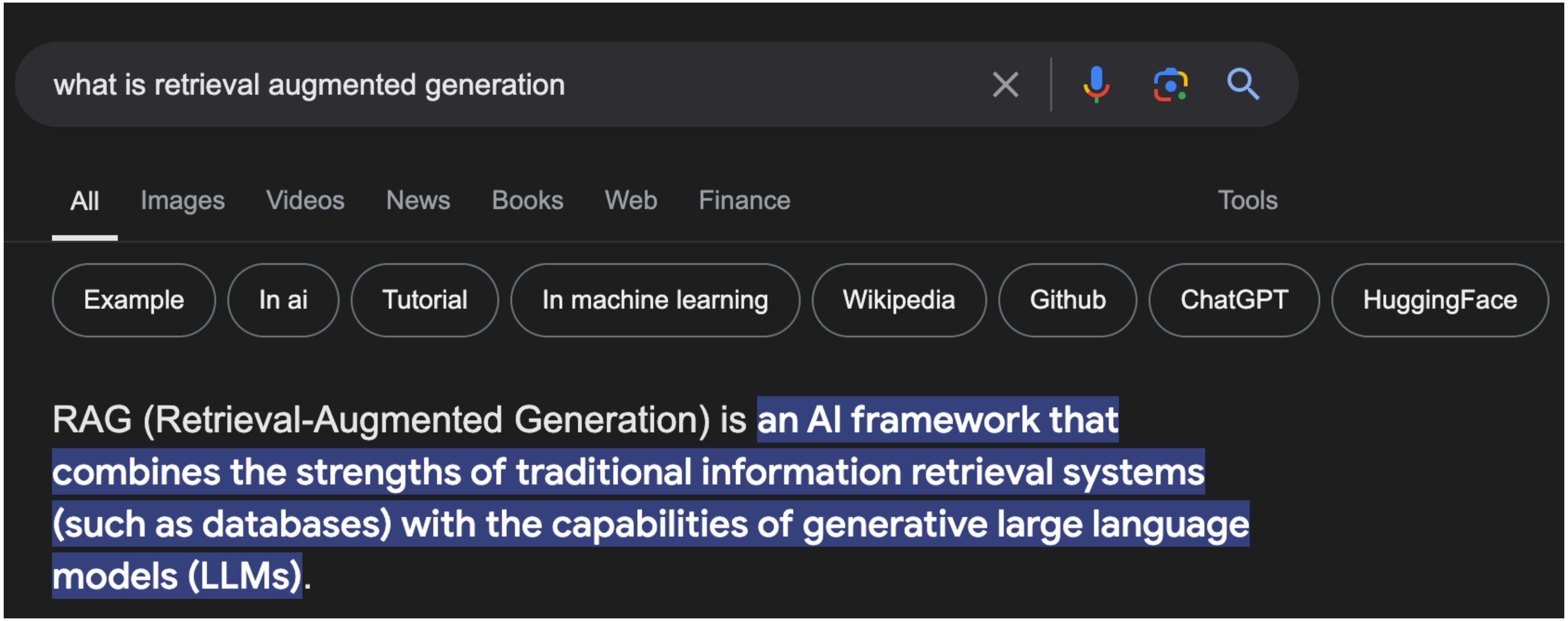
Retrieval Augmented Generation

ATHENA
Research & Innovation
Information Technologies

What is RAG?

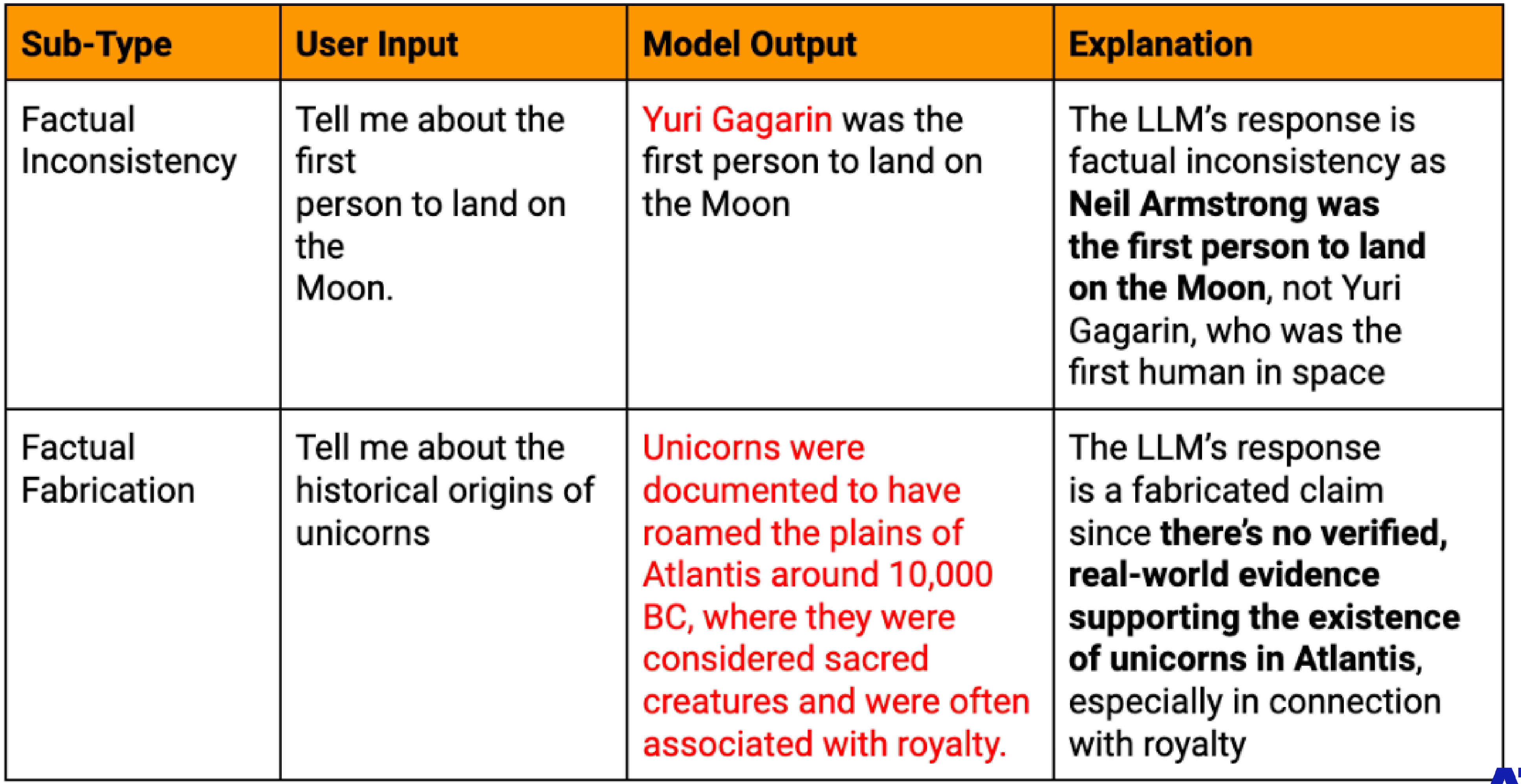


RAG (Retrieval-Augmented Generation) is an AI framework that combines the strengths of traditional information retrieval systems (such as databases) with the capabilities of generative large language models (LLMs).

ATHENA Research & Innovation Information Technologies

Why RAG?

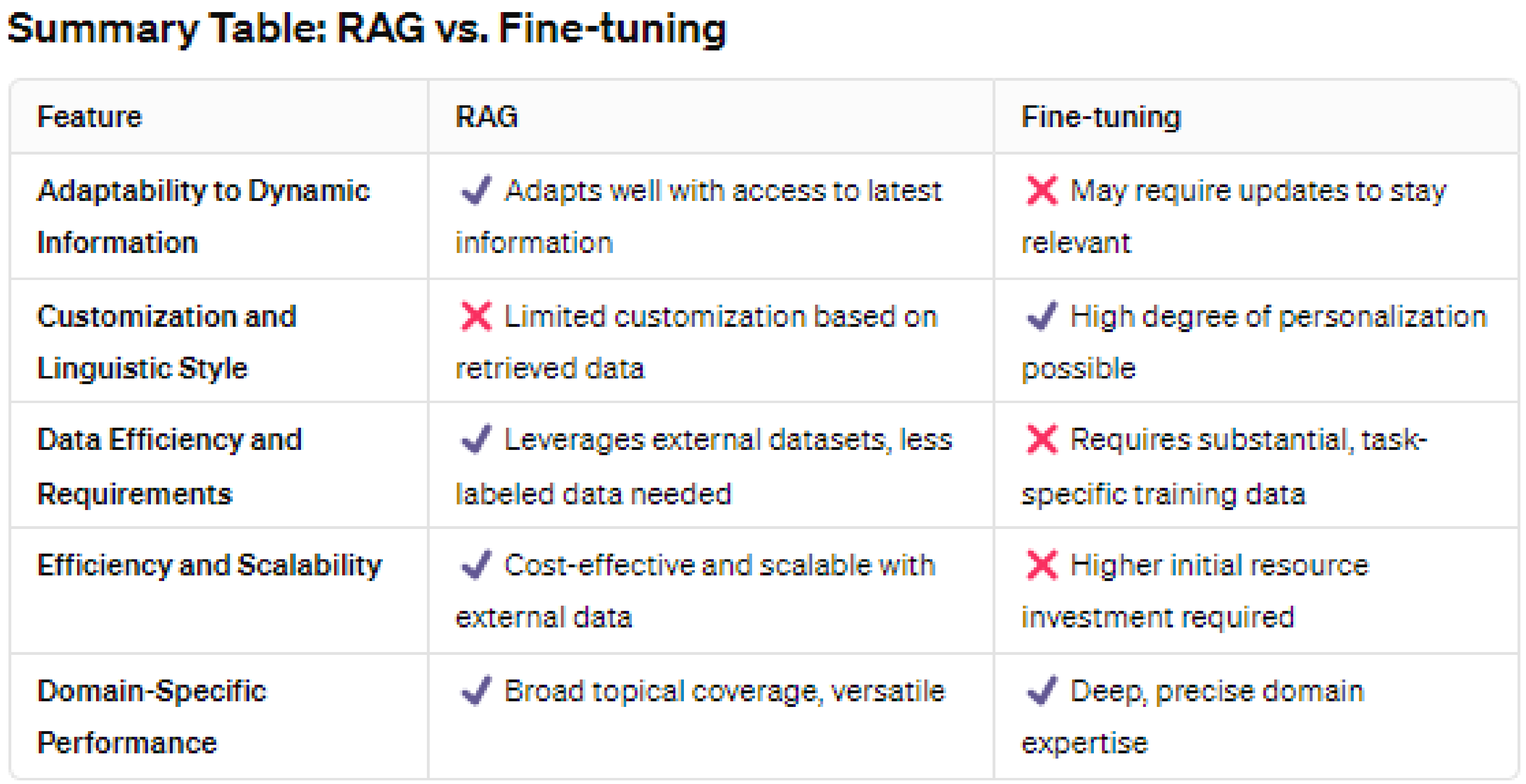
- LLM Hallucinations



Sub-Type	User Input	Model Output	Explanation
Factual Inconsistency	Tell me about the first person to land on the Moon.	Yuri Gagarin was the first person to land on the Moon	The LLM's response is factual inconsistency as **Neil Armstrong was the first person to land on the Moon**, not Yuri Gagarin, who was the first human in space
Factual Fabrication	Tell me about the historical origins of unicorns	Unicorns were documented to have roamed the plains of Atlantis around 10,000 BC, where they were considered sacred creatures and were often associated with royalty.	The LLM's response is a fabricated claim since **there's no verified, real-world evidence supporting the existence of unicorns in Atlantis,** especially in connection with royalty

Why RAG?

- Extend LLM knowledge to new sources without fine-tuning



Summary Table: RAG vs. Fine-tuning

Feature	RAG	Fine-tuning
Adaptability to Dynamic Information	✓ Adapts well with access to latest information	✗ May require updates to stay relevant
Customization and Linguistic Style	✗ Limited customization based on retrieved data	✓ High degree of personalization possible
Data Efficiency and Requirements	✓ Leverages external datasets, less labeled data needed	✗ Requires substantial, task-specific training data
Efficiency and Scalability	✓ Cost-effective and scalable with external data	✗ Higher initial resource investment required
Domain-Specific Performance	✓ Broad topical coverage, versatile	✓ Deep, precise domain expertise

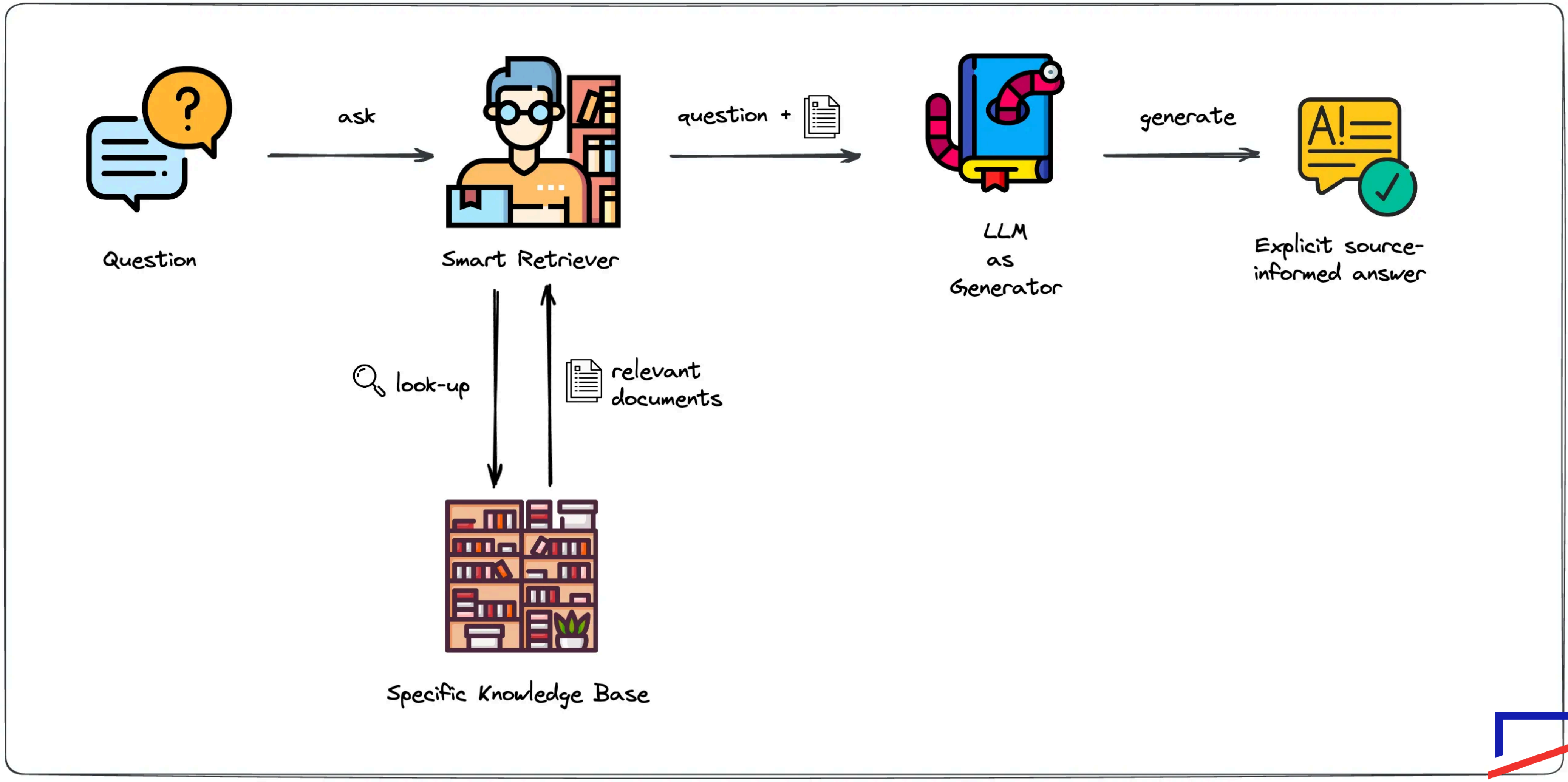
ATHENA
Research & Innovation
Information Technologies

Why RAG?

- Add factual context from authoritative sources in the LLM prompt
- The context is obtained from (usually unstructured) operational data
- Store data in the LLM vs store data in a knowledge base
  - Only use the LLM for what it’s good at (generation)
  - Use knowledge base to store knowledge.

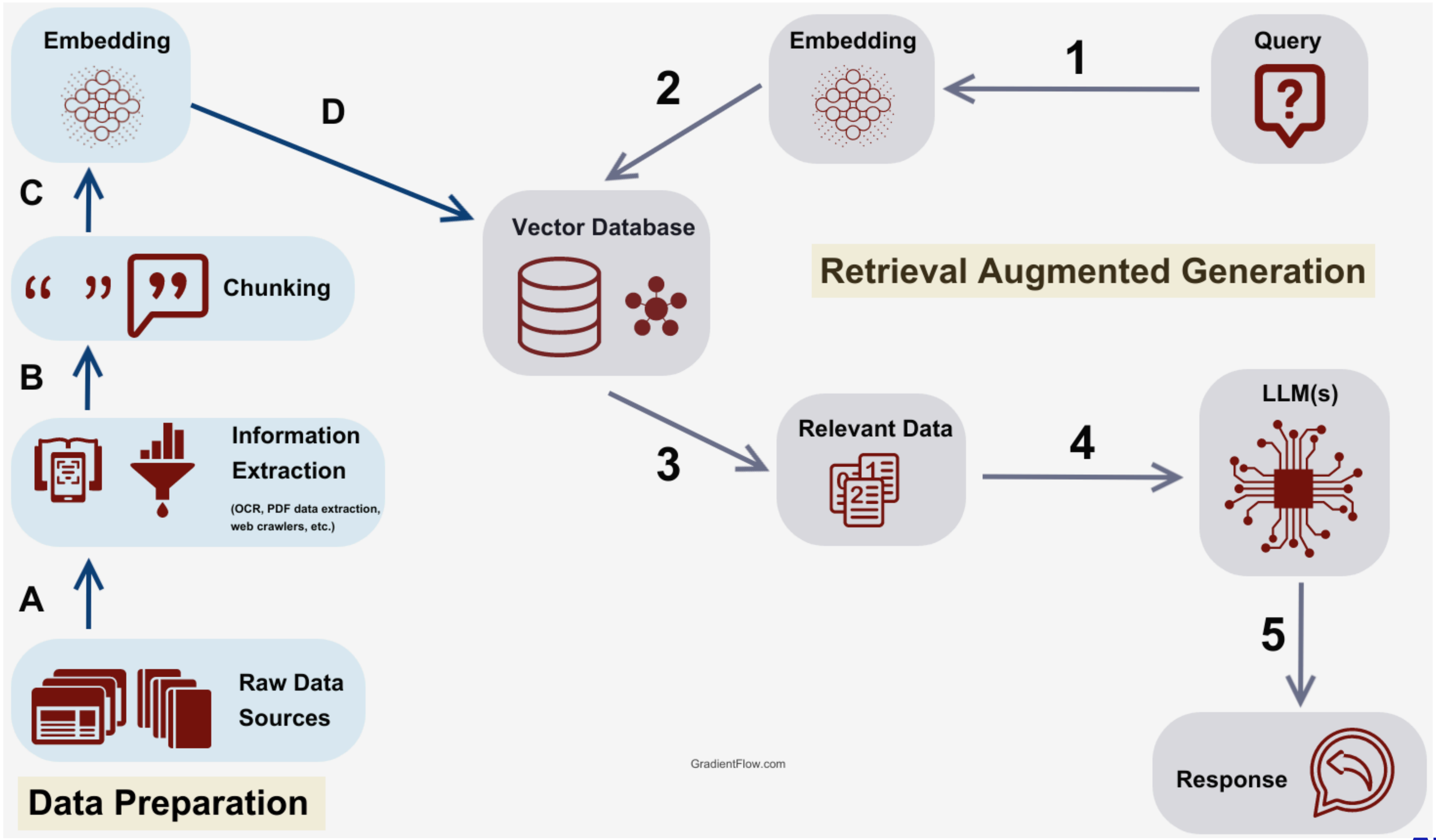
ATHENA
Research & Innovation
Information Technologies

RAG overview



ATHENA Research & Innovation Information Technologies

Complete RAG pipeline



ATHENA Research & Innovation Information Technologies

Phase 1: Preparation

- Need to do once
- Step A: Find data sources for your use case
  - Operational data, domain specific knowledge
- Step B: Ingest documents
  - May be in a variety of formats (web pages, pdfs, audio, etc.)
- Step C: Split documents into chunks of manageable length
- Step D: Extract embeddings & index vector database
- Repeat the process for new data and add them to the vector database

ATHENA
Research & Innovation
Information Technologies

Data ingestion

- Goal: Convert unstructured data to useable text
- Relevant technologies: Web scraping, pdf parsing, speech recognition, OCR
- Not the focus of this lecture
- Most RAG frameworks contain parsers for all the popular formats, e.g.:

```python
# Automagically load all useable files from the data folder
# File formats can be pdf, pptx, html, word, text, etc.
from llama_index.core import SimpleDirectoryReader
documents = SimpleDirectoryReader("./data").load_data()
```

Chunking

- Split documents into manageable chunks
- Chunks must contain contextually relevant information
- The system performance can be sensitive to the chunking algorithm
- For a fixed token budget, we need to balance
  - Chunk size: How much context is included in each chunk
  - Number of chunks in prompt: How many possibly contextually relevant chunks we present to the LLM

ATHENA
Research & Innovation
Information Technologies

Chunking algorithms

1. Sentence-based chunking
    - Split the document into sentences. Each chunk corresponds to one sentence.
2. Fixed-size chunking
    - Split the document into chunks of (nearly) equal size
3. Semantic chunking
    - First split into sentences, then merge chunks that are semantically coherent using embedding similarity
4. Document specific chunking
    - For some document types (e.g., Markdown, HTML, code) etc., make use of the known structure to split the document.
5. Agent chunking
    - Use an LLM to do the heavy lifting and decide how the document is better split. Not yet production ready.

Embeddings

- Convert text into vectors with nice properties
  - Vectors that correspond to semantically relevant chunks have small cosine distance
- Commercial options: OpenAI `text-embedding-3`, Google Gemini
- Open-source options: `sentence-transformers paraphrase-multilingual-MiniLM-L12-v2`

Vector databases

- A database that performs fast similarity search on vectors using Approximate Nearest Neighbors
- Need to compare
  - Performance (throughput / latency)
  - Embedding support (can it be extended for our embeddings of choice?)
  - Self-host vs managed
  - Features (e.g., metadata filtering, hybrid search, geo search)
  - Integrations (e.g., llama-index, langchain)

ATHENA Research & Innovation Information Technologies

	Pinecone	Weaviate	Milvus	Qdrant	Chroma	Elasticsearch	PGvector
Is open source	❌	✅	✅	✅	✅	❌	✅
Self-host	❌	✅	✅	✅	✅	✅	✅
Cloud management	✅	✅	✅	✅	❌	✅	(✓)
Purpose-built for Vectors	✅	✅	✅	✅	✅	❌	❌
Developer experience	👍👍👍	👍👍	👍👍	👍👍	👍👍	👍	👍
Community	Community page & events	8k☆ github, 4k slack	23k☆ github, 4k slack	13k☆ github, 3k discord	9k☆ github, 6k discord	23k slack	6k☆ github
Queries per second (using text nytimes-256-angular)	150 *for p2, but more pods can be added	791	2406	326	?	700-100 *from various reports	141
Latency, ms (Recall/Percentile 95 (millis), nytimes-256-angular)	1 *batched search, 0.99 recall, 200k SBERT	2	1	4	?	?	8
Supported index types	?	HNSW	Multiple (11 total)	HNSW	HNSW	HNSW	HNSW/IVFFlat
Hybrid Search (i.e. scalar filtering)	✅	✅	✅	✅	✅	✅	✅
Disk index support	✅	✅	✅	✅	✅	❌	✅
Role-based access control	✅	❌	✅	❌	❌	✅	❌
Dynamic segment placement vs. static data sharding	?	Static sharding	Dynamic segment placement	Static sharding	Dynamic segment placement	Static sharding	-
Free hosted tier	✅	✅	✅	(free self-hosted)	(free self-hosted)	(free self-hosted)	(varies)
Pricing (50k vectors @1536)	$70	fr. $25	fr. $65	est. $9	Varies	$95	Varies
Pricing (20M vectors, 20M req. @768)	$227 ($2074 for high performance)	$1536	fr. $309 ($2291 for high performance)	fr. $281 ($820 for high performance)	Varies	est. $1225	Varies

ATHENA
Research & Innovation
Information Technologies

Phase 2: Inference

- Step 1: Extract question embedding
- Step 2: Retrieve relevant documents from vector database using question embedding
- Step 3: Prompt engineering
    - Prepend the context to the question and send augmented prompt to the LLM

ATHENA
Research & Innovation
Information Technologies

RAG prompt template

```
CONTEXT:
{chunks}

QUESTION:
{query}

INSTRUCTIONS:
Answer the user's QUESTION using the CONTEXT chunks above.
Keep your answer grounded in the facts of the CONTEXT.
If the CONTEXT doesn’t contain the facts to answer the QUESTION simply  answer
"I don't know the answer to your question".
```

ATHENA
Research & Innovation
Information Technologies

RAG frameworks

- Langchain
  - The most flexible framework
  - Focuses more on agentic workflows and orchestration
  - The API may require a bit steeper learning curve
- Haystack
  - Framework focused on RAG and question answering
  - High-level API using pipeilnes
  - Debugging can be a bit hard
- Llamaindex
  - Framework focused on RAG, with some agentic functionalities
  - Clean API
  - Llama Hub for community-driven integrations, parsers, agents etc.
  - Some agentic workflows but more limited than langchain

RAG evaluation

- Use an LLM to judge the system outputs
- Metrics to evaluate the retrieval system
  - Context precision: Measure if relevant information from ground truth appears highly ranked in the context
  - Context recall: Measure the extent to which the retrieved context aligns with the annotated answer
- Metrics to evaluate the generated answer
  - Faithfulness: Measures the factual consistency of the generated answer against the given context
  - Answer relevance: measure how pertinent the generated answer is to the given prompt. Low score for incomplete or redundant answers.

ATHENA Research & Innovation Information Technologies

Ragas

- Framework for LLM evaluation
- Use LLM to generate synthetic test set
- Calculate RAG metrics
- Iteratively improve the model