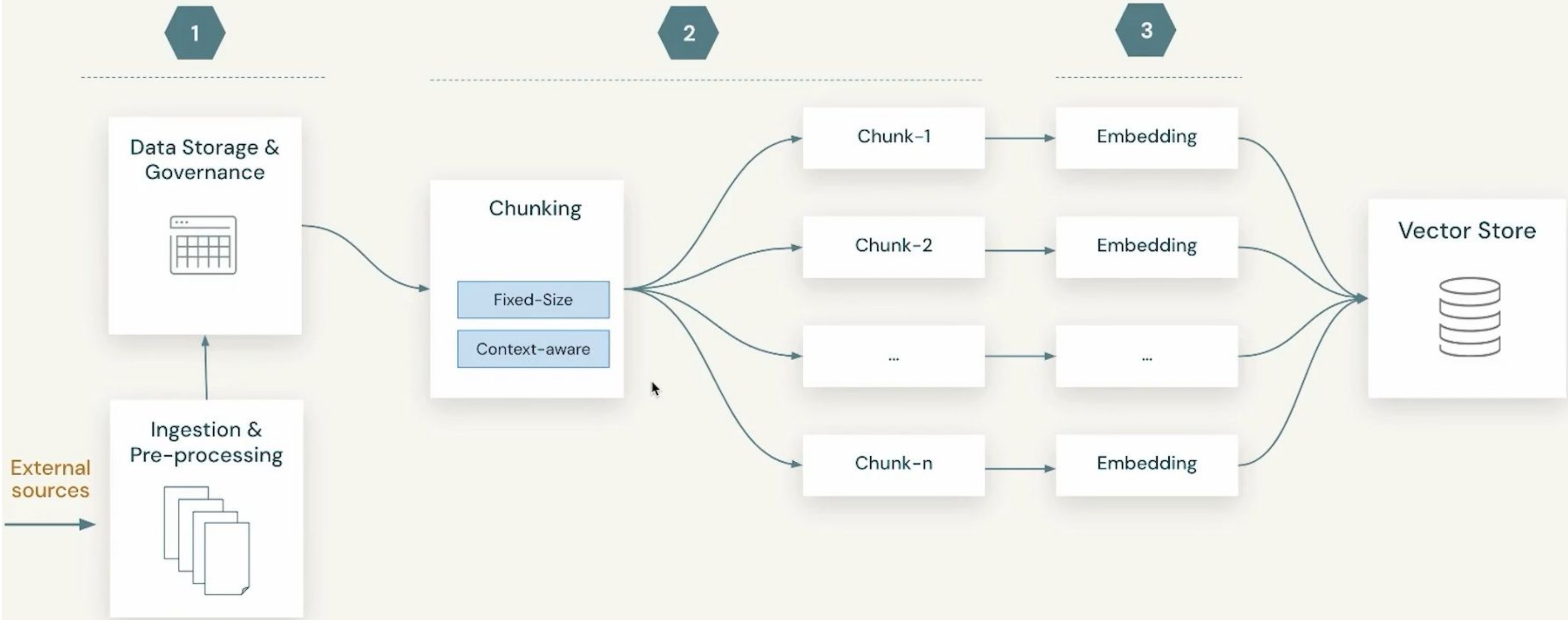
Why is Data Prep Important for RAG?

Potential **issues** when data is prepared improperly

- **Poor quality model output:** If data is inaccurate, incomplete, or biased, the RAG system is more likely to produce misleading or incorrect responses.
- **"Lost in the middle"**: In long context, LLMs tend to overlook the documents placed in the middle.(Related Research Paper and needle in haystack test repo).
- **Inefficient retrieval**: Poorly prepared data would decrease the accuracy and precision of retrieving relevant information from knowledge base.
- **Exposing data**: Poor data governance could lead to exposing data during the retrieval process.
- **Wrong embedding model**: Wrong embedding model would decrease the quality of embeddings and retrieval accuracy.

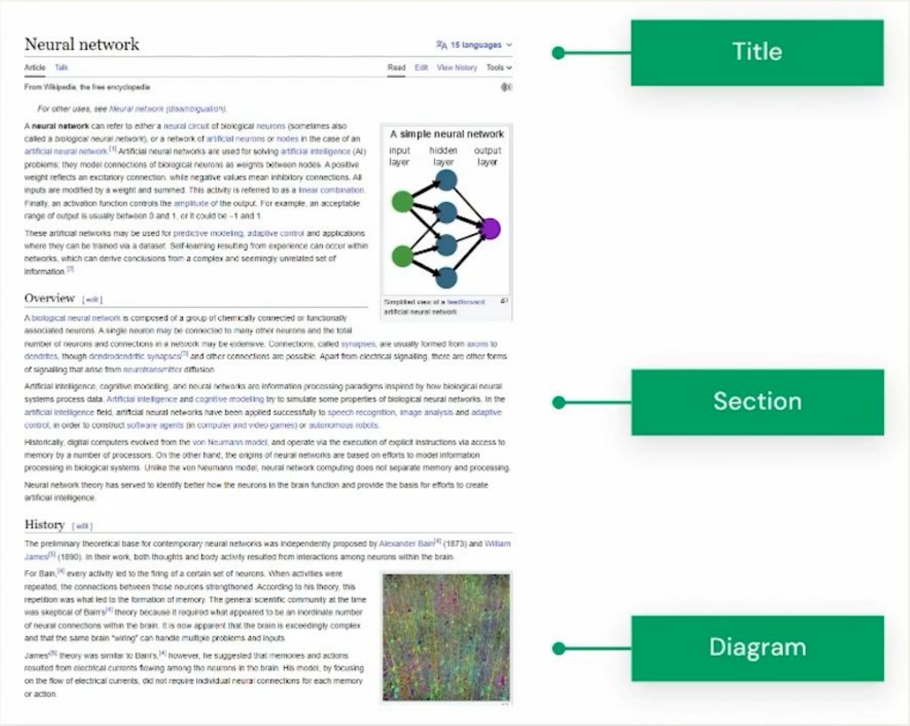
Data Prep Process Overview

A simple data prep process



How to Chunk Data?

How should we organise it?



Context-aware Chunking:

- Chunk by sentence/paragraph/section
- Leverage special punctuation (i.e. '.', '\n')
- Include/Inject metadata/tags/title(s)

&/OR

Fixed-size Chunking:

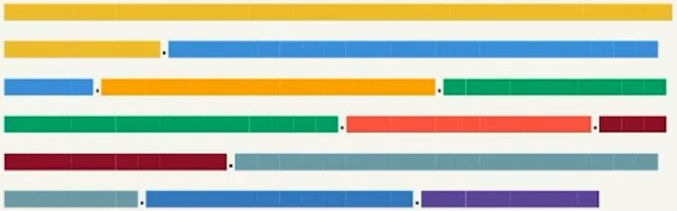
- Divide by a specific number of tokens
- Simple and computationally cheap method

Chunking Strategy is Use-Case Specific

Another iterative step! Experiment with different chunk sizes and approaches

- How long are our documents?
  - 1 sentence?
  - N sentences?
- If 1 chunk = **1 sentence**, embeddings focus on specific meaning
- If 1 chunk = **multiple paragraphs**, embeddings capture broader theme
  - How about splitting by headers?

Chunking by sentence:



Chunking by Paragraph:

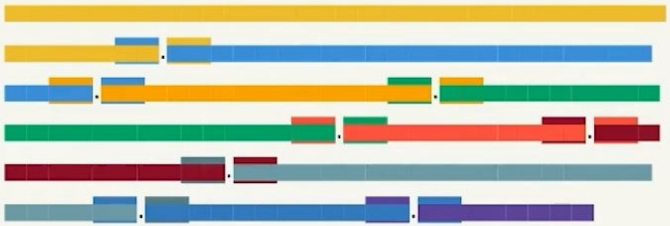
Use ChunkViz helper tool

Chunking Strategy is Use-Case Specific

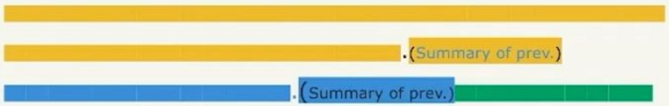
Another iterative step! Experiment with different chunk sizes and approaches

- Chunk **overlap** defines the amount of overlap between consecutive chunks, ensuring that no contextual information is lost between them.
- **Windowed summarization** is a ‘context-enriching’ chunking method where each chunk includes a ‘windowed summary’ of previous few chunks.

Chunk overlap:



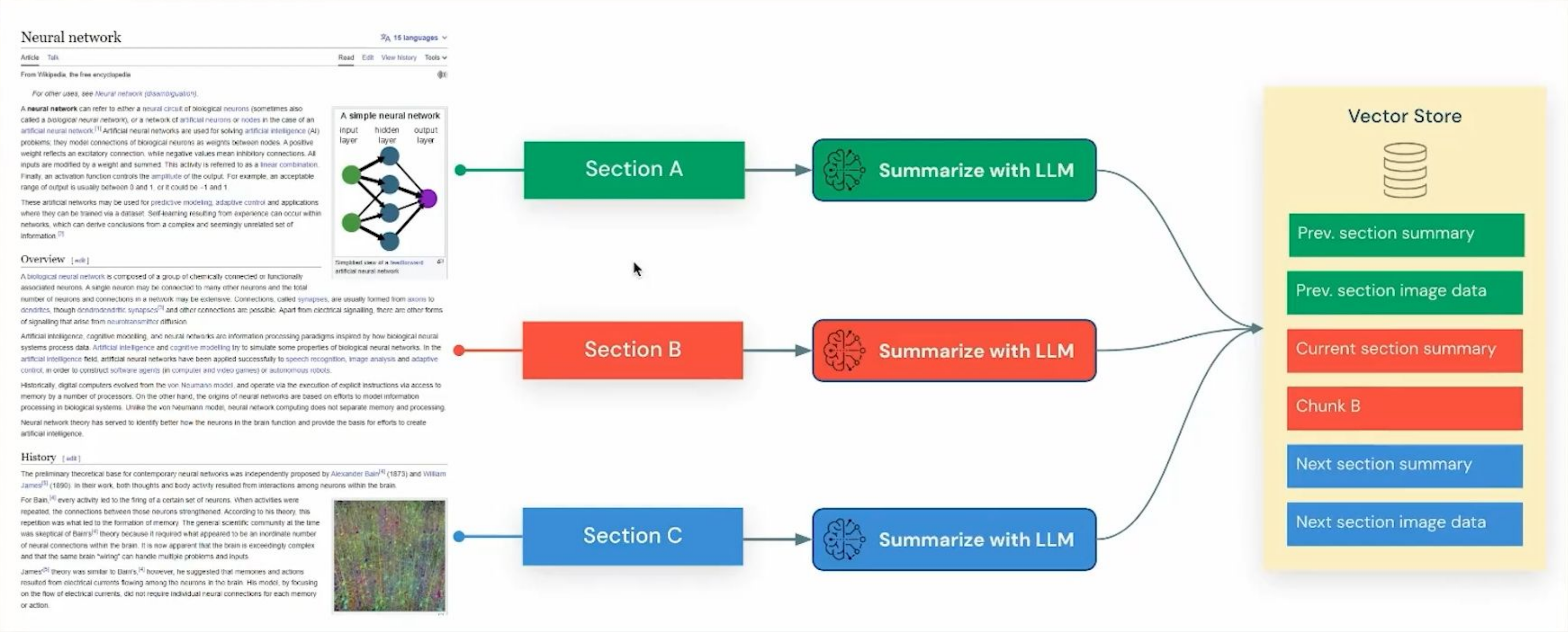
Windowed summarization:



- Prior knowledge of user’s query patterns can be helpful *(i.e. query length?)*
  - While long queries may have better aligned embeddings to returned chunks, shorter queries could be more precise

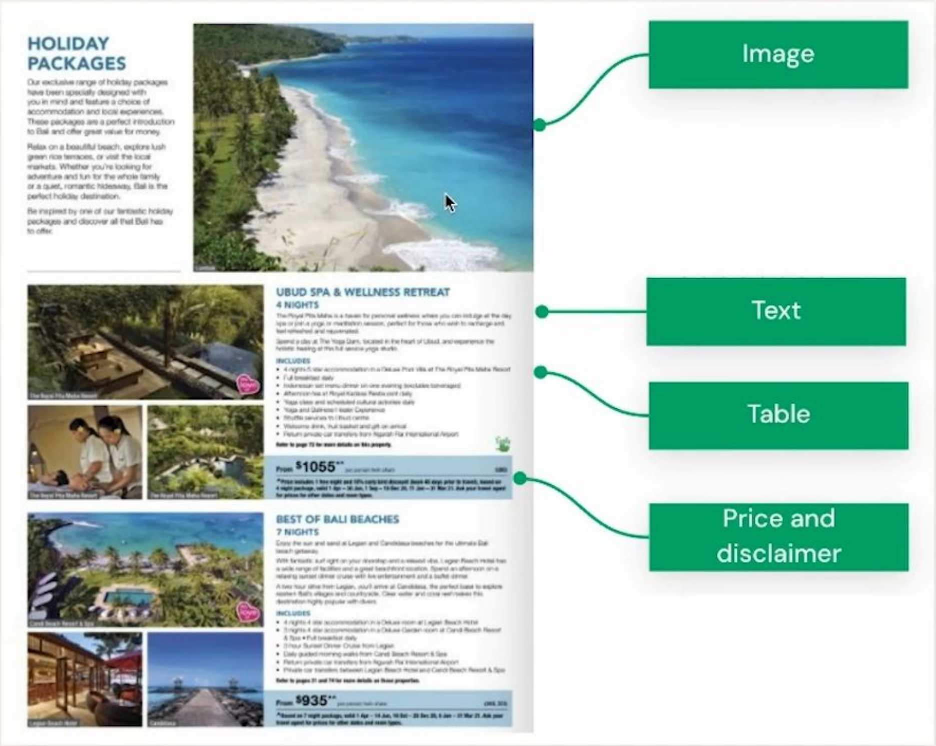
Advanced Chunking Strategies

Summarization with metadata



Data Extraction and Chunking Challenges

Working with complex documents

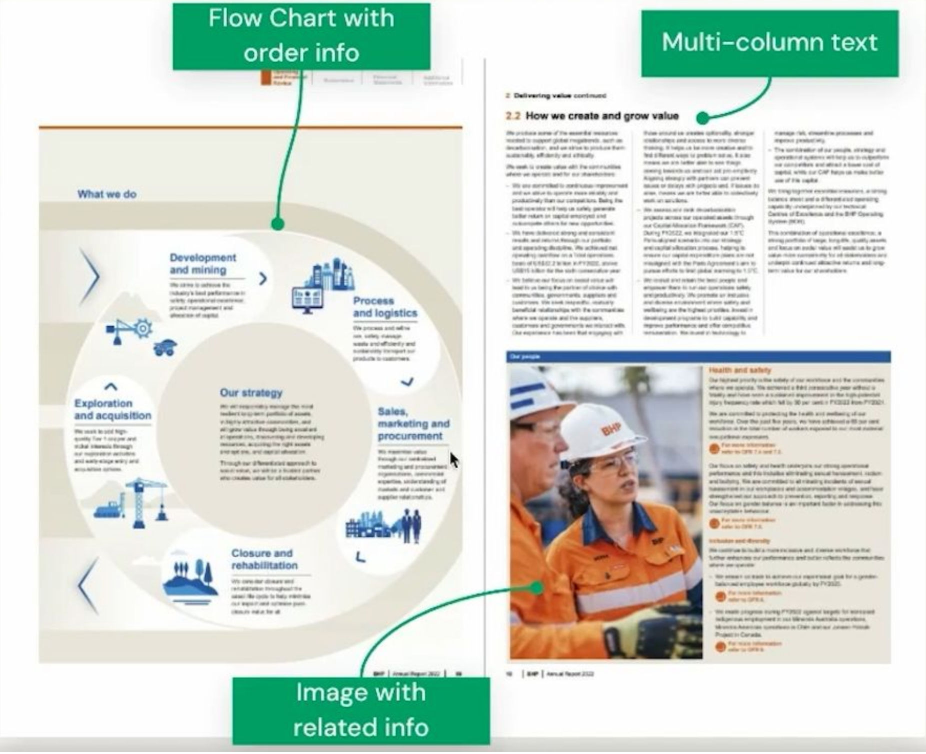


Other challenges:

- Text mixed with image
- Irregular placement of text
- Color encoded focus *(Important for context)*

Data Extraction and Chunking Challenges

Working with complex documents



Other challenges:

- Chart with hierarchical information. Keeping the order of the information is critical.
- Multi-column text and the order of columns if crucial.
- Keeping images with related information is crucial.

General Approaches

Approaches to address unstructured/complex raw text documents

Traditional Approach

Libraries:

- PyMuPDF
- PyPDF

Features:

- Breaks down text to into raw constructs
- Very low level requires hard coding rules

Use a layout model

Libraries:

- Hugging Face
  - LayoutLMv3
- doctr
- Donut
- Unstructured

Features:

- Apply Deep learning models built to do text extraction and context extraction

Multi-Modal Models

Models:

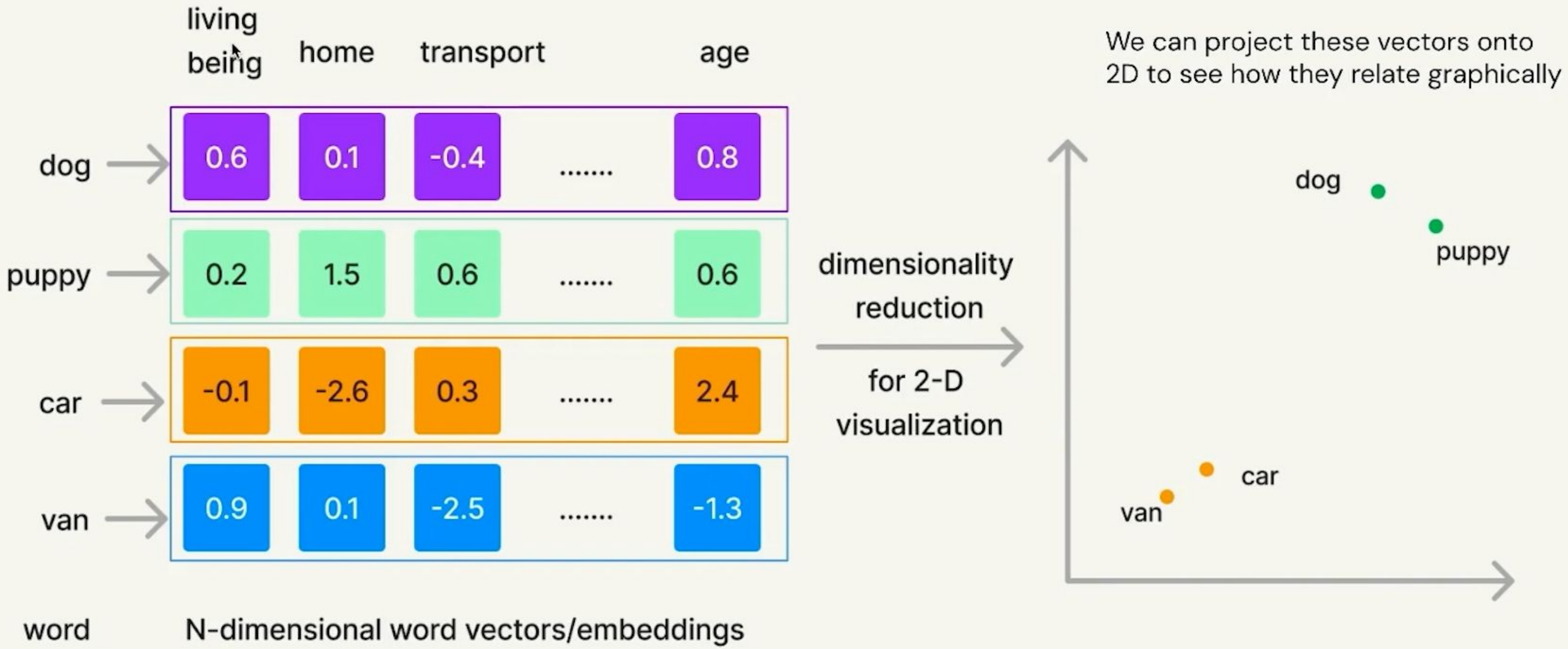
- OpenAI's GPT-4o (and beyond)
- Alphabet's Gemini1.5 (and beyond)
- Other OSS models (i.e. Dolphin's Series, OpenFlamingo, Llava, OLMo)

Features:

- Multimodal LLMs intrinsically understand images but are still more experimental at this stage

Refresher: Representing Words with Vectors

Embedding: A numerical representation of content



Embedding Models

Choosing the **right model for your application**

- Data/Text properties:
  - Vocabulary size in your text/documents (some models handles more diverse words)
  - Domain/Topic (i.e. finance, medical, news etc.)
  - Text length: typical length of chunks/docs to be embedded
- Model capabilities:
  - Multi-Language support
  - Embedding dimensions/size: more storage cost for higher dimensions

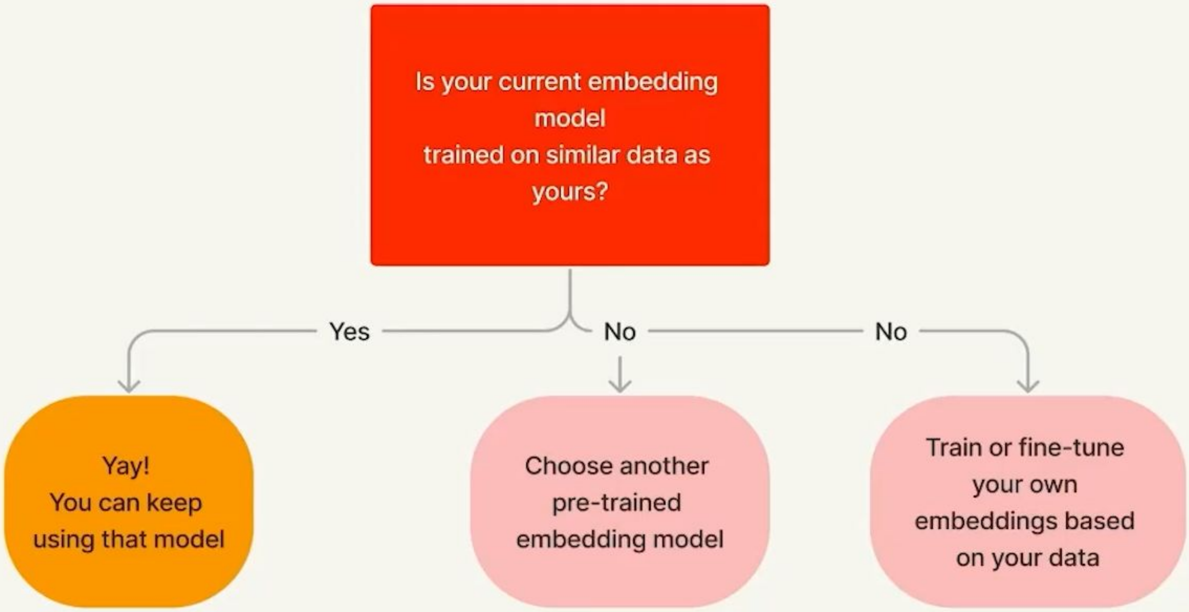
Practical considerations:

- Be aware of context window limitations. Many embedding models will **ignore text beyond their context window limits**.
- Privacy and cost/licensing when using proprietary API-based models.

⇒ benchmark multiple models & choose the one that strikes the best balance.

Tip 1: Choose Your Embedding Model Wisely

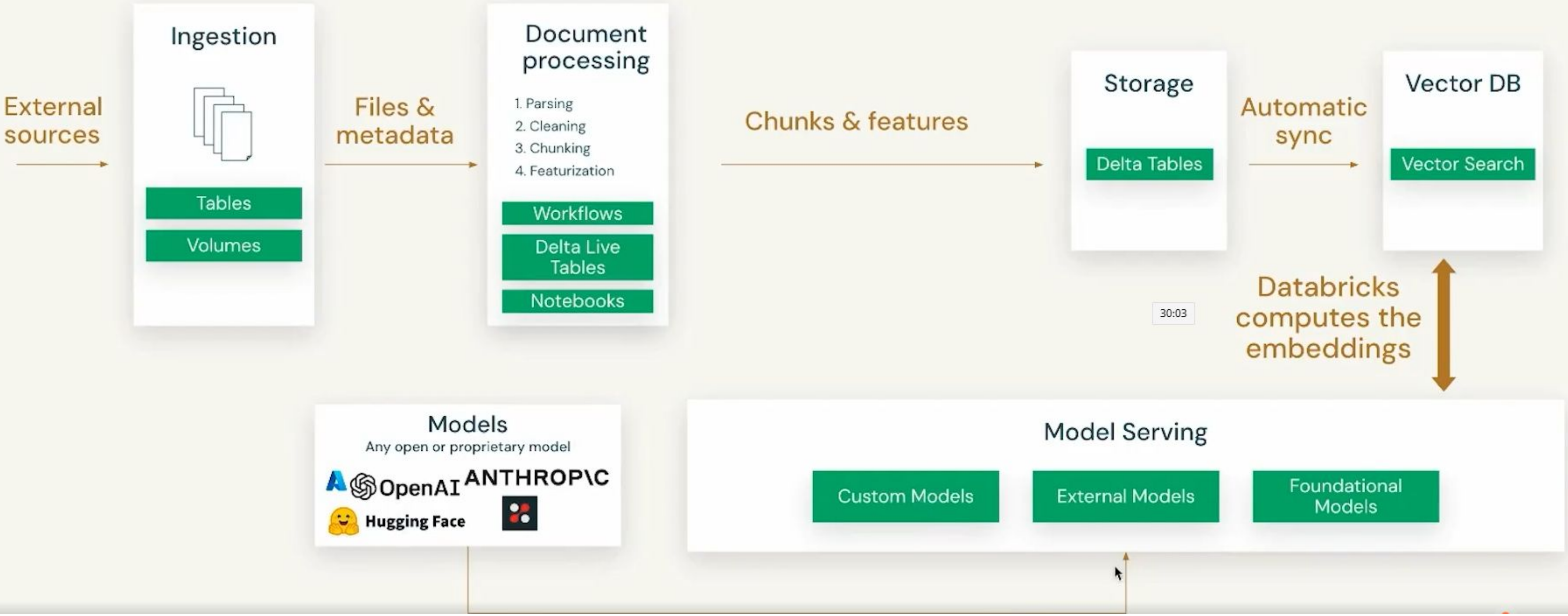
The embedding model should represent **BOTH queries and documents**



This practice has been around for years in NLP.
Example: Fine-tune BERT embeddings

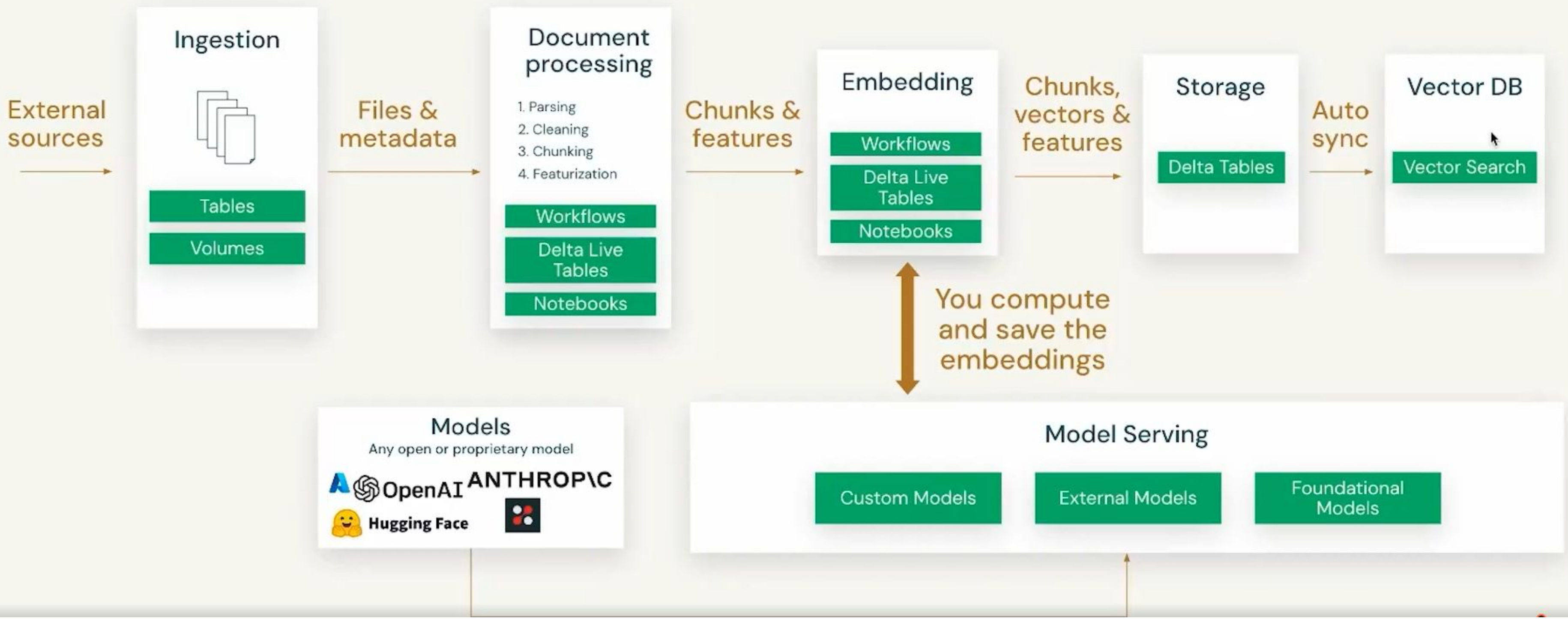
Unstructured Data Prep

Vector Search with Databricks-managed embeddings



Unstructured Data Prep

Vector Search with user-managed embeddings



Structured Data Prep

Feature Serving and Online Tables