DATA+AI SUMMIT
BY databricks

Building Advanced RAG Over Complex Documents

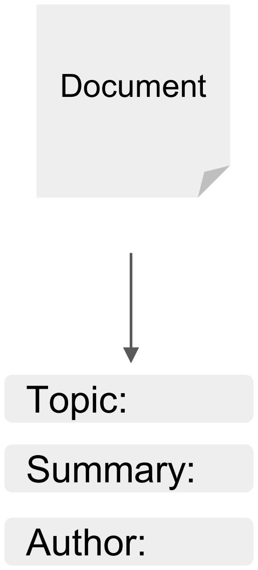
Jerry Liu
June 11, 2024

Agenda

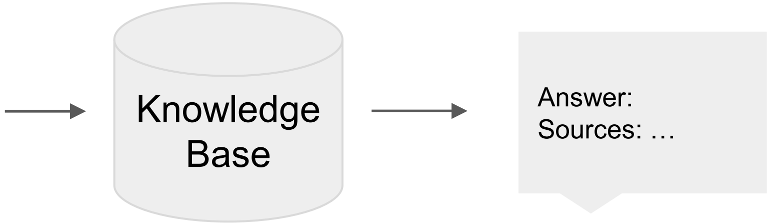
1. Building a Knowledge Assistant
2. RAG Overview: Basic RAG and where it goes wrong
3. Improving Data Quality:
    - Improve LLM reasoning over complex data
    - **Workshop:** LlamaParse over Complex Documents
4. **Improving Query Complexity:** from RAG to agents
    - **Workshop:** LlamaParse-powered document agent
5. What's next?

Enterprise Use Cases

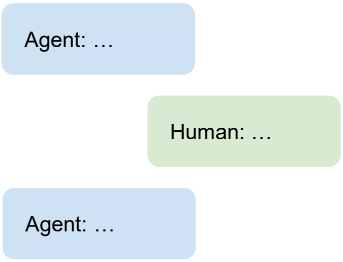
Enterprise Use Cases



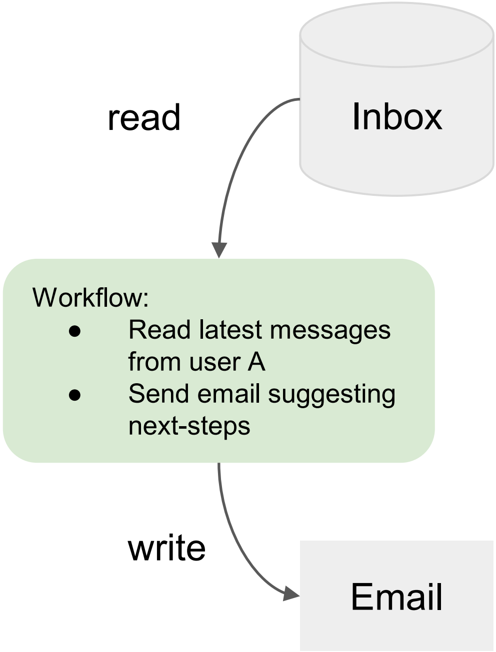
Document Processing
Tagging & Extraction



Knowledge Search & QA (RAG)

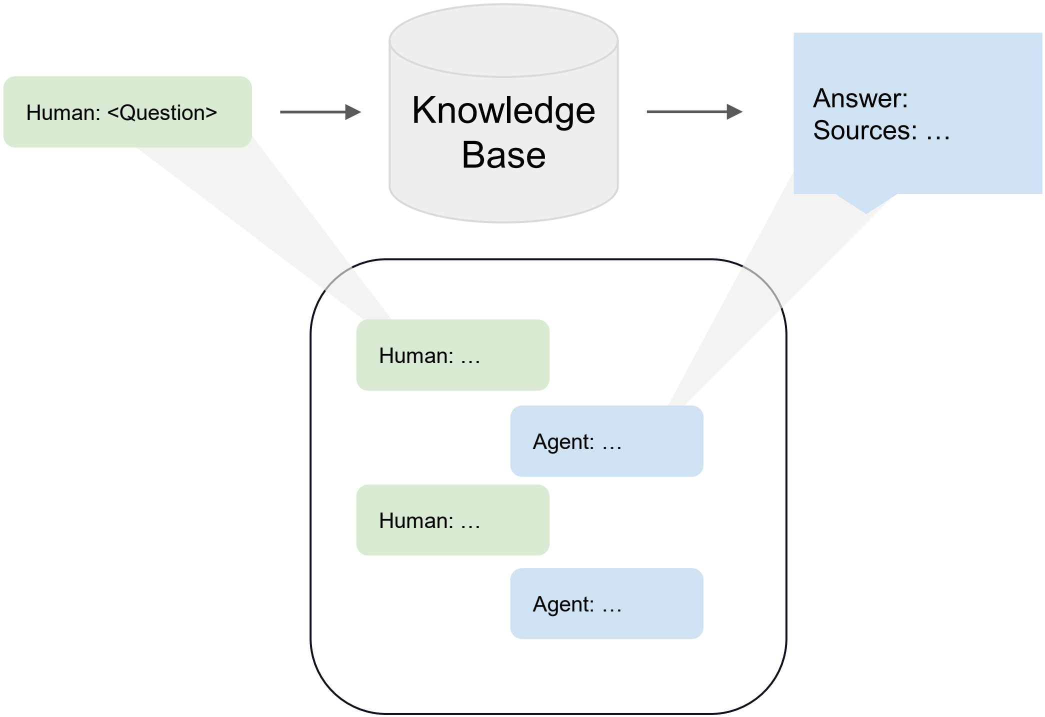


Conversational Agent

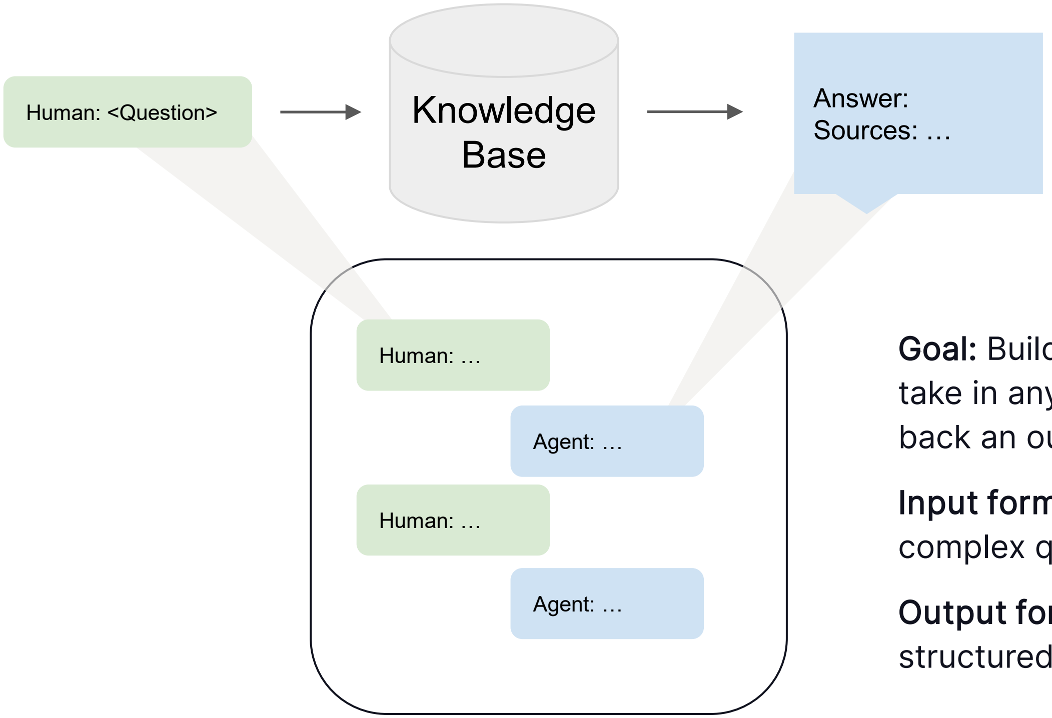


Workflow Automation

Building a Knowledge Assistant



Building a Knowledge Assistant



Goal: Build an interface that can take in any task as input and give back an output.

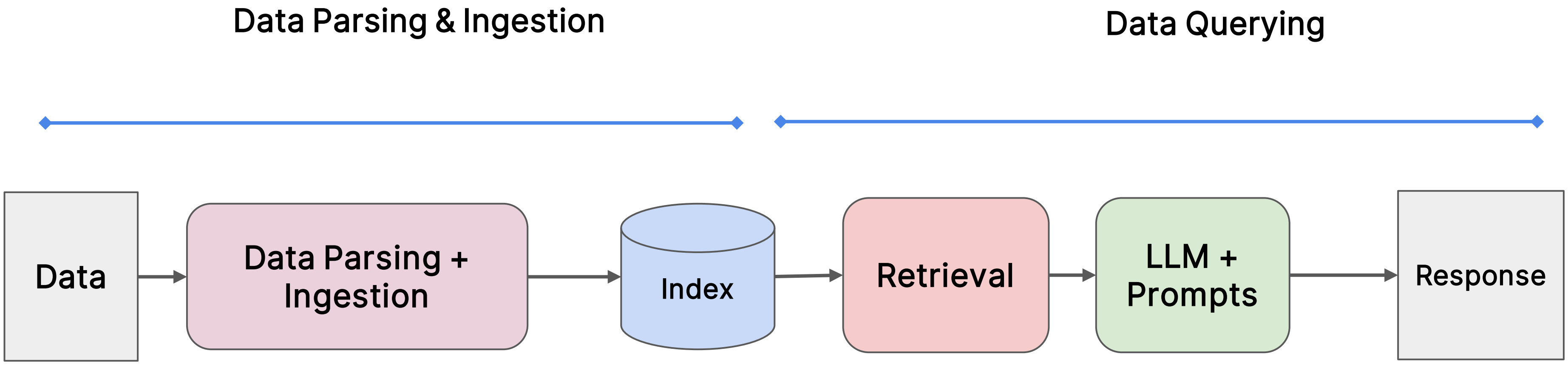
Input forms: simple questions, complex questions, research tasks

Output forms: short answer, structured output, research report

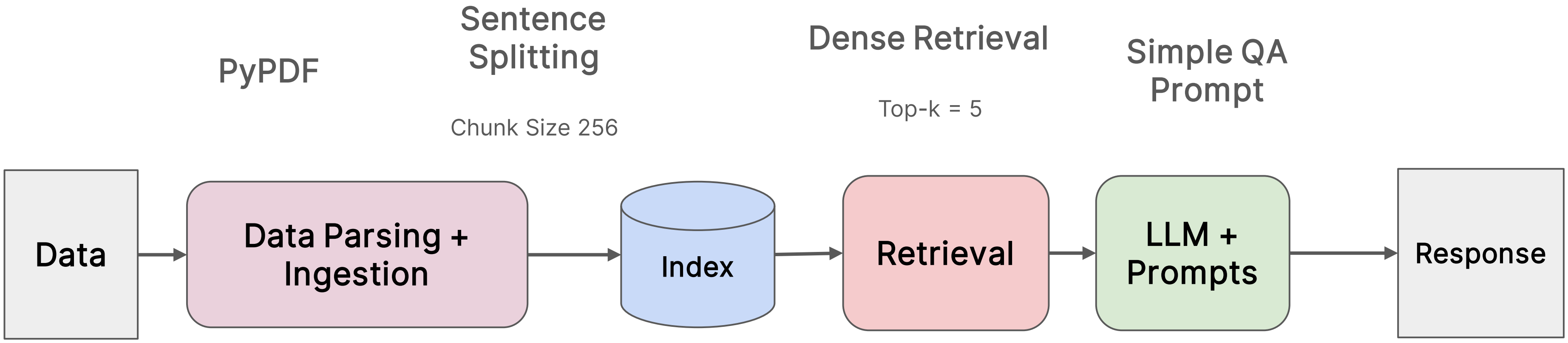
RAG

Retrieval Augmented Generation (RAG)

An overview of a RAG Pipeline



Naive RAG


PyPDF
Sentence Splitting
Chunk Size 256
Dense Retrieval
Top-k = 5
Simple QA Prompt
Data
Data Parsing + Ingestion
Index
Retrieval
LLM + Prompts
Response

Challenges with Naive RAG

Easy to Prototype, Hard to Productionize

Naive RAG approaches tend to work well for **simple** questions over a **simple, small** set of documents.

- “What are the main risk factors for Tesla?” (over Tesla 2021 10K)
- “What did the author do during his time at YC?” (Paul Graham essay)

Easy to Prototype, Hard to Productionize

But productionizing RAG over more questions and a larger set of data **is hard**!

Easy to Prototype, Hard to Productionize

Failure Modes:

- Simple Questions over Complex Data
- Simple Questions over Multiple Documents
- Complex Questions

Easy to Prototype, Hard to Productionize

Failure Modes:

- Simple Questions over Complex Data
- Simple Questions over Multiple Documents
- Complex Questions

The top priority goal should be figuring out how to **get high-response quality** from the set of representative questions you want to ask.

Can we do more?

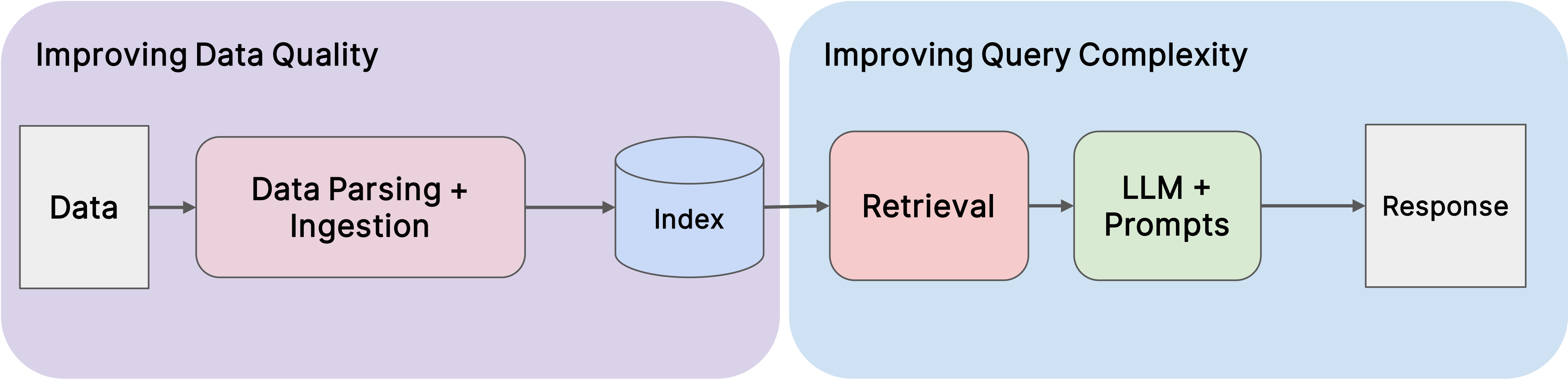
In the naive setting, RAG is boring.

🚫 It's just a glorified search system

🚫 There's many questions/tasks that naive RAG can't give an answer to.

💡 Can we go beyond simple search/QA to building a general **context-augmented research assistant?**

Main Focus Areas

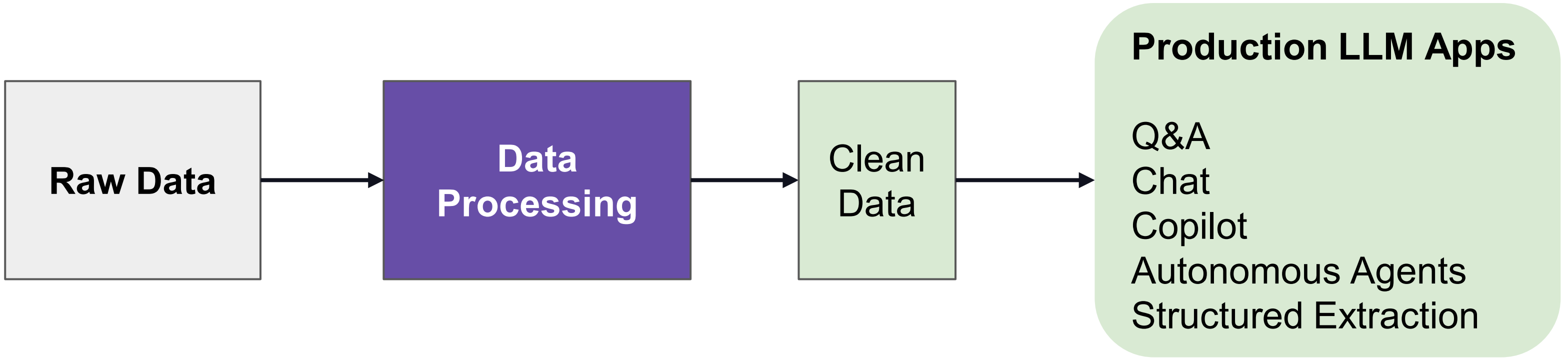


Improving Data Quality

RAG is only as Good as your Data

Garbage in = garbage out

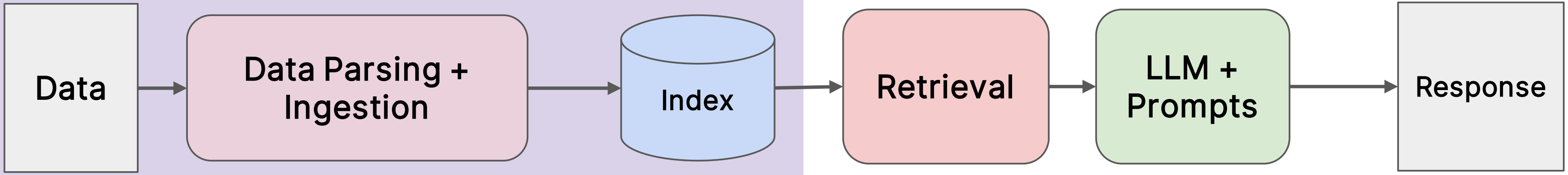
Good data quality is a **necessary** component of any production LLM app.



RAG is only as Good as your Data

Main Components of Data Processing:

- Parsing
- Chunking
- Indexing



General Principles

Parsing:

- Bad parsers are a key cause of garbage in == garbage out.
- Badly formatted text/tables confuse even the best LLMs

Chunking:

- Try to preserve semantically similar content.
  - 5 Levels of Text Splitting
- **Strong baseline:** page-level chunking.

Indexing:

- Raw text oftentimes confuse the embedding model.
- Don’t just embed the raw text, embed **references.**
- Having multiple embeddings point to the same chunk is a **good practice!**

Case Study: Complex Documents

A lot of documents can be classified as **complex:**

- Embedded Tables, Charts, Images
- Irregular Layouts
- Headers/Footers

Naive RAG indexing pipelines fail over these documents.

Let's build an **advanced RAG indexing pipeline.**

Liabilities

(in 000's of CHF)

Item	31 Dec 2022	31 Dec 2021	Change
Payables and accruals	4,685	4,066	619
Employee benefits	127,215	84,676	42,539
Contributions received in advance	6,975	10,192	(3,217)
Unearned revenue from exchange transactions	20	651	(631)
Deferred Revenue	71,301	55,737	15,564
Borrowings	28,229	29,002	(773)
Funds held in trust	30,373	29,014	1,359
Provisions	1,706	$\boxed{1,910}$	(204)
Total Liabilities	270,504	215,248	55,256

Most PDF Parsing is Inadequate

Extracts into a messy format that is impossible to pass down into more advanced ingestion/retrieval algorithms.

Please find below AXA's rankings and market shares in the main countries where it operates:

		Property & Casualty		Life & Savings		
		Ranking	Market share *(in %)*	Ranking	Market share *(in %)*	Sources
Main Developed Markets	France	2	12.9	3	8.4	"France Assureurs" as of December 31, 2022.
	Switzerland	1	13.3	4	7.8	Market share based on statutory premiums and market estimations by SIA (Swiss Insurance Association) figures as of January 31, 2023.
	Germany	6	4.8	8	3.4	GDV (German association of Insurance companies) as of December 31, 2021.
	Belgium	1	17.7	4	8.7	Assuralia (Belgium Professional Union of Insurance companies) based on gross written premium as of September 30, 2022.
	United Kingdom	4	8.2	n/a	n/a	UK General Insurance: Competitor Analytics 2021, Global Data, as of December 31, 2021.
	Ireland	1	31.9	n/a	n/a	Insurance Ireland P&C Statistics 2021 as of December 31, 2021.
	Spain	5	4.9	9	3.1	Spanish Association of Insurance Companies. ICEA as of December 31, 2022.
	Italy	5	5.8	9	3.9	Associazione Nazionale Imprese Assicuratrici (ANIA) as of December 31, 2021.
	Japan	13	0.6	9	5.0	Disclosed financial reports (excluding Kampo Life) for the 12 months ended September 30, 2022.
	Hong Kong	1	7.0	7	5.0	Insurance Authority statistics based on gross written premiums as of September 30, 2022.
	XL Insurance in the United States	16	1.8	n/a	n/a	AM Best 2021 as of December 31, 2021, in the United States in Commercial lines.
	XL Reinsurance worldwide	14	2.3	n/a	n/a	AM Best 2021 as of December 31, 2021.
Main Emerging Markets	Thailand	18	1.8	5	7.2	TGIA (Thai General Insurance Association) as of December 31, 2022 and TLAA (Thai Life Assurance Association) as of November 30, 2022.
	Indonesia	n/a	n/a	2	8.7	AAJI Statistic measured on Weighted New Business Premium as of September 30, 2022.
	Philippines	n/a	n/a	6	8.6	Insurance Commission measured on total premium income as of June 30, 2022.
	China	n/a	0.4	n/a	n/a	CBIRC (China Banking and Insurance Regulatory Commission) as of December 31, 2022 [(a)].
	Mexico	3	8.0	12	2.0	AMIS (Asociación Mexicana de Instituciones de Seguros) as of September 30, 2022.
	Brazil	15	1.4	n/a	n/a	SUSEP (Superintendência de Seguros Privados) as of September 2022.

(a) For Property & Casualty insurance market, CBIRC did not disclose information on ranking. For Life & Savings insurance market, CBIRC did not disclose information on market shares and ranking.

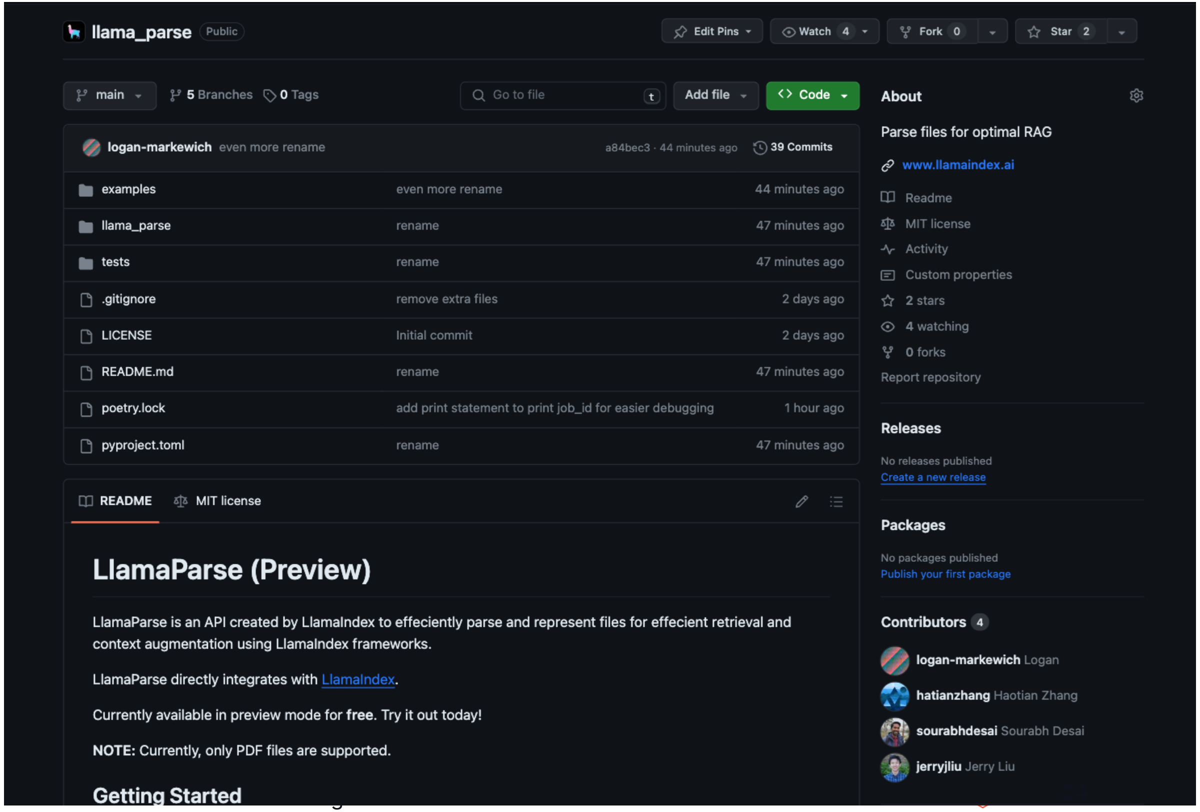
PyPDF

```
Please find below AXA’s rankings and market shares in the main countries where it operates:
Property & Casualty  Life & Savings
Market
shar e
(in %) Market
share
(in %) Ranking Ranking Sources
France 2 12.9 3 8.4 “France Assureurs” as of December 31, 2022.
Market share based on statutory premiums and market
estima
tions by SIA (Swiss Insurance Association) figures
as of January 31, 2023.  Switzerland  1 13.3 4 7.8
GDV (German association of Insur ance companies)
as of December 31, 2021. Germany 6 4.8 8 3.4
Assuralia (Belgium Professional Union of Insurance
comp
anies) based on gross written premium
as of September 30, 2022.s t e k  Mar loped Main DeveBelgium 1 17.7 4 8.7
UK Gener al Insuranc e: Competitor Analytics 2021, Global Data,
as of December 31, 2021. United Kingdom  4 8.2 n/a n/a
Ireland 1 31.9 n/a n/a Insurance Ireland P&C Statistics 2021 as of December 31, 2021.
Spanish Associa tion of Insurance Companies. ICEA
as of December 31, 2022. Spain 5 4.9 9 3.1
Associazione Nazionale Imprese A ssicuratrici (ANIA)
as of December 31, 2021. Italy 5 5.8 9 3.9
Disclosed financial r eports (ex cluding Kampo Life)
for the 12 months ended September 30, 2022. Japan 13 0.6 9 5.0
Insur ance A uthority statistics based on gross written premiums
as of September 30, 2022. Hong Kong 1 7.0 7 5.0
XL Insurance in
the United S
tates AM Best 2021 as of December 31, 2021, in the United States in Commercial lines. 1
6 1.8 n/a n/a
XL Reinsurance worldwide 14 2.3 n/a n/a AM Best 2021 as of December 31, 2021.
Thailand 18 1.8 5 7.2 TGIA (Thai General Insurance Association) as of December 31, 2022 and TL
AA (Thai Life
Assurance Association) as of November 30, 2022.  ts e rk ing Ma g Emer Main
Indonesia n/a n/a 2 8.7 AAJI Statistic measured on Weighted New Business Premium as of Sep tember 30, 2022.
Philippines n/
a n/a 6 8.6 Insurance Commission measured on total premium income as of June 30, 2022.
China n/a 0.4 n/a n/a CBIRC (China Banking and Insurance Regulatory Commission) as of Dec ember 31, 2022
(a).
Mexico 3 8.0 12 2.0 AMIS (Asociación Mexicana de Instituciones de Seguros) as of Sept
ember 30, 2022.
Brazil 15 1.4 n/a n/a SUSEP (Superintendência de Seguros Privados) as of September 2022.
```

LlamaParse

A special **Document Parser** designed to let you build RAG over Complex docs

https://github.com/run-llama/llama_parse



LlamaParse

Capabilities

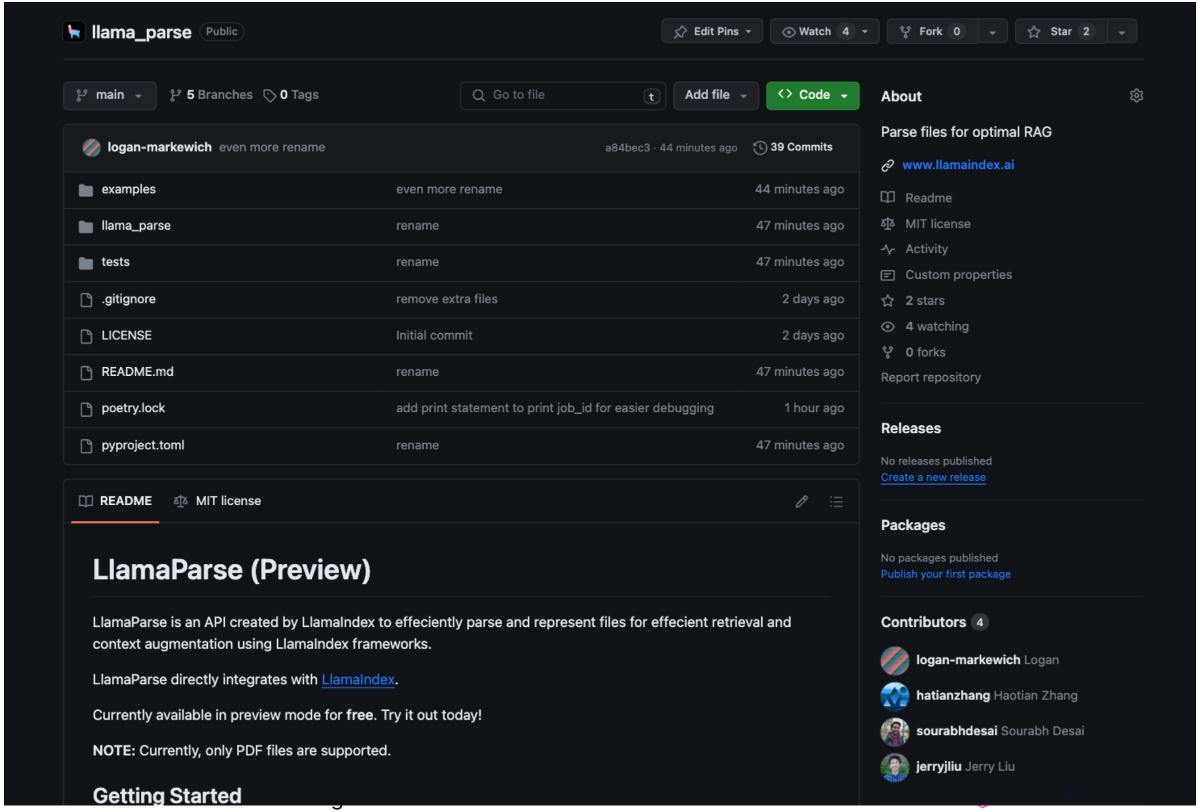
✅ Extracts tables / charts

✅ Input natural language parsing instructions

✅JSON mode

✅Image Extraction

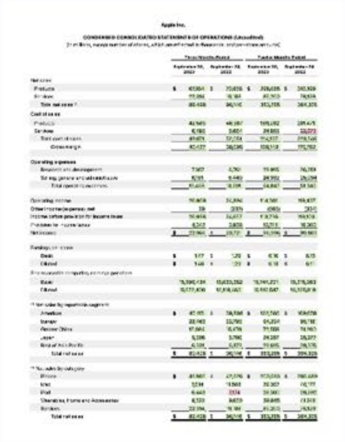
✅Support for ~10+ document types (.pdf, .pptx, .docx, .xml)



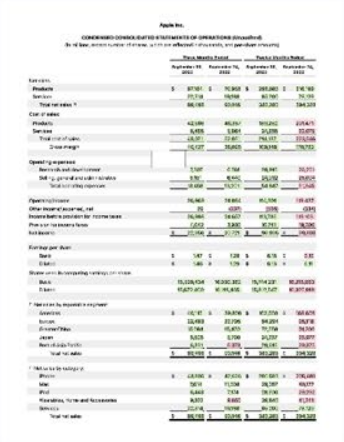
LlamaParse Results

Expanded:

LlamaParse



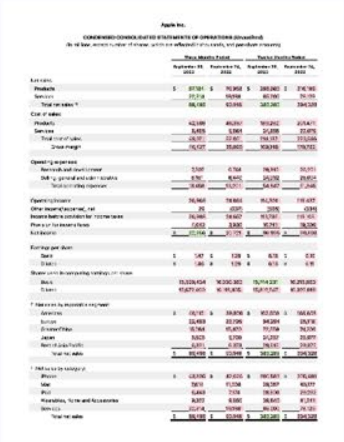
PyPDF



PyMuPDF

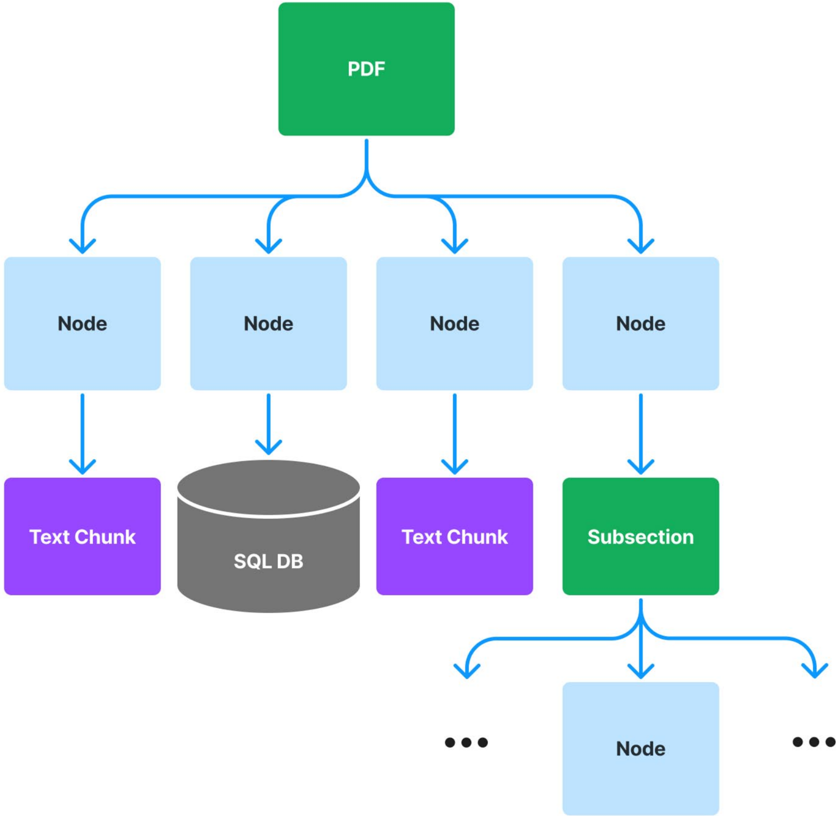
Textract

PdfMiner



LlamaParse + Advanced Indexing

1. Use LlamaParse to parse a document into a semi-structured markdown representation (text + tables)
2. Use a markdown parser to extract out text and table chunks
3. Use an LLM to extract a summary from each table → link to underlying table chunk.
4. Index a **graph** of text and table chunks.



Advanced Table Understanding

https://github.com/run-llama/llama_parse/blob/main/examples/demo_advanced.ipynb

```
[54]: query = "what is the Repayments of debt in the Cash flows from financing activities for Netflix?"

response_0 = baseline_pdf_query_engine.query(query)
print("***********Baseline PDF Query Engine***********")
print(response_0)


response_1 = raw_query_engine.query(query)
print("\n***********New LlamaParse+ Basic Query Engine***********")
print(response_1)

response_2 = recursive_query_engine.query(query)
print("\n***********New LlamaParse+ Recursive Retriever Query Engine***********")
print(response_2)
```

```
***********Baseline PDF Query Engine***********
The Repayments of debt in the Cash flows from financing activities for Netflix is not provided in the given information.

***********New LlamaParse+ Basic Query Engine***********
The repayments of debt for Netflix in the Cash flows from financing activities were $700,000 for the year ended as of December 31, 20
22, and $500,000 for the year ended as of December 31, 2021.

***********New LlamaParse+ Recursive Retriever Query Engine***********
The repayments of debt in the cash flows from financing activities for the year ended December 31, 2021 was $500,000.
```

Net cash provided by operating activities	[illegible]	[illegible]	[illegible]
Cash flows from investing activities:			
Purchases of property and equipment	(407,729)	(524,585)	(497,923)
Change in other assets	—	(26,919)	(7,431)
Acquisitions	(757,387)	(788,349)	—
Purchases of short-term investments	(911,276)	—	—
Net cash used in investing activities	(2,076,392)	(1,339,853)	(505,354)
Cash flows from financing activities:			
Proceeds from issuance of debt	—	—	1,009,464
Debt issuance costs	—	—	(7,559)
Repayments of debt	(700,000)	(500,000)	—
Proceeds from issuance of common stock	35,746	174,414	235,406
Repurchases of common stock	—	(600,022)	—
Taxes paid related to net share settlement of equity awards	—	(224,168)	—
Net cash provided by (used in) financing activities	(664,254)	(1,149,776)	1,237,311

Parsing Instructions

https://colab.research.google.com/drive/1dO2cwDCXjj9pS9yQDZ2vjg-0b5sRXQYo?usp=sharing

CALCULATING THE DERIVATIVE OF A CONSTANT, LINEAR, OR QUADRATIC FUNCTION

1. **Let's find the derivative of constant function $f(x) = \alpha$. The differential coefficient of $f(x)$ at $x = a$ is**

$$\lim_{\varepsilon \to 0} \frac{f(a+\varepsilon) - f(a)}{\varepsilon} = \lim_{\varepsilon \to 0} \frac{\alpha - \alpha}{\varepsilon} = \lim_{\varepsilon \to 0} 0 = 0$$

Thus, the derivative of $f(x)$ is $f'(x) = 0$. This makes sense, since our function is constant—the rate of change is 0.

To parse this, we take the same instructions as before and add one sentence: `Output any math equation in LATEX markdown (between $$)`. The result of parsing is clear LaTeX instructions, which render the equations perfectly:

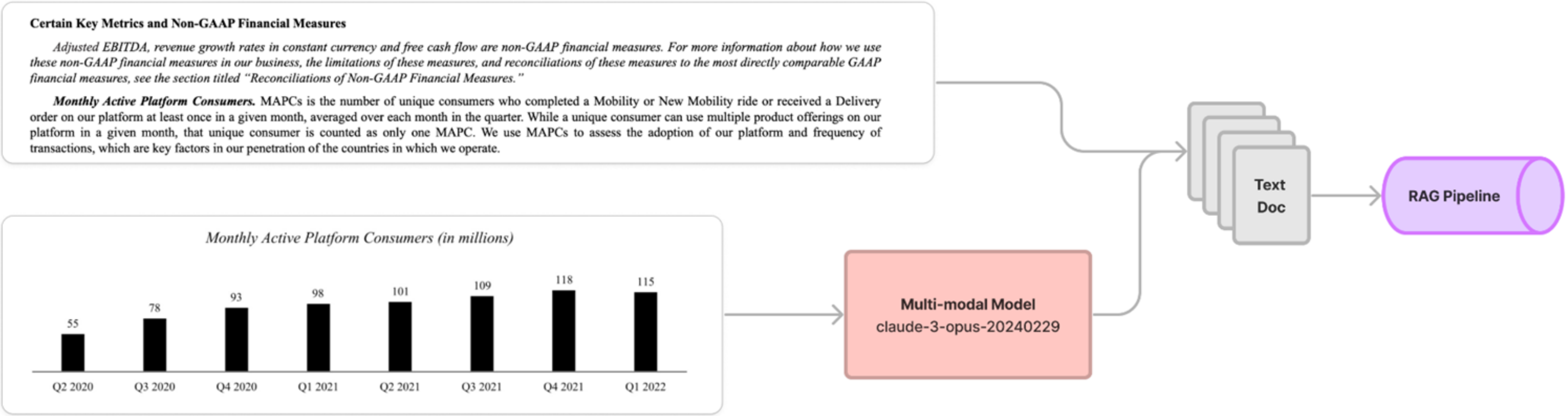
Calculating the Derivative of a Constant, Linear, or Quadratic Function

1. Let's find the derivative of constant function f(x) = α. The differential coefficient of f(x) at x = a is

$$\lim_{\varepsilon \to 0} \left(\frac{f(a+\varepsilon) - f(a)}{\varepsilon} \right) = \lim_{\varepsilon \to 0} \left(\frac{\alpha - \alpha}{\varepsilon} \right) = \lim_{\varepsilon \to 0} 0 = 0$$ Thus, the derivative of f(x) is f'(x) = 0. This makes sense, since our function is constant—the rate of change is 0.

JSON Mode

https://github.com/run-llama/llama_parse/blob/main/examples/demo_json.ipynb



RAG over Powerpoints

https://github.com/run-llama/llama_parse/blob/main/examples/other_files/demo_ppt_financial.ipynb

```
[27]: print(llama_parse_documents[0].get_content()[-2800:-2300])

ation and mitigation
---
|Item|31 Dec 2022|31 Dec 2021|Change|
|---|---|---|---|
|Payables and accruals|4,685|4,066|619|
|Employee benefits|127,215|84,676|42,539|
|Contributions received in advance|6,975|10,192|(3,217)|
|Unearned revenue from exchange transactions|20|651|(631)|
|Deferred Revenue|71,301|55,737|15,564|
|Borrowings|28,229|29,002|(773)|
|Funds held in trust|30,373|29,014|1,359|
|Provisions|1,706|1,910|(204)|
|Total Liabilities|270,504|215,248|55,256|
---
## Liabilities

Employee Ben
```

Compared against the original slide image.

Liabilities

(in 000's of CHF)

Item	31 Dec 2022	31 Dec 2021	Change
Payables and accruals	4,685	4,066	619
Employee benefits	127,215	84,676	42,539
Contributions received in advance	6,975	10,192	(3,217)
Unearned revenue from exchange transactions	20	651	(631)
Deferred Revenue	71,301	55,737	15,564
Borrowings	28,229	29,002	(773)
Funds held in trust	30,373	29,014	1,359
Provisions	1,706	1,910	(204)
Total Liabilities	270,504	215,248	55,256

Workshop

Let's build a RAG pipeline with Databricks LLMs + local embeddings

Improving Query Complexity

Complex Questions

There's certain questions we want to ask where naive RAG will fail.

Examples:

- **Summarization Questions:** "Give me a summary of the entire <company> 10K annual report"

Complex Questions

There's certain questions we want to ask where naive RAG will fail.

Examples:

- **Summarization Questions:** "Give me a summary of the entire <company> 10K annual report"
- **Comparison Questions:** "Compare the open-source contributions of candidate A and candidate B"

Complex Questions

There's certain questions we want to ask where naive RAG will fail.

Examples:

- **Summarization Questions:** "Give me a summary of the entire <company> 10K annual report"
- **Comparison Questions:** "Compare the open-source contributions of candidate A and candidate B"
- **Structured Analytics + Semantic Search:** "Tell me about the risk factors of the highest-performing rideshare company in the US"

Complex Questions

There's certain questions we want to ask where naive RAG will fail.

Examples:

- **Summarization Questions:** "Give me a summary of the entire <company> 10K annual report"
- **Comparison Questions:** "Compare the open-source contributions of candidate A and candidate B"
- **Structured Analytics + Semantic Search:** "Tell me about the risk factors of the highest-performing rideshare company in the US"
- **General Multi-part Questions:** "Tell me about the pro-X arguments in article A, and tell me about the pro-Y arguments in article B, make a table based on our internal style guide, then generate your own conclusion based on these facts."

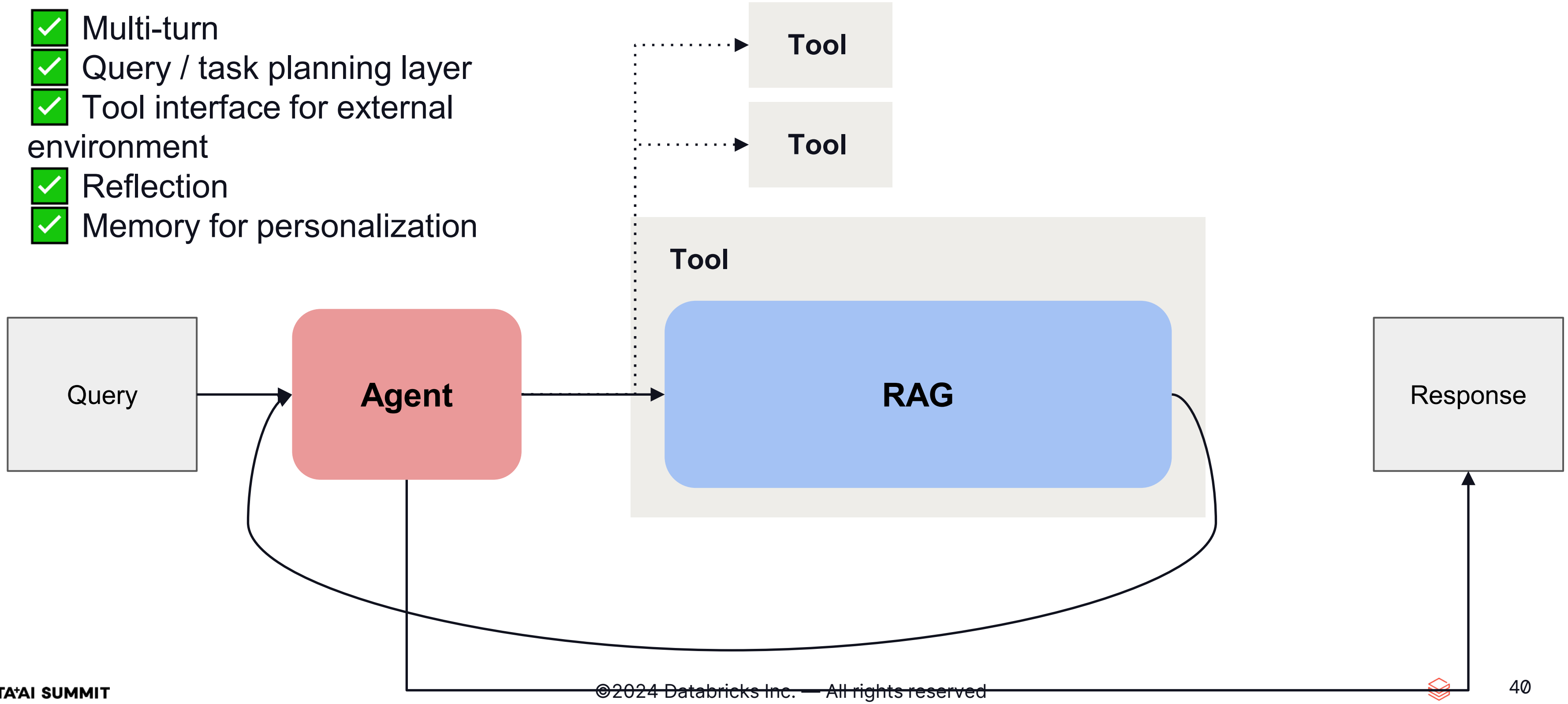
From RAG to Agents

From RAG to Agents

⚠ Single-shot
⚠ No query understanding/planning
⚠ No tool use
⚠ No reflection, error correction
⚠ No memory (stateless)

From RAG to Agents

- ✅ Multi-turn
- ✅ Query / task planning layer
- ✅ Tool interface for external environment
- ✅ Reflection
- ✅ Memory for personalization



From Simple to Advanced Agents

Agent Ingredients

Routing

Tool Use

One-Shot Query Planning

Conversation Memory

Full Agents

ReAct

Dynamic Planning + Execution

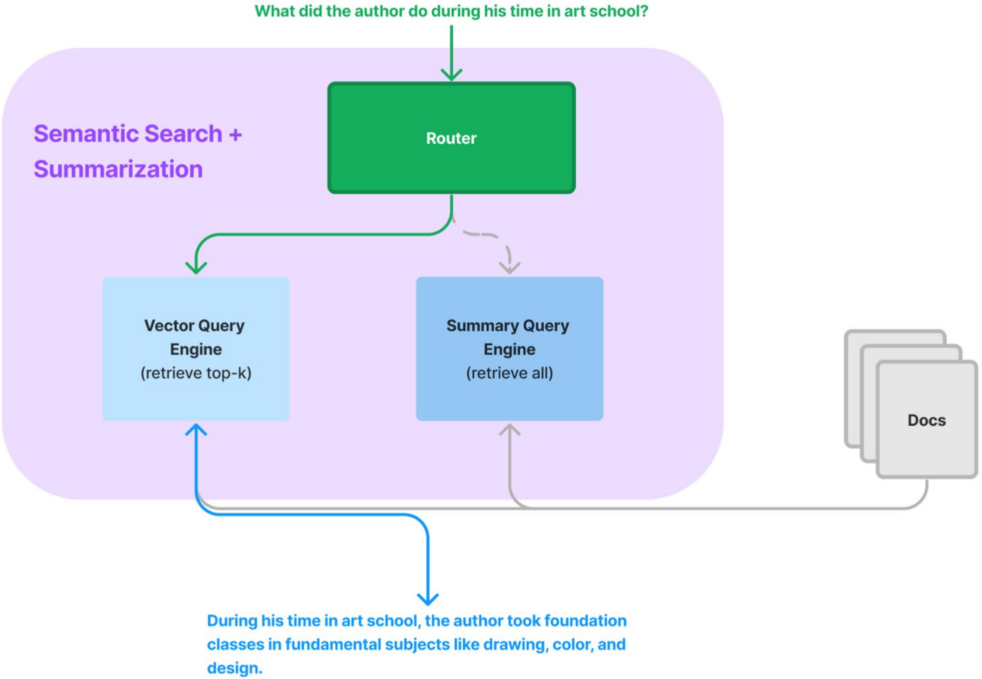
Simple
Lower Cost
Lower Latency

Advanced
Higher Cost
Higher Latency

Routing

Simplest form of agentic reasoning.

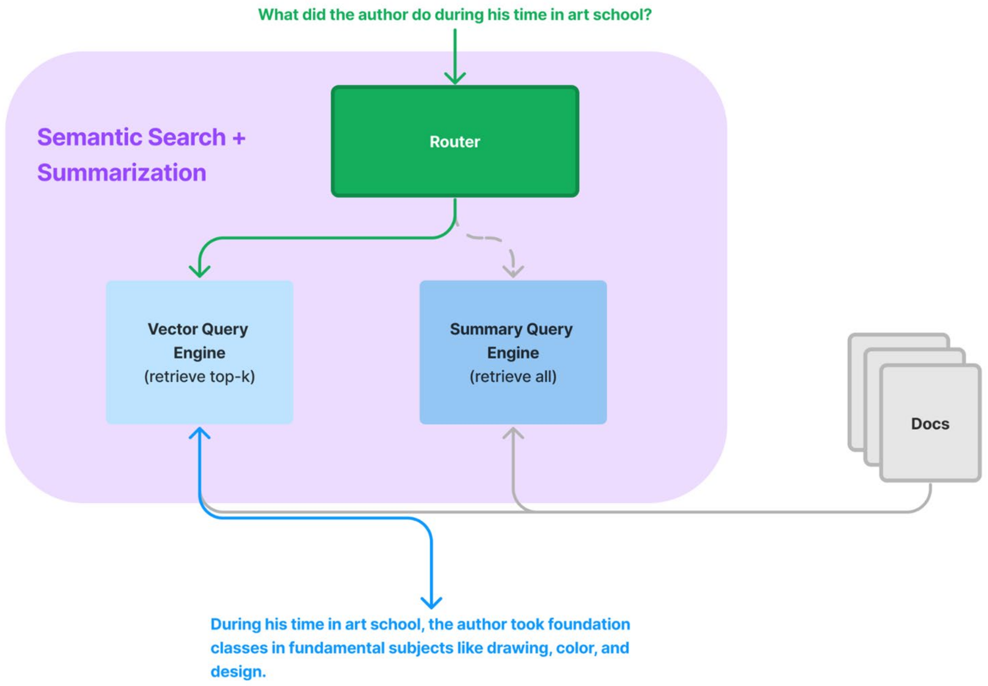
Given user query and set of choices, output subset of choices to route query to.



Routing

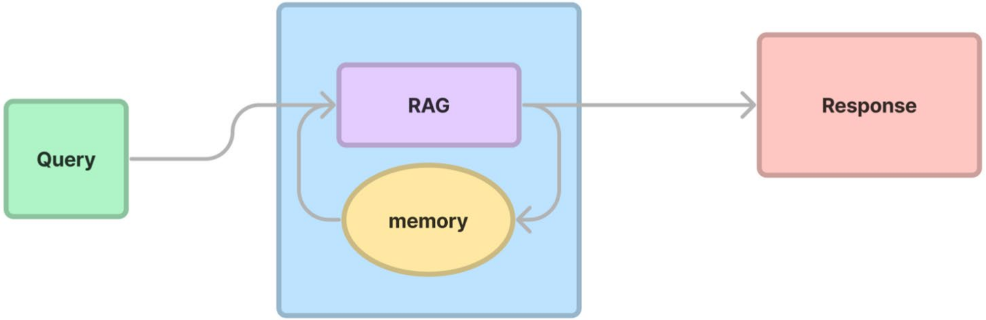
Use Case: Joint QA and Summarization

Guide



Conversation Memory

In addition to current query, take int
input to your RAG pipeline.

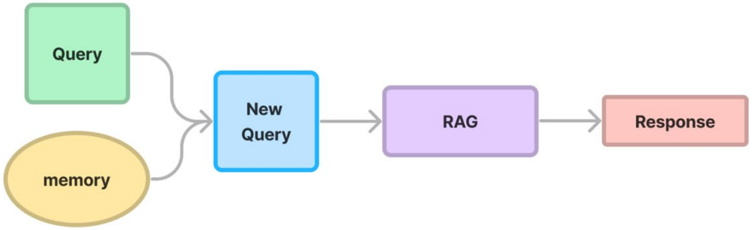


Conversation Memory

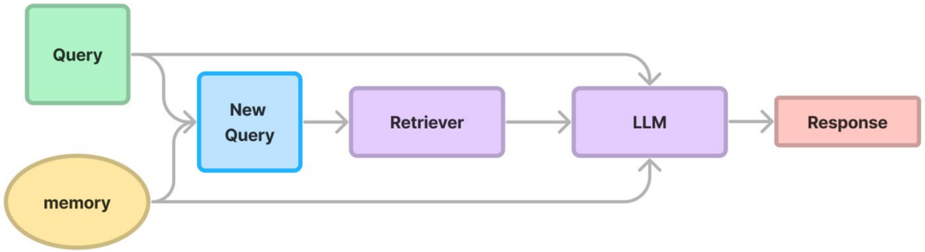
How to account for conversation history in a RAG pipeline?

- Condense question
- Condense question + context

Condense Question



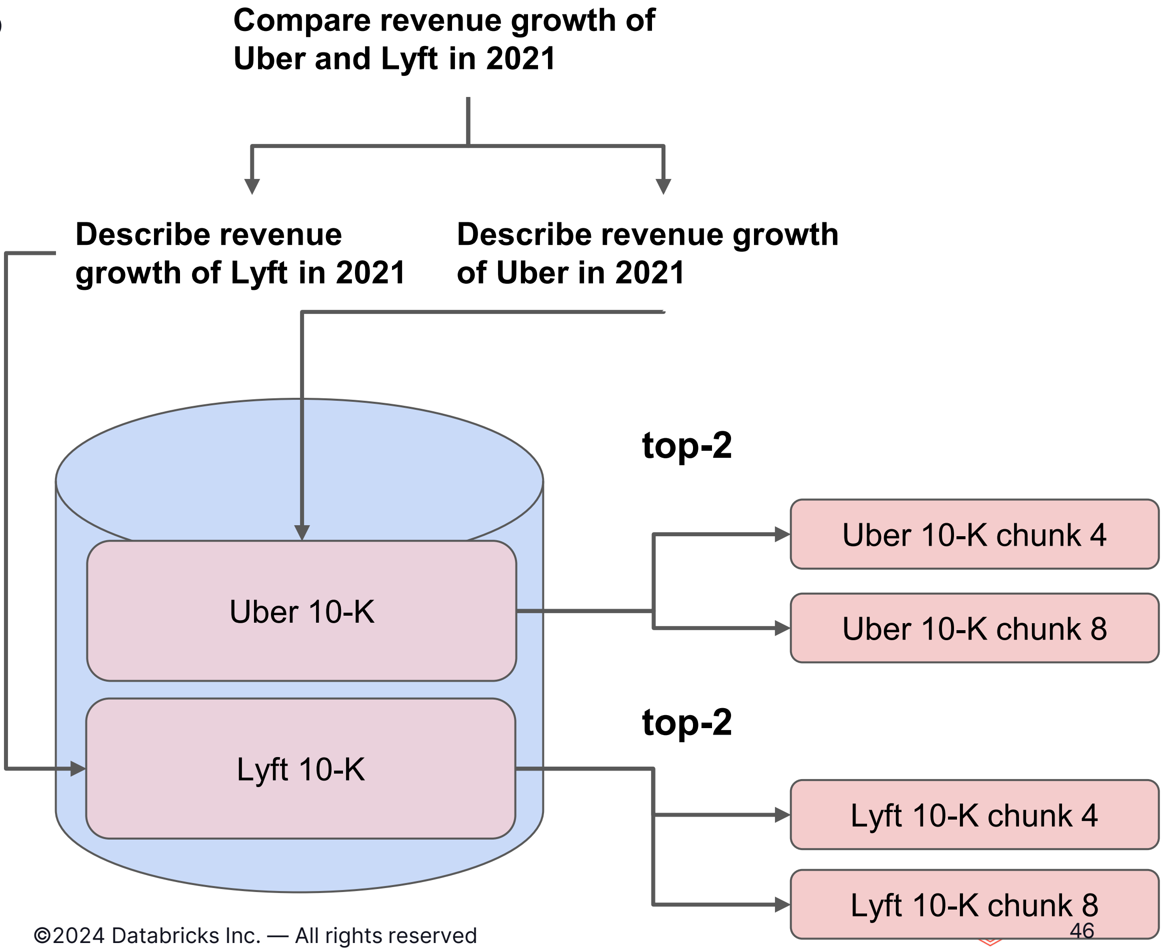
Condense Question for Context Retrieval



Query Planning

Break down query into parallelizable sub-queries.

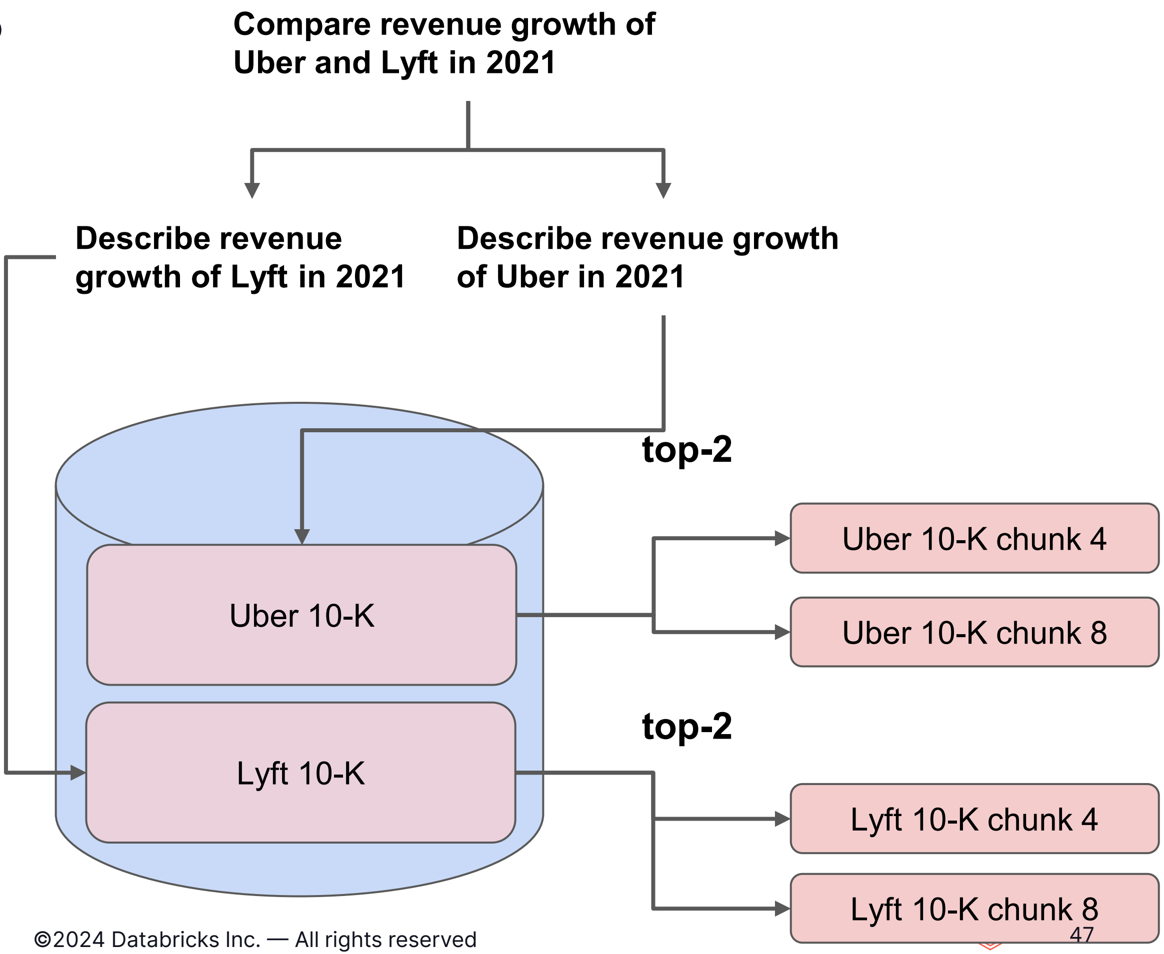
Each sub-query can be executed against any set of RAG pipelines



Query Planning

Example: Compare revenue of Uber and Lyft in 2021

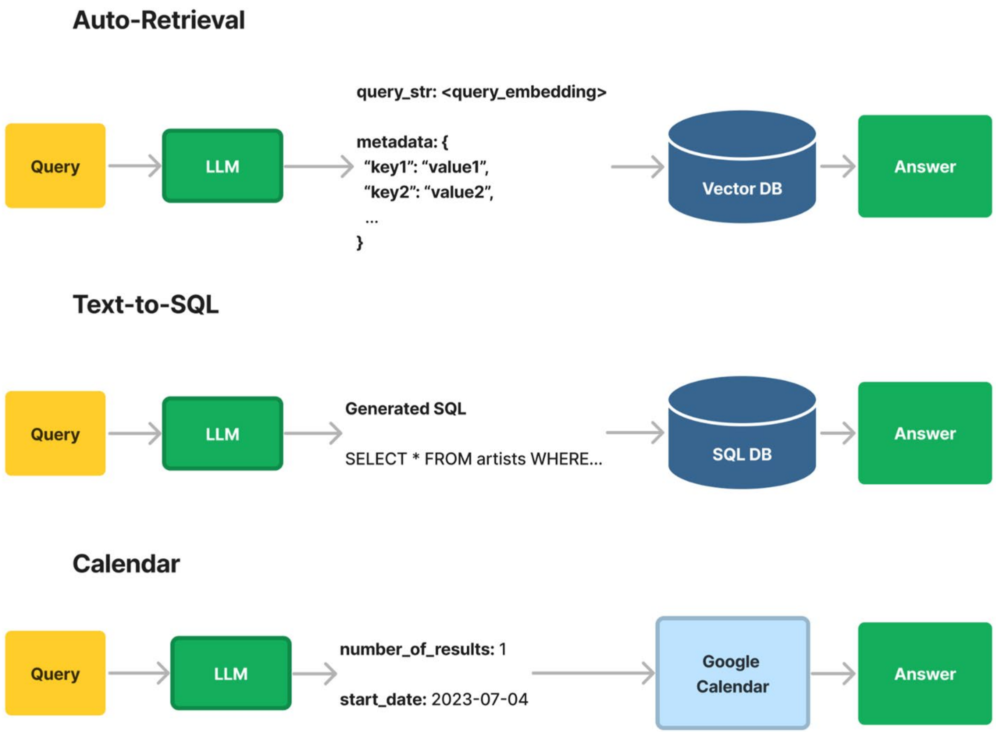
Query Planning Guide



Tool Use

Use an LLM to call an API

Infer the parameters of that API

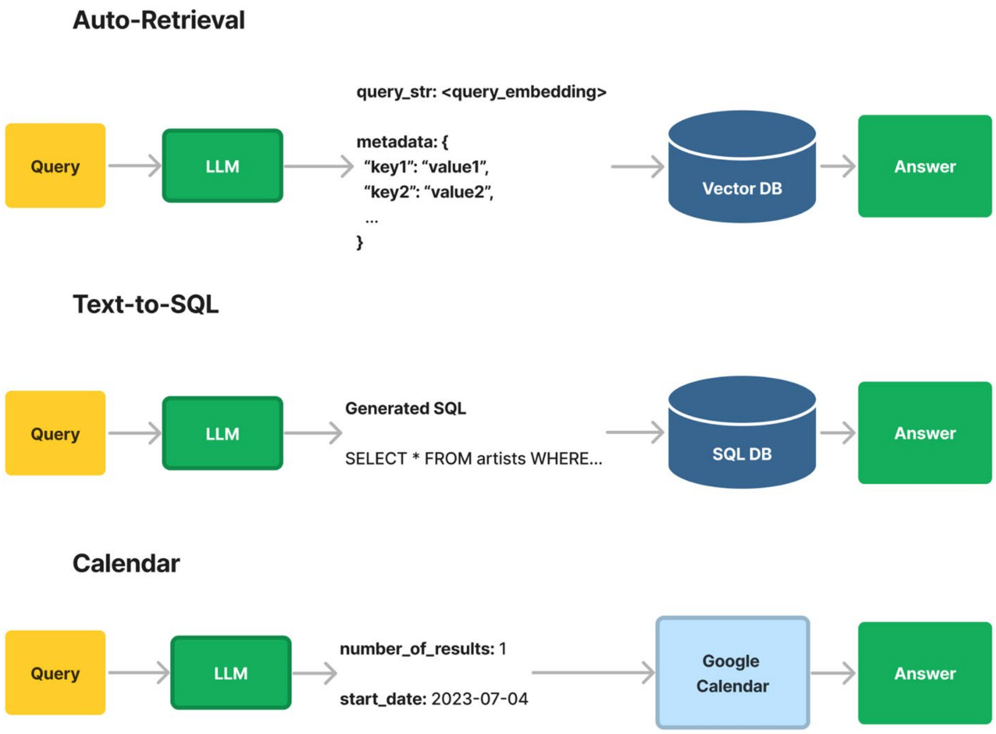


Tool Use

In normal RAG you just pass through the query.

But what if you used the LLM to infer all the parameters for the API interface?

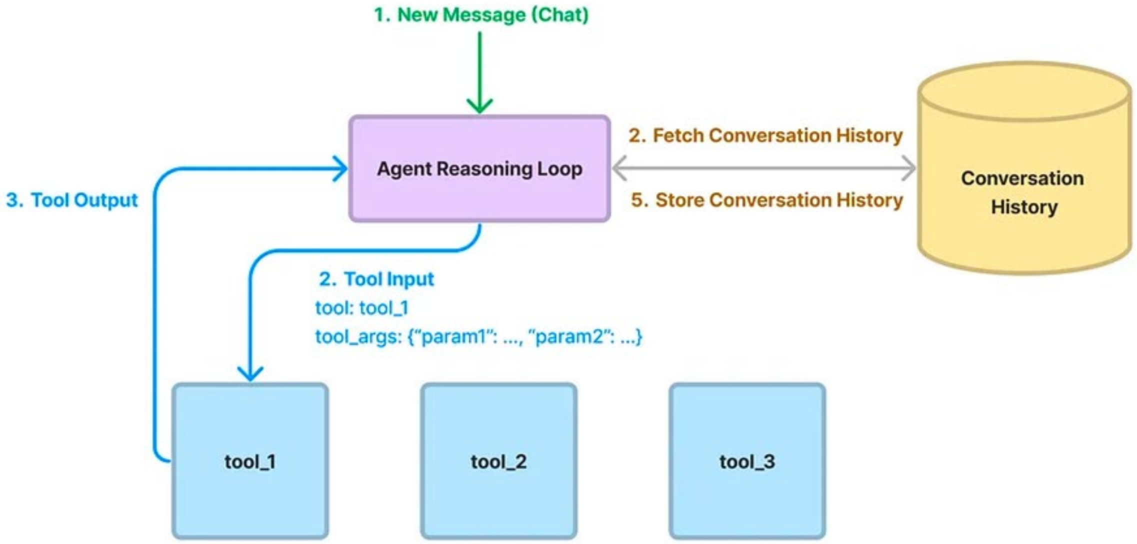
A key capability in many QA use cases (auto-retrieval, text-to-SQL, and more)



Let's put them together

- All of these are **agent ingredients**
- Let's put them together for a full agent system
    - Query planning
    - Memory
    - Tool Use
- Let's add additional components
    - Reflection
    - Controllability
    - Observability

Core Components of a Full Agent



Minimum necessary ingredients:

- Query planning
- Memory
- Tool Use

Agent Reasoning Loops

Sequential: Generate next step given previous steps (chain-of-thought prompt)

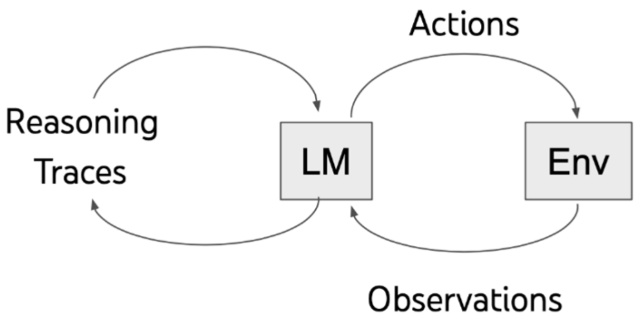
DAG-based planning (deterministic): Generate a deterministic DAG of steps. Replan if steps don't reach desired state.

Tree-based planning (stochastic): Sample multiple future states at each step. Run Monte-Carlo Tree Search (MCTS) to balance exploration vs. exploitation.

Agent Reasoning: Sequential

ReAct: Chain-of-thought and tool use through prompting.

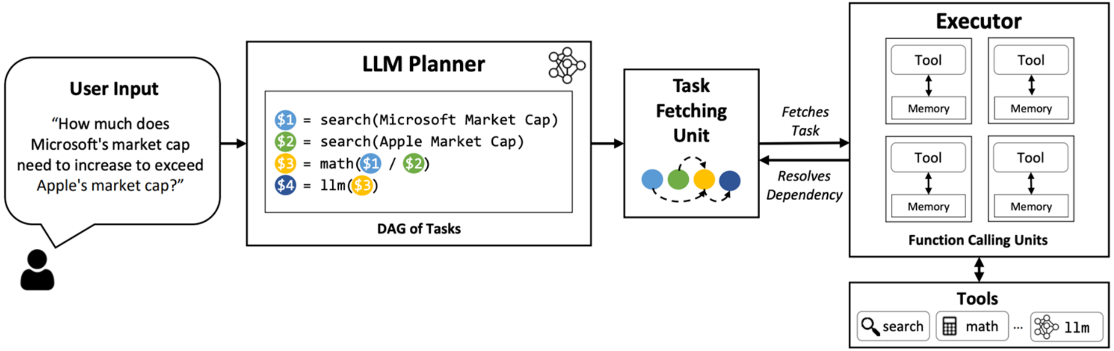
Function Calling Loop: Call LLM Function Calling APIs in a loop until done.



ReAct (Reason + Act)

Agent Reasoning: DAG-based Planning

LLM Compiler (Kim et al. 2023): An agent compiler for parallel multi-function planning + execution.



Agent Reasoning: Tree-based Planning

Tree of Thoughts (Yao et al. 2023)

Reasoning via Planning (Hao et al. 2023)

Language Agent Tree Search (Zhou et al. 2023)

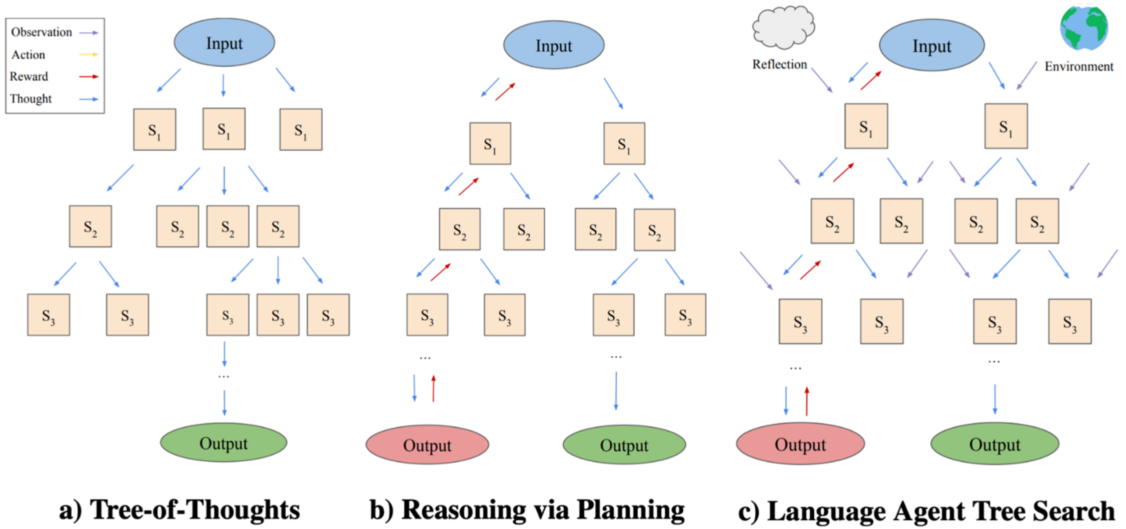


Figure 2: An overview of the differences between LATS and recently proposed LM search algorithms ToT (Yao et al., 2023a) and RAP (Hao et al., 2023). LATS leverages environmental feedback and self-reflection to further adapt search and improve performance.

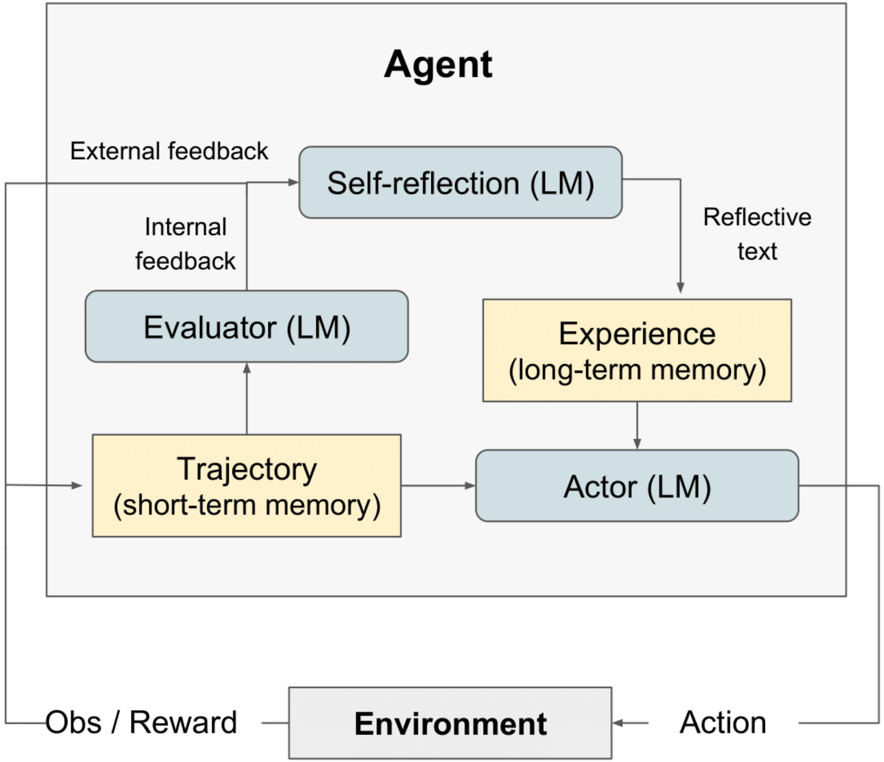
Self-Reflection

Use feedback to improve agent execution and reduce errors

🗣️ Human feedback

🤖 LLM feedback

Use few-shot examples instead of retraining the model!



Reflexion: Language Agents with Verbal Reinforcement Learning, by Shinn et al. (2023)

Additional Requirements

- **Observability:** see the full trace of the agent
    - Observability Guide
- **Control:** Be able to guide the intermediate steps of an agent *step-by-step*
    - Lower-Level Agent API
- **Customizability:** Define your own agentic logic around any set of tools.
    - Custom Agent Guide
    - Custom Agent with Query Pipeline Guide
- **Multi-agents:** Define multi-agent interactions!
    - Synchronously: Define an explicit flow between agents
    - Asynchronously: Treat each agent as a microservice that can communicate with each other.
        - Upcoming in LlamaIndex!
    - Current Frameworks: Autogen, CrewAI

Workshop

Let's extend our RAG pipeline into an agent!