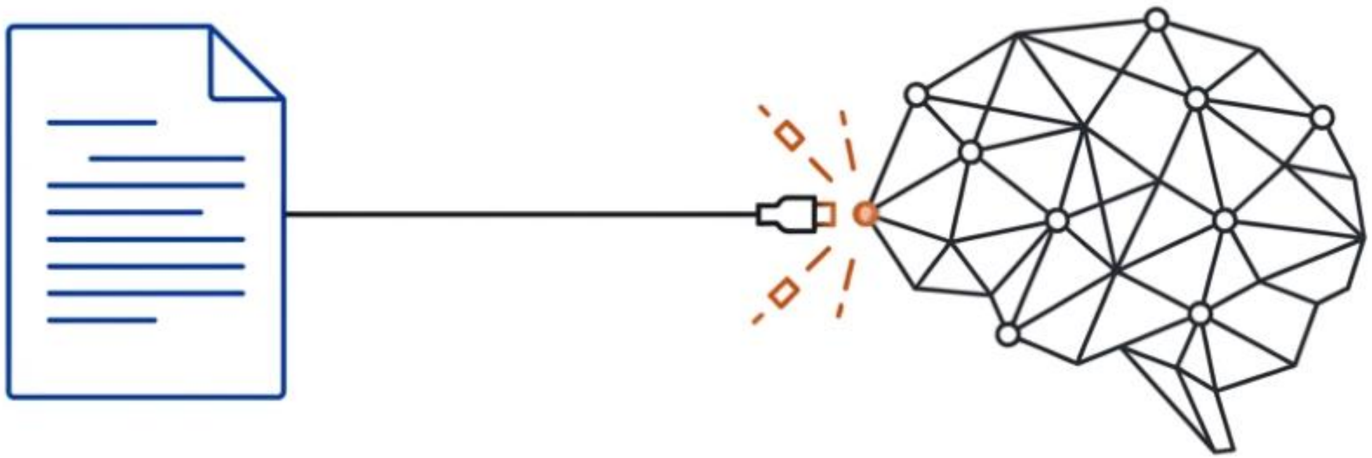


Introduction to RAG

Introducing RAG (What/Why/When) + RAG vs Traditional LLMs + Core Components (overview). Define RAG and purpose. Why retrieval helps. When RAG beats plain LLMs. High-level roles of retriever, store, generator.

Sakharam Gawade
AI Productization / Platform Intelligence Team

https://tinyurl.com/rag-augmented

We use LLMs daily

Learn/Explain
Functioning as a quick tutor for new concepts.

Debug
Interpreting errors and fixing code.

Write/Rewrite
Drafting emails, documentation, and posts.

Summarize
Digesting PDFs, articles, and meeting notes.

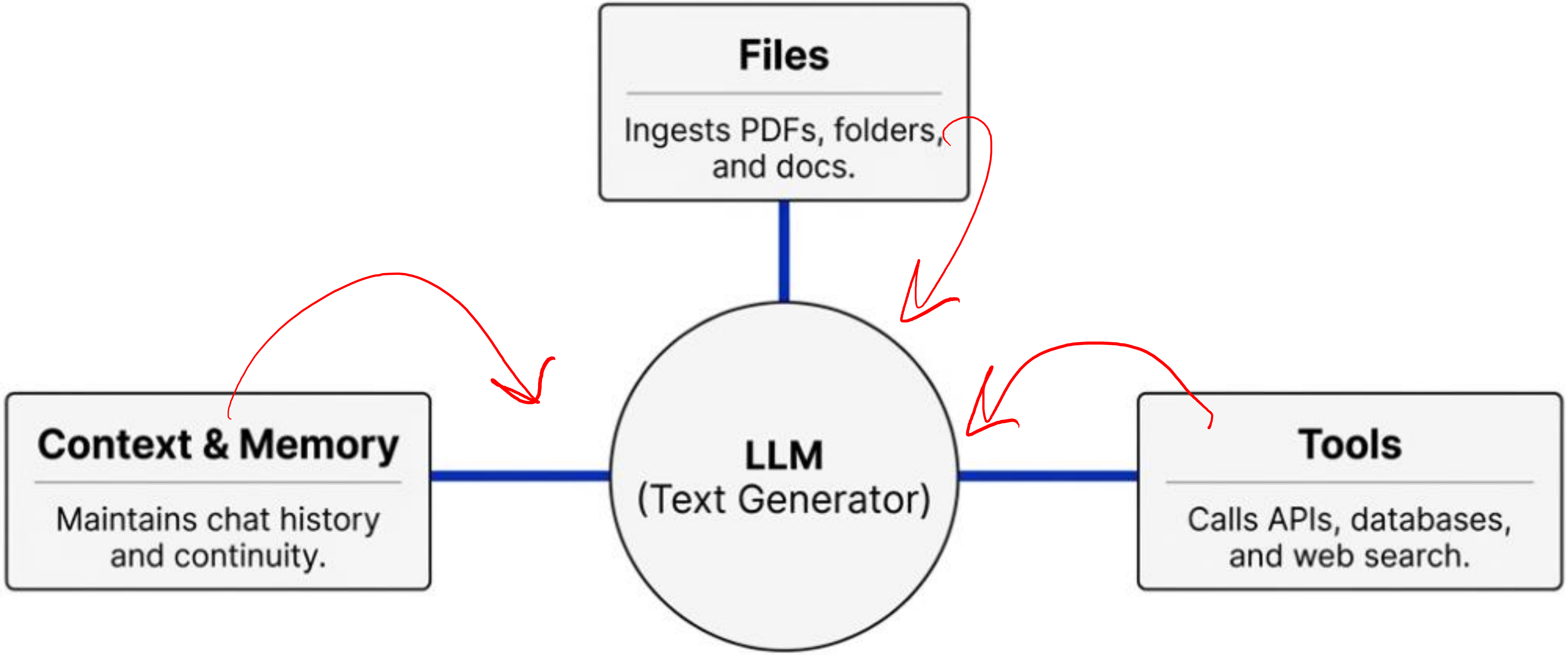
Search-like Q&A
Fact lookups and option comparisons.

Plan
Creating itineraries, study plans, and workflows.

Generate
Creating images, captions, and prompts.

Convert
Transforming formats (JSON, YAML, SQL).

Modern assistants are systems, not just text generators



Key Insight: The model is no longer isolated; it reaches outside its training weights.

The fundamental limitations of a static Model

1. Training Data Gaps

Facts may be missing, rare (long-tail), or domain-specific.

2. Private Knowledge

The model does not know your internal policies, runbooks, tickets, or codebase documentation.

3. Freshness

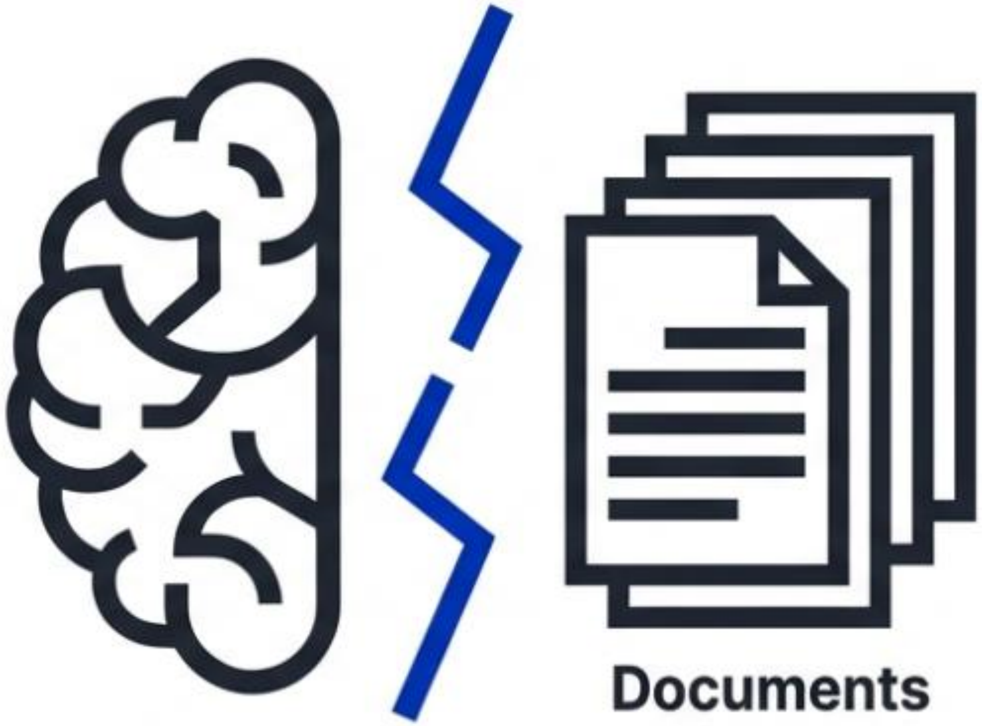
Models are frozen in time. They do not know about recent API changes or new product versions.

4. Context Window

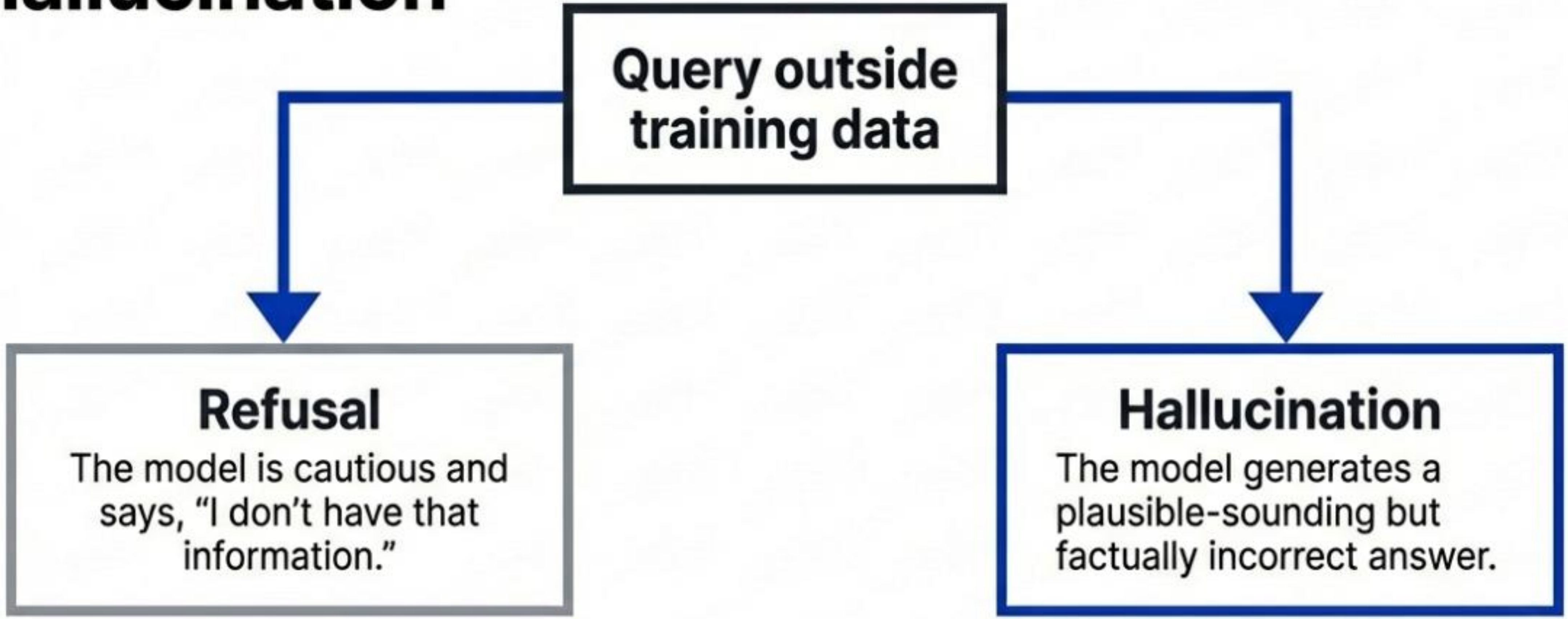
You cannot simply "paste" an entire corporation's knowledge base into a prompt. It doesn't scale.

5. No Built-in Truth

A plain LLM cannot naturally cite sources or point to evidence.



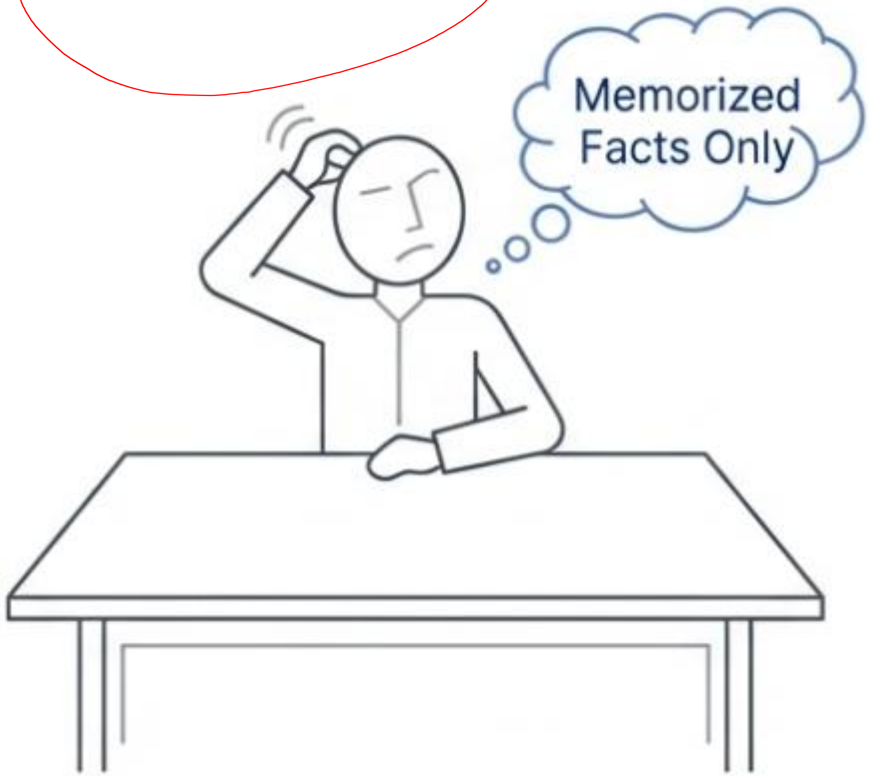
The Consequence: Silence or Hallucination



The Enterprise Failure Mode: A fluent answer + wrong details + no traceability.

The Goal: We need to make the assistant grounded in actual sources.

Closed-Book Approach



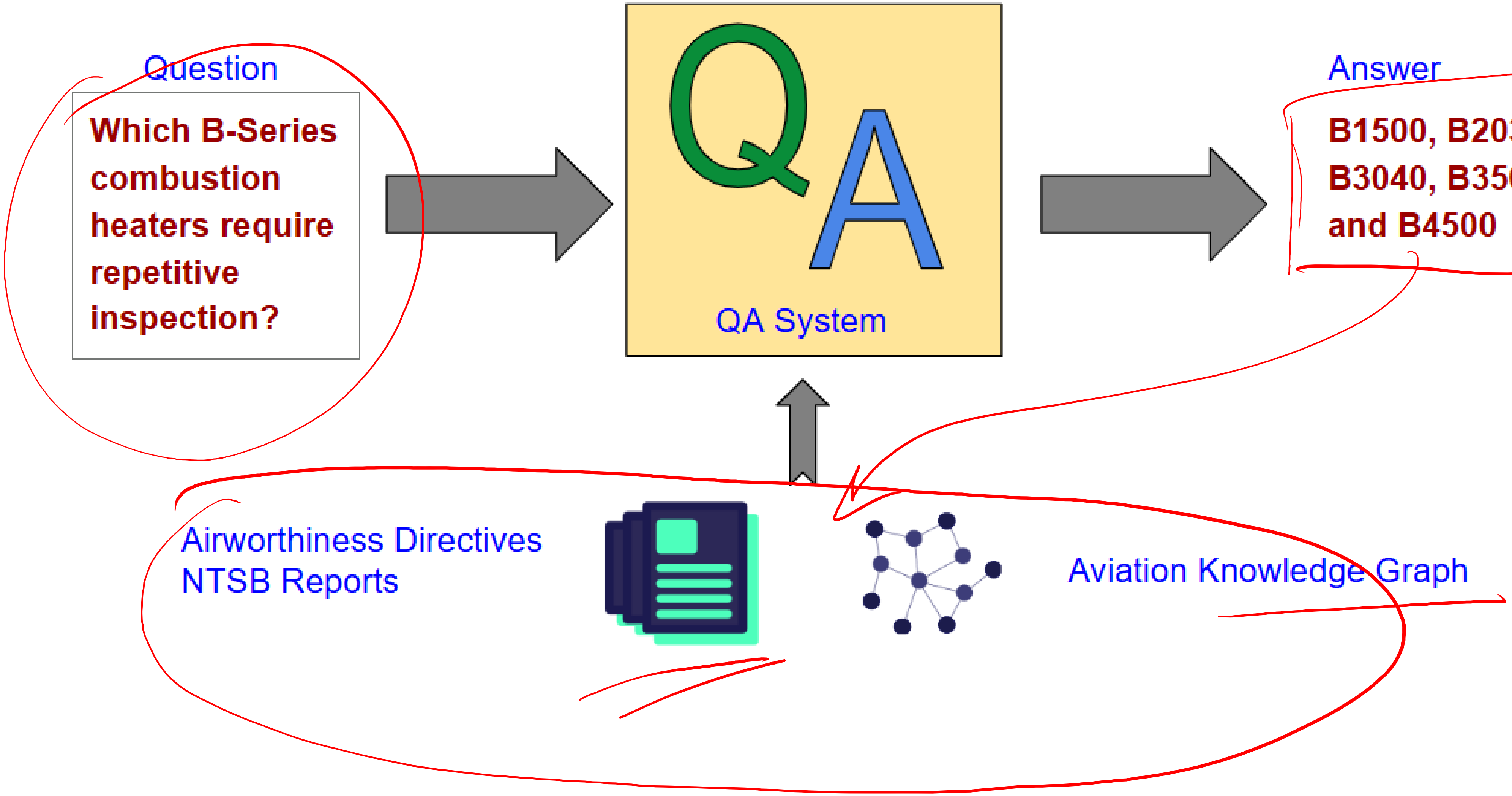
Answering from memory (Weights).
High risk of guessing.

Open-Book Approach (RAG)

Looking up info first, then answering.

RAG is the standard pattern for open-book answers over documents.

Aviation IR System



S

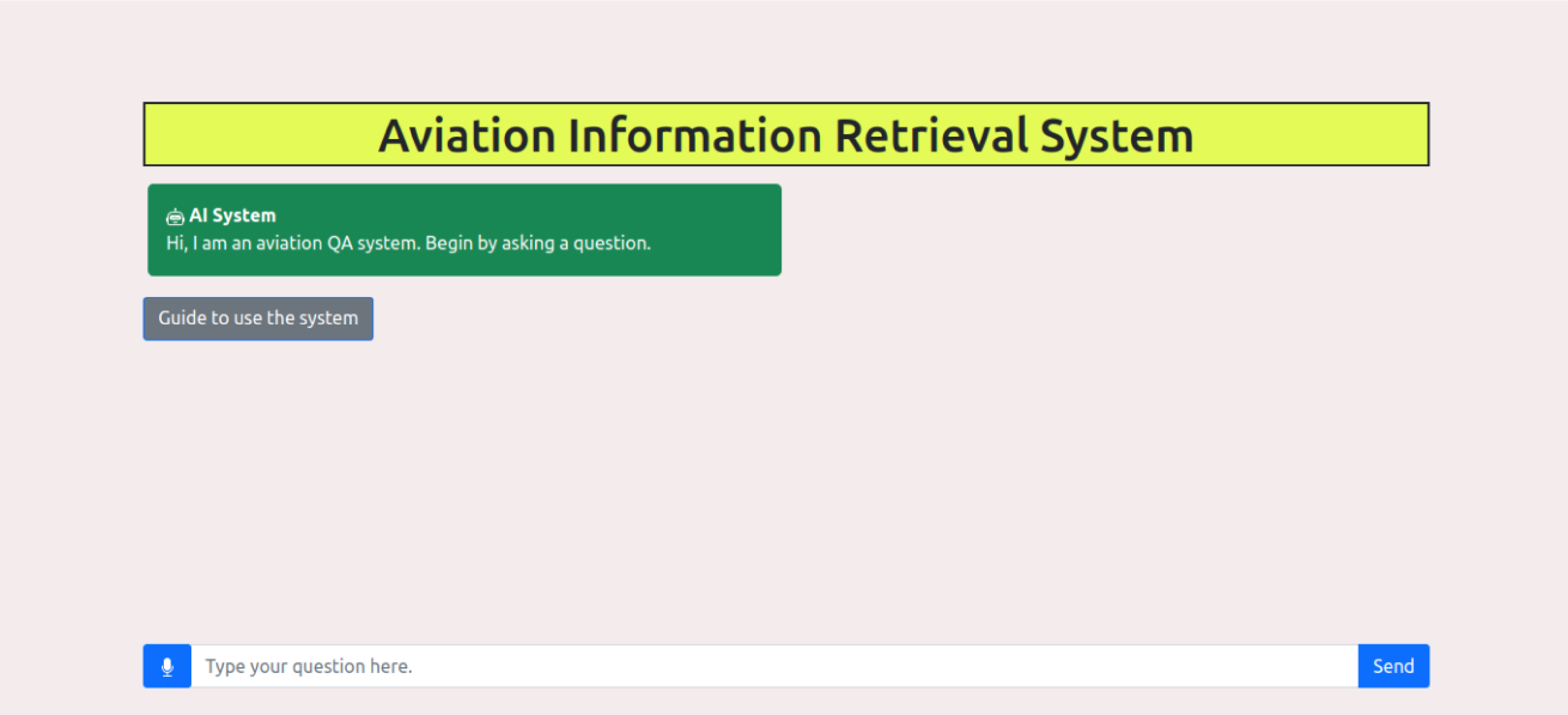
Which B-Series combustion heaters require repetitive inspection?

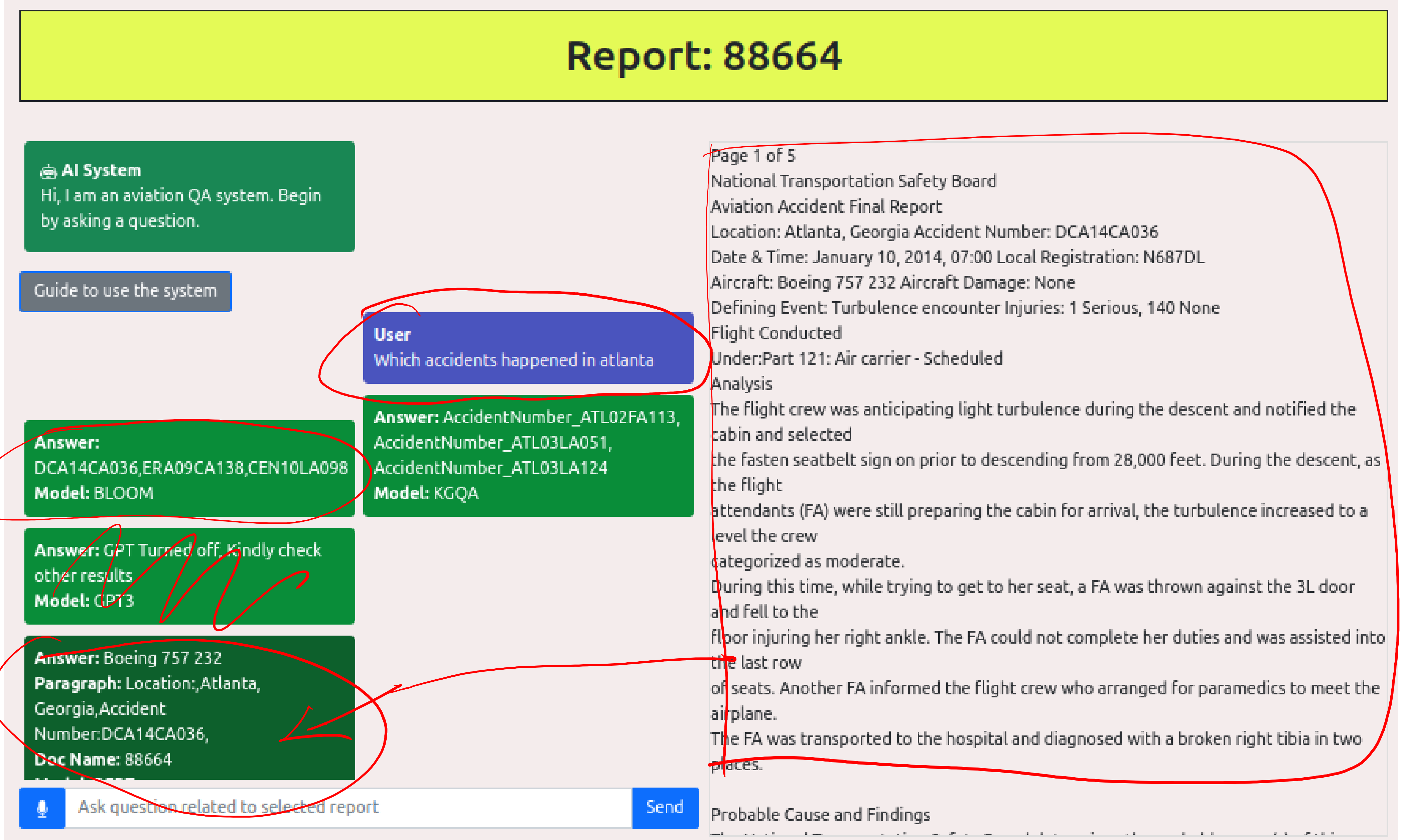
As of my knowledge cutoff in September 2021, there are no specific B-Series combustion heaters that require repetitive inspection universally. The need for repetitive inspection can vary depending on the specific model, brand, and applicable regulations or safety standards in different countries or regions.

However, it's important to note that regular maintenance and inspection are generally recommended for combustion heaters to ensure safe and efficient operation. The manufacturer's guidelines and local regulations should be followed to determine the appropriate inspection intervals and procedures for a specific B-Series combustion heater.

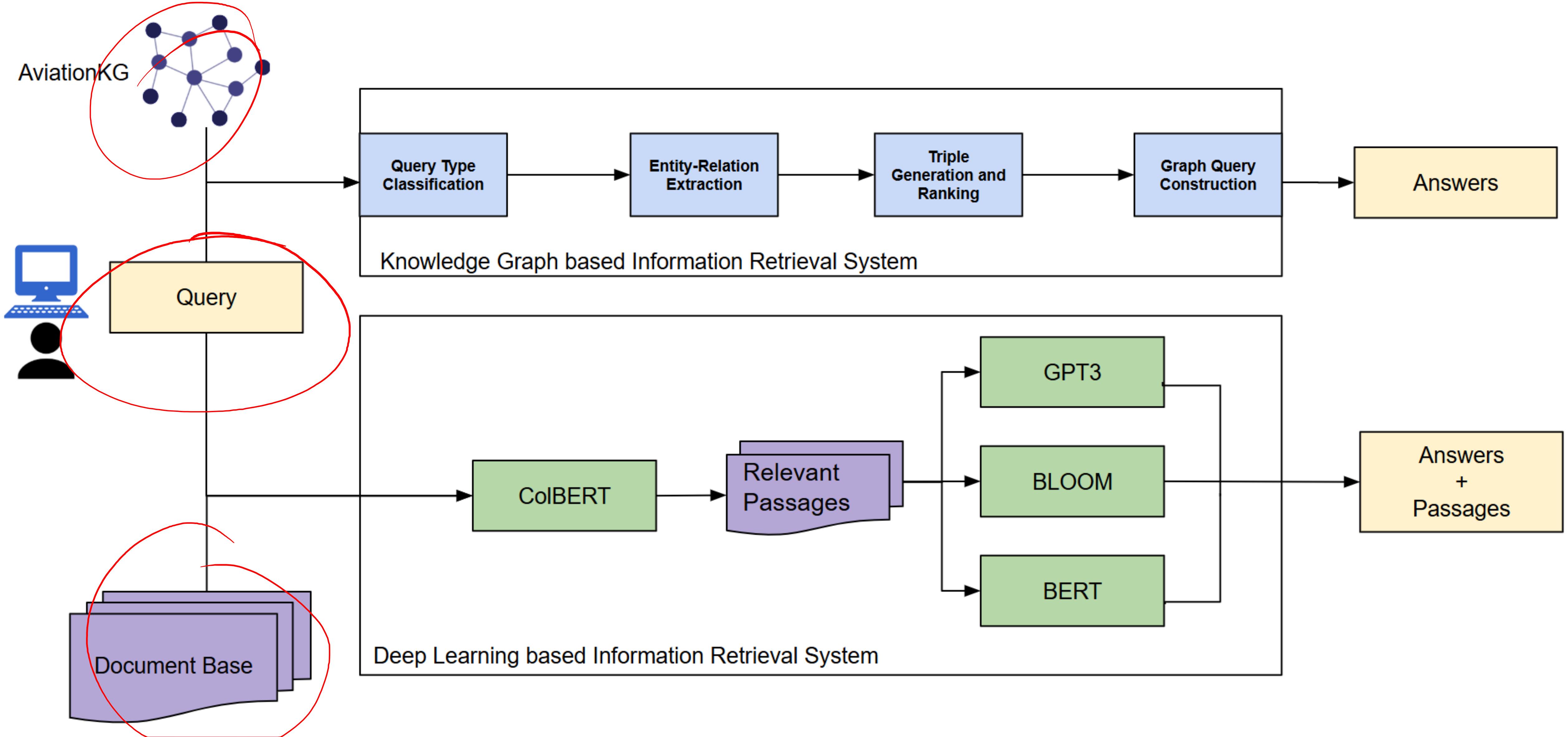
To obtain the most accurate and up-to-date information on which B-Series combustion heaters require repetitive inspection, I recommend consulting the manufacturer's documentation or contacting the manufacturer directly. They can provide detailed information about the maintenance and inspection schedules for their specific products.

Aviation IR System




Report: 88664
AI System
Hi, I am an aviation QA system. Begin by asking a question.
Guide to use the system
User
Which accidents happened in atlanta
Answer: AccidentNumber_ATL02FA113, AccidentNumber_ATL03LA051, AccidentNumber_ATL03LA124
Model: KGQA
Answer: DCA14CA036,ERA09CA138,CEN10LA098
Model: BLOOM
Answer: GPT Turned off, Kindly check other results
Model: GPT3
Answer: Boeing 757 232
Paragraph: Location:,Atlanta, Georgia,Accident Number:DCA14CA036,
Doc Name: 88664
Ask question related to selected report
Send
Page 1 of 5
National Transportation Safety Board
Aviation Accident Final Report
Location: Atlanta, Georgia Accident Number: DCA14CA036
Date & Time: January 10, 2014, 07:00 Local Registration: N687DL
Aircraft: Boeing 757 232 Aircraft Damage: None
Defining Event: Turbulence encounter Injuries: 1 Serious, 140 None
Flight Conducted
Under:Part 121: Air carrier - Scheduled
Analysis
The flight crew was anticipating light turbulence during the descent and notified the cabin and selected
the fasten seatbelt sign on prior to descending from 28,000 feet. During the descent, as the flight
attendants (FA) were still preparing the cabin for arrival, the turbulence increased to a level the crew
categorized as moderate.
During this time, while trying to get to her seat, a FA was thrown against the 3L door and fell to the
floor injuring her right ankle. The FA could not complete her duties and was assisted into the last row
of seats. Another FA informed the flight crew who arranged for paramedics to meet the airplane.
The FA was transported to the hospital and diagnosed with a broken right tibia in two places.
Probable Cause and Findings

Aviation IR System



Retrieval-Augmented Generation

RETRIEVE + GENERATE = RAG

Fetch relevant external context (documents) at query time.

LLM writes an answer grounded in that context.

The Purpose

Keeps knowledge in documents (non-parametric memory) rather than model weights. Allows for easy updates, private data use, and traceable citations.

“Search the right pages, then write the answer.”

When to use RAG?

Debugging

Fixes depend on specific repo docs or config files.

Writing

Content must adhere to brand guidelines or policy.

Use RAG whenever “truth” lives in documents.

Planning

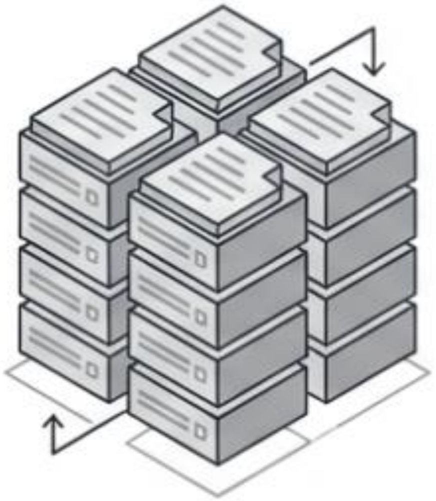
Plans must respect internal schedules or constraints.

Format Conversion

Output must match a specific schema.

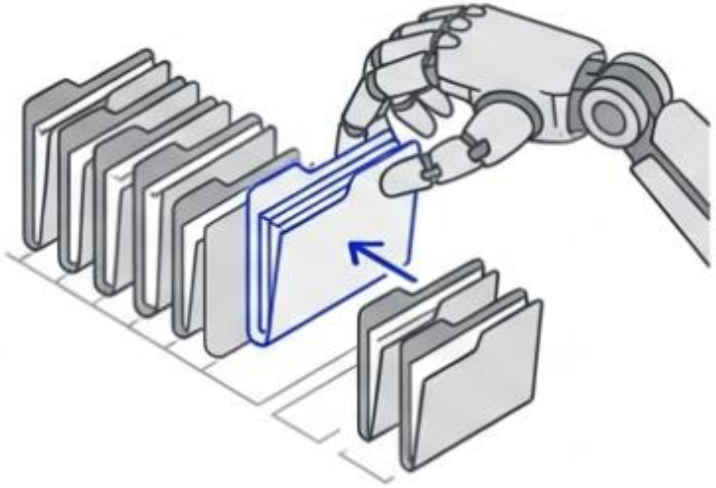
A lot of “chat” experiences are actually “Retrieve -> Generate” under the hood.

The Store
(The Library)



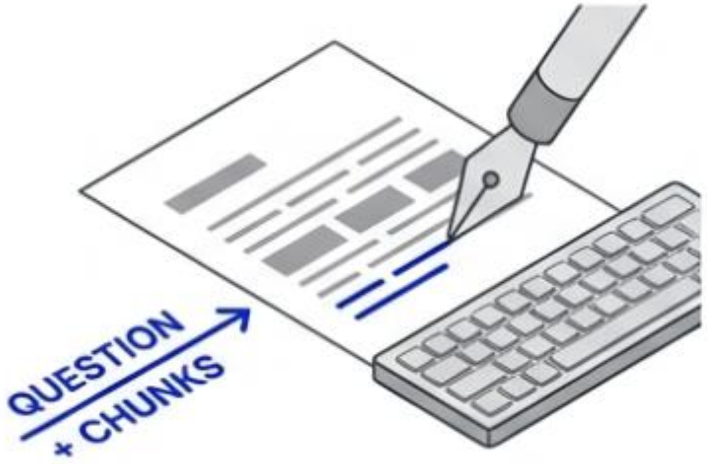
Holds your documents in a retrieval-friendly form.

The Retriever
(The Librarian)



Fetches the top relevant chunks based on a query.

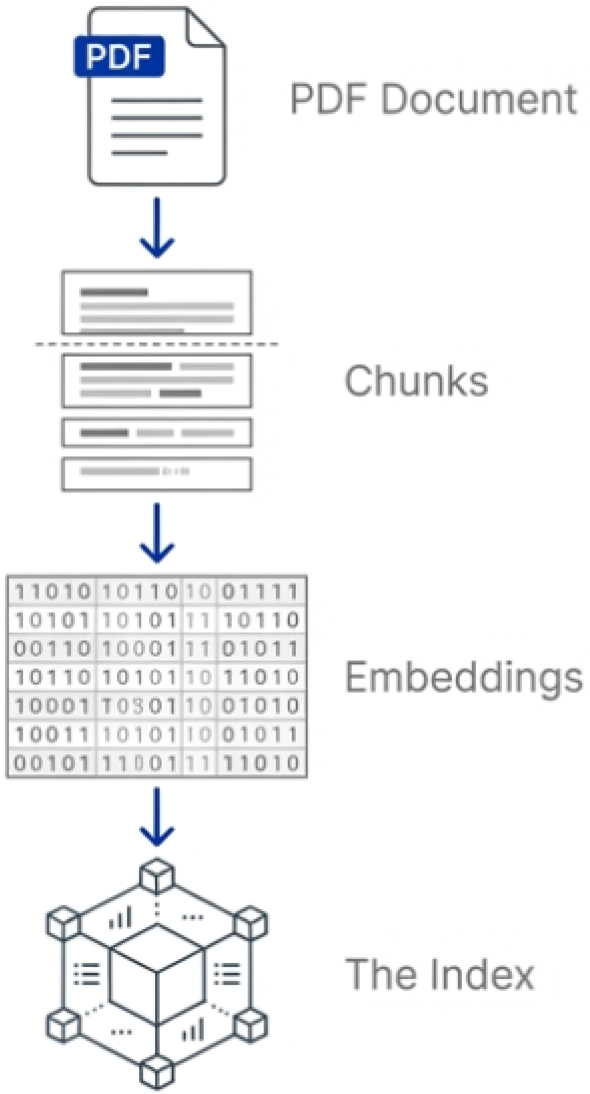
The Generator
(The Writer)



Reads the question + chunks to synthesize an answer.

Component 1: The Store (Knowledge Base)

- **Ingestion**: Parsing PDFs, docs, wikis.
- **Chunking**: Splitting large documents into smaller pieces.
- **Embeddings**: Converting text into numbers.
- **Indexing**: Organizing data for fast access.

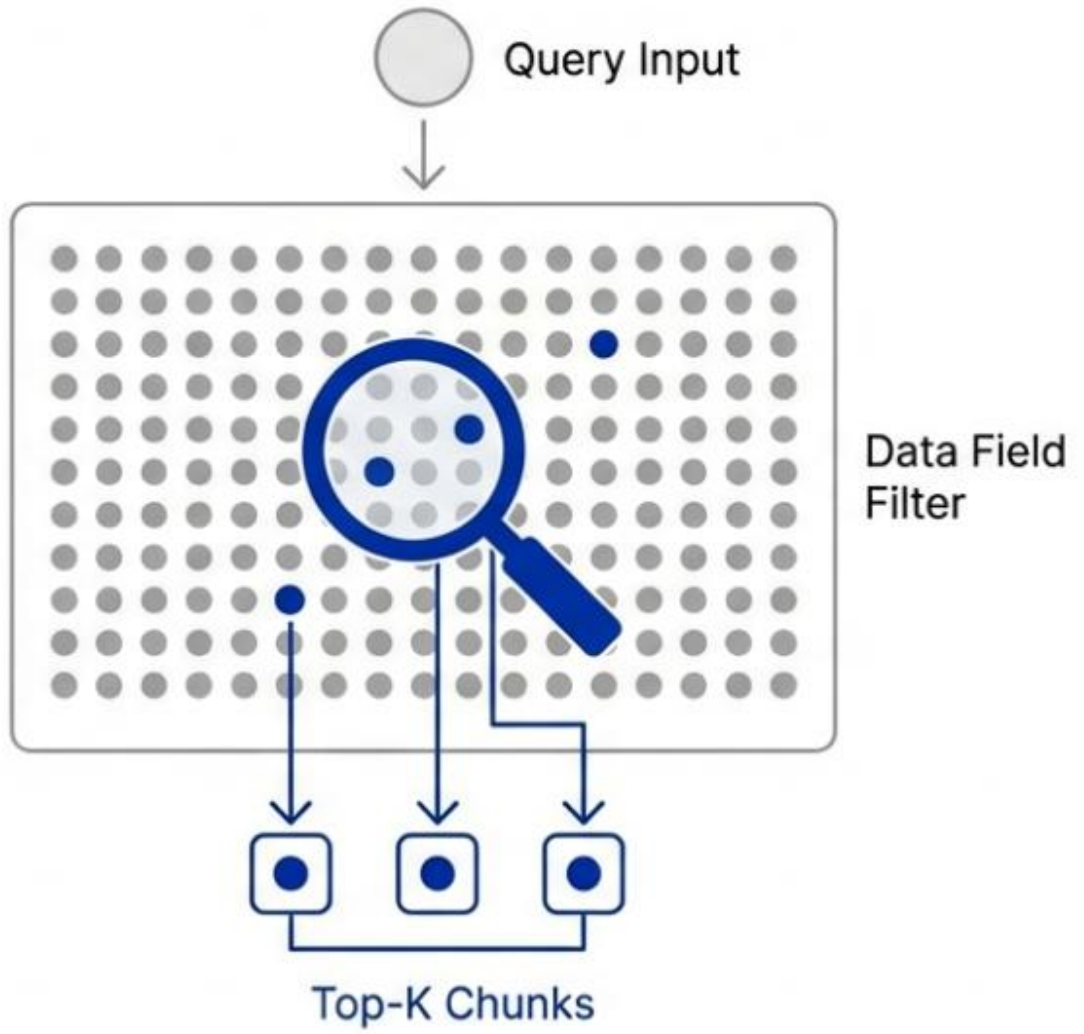


Role: The library catalog. It holds non-parametric memory ready for search.

Component 2: The Retriever

The engine that finds the needle in the haystack. Given a query, it fetches the "Top-K" most relevant chunks.

- **Lexical Search:** Keyword matching (Exact words).
- **Dense Search:** Embedding similarity (Meaning).
- **Hybrid Search:** Combination of both.

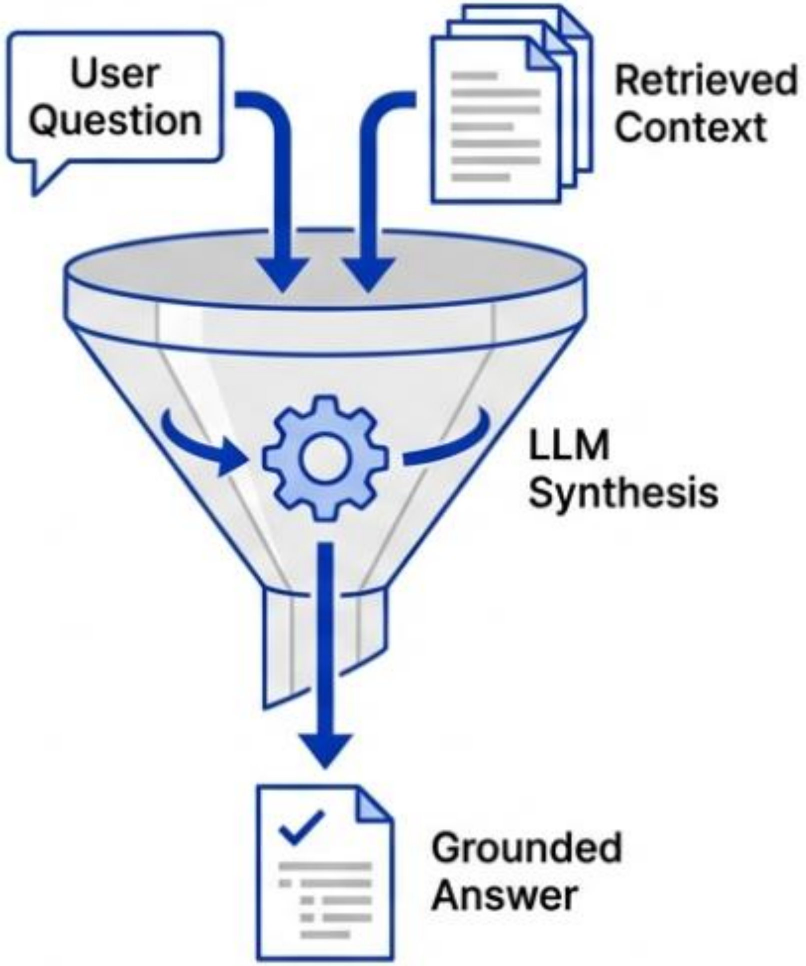


Role: The librarian who decides which evidence to put on the desk.

Component 3: The Generator (LLM)

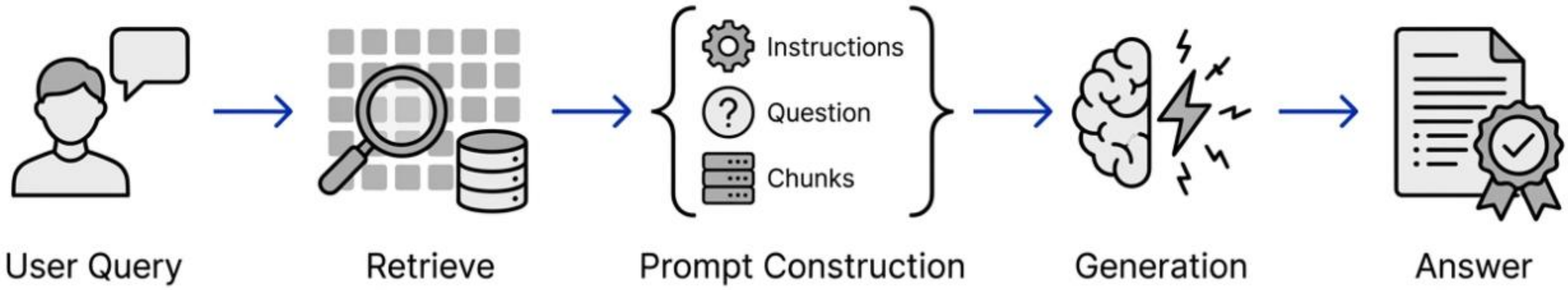
The synthesis engine. It receives the User Question + The Retrieved Chunks.

- Synthesize facts across multiple chunks.
- Stay grounded (do not invent beyond evidence).
- Handle conflicts or uncertainty.
- Cite sources.



Role: The writer who must use **only* the material on the desk.

The End-to-End RAG Flow



This flow turns a creative writing machine into a grounded research assistant.

The Reality Check: What Can Go Wrong?

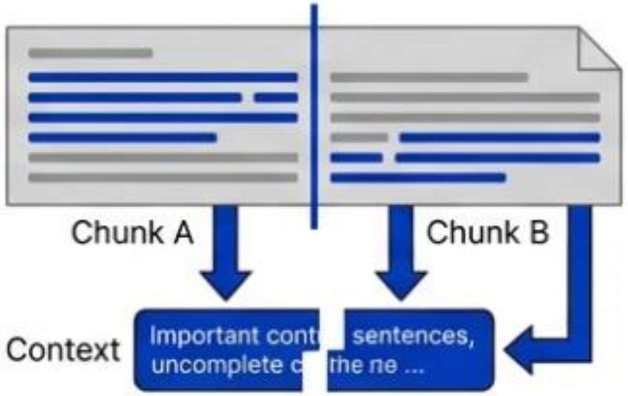
RAG reduces hallucinations, but introduces Retrieval Failures.

Wrong Chunks

Retriever fetches irrelevant information.

Bad Chunking

Important context is split across two chunks.

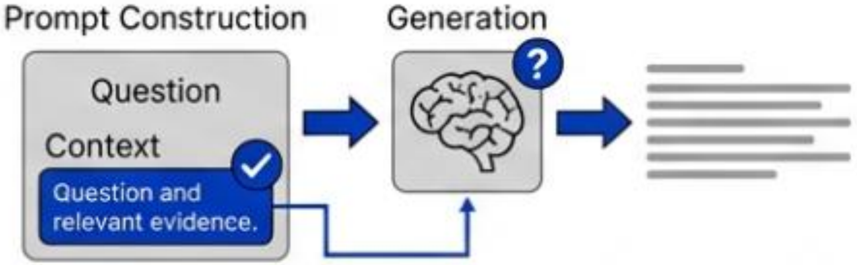


Missing Docs

The answer exists, but the document wasn't indexed.

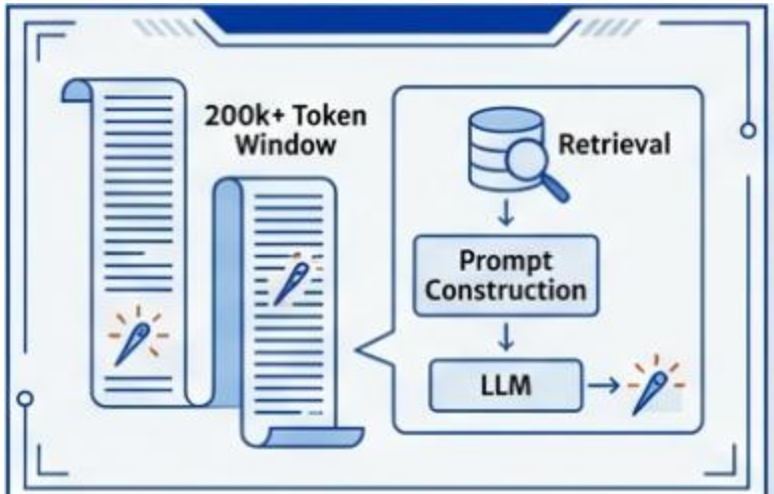
Generator Failure

LLM ignores the evidence provided in the prompt.



RAG is an engineering pattern that requires tuning, not a magic switch.

And there is more…

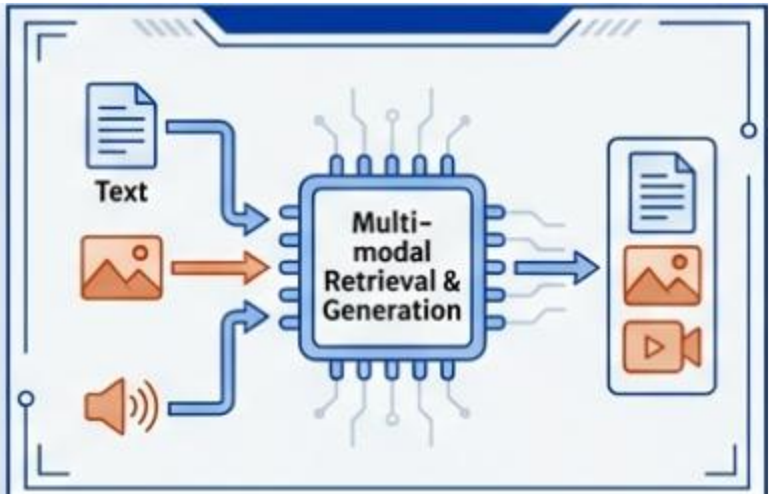


Context Length vs. RAG

Even with 200k+ token windows, RAG is necessary for cost, latency, and 'needle-in-a-haystack" precision.

Robustness

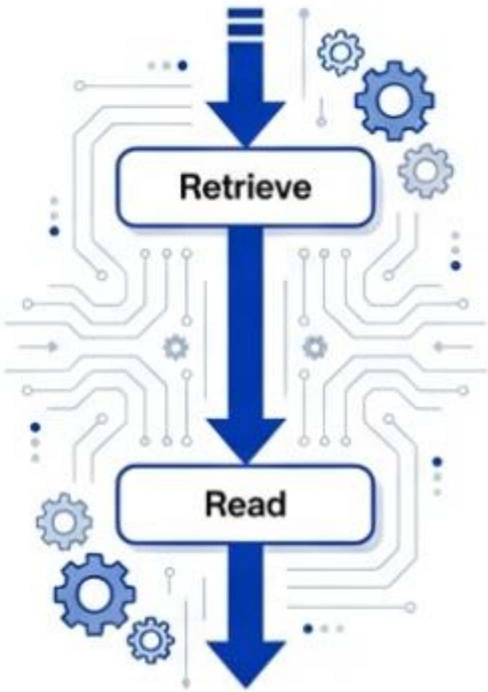
Handling contradictory or 'poisoned' information in retrieved docs.



Multi-modal RAG

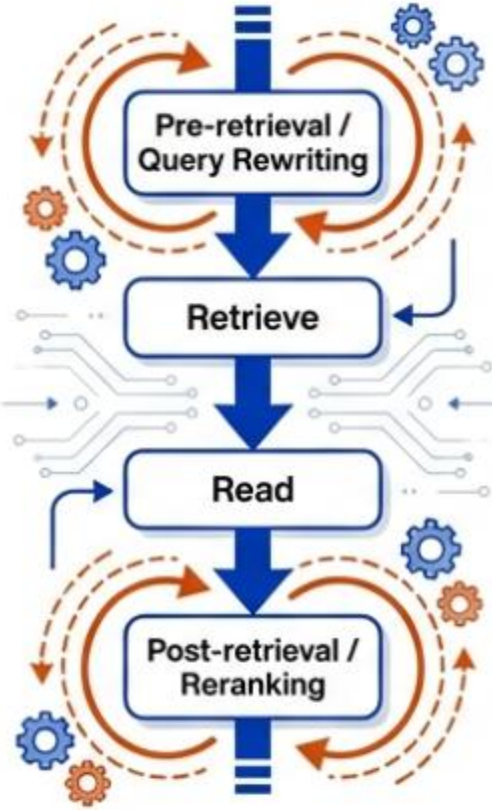
Moving beyond text: Retrieving and generating Images, Audio, and Video.

Naive RAG



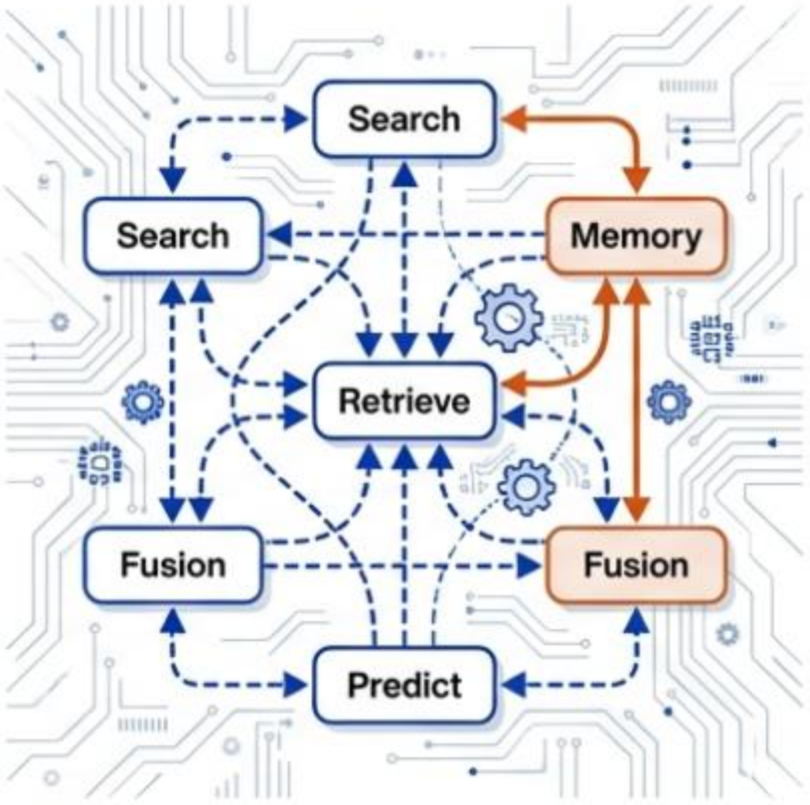
Basic flow. Prone to precision issues.

Advanced RAG



Enhanced flow with pre- and post-retrieval optimizations.

Modular RAG

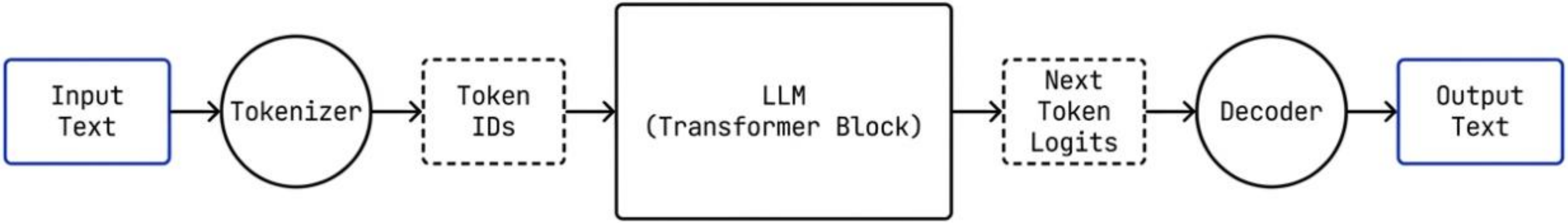


Flexible, adaptive system with multiple connected modules.

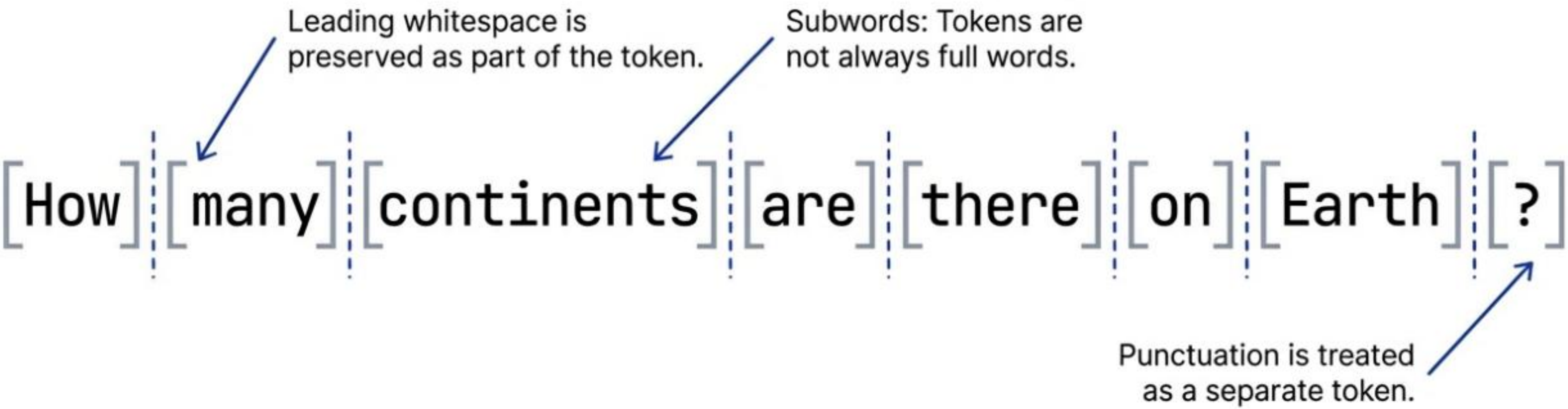
LLM Inference Simplified

```
> INPUT: How many continents are there on Earth?

> TARGET OUTPUT: Seven.
```



The Tokenizer breaks text into discrete sequence chunks.



The Tokenizer splits text into a sequence of tokens based on specific rules.
It is no longer a sentence, but a list of strings.

The system maps every token to a fixed integer ID.

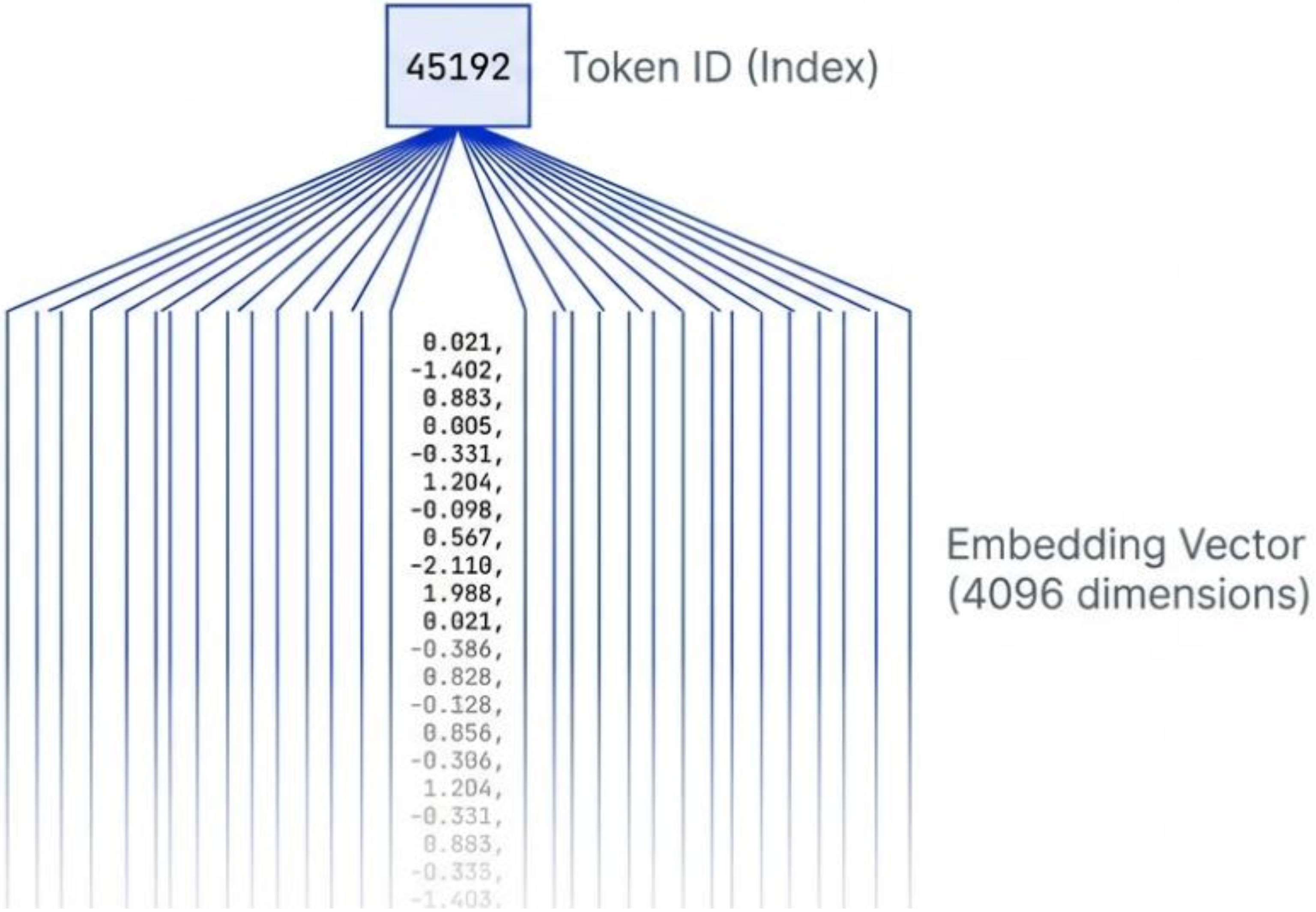
Running

Token		ID
"How"		8123
" many"		203
" continents"		45192
" are"	Vocabulary Lookup	38
" there"		112
" on"		19
" Earth"		9554
"?"		30

```
Input_IDs = [8123, 203, 45192, 38, 112, 19, 9554, 30]
```

The Tokenizer relies on a fixed vocabulary (the model's dictionary). Each unique token maps to a specific, permanent integer ID. The model no longer sees 'How'; it sees '8123'.

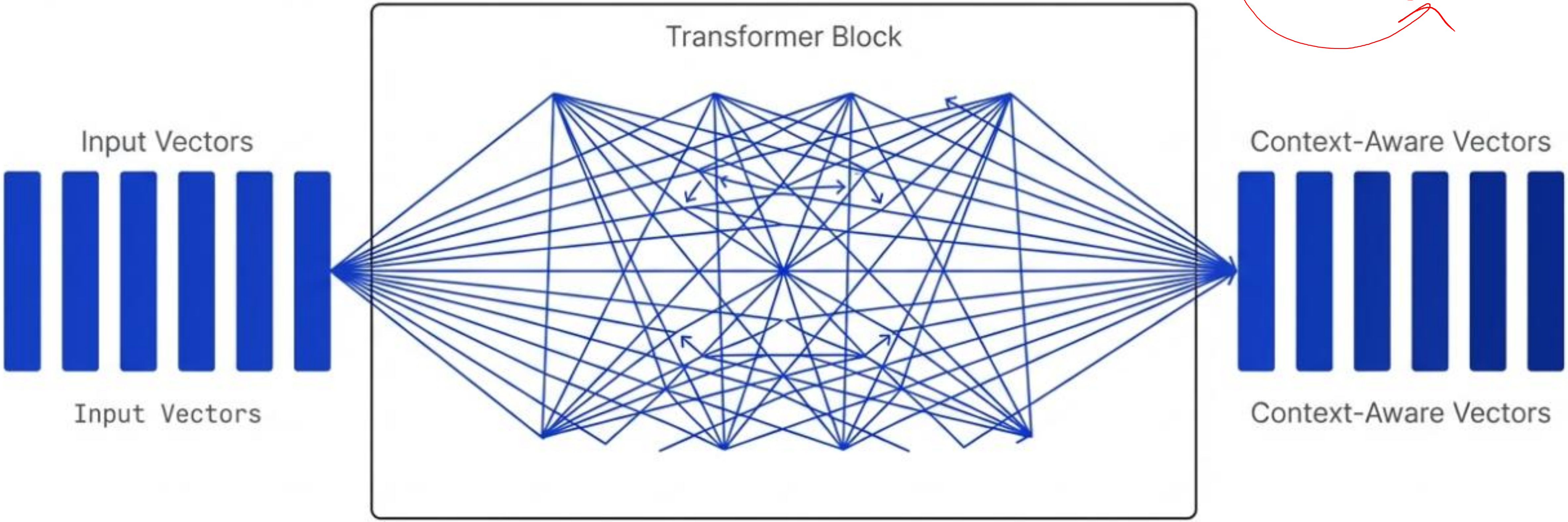
Integers are expanded into multi-dimensional vectors.



Integers are just indexes. To process meaning, the system looks up each ID in an 'embedding table' and converts it into a vector sequence. These vectors capture the semantic relationships between words.

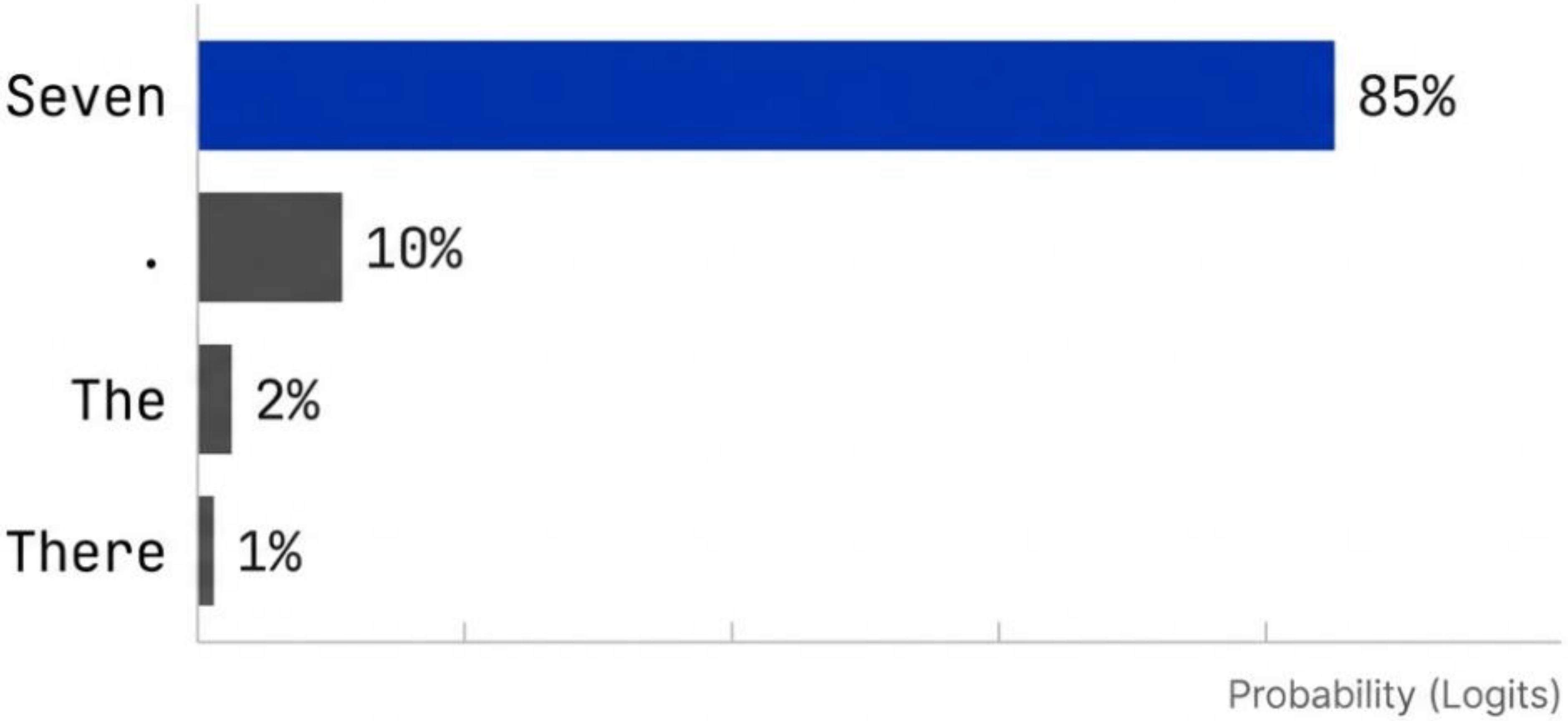
The Transformer layers process the vector sequence.

In. Embedding
tokenid → Emb



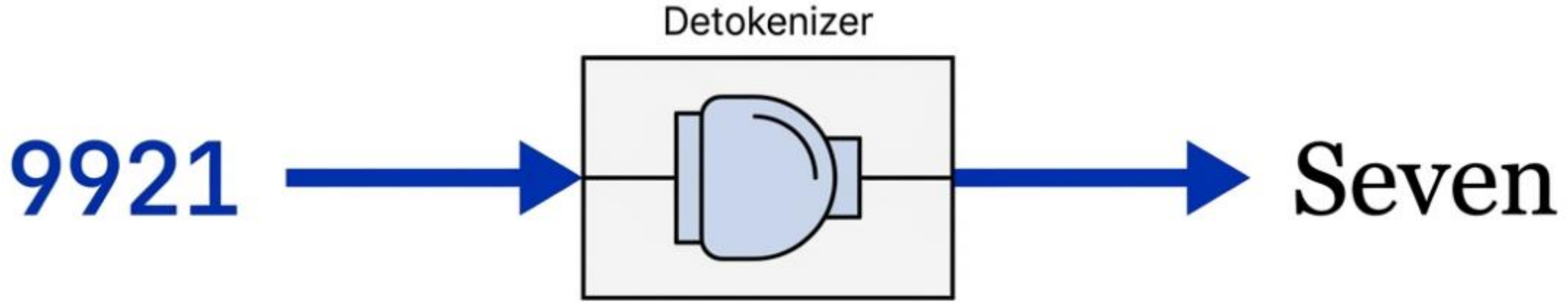
Inside the Transformer layers, the model analyzes the relationships between all vectors simultaneously. It calculates how "continents" relates to 'Earth' and 'How many' to build a contextual understanding.

The model generates probabilities for the next token.



The model does not "know" the answer. It predicts the most likely next token based on statistical patterns learned during training.

The chosen ID is decoded back into text.

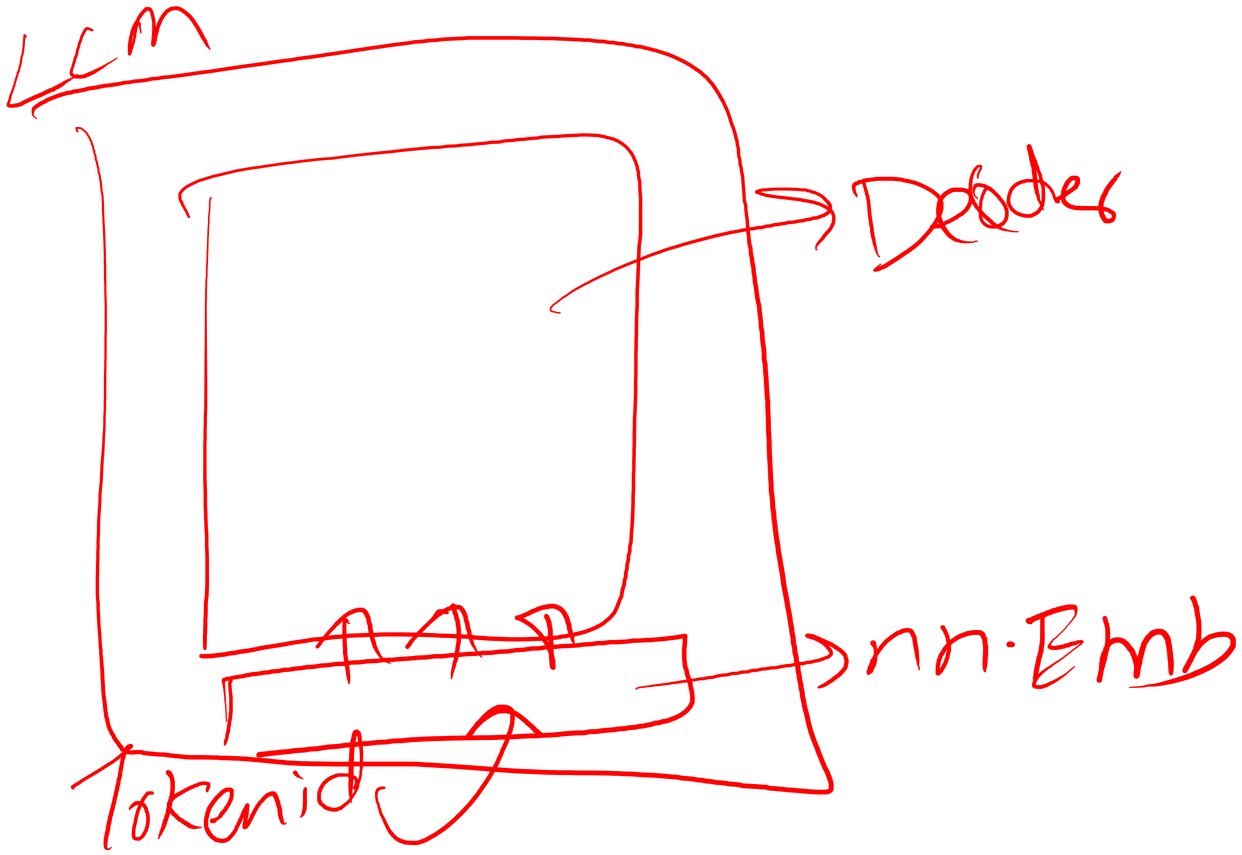


Result Block

Output Stream:	How many continents are there on Earth? **Seven**

Detokenization reverses the initial mapping, turning the integer back into a human-readable string.

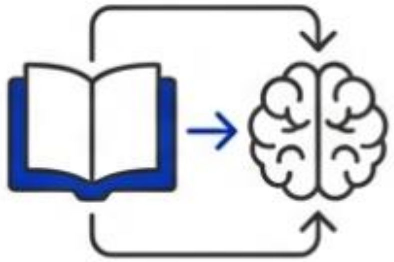
Lab



https://tinyurl.com/rag-augmented

Thank You

Appendix

1. Prompt-Only	2. Fine-Tuning	3. RAG
		
• Pros: Simple, Fast.	• Pros: Consistent style/behavior.	• Pros: Fresh data, private docs, citable sources.
• Cons: Stale knowledge, hallucinations, context limits.	• Cons: Expensive, hard to update facts, private data concerns.	• Cons: Retrieval complexity, infrastructure needs.
• Best For: Creative writing, generic explanations.	• Best For: Behavior shaping, format discipline, domain language.	• Best For: Enterprise Q&A, policy assistants, doc-heavy workflows.

Takeaway: Fine-tuning changes HOW it behaves. RAG changes WHAT it can look up.