

CSE/DSC 234

Data-Centric AI and AI Engineering

Arun Kumar

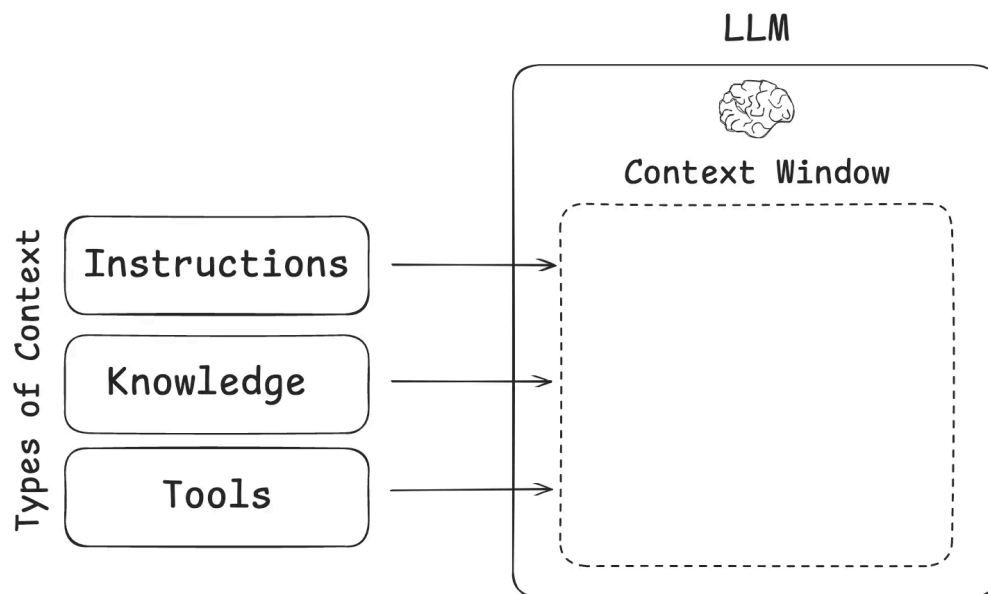
Topic 2: Retrieval and Context Engineering

Outline

- ➔ ❖ Introduction
 - ❖ Prompt Engineering
 - ❖ Retrieval and Context Engineering
 - ❖ Retrieval vs. Long Context
 - ❖ VectorRAG
 - ❖ GraphRAG
 - ❖ Multimodal and Other Forms of RAG

In-Context Learning

- ❖ LLMs resuscitated an age-old AI fact: “learning” $\neq$ “training”!
 - ❖ E.g., k-NN is an ML model with no “training” phase
- ❖ The “context window” of a frozen LLM is itself a canvas for learning at inference time, aka In-Context Learning (ICL)



In-Context Learning

- ❖ ICL is the algorithmic phenomenon that led to new AI engineering mechanisms to improve LLM applications:
 - ❖ Prompt eng. -> Retrieval eng. -> Context eng. -> Agent eng.
 - ❖ **Prompt eng.:** Crafting the input (instructions, examples, format) to steer LLM behavior via ICL
 - ❖ **Retrieval eng.:** Designing systems that dynamically fetch external info. and inject it into prompt at inference time
 - ❖ **Context eng.:** Assembling and managing everything in the context window at inference time—instructions, retrieved data, tool outputs, prior turns, memory, state, etc.—under token and latency budgets.

Outline

- ❖ Introduction
- ➔ ❖ Prompt Engineering
- ❖ Retrieval and Context Engineering
 - ❖ Retrieval vs. Long Context
 - ❖ VectorRAG
 - ❖ GraphRAG
 - ❖ Multimodal and Other Forms of RAG

Prompt Engineering

- ❖ Typically, Prompt = Task + Context + Examples + Output Format
 - ❖ Can add Guardrails, Persona, Tone, etc. too
- ❖ Good prompts are *specifications*, not “incantations”
- ❖ Under-specification is the #1 failure mode of LLMs (Gulf of Spec.)



Prompting Methodologies

- ❖ **Zero-Shot:** Just the task, no (input, output) examples
- ❖ **One/Few-Shot:** Provide one or few (input, output) examples for LLM to learn with ICL
- ❖ **Chain of Thought (COT):** Demonstrate step by step reasoning, not just answers, for LLM to learn with ICL

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

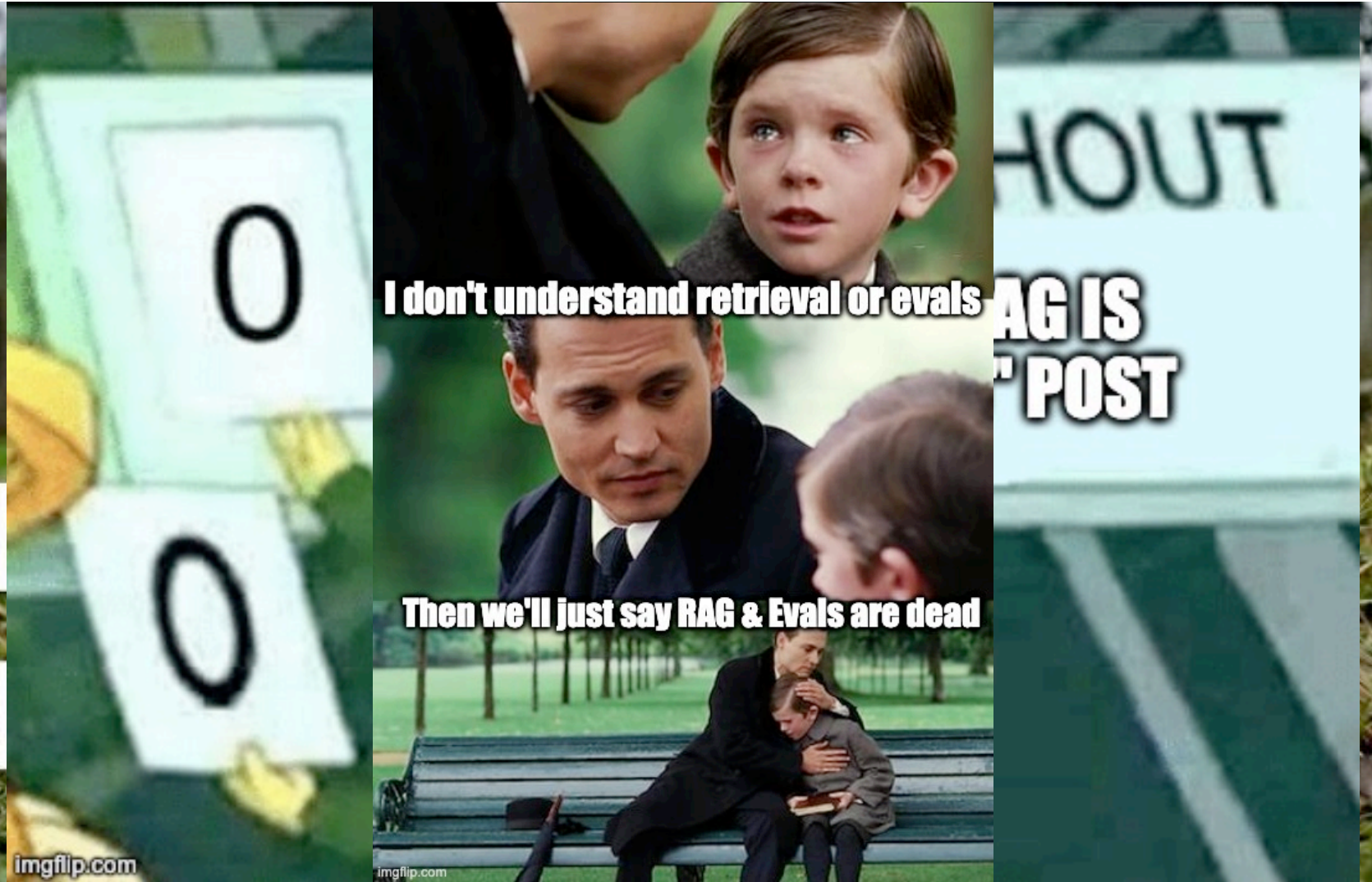
Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

Outline

- ❖ Introduction
- ❖ Prompt Engineering
-  ❖ Retrieval and Context Engineering
 - ❖ Retrieval vs. Long Context
 - ❖ VectorRAG
 - ❖ GraphRAG
 - ❖ Multimodal and Other Forms of RAG

RAG is Dead!



Outline

- ❖ Introduction
- ❖ Prompt Engineering
- ❖ Retrieval and Context Engineering
- ➔ ❖ Retrieval vs. Long Context
 - ❖ VectorRAG
 - ❖ GraphRAG
 - ❖ Multimodal and Other Forms of RAG

So, Is RAG Really Dead?

❖ Yes and No. :)

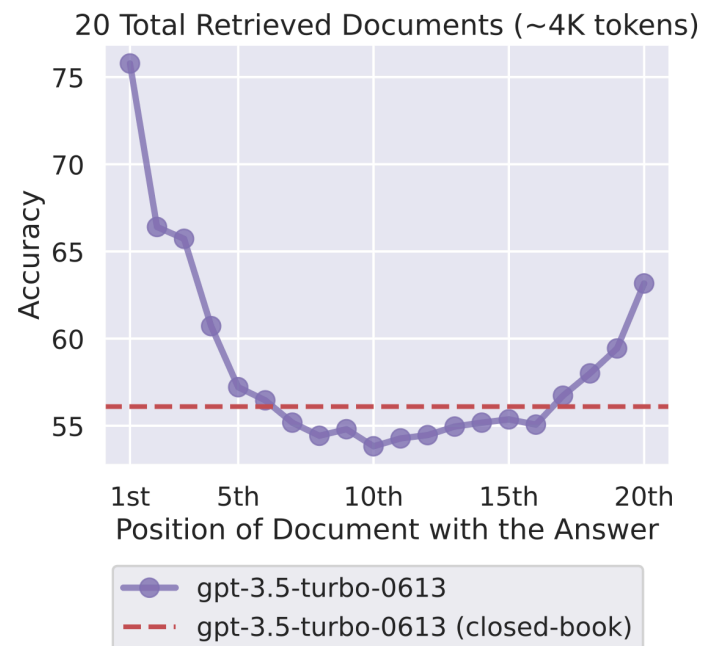
❖ Yes: The specific algorithm proposed in Piktus et al. 2021

❖ No: Idea/mechanism of “retrieval augmentation” for generation

Q: But won't ever growing context windows obviate retrieval?

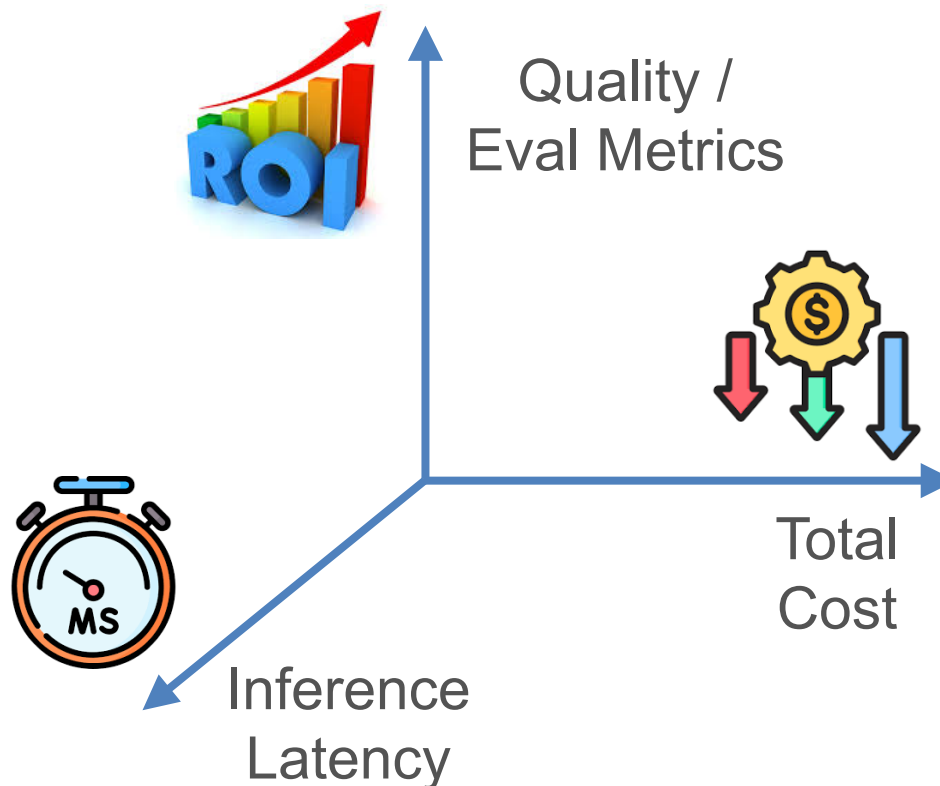
LLM	Input Limit
Claude Opus 4.7	1M
Claude Sonnet 4.6	1M
GPT-5.4	1M
Gemini 3.1 Pro	1M
Llama 4 Scout	10M

Vs.



So, Is RAG Really Dead?

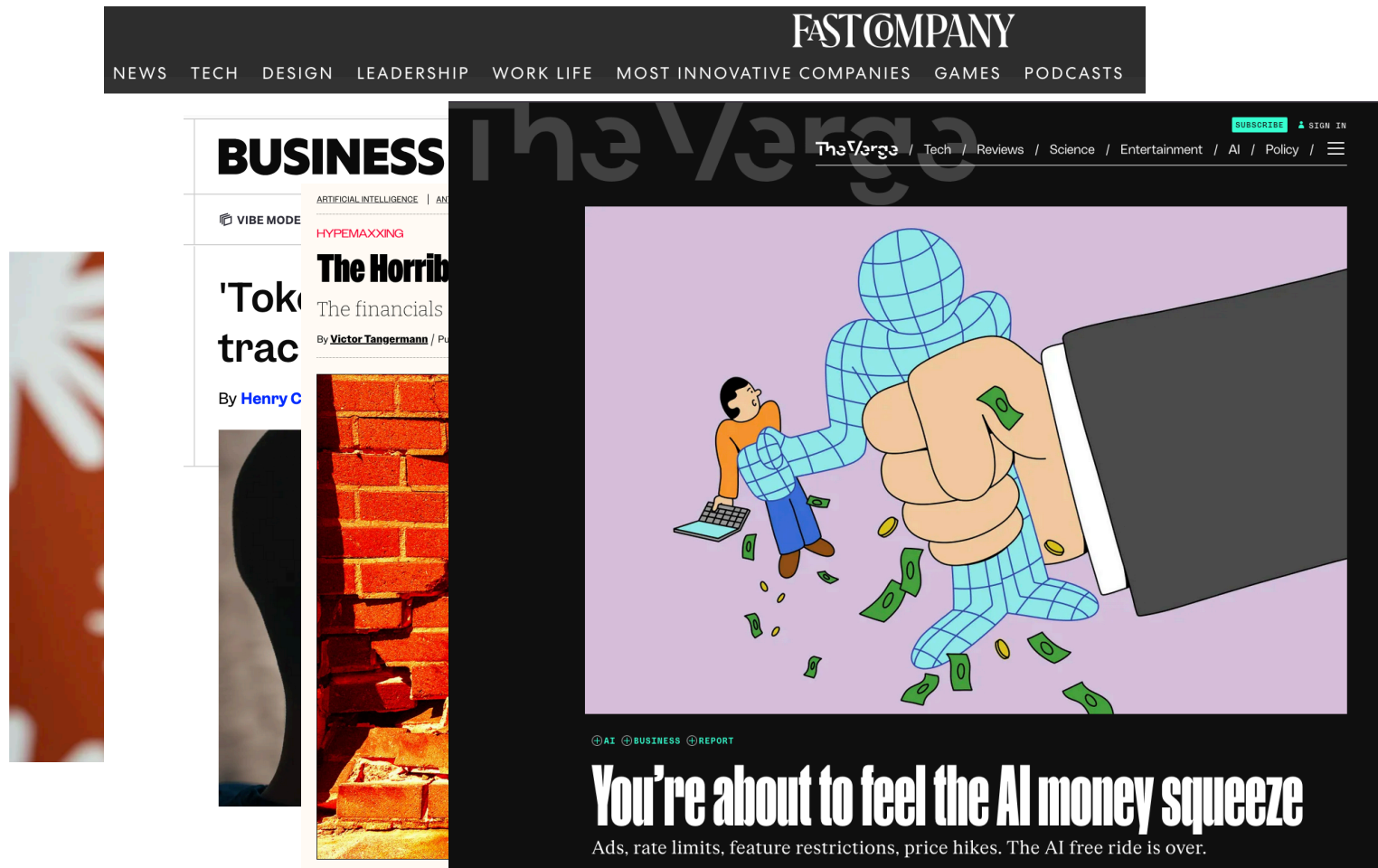
- ❖ Real-world AI applications: revenge of the Pareto Frontier!
- ❖ Long context dominates quality when content fits cleanly; retrieval dominates cost and latency at scale; corpus size and query type set the operating point



So, Is RAG Really Dead?

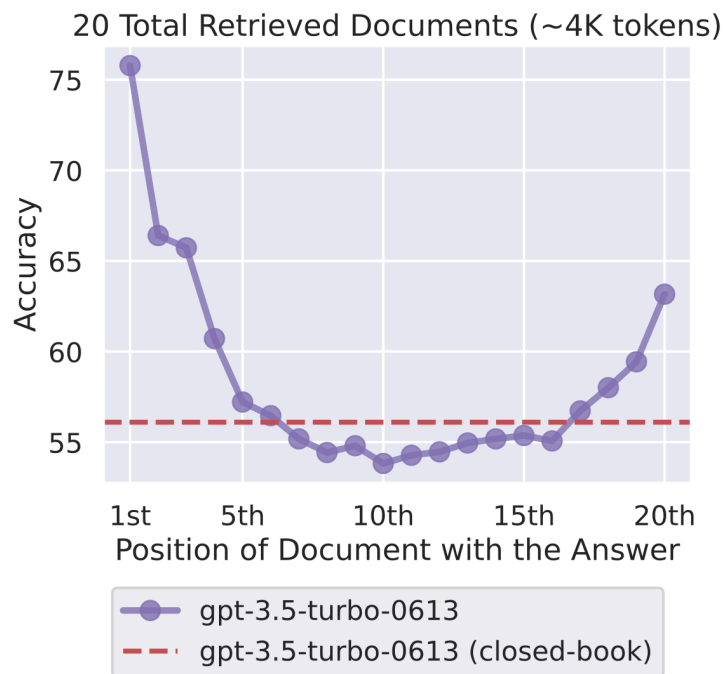
- ❖ Real-world AI applications: revenge of the Pareto Frontier!

B B C



Lost-in-the-Middle Problem

- ❖ **Setting:** Multi-document QA + key-value retrieval
 - ❖ Vary the position of relevant info. *inside* context window.



- ❖ **Key findings:**

- ❖ Accuracy varies non-monotonically based on position of ground truth doc! “U” curve
- ❖ Even long-context models degrade when doc in the middle
- ❖ Just because longer context feasible, not necessarily useful

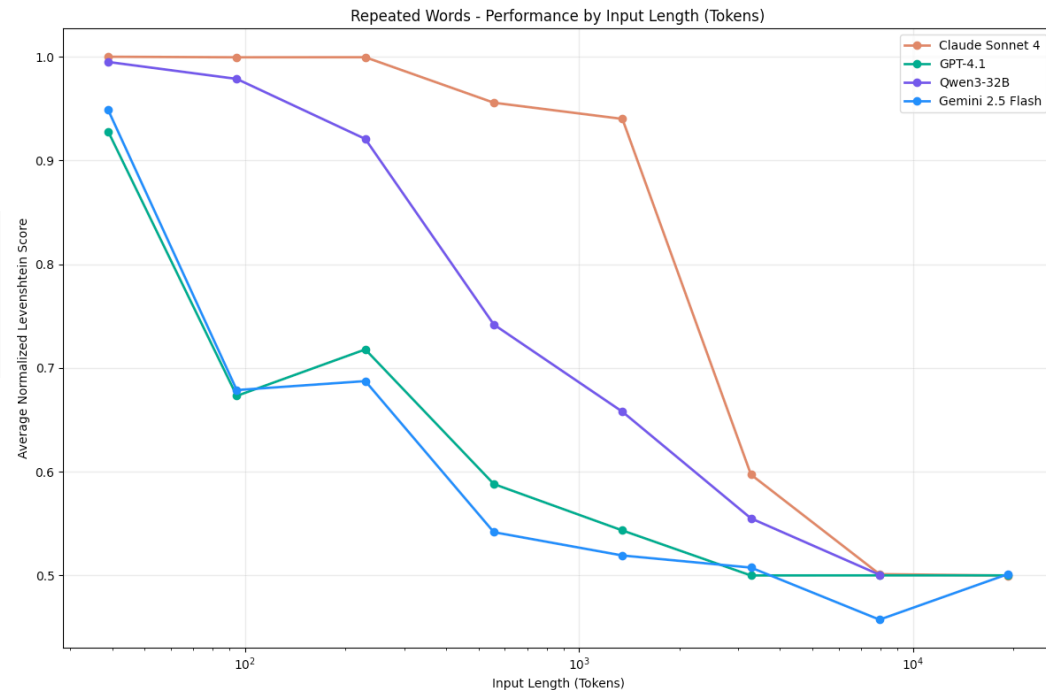
Context Rot

- ❖ Worse, it turns out *length* itself might hurt, not just position
- ❖ **Setting:** 18 frontier models on simple tasks with task complexity held constant and only input length varied

Repeated Words Task:

Simply replicate the following text, output the exact same text: apple apple apple apple **apples**
apple apple apple apple apple apple apple apple apple apple apple apple apple apple apple
apple apple apple apple

Repeated Words - Sample Prompt Containing 'apple' as the repeated word, and 'apples' as the unique word



Context Rot: How Increasing Input Tokens Impacts LLM Performance

<https://www.trychroma.com/research/context-rot>

Context Rot

❖ Key findings:

- ❖ All LLMs degrade as input length grows, even on trivial tasks!
- ❖ Long context is not uniformly usable memory; measurable drops even 1K -> 4K tokens
- ❖ Non-lexical retrieval degrades faster than lexical
- ❖ Even 1 distractor in context hurts
- ❖ Claude tends to abstain more under ambiguity; GPT hallucinates confidently more often! :)
- ❖ Seems to be an architectural limitation, not just capability gap

When to Use Which Mechanism

❖ Long context vs. retrieval is a false dichotomy!

Attribute of Use Case	Long Context Likely Suffices	Retrieval Likely Useful in Addition
Base data/doc corpus size	If it fits in context window	Exceeds the window
Query scope	Whole-document reasoning	Narrow lookup or multi-hop across many docs
Effective recall	Below threshold of context rot	When at risk of lost-in-the-middle
Update frequency	Static/session-scoped	Frequently updated
Cost and latency	Low-volume, loose budget	High-volume, tight per-query budget
Provenance	Implicit (the doc itself)	Citations and audit trail required
Working memory	Within-session state	Persistent across sessions / users

Outline

- ❖ Introduction
- ❖ Prompt Engineering
- ❖ Retrieval and Context Engineering
 - ❖ Retrieval vs. Long Context
-  ❖ VectorRAG
- ❖ GraphRAG
- ❖ Other Forms of RAG

The Original RAG Architecture

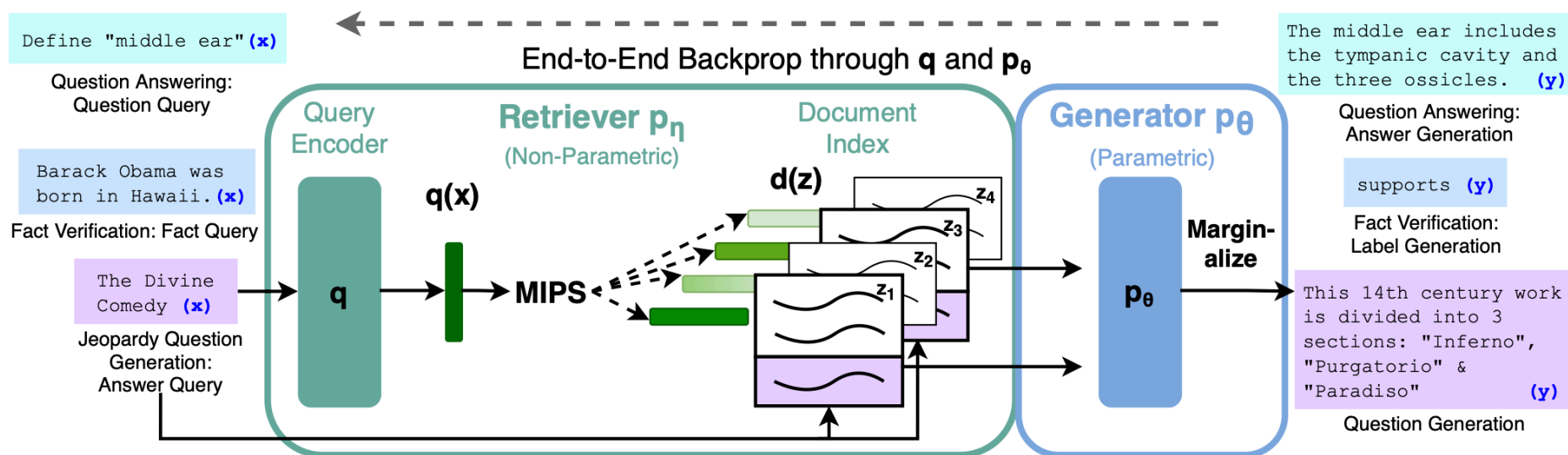


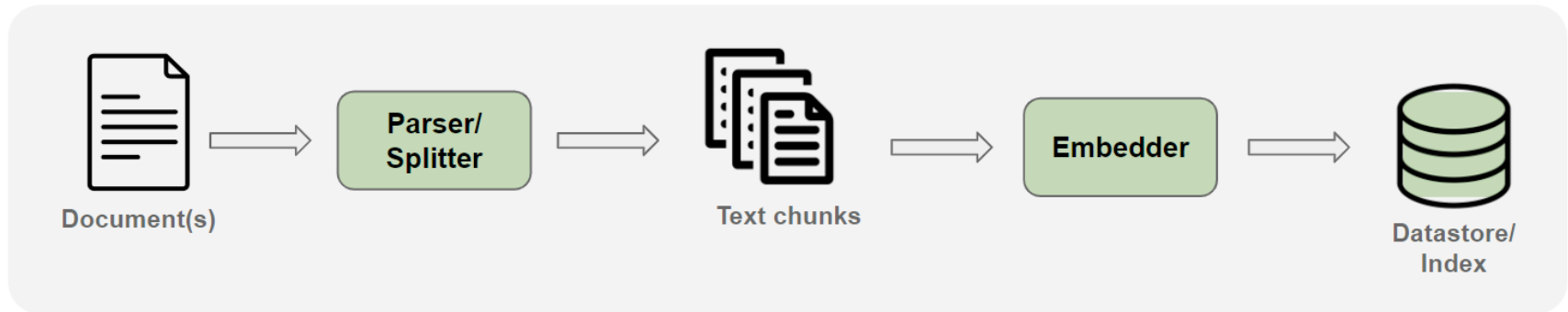
Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x , we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i . For final prediction y , we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

The Original RAG Architecture

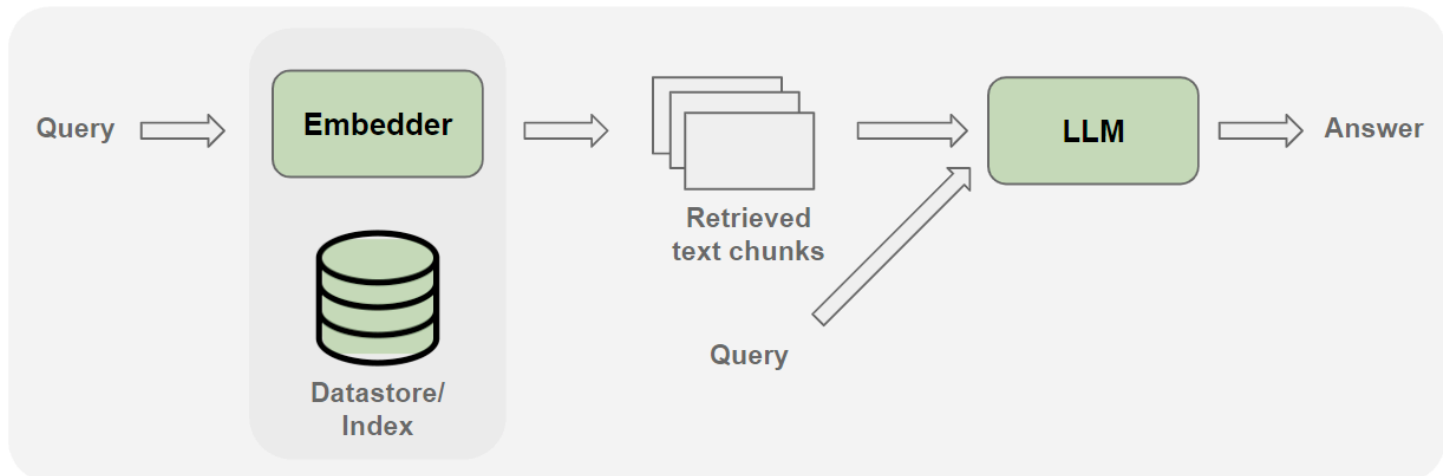
- ❖ Distinguishes 2 types of “memory” for LLM’s generation:
 - ❖ **Parametric:** Models’ weights; handles *how to reason*
 - ❖ **Non-parametric:** Dense vector index; tells *what is true*
- ❖ “RAG-Sequence” approach retrieves passages from doc corpus for a given query at query-time
- ❖ Original paper: End to end training of retriever and generator
- ❖ Modern RAG practice:
 - ❖ Both retriever and LLM are frozen after training
 - ❖ Orchestrate retrieval + generation at inference time
 - ❖ Same architecture, much simpler plumbing

The Canonical VectorRAG

Indexing

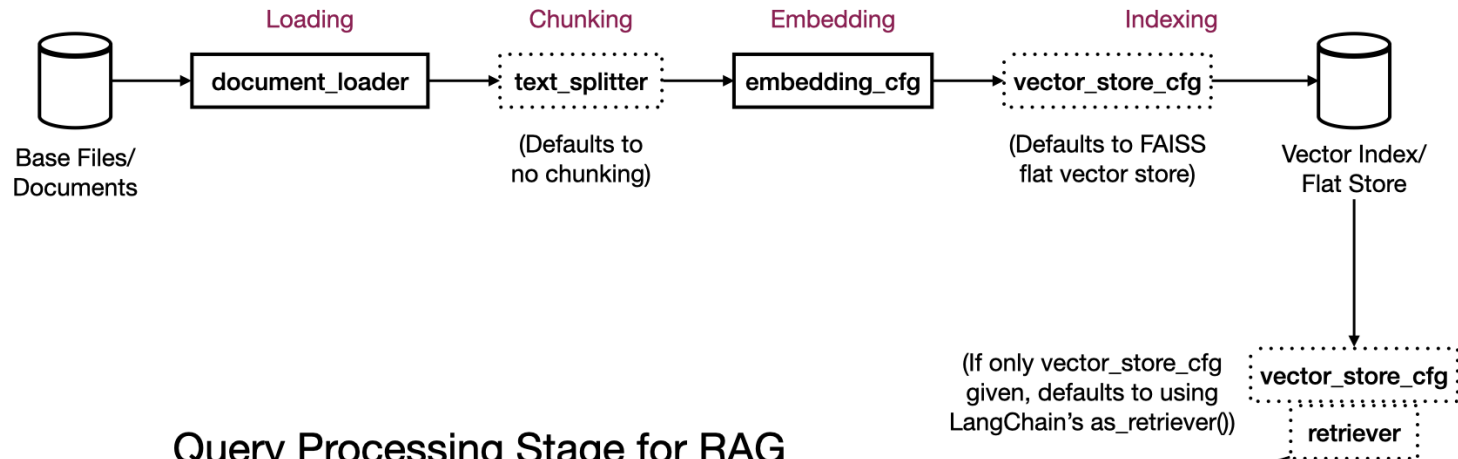


Generating

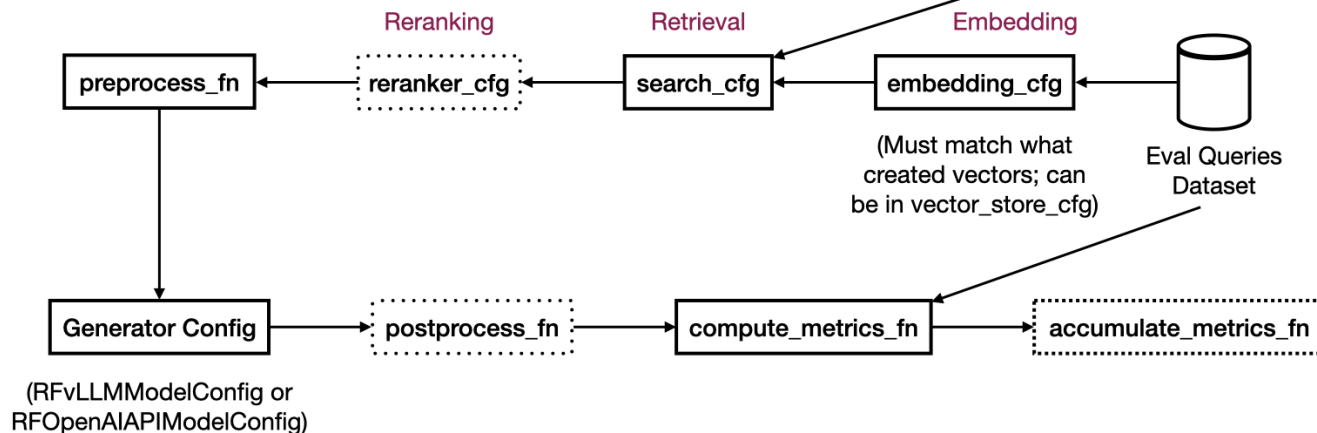


VectorRAG Pipeline in RapidFire AI

(Optional) Document Preprocessing Stage for RAG



Query Processing Stage for RAG



Parts of Canonical VectorRAG

1. **Parsing and Splitting/Chunking:** Turn raw data/docs into retrieval units
2. **Embedding:** Encode each chunk as a dense vector
3. **Indexing:** Store vectors for fast (approximate) nearest-neighbor (NN/ANN) search
4. **Retrieval:** Fetch top-K candidates from one or more sources; fuse into a single ranked list
5. **Reranking:** Rescore top-K to reorder chunks by relevance to query
6. **Context Stitching:** Assemble the final prompt—system instructions plus ordered, deduped, formatted, budgeted context
7. **Generation:** LLM produces a grounded answer with citations or abstains

Peer Instruction Activity

(Switch to Google Form)