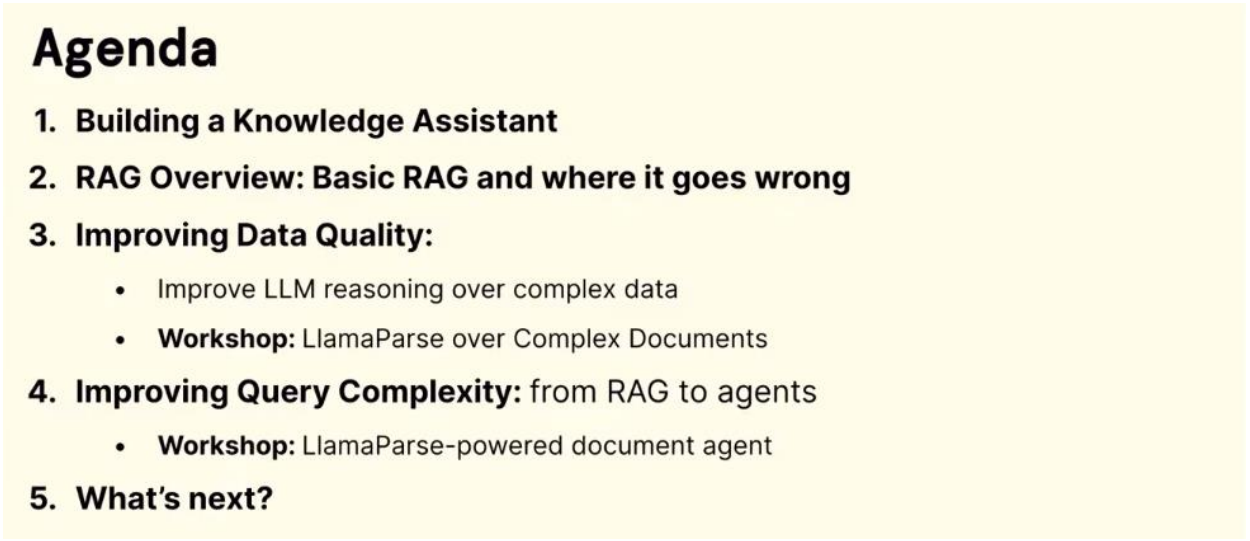
Building Production RAG Over Complex Documents

Databricks
142K subscribers
Subscribe

1.7K | Share | Ask | Download | ...

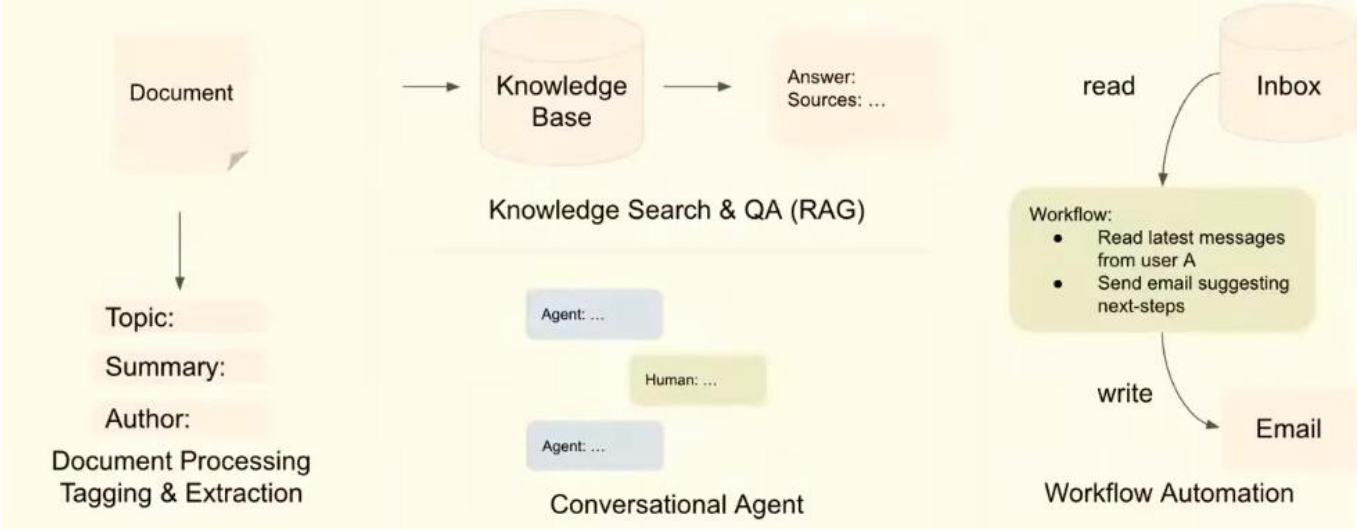
61,492 views Jul 23, 2024

Large Language Models (LLMs) are revolutionizing how users search for, interact with, and generate new content. Some recent stacks and toolkits around Retrieval-Augmented Generation (RAG) have emerged, enabling users to build applications such as chatbots using LLMs on their private data. However, while setting up naive RAG is straightforward, building production RAG is very challenging, especially as users scale to larger and more complex data sources. A classic example is a large number of PDFs with embedded tables. RAG is only as good as your data, and developers must carefully consider how to parse, ingest, and retrieve their data to successfully build RAG over complex documents. This session provides an in-depth exploration of this entire process; you will get an overview of the process around building a RAG pipeline that can handle messy, complicated PDF documents. This includes implementing a parsing strategy for parsing a complex document with embedded objects. This consists of an indexing strategy to process these documents beyond simple chunking techniques. We will then explore various advanced retrieval algorithms to handle questions about the tabular and unstructured data and discuss their use cases and tradeoffs.

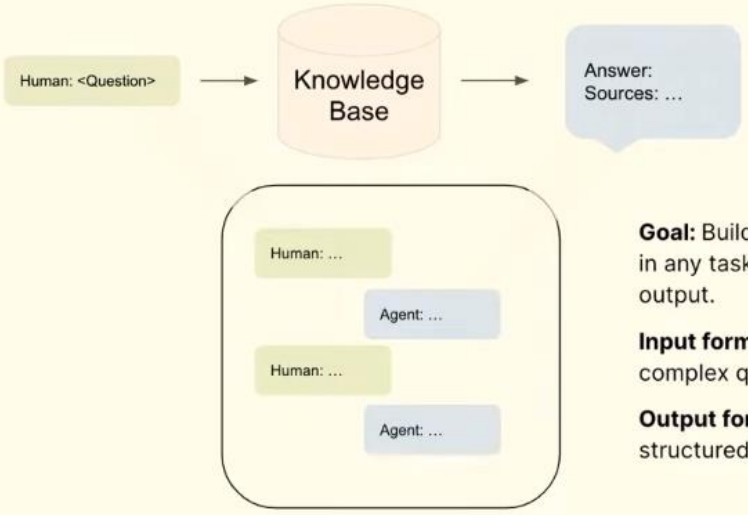


We will be seeing how to build a full end to end RAG pipeline from the data side to the query side.

Enterprise Use Cases



Building a Knowledge Assistant



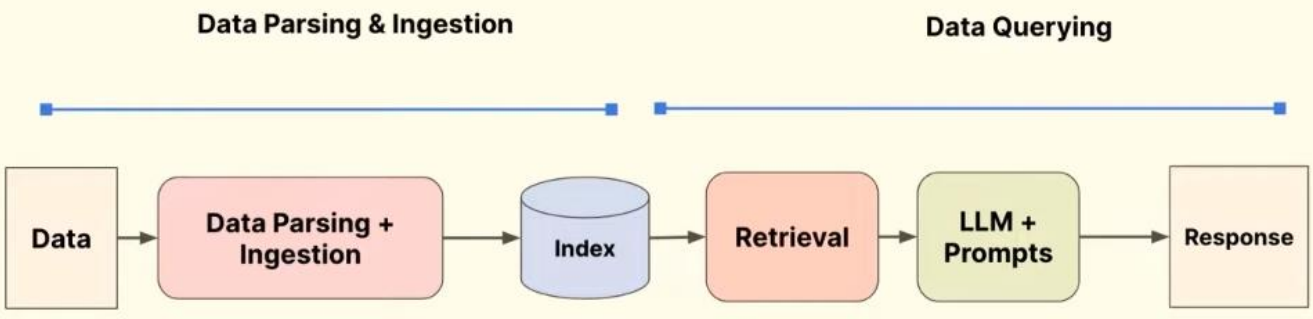
Goal: Build an interface that can take in any task as input and give back an output.

Input forms: simple questions, complex questions, research tasks

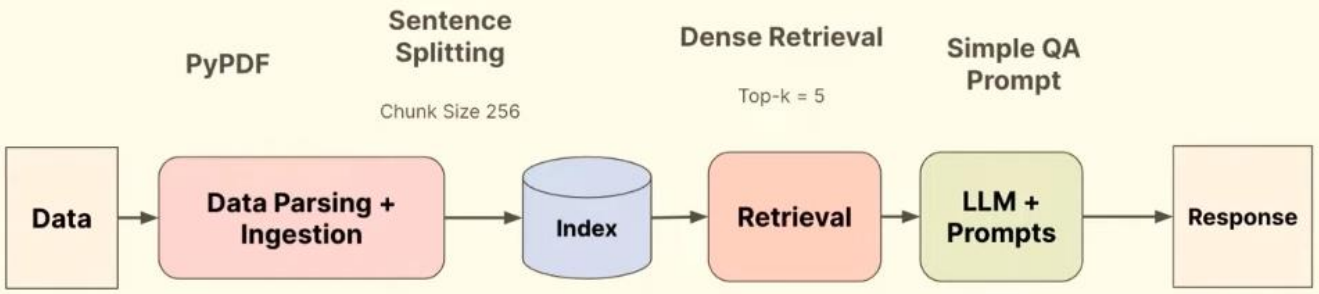
Output forms: short answer, structured output, research report

Retrieval Augmented Generation (RAG)

An overview of a RAG Pipeline



Naive RAG



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)

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

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?

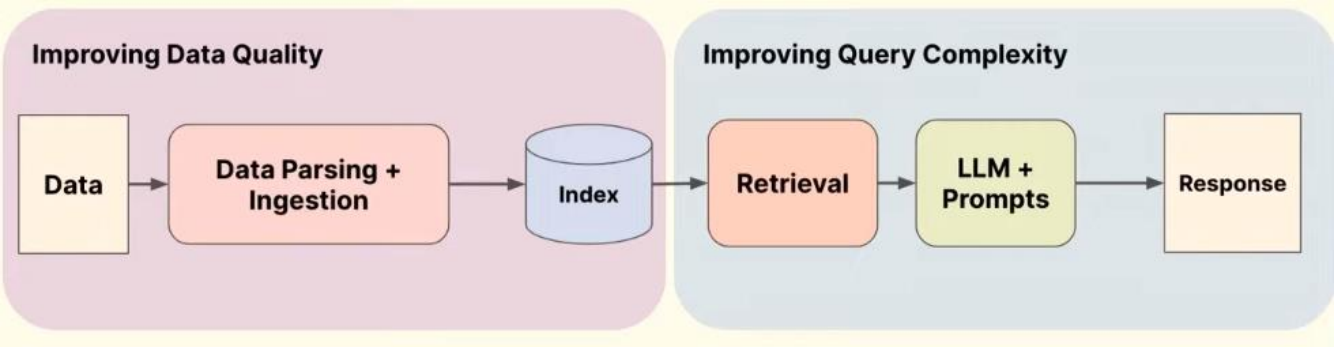
In the naive setting, RAG is boring.

🚫 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

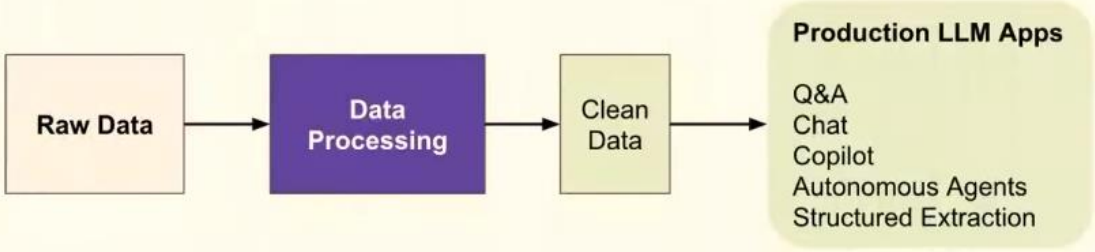


Improving Data Quality

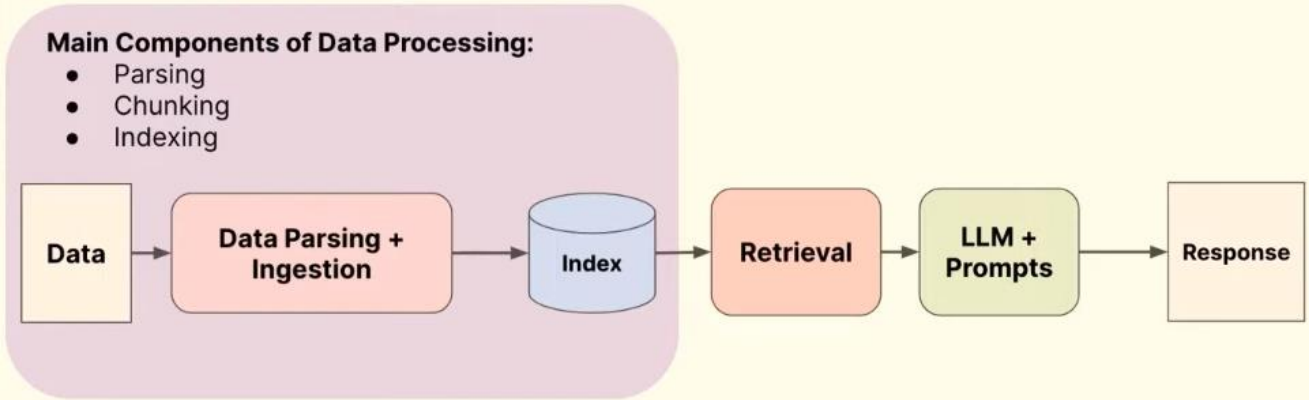
RAG is only as Good as your Data

Garbage in = garbage out

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



RAG is only as Good as your Data



You can add metadata to your chunks to provide more details about the parent document ID.

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.**

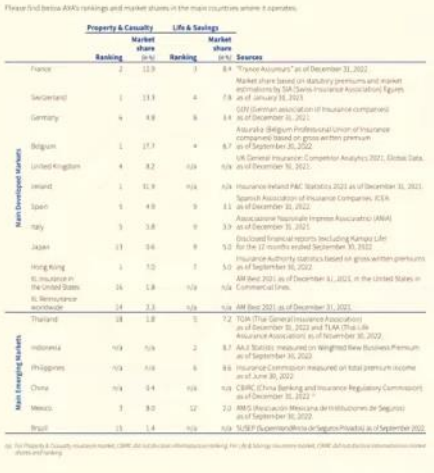
Liabilities

(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	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.



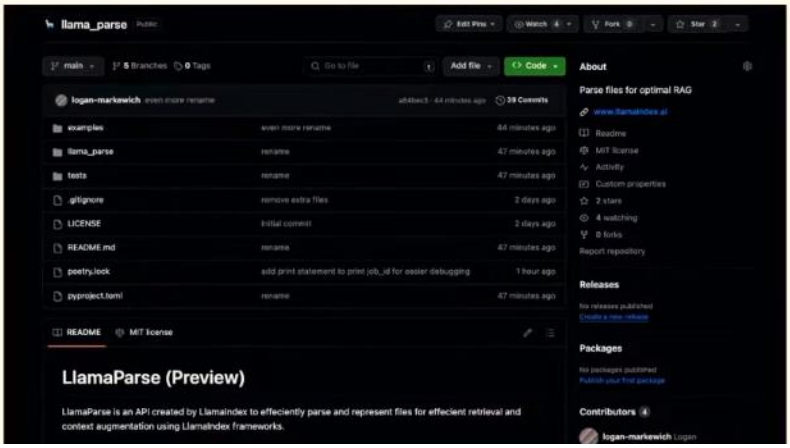
PyPDF

```
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
estimate
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.  ta x rk ing Ma p 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 n/a 4 n/a CBIRC (China Banking and Insurance Regulatory Commission) as of Dec ember 31, 2022
(a).
Mexico 3 8.8 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

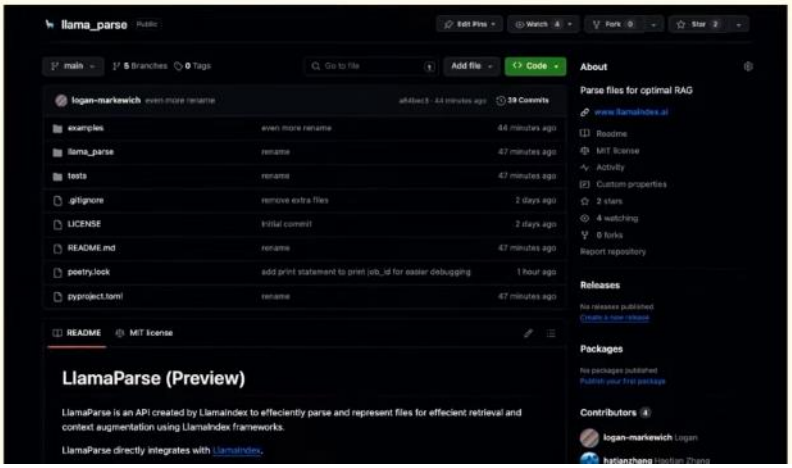
✅ Extracts tables / charts

✅ Input natural language parsing instructions

✅ JSON mode

✅ Image Extraction

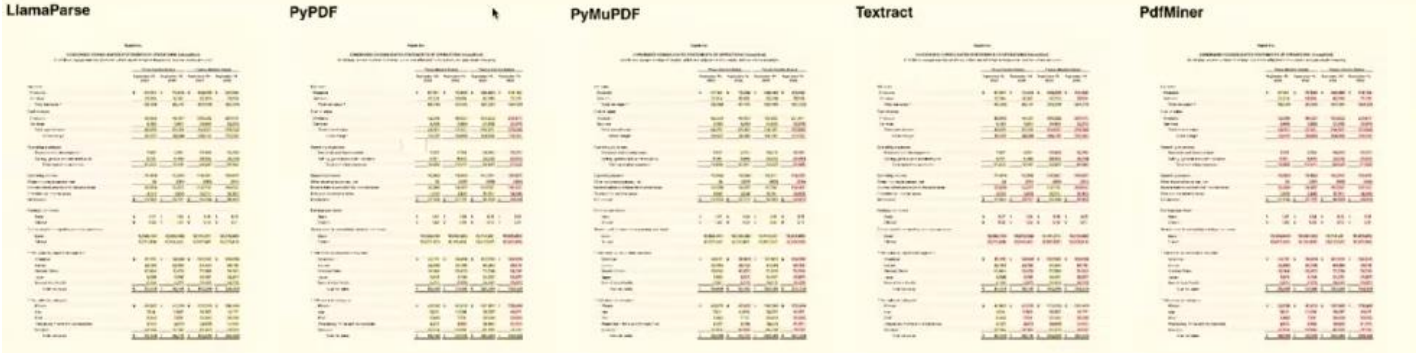
✅ Support for ~10+ document types (.pdf, .pptx, .docx, .xml)



LlamaParse Results

Expanded:

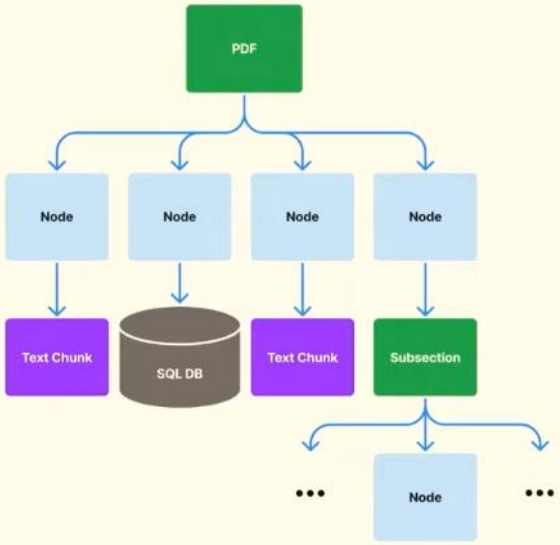
LlamaParse | PyPDF | PyMuPDF | Textract | PdfMiner



	September 30, 2023	September 24, 2022	September 30, 2023	September 24, 2022
Net sales:				
Products	$ 67,184	$ 70,958	$ 298,085	$ 316,199
Services	22,314	19,188	85,200	78,129
Total net sales [1]	89,498	90,146	383,285	394,328
Cost of sales:				
Products	42,586	46,387	189,282	201,471
Services	6,485	5,664	24,855	22,075
Total cost of sales	49,071	52,051	214,137	223,546
Gross margin	40,427	38,095	169,148	170,782
Operating expenses:				
Research and development	7,307	6,761	29,915	26,251
Selling, general and administrative	6,151	6,440	24,932	25,094
Total operating expenses	13,458	13,201	54,847	51,345
Operating income	26,969	24,894	114,301	119,437
Other income/(expense), net	29	(237)	(565)	(334)
Income before provision for income taxes	26,998	24,657	113,736	119,103
Provision for income taxes	4,042	3,936	16,741	19,300
Net income	$ 22,956	$ 20,721	$ 96,995	$ 99,803
Earnings per share:				
Basic	$ 1.47	$ 1.29	$ 6.16	$ 6.15
Diluted	$ 1.46	$ 1.29	$ 6.13	$ 6.11
Shares used in computing earnings per share:				
Basic	15,599,434	16,030,382	15,744,231	16,215,963
Diluted	15,672,400	16,118,465	15,812,547	16,325,819
[1] Net sales by reportable segment:				
Americas	$ 40,115	$ 39,808	$ 162,560	$ 169,658

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

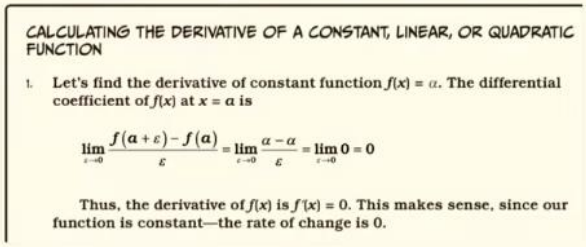
```
[54]: query = "what is the Repayments of debt in the Cash flows from financing activities for Netflix?"

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)

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, 2022, 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.
```

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

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

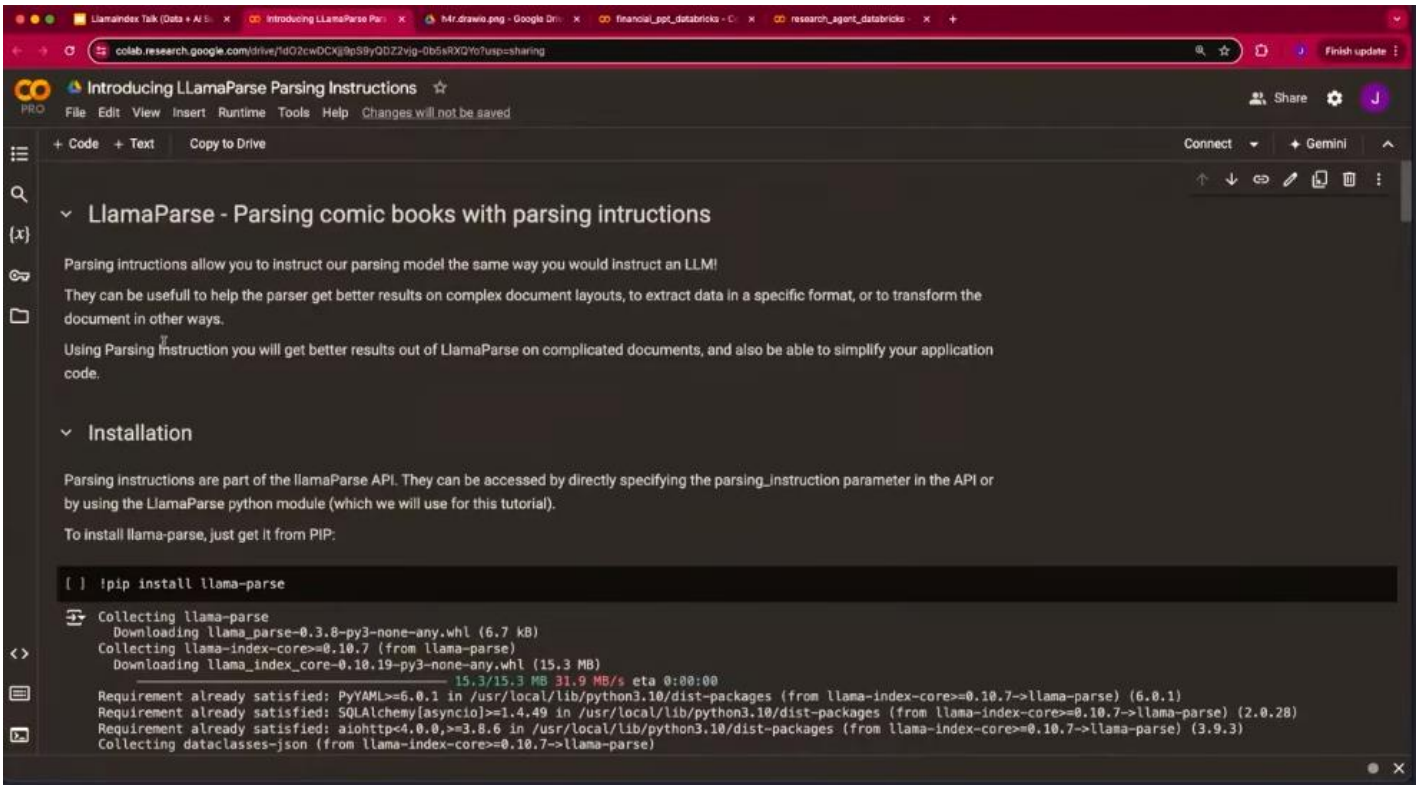
Thus, the derivative of $f(x)$ is $f'(x) = 0$. This makes sense, since our function is constant—the rate of change is 0.

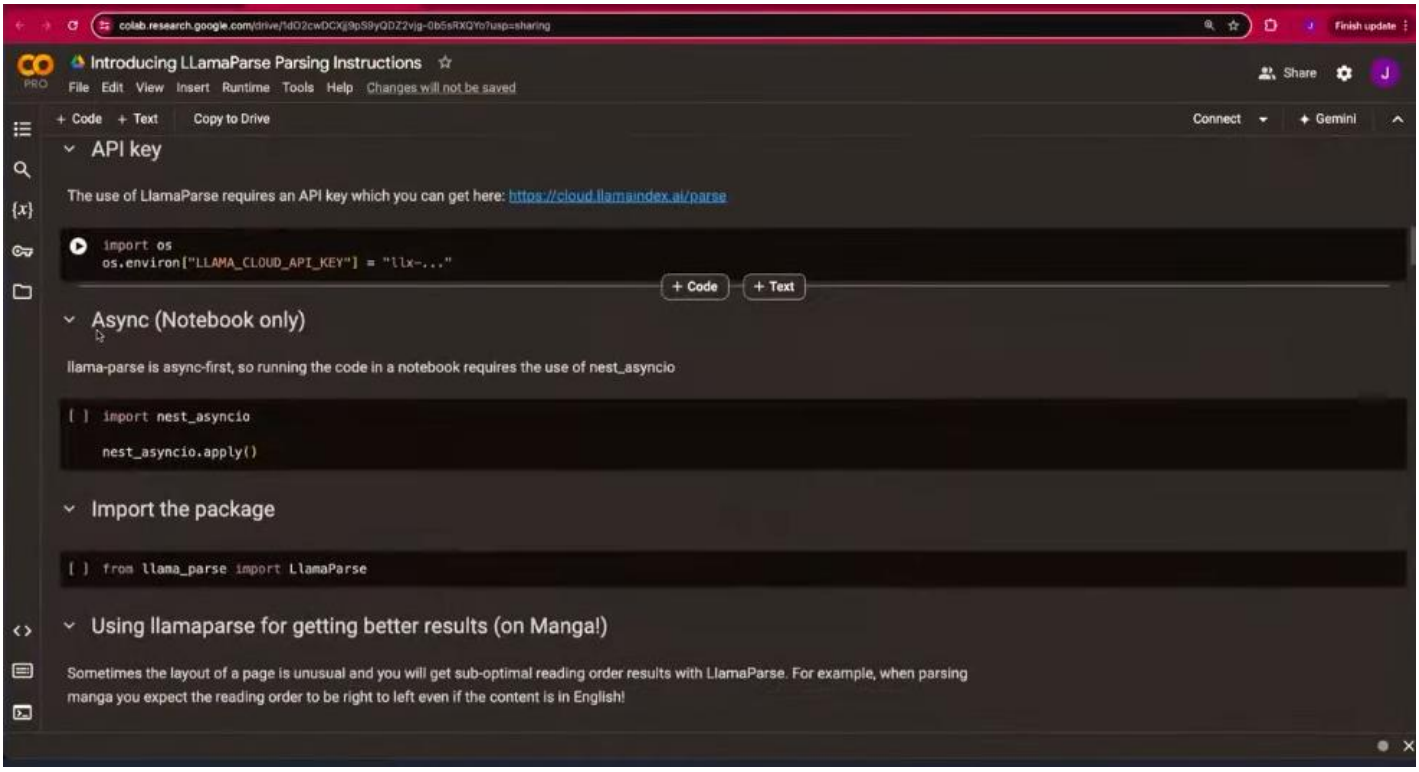
To parse this, we take the same instructions as before and add one sentence: `Output any math equation in LATEX markdown (between $$)` . The result of parsing is clear LaTeX instructions, which render the equations perfectly:

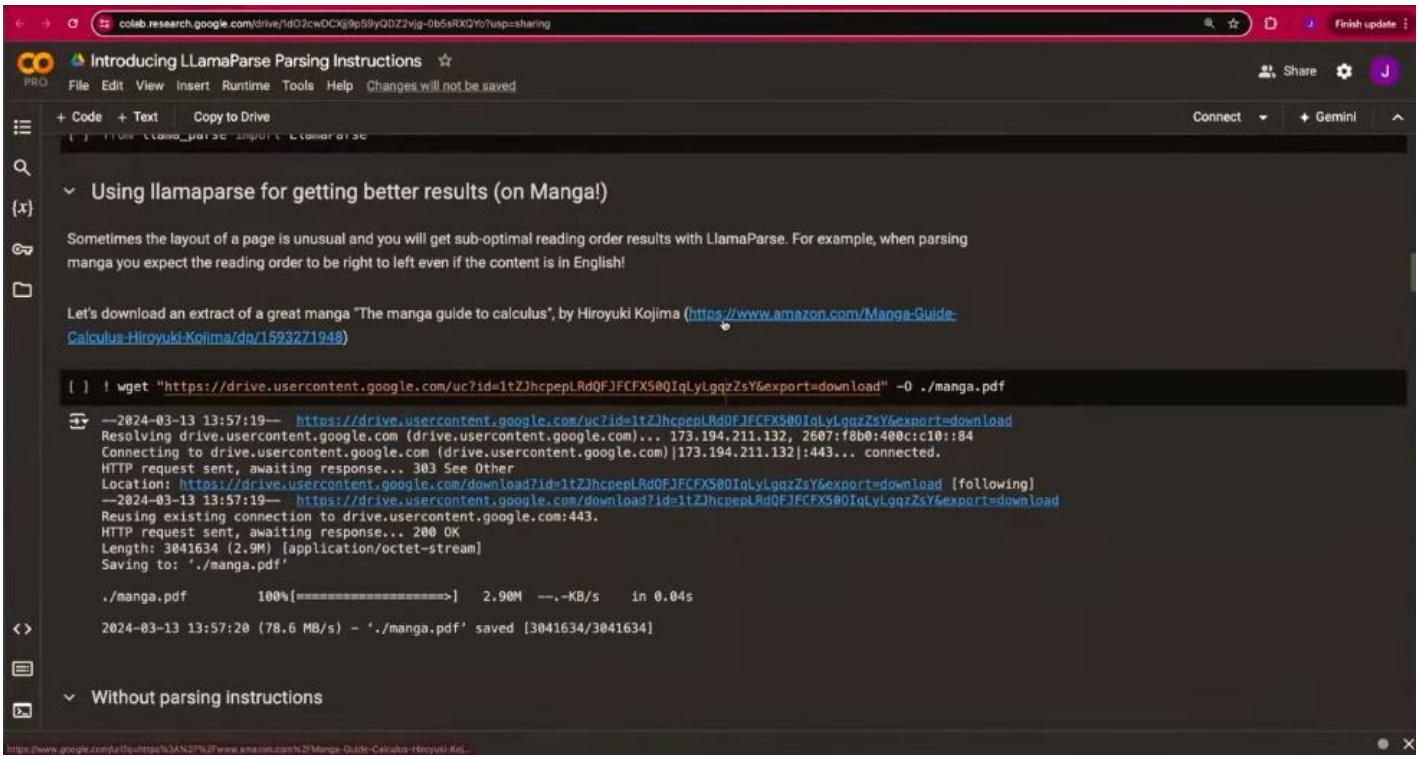
Calculating the Derivative of a Constant, Linear, or Quadratic Function

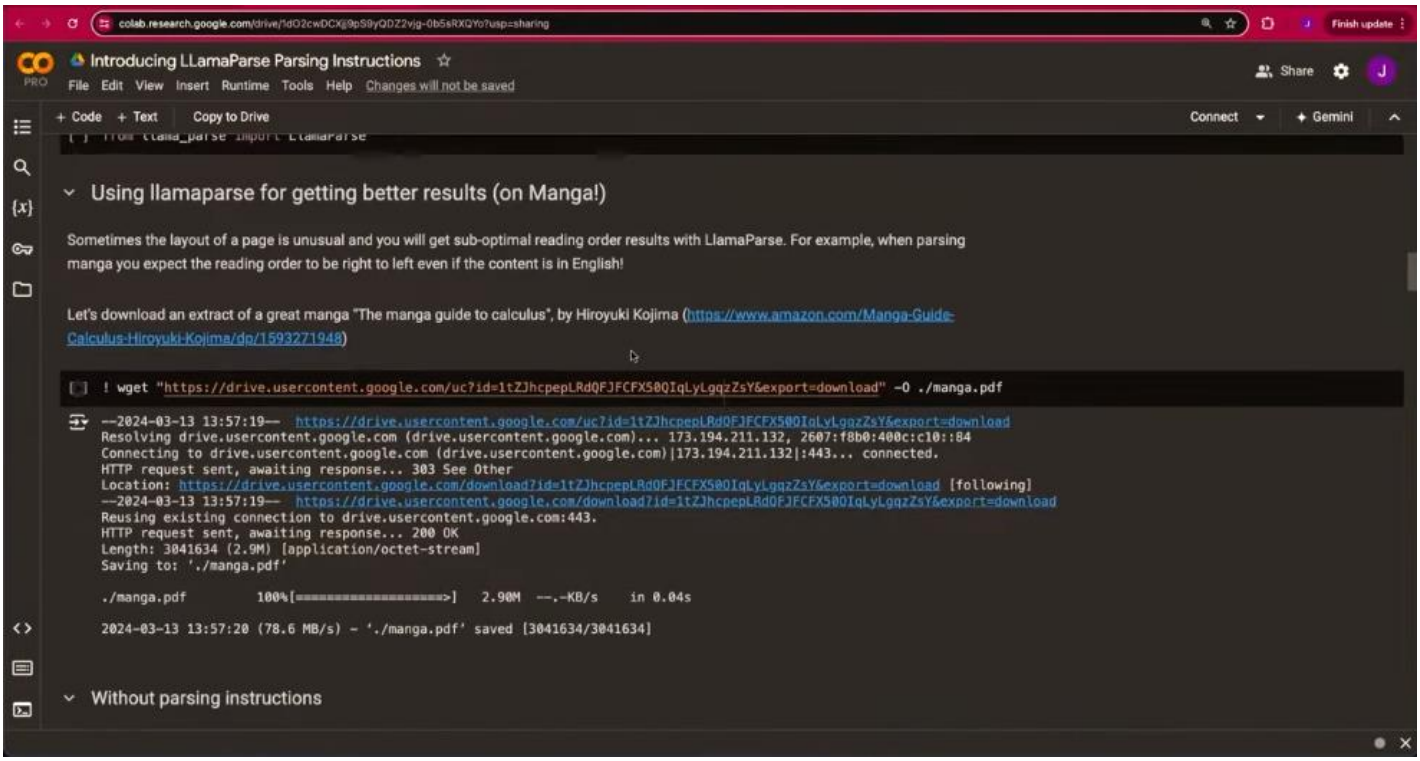
1. Let's find the derivative of constant function f(x) = α. The differential coefficient of f(x) at x = a is

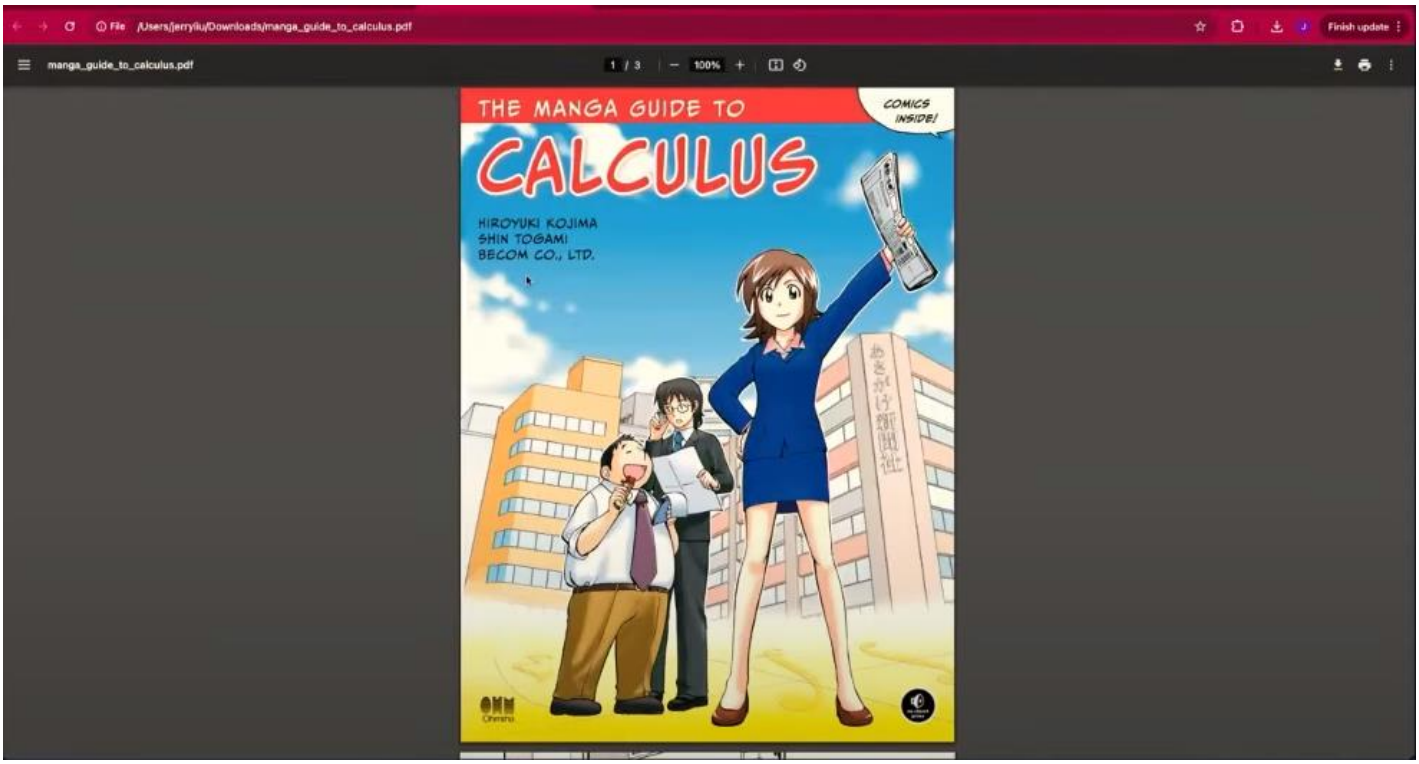
$$\lim_{\varepsilon \to 0} \left(\frac{f(a+\varepsilon) - f(a)}{\varepsilon} \right) = \lim_{\varepsilon \to 0} \left(\frac{\alpha - \alpha}{\varepsilon} \right) = \lim_{\varepsilon \to 0} 0 = 0$$ Thus, the derivative of f(x) is f'(x) = 0. This makes sense, since our function is constant—the rate of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

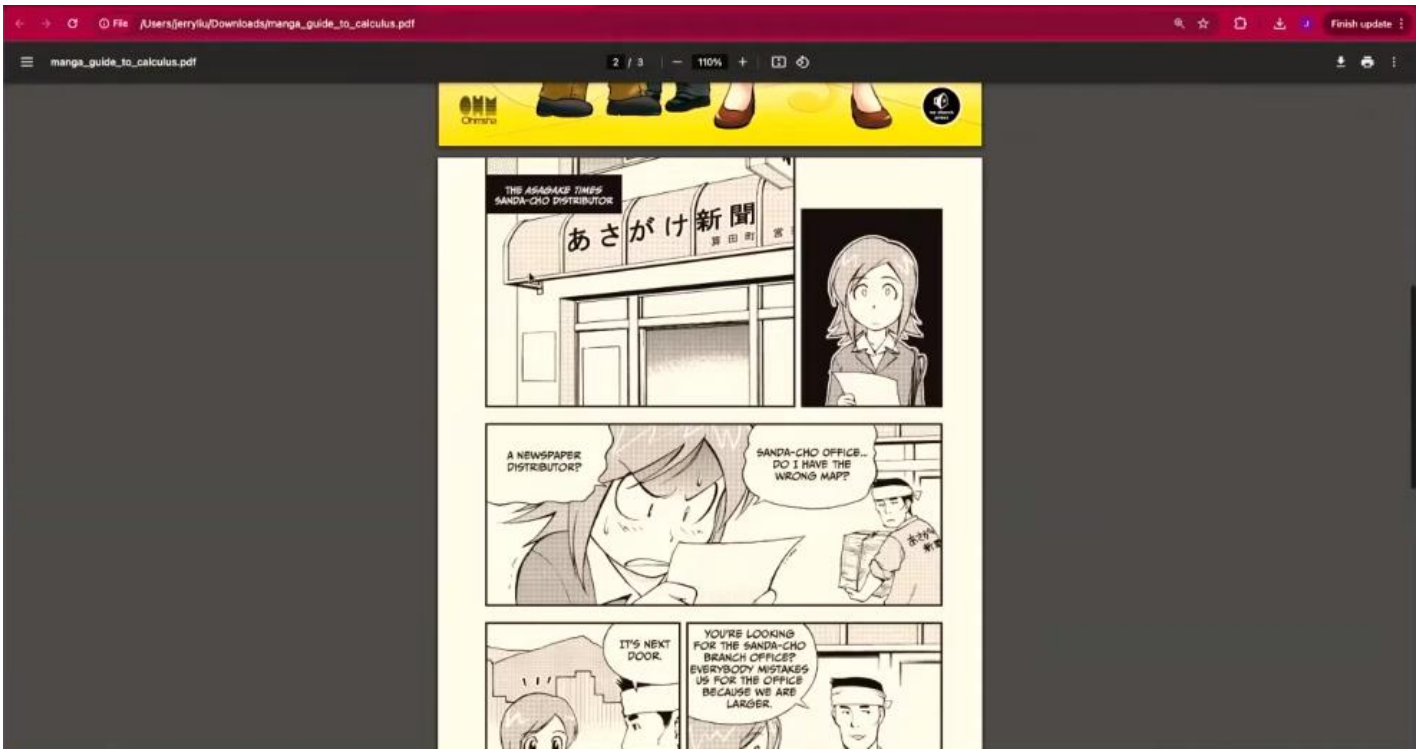

Introducing LLamaParse Parsing Instructions
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect + Gemini
LlamaParse - Parsing comic books with parsing intructions
Parsing intructions allow you to instruct our parsing model the same way you would instruct an LLM!
They can be usefull to help the parser get better results on complex document layouts, to extract data in a specific format, or to transform the document in other ways.
Using Parsing instruction you will get better results out of LlamaParse on complicated documents, and also be able to simplify your application code.
Installation
Parsing instructions are part of the llamaParse API. They can be accessed by directly specifying the parsing_instruction parameter in the API or by using the LlamaParse python module (which we will use for this tutorial).
To install llama-parse, just get it from PIP:
[ ] !pip install llama-parse
Collecting llama-parse
Downloading llama_parse-0.3.8-py3-none-any.whl (6.7 kB)
Collecting llama-index-core>=0.10.7 (from llama-parse)
Downloading llama_index_core-0.10.19-py3-none-any.whl (15.3 MB)
15.3/15.3 MB 31.9 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=6.0.1 in /usr/local/lib/python3.10/dist-packages (from llama-index-core>=0.10.7->llama-parse) (6.0.1)
Requirement already satisfied: SQLAlchemy[asyncio]>=1.4.49 in /usr/local/lib/python3.10/dist-packages (from llama-index-core>=0.10.7->llama-parse) (2.0.28)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.6 in /usr/local/lib/python3.10/dist-packages (from llama-index-core>=0.10.7->llama-parse) (3.9.3)
Collecting dataclasses-json (from llama-index-core>=0.10.7->llama-parse)


Introducing LLamaParse Parsing Instructions
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect + Gemini
API key
The use of LlamaParse requires an API key which you can get here: https://cloud.llamaindex.ai/parse
import os
os.environ["LLAMA_CLOUD_API_KEY"] = "llx-..."
+ Code + Text
Async (Notebook only)
llama-parse is async-first, so running the code in a notebook requires the use of nest_asyncio
[ ] import nest_asyncio
nest_asyncio.apply()
Import the package
[ ] from llama_parse import LlamaParse
Using llamaparse for getting better results (on Manga!)
Sometimes the layout of a page is unusual and you will get sub-optimal reading order results with LlamaParse. For example, when parsing manga you expect the reading order to be right to left even if the content is in English!


colab.research.google.com/drive/1dO2cwDCXj9pS9yQDZ2vjg-0b5sRXQYo?usp=sharing
Finish update
Introducing LLamaParse Parsing Instructions
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect
+ Gemini
Using llamaparse for getting better results (on Manga!)
Sometimes the layout of a page is unusual and you will get sub-optimal reading order results with LlamaParse. For example, when parsing manga you expect the reading order to be right to left even if the content is in English!
Let's download an extract of a great manga "The manga guide to calculus", by Hiroyuki Kojima (https://www.amazon.com/Manga-Guide-Calculus-Hiroyuki-Kojima/dp/1593271948)
[ ] ! wget "https://drive.usercontent.google.com/uc?id=1tZJhcpepLRdQFJFCFX50QIqLyLgqzZsY&export=download" -O ./manga.pdf
--2024-03-13 13:57:19-- https://drive.usercontent.google.com/uc?id=1tZJhcpepLRdQFJFCFX50QIqLyLgqzZsY&export=download
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 173.194.211.132, 2607:f8b0:400c:c10::84
Connecting to drive.usercontent.google.com (drive.usercontent.google.com)|173.194.211.132|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=1tZJhcpepLRdQFJFCFX50QIqLyLgqzZsY&export=download [following]
--2024-03-13 13:57:19-- https://drive.usercontent.google.com/download?id=1tZJhcpepLRdQFJFCFX50QIqLyLgqzZsY&export=download
Reusing existing connection to drive.usercontent.google.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 3041634 (2.9M) [application/octet-stream]
Saving to: './manga.pdf'
./manga.pdf 100%[==========>] 2.90M --.-KB/s in 0.04s
2024-03-13 13:57:20 (78.6 MB/s) - './manga.pdf' saved [3041634/3041634]
Without parsing instructions


colab.research.google.com/drive/1dO2cwDCXj9pS9yQDZ2vjg-0b5sRXQYo?usp=sharing
Finish update
Introducing LLamaParse Parsing Instructions
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect
+ Gemini
Using llamaparse for getting better results (on Manga!)
Sometimes the layout of a page is unusual and you will get sub-optimal reading order results with LlamaParse. For example, when parsing manga you expect the reading order to be right to left even if the content is in English!
Let's download an extract of a great manga "The manga guide to calculus", by Hiroyuki Kojima (https://www.amazon.com/Manga-Guide-Calculus-Hiroyuki-Kojima/dp/1593271948)
[ ] ! wget "https://drive.usercontent.google.com/uc?id=1tZJhcpepLRdQFJFCFX50QIqLyLgqzZsY&export=download" -O ./manga.pdf
--2024-03-13 13:57:19-- https://drive.usercontent.google.com/uc?id=1tZJhcpepLRdQFJFCFX50QIqLyLgqzZsY&export=download
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 173.194.211.132, 2607:f8b0:400c:c10::84
Connecting to drive.usercontent.google.com (drive.usercontent.google.com)|173.194.211.132|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=1tZJhcpepLRdQFJFCFX50QIqLyLgqzZsY&export=download [following]
--2024-03-13 13:57:19-- https://drive.usercontent.google.com/download?id=1tZJhcpepLRdQFJFCFX50QIqLyLgqzZsY&export=download
Reusing existing connection to drive.usercontent.google.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 3041634 (2.9M) [application/octet-stream]
Saving to: './manga.pdf'
./manga.pdf 100%[==========>] 2.90M --.-KB/s in 0.04s
2024-03-13 13:57:20 (78.6 MB/s) - './manga.pdf' saved [3041634/3041634]
Without parsing instructions


File /Users/jerryliu/Downloads/manga_guide_to_calculus.pdf
Finish update
manga_guide_to_calculus.pdf
1 / 3
100%
THE MANGA GUIDE TO
CALCULUS
COMICS INSIDE!
HIROYUKI KOJIMA
SHIN TOGAMI
BECOM CO., LTD.


File /Users/jerryliu/Downloads/manga_guide_to_calculus.pdf
Finish update
manga_guide_to_calculus.pdf
2 / 3
110%
THE ASAGAKE TIMES SANDA-CHO DISTRIBUTOR
あさがけ新聞
A NEWSPAPER DISTRIBUTOR?
SANDA-CHO OFFICE... DO I HAVE THE WRONG MAP?
IT'S NEXT DOOR.
YOU'RE LOOKING FOR THE SANDA-CHO BRANCH OFFICE? EVERYBODY MISTAKES US FOR THE OFFICE BECAUSE WE ARE LARGER.

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.

NOTE The *differential coefficient* of $f(x)$ at $x = a$ is often simply called the derivative of $f(x)$ at $x = a$, or just $f'(a)$.

2. Let's calculate the derivative of linear function $f(x) = \alpha x + \beta$. The derivative of $f(x)$ at $x = \alpha$ is

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

Thus, the derivative of $f(x)$ is $f'(x) = \alpha$, a constant value. This result should also be intuitive—linear functions have a constant rate of change by definition.

3. Let's find the derivative of $f(x) = x^2$, which appeared in the story. The differential coefficient of $f(x)$ at $x = a$ is

$$\lim_{\varepsilon \to 0} \frac{f(a+\varepsilon) - f(a)}{\varepsilon} = \lim_{\varepsilon \to 0} \frac{(a+\varepsilon)^2 - a^2}{\varepsilon} = \lim_{\varepsilon \to 0} \frac{2a\varepsilon + \varepsilon^2}{\varepsilon} = \lim_{\varepsilon \to 0} (2a + \varepsilon) = 2a$$

Thus, the differential coefficient of $f(x)$ at $x = a$ is $2a$, or $f'(a) = 2a$. Therefore, the derivative of $f(x)$ is $f'(x) = 2x$.

SUMMARY

- The calculation of a limit that appears in calculus is simply a formula calculating an error.
- A limit is used to obtain a derivative.
- The derivative is the slope of the tangent line at a given point.
- The derivative is nothing but the rate of change.

File /Users/jerryliu/Downloads/manga_guide_to_calculus.pdf
manga_guide_to_calculus.pdf
3 / 3
204%
Finish update

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

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

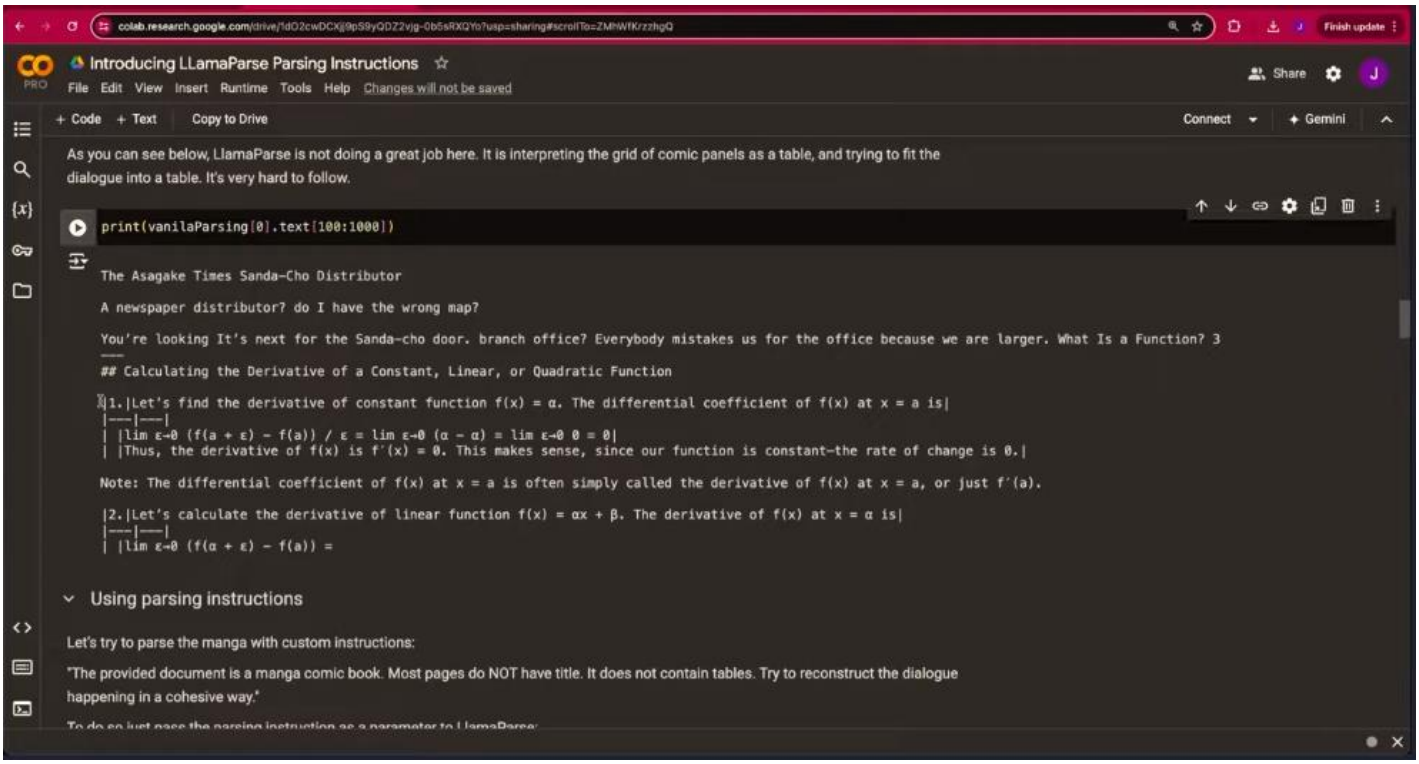
Thus, the derivative of $f(x)$ is $f'(x) = 0$. This makes sense, since our function is constant—the rate of change is 0.

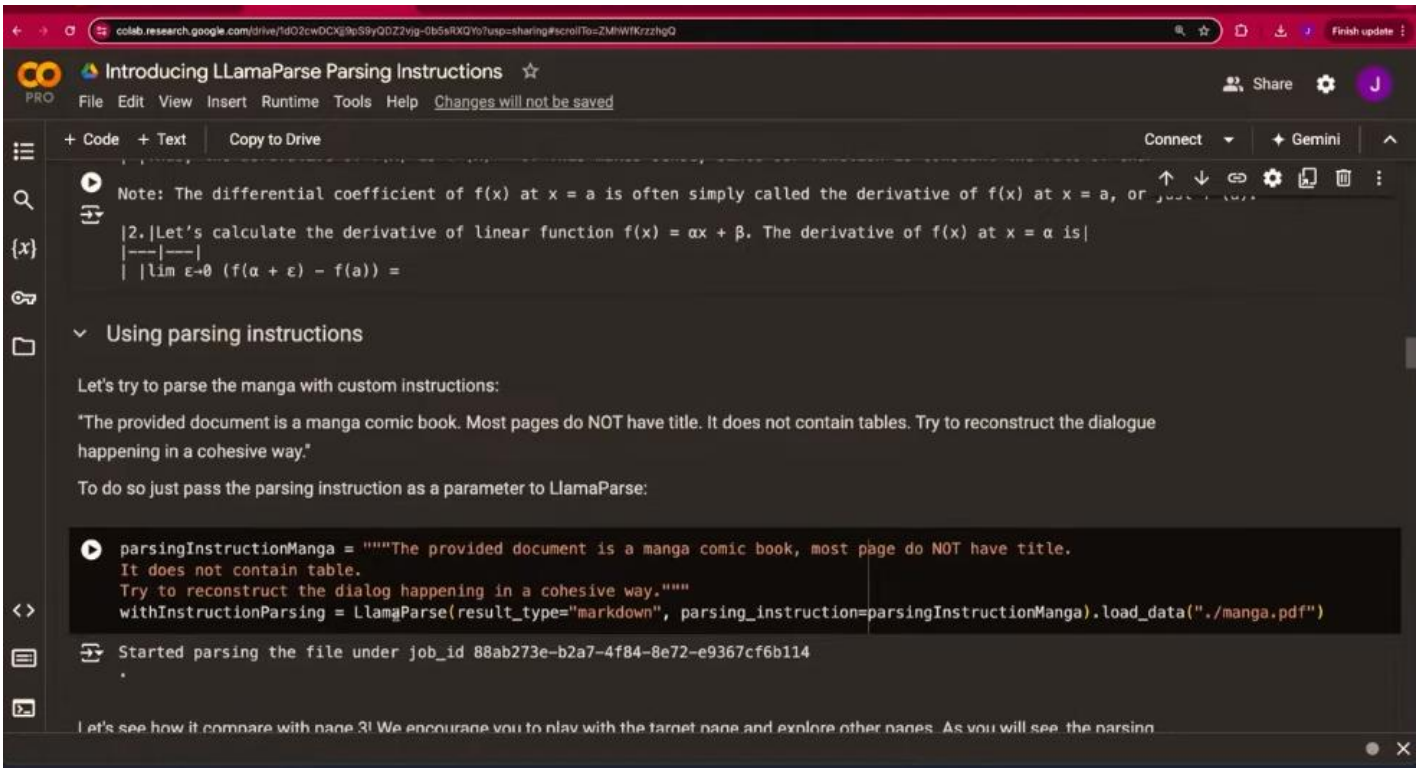
NOTE The *differential coefficient* of $f(x)$ at $x = a$ is often simply called the derivative of $f(x)$ at $x = a$, or just $f'(a)$.

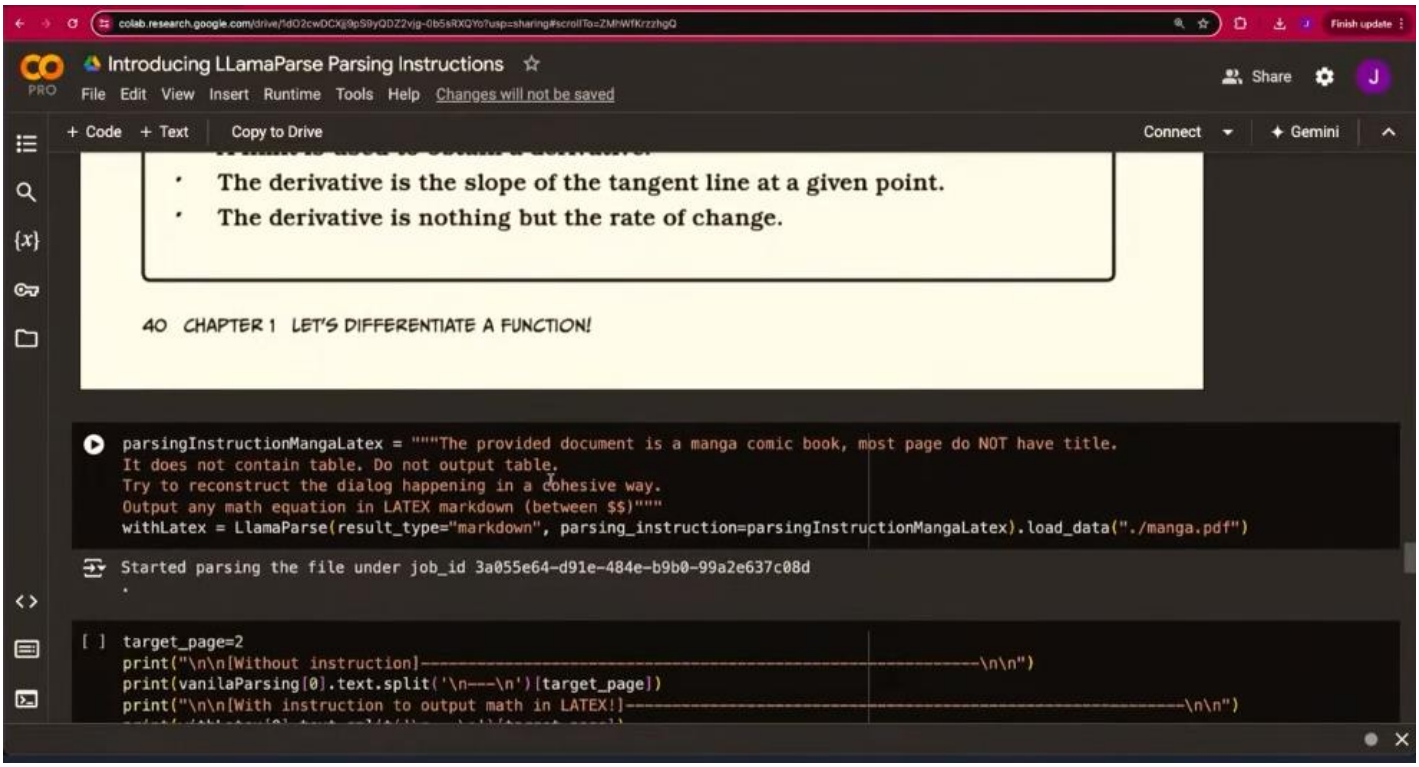
2. Let's calculate the derivative of linear function $f(x) = \alpha x + \beta$. The derivative of $f(x)$ at $x = \alpha$ is

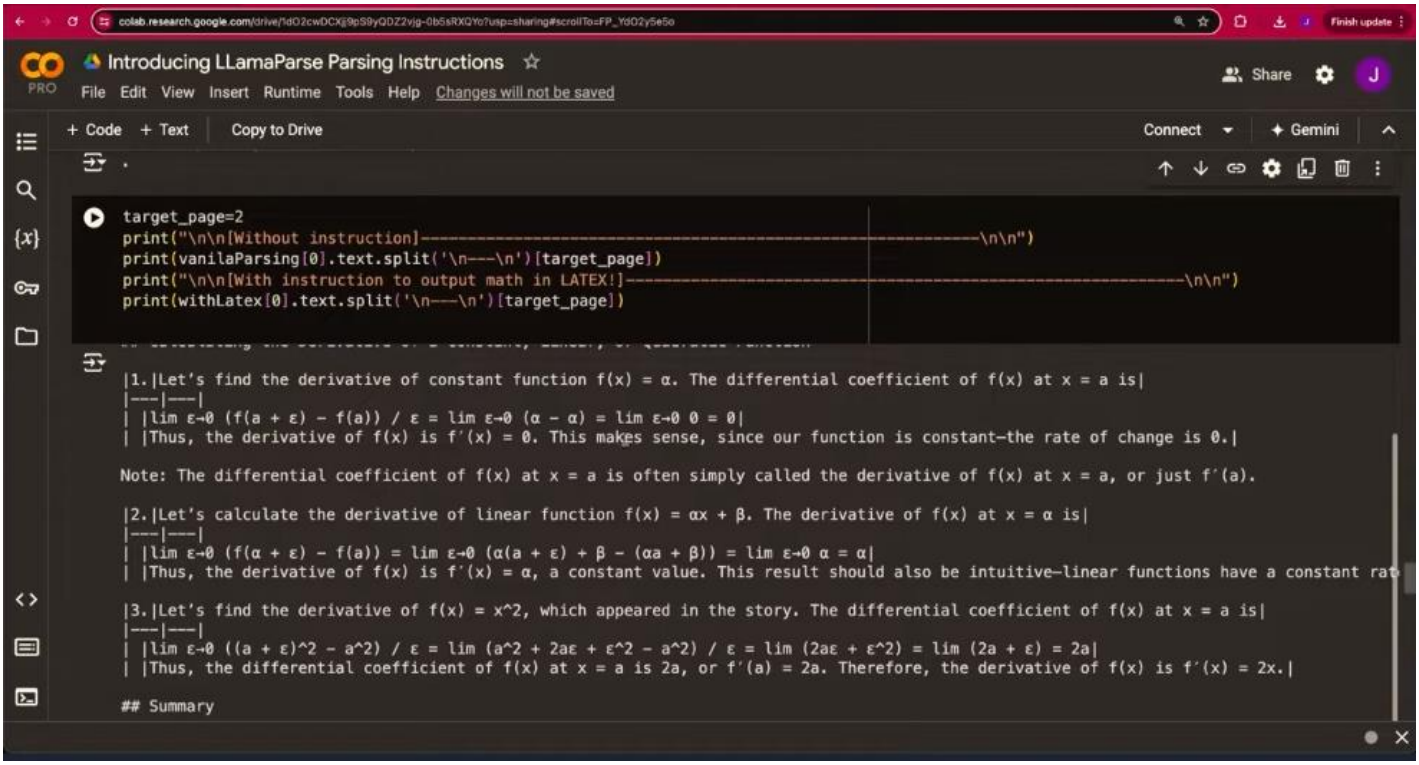
$$\lim_{\varepsilon \to 0} \frac{f(\alpha+\varepsilon) - f(\alpha)}{\varepsilon} = \lim_{\varepsilon \to 0} \frac{\alpha(a+\varepsilon) + \beta - (\alpha a + \beta)}{\varepsilon} = \lim_{\varepsilon \to 0} \alpha = \alpha$$

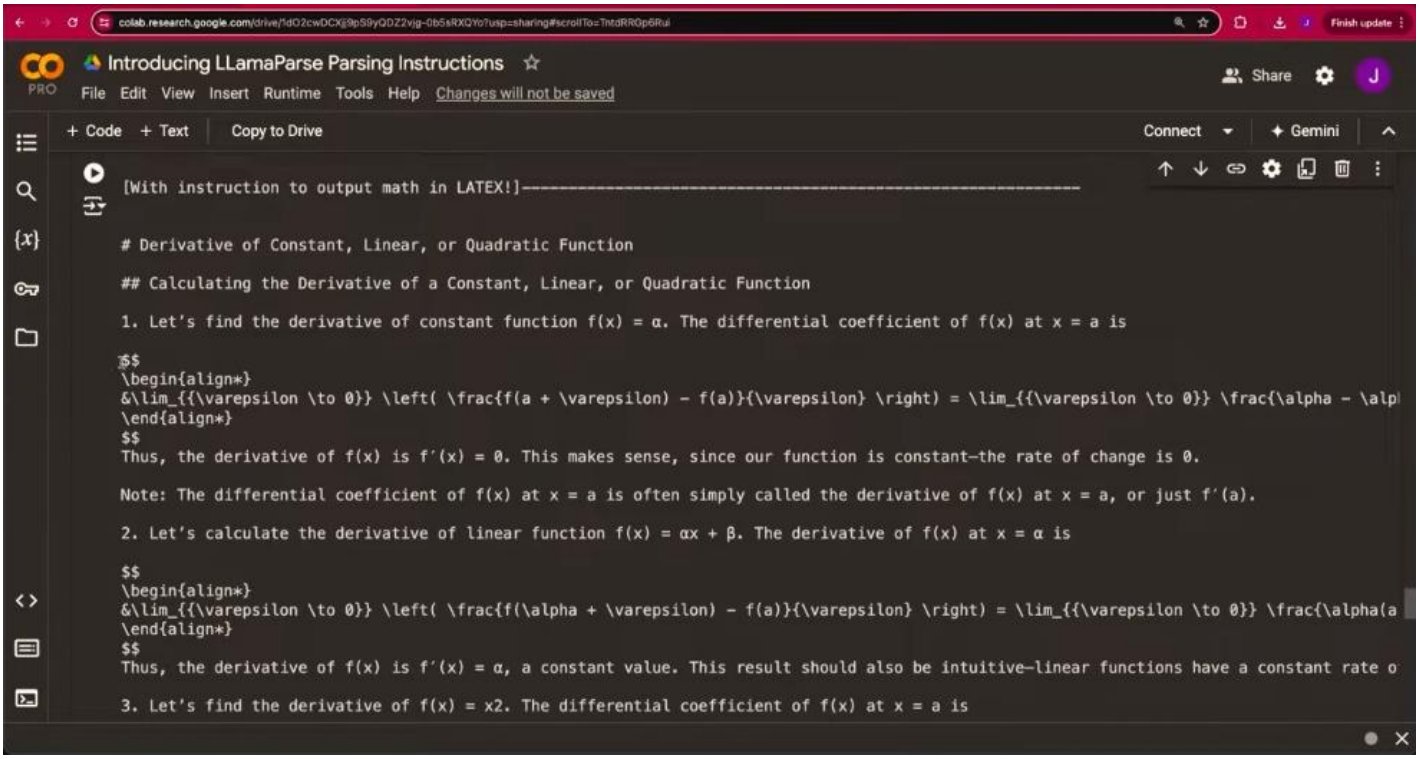
Thus, the derivative of $f(x)$ is $f'(x) = \alpha$, a constant value. This result should also be intuitive—linear functions have a constant rate of change


Introducing LLamaParse Parsing Instructions
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Gemini
As you can see below, LlamaParse is not doing a great job here. It is interpreting the grid of comic panels as a table, and trying to fit the dialogue into a table. It's very hard to follow.
print(vanilaParsing[0].text[100:1000])
The Asagake Times Sanda-Cho Distributor
A newspaper distributor? do I have the wrong map?
You're looking It's next for the Sanda-cho door. branch office? Everybody mistakes us for the office because we are larger. What Is a Function? 3
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 ε→0 (f(a + ε) − f(a)) / ε = lim ε→0 (α − α) = lim ε→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.
Note: The differential coefficient of f(x) at x = a is often simply called the derivative of f(x) at x = a, or just f'(a).
2.	Let's calculate the derivative of linear function f(x) = αx + β. The derivative of f(x) at x = α is
	lim ε→0 (f(α + ε) − f(a)) =
Using parsing instructions
Let's try to parse the manga with custom instructions:
"The provided document is a manga comic book. Most pages do NOT have title. It does not contain tables. Try to reconstruct the dialogue happening in a cohesive way."

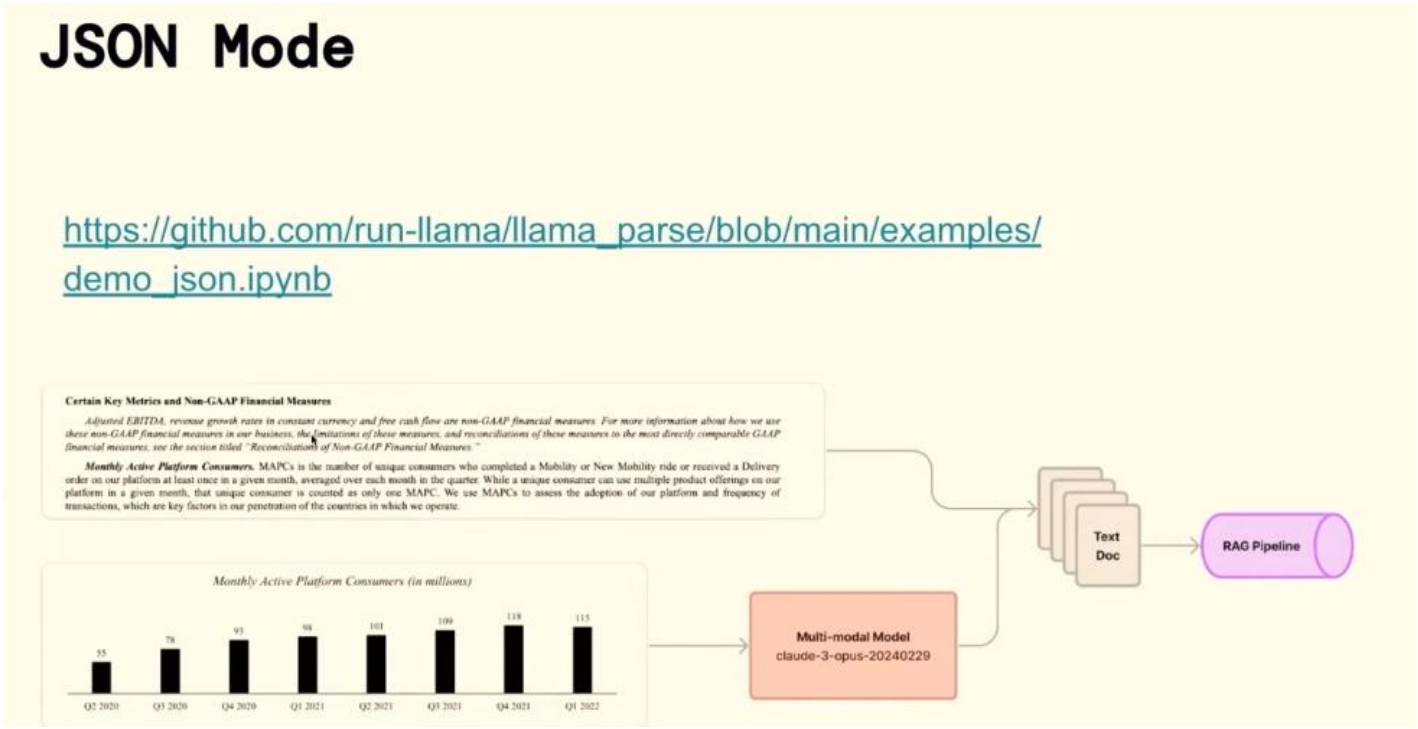

Introducing LLamaParse Parsing Instructions
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Gemini
Note: The differential coefficient of f(x) at x = a is often simply called the derivative of f(x) at x = a, or
2.	Let's calculate the derivative of linear function f(x) = αx + β. The derivative of f(x) at x = α is
	lim ε→0 (f(α + ε) − f(a)) =
Using parsing instructions
Let's try to parse the manga with custom instructions:
"The provided document is a manga comic book. Most pages do NOT have title. It does not contain tables. Try to reconstruct the dialogue happening in a cohesive way."
To do so just pass the parsing instruction as a parameter to LlamaParse:
parsingInstructionManga = """The provided document is a manga comic book, most page do NOT have title.
It does not contain table.
Try to reconstruct the dialog happening in a cohesive way."""
withInstructionParsing = LlamaParse(result_type="markdown", parsing_instruction=parsingInstructionManga).load_data("./manga.pdf")
Started parsing the file under job_id 88ab273e-b2a7-4f84-8e72-e9367cf6b114
Let's see how it compare with page 3! We encourage you to play with the target page and explore other pages. As you will see, the parsing


Introducing LLamaParse Parsing Instructions
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Gemini
The derivative is the slope of the tangent line at a given point.
The derivative is nothing but the rate of change.
40 CHAPTER 1 LET'S DIFFERENTIATE A FUNCTION!
parsingInstructionMangaLatex = """The provided document is a manga comic book, most page do NOT have title.
It does not contain table. Do not output table.
Try to reconstruct the dialog happening in a cohesive way.
Output any math equation in LATEX markdown (between $$)"""
withLatex = LlamaParse(result_type="markdown", parsing_instruction=parsingInstructionMangaLatex).load_data("./manga.pdf")
Started parsing the file under job_id 3a055e64-d91e-484e-b9b0-99a2e637c08d
target_page=2
print("\n\n[Without instruction]----------\n\n")
print(vanilaParsing[0].text.split('\n---\n')[target_page])
print("\n\n[With instruction to output math in LATEX!]----------\n\n")


Introducing LLamaParse Parsing Instructions
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Gemini
target_page=2
print("\n\n[Without instruction]----------\n\n")
print(vanilaParsing[0].text.split('\n---\n')[target_page])
print("\n\n[With instruction to output math in LATEX!]----------\n\n")
print(withLatex[0].text.split('\n---\n')[target_page])
1.	Let's find the derivative of constant function f(x) = α. The differential coefficient of f(x) at x = a is
	lim ε→0 (f(a + ε) − f(a)) / ε = lim ε→0 (α − α) = lim ε→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.
Note: The differential coefficient of f(x) at x = a is often simply called the derivative of f(x) at x = a, or just f′(a).
2.	Let's calculate the derivative of linear function f(x) = αx + β. The derivative of f(x) at x = α is
	lim ε→0 (f(α + ε) − f(a)) = lim ε→0 (α(a + ε) + β − (αa + β)) = lim ε→0 α = α
	Thus, the derivative of f(x) is f′(x) = α, a constant value. This result should also be intuitive—linear functions have a constant rat
3.	Let's find the derivative of f(x) = x^2, which appeared in the story. The differential coefficient of f(x) at x = a is
	lim ε→0 ((a + ε)^2 − a^2) / ε = lim (a^2 + 2aε + ε^2 − a^2) / ε = lim (2aε + ε^2) = lim (2a + ε) = 2a
	Thus, the differential coefficient of f(x) at x = a is 2a, or f′(a) = 2a. Therefore, the derivative of f(x) is f′(x) = 2x.
Summary



Using a prompt, we now get back the equations in Latex as above



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

https://colab.research.google.com/drive/1iB3HPOZPpY4qMFpw1Dtr1rJDb44k1TL2?usp=sharing

Improving Query Complexity

Complex Questions

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

Examples:

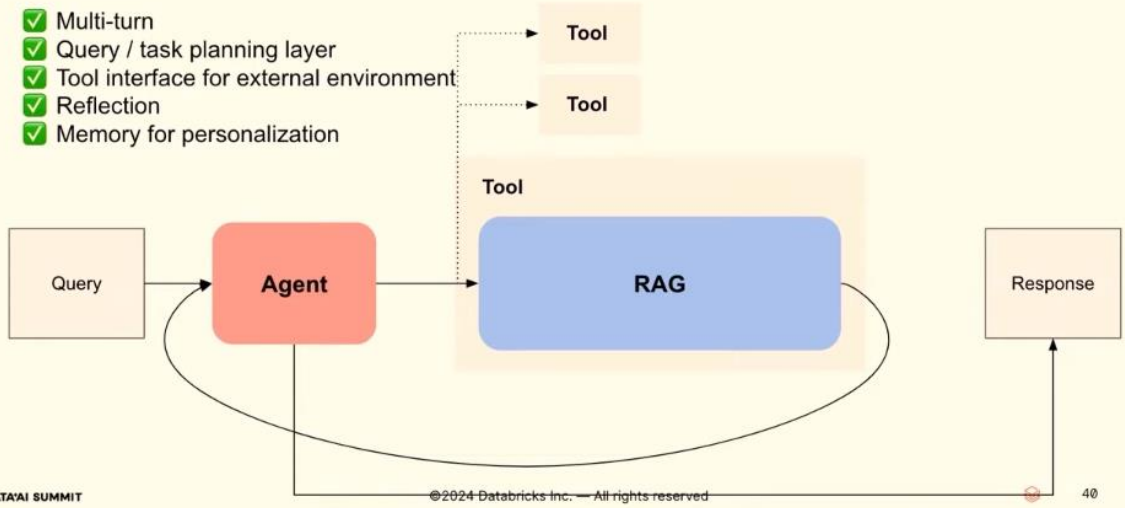
- **Summarization Questions:** "Give me a summary of the entire <company> 10K annual report"
- **Comparison Questions:** "Compare the open-source contributions of candidate A and candidate B"
- **Structured Analytics + Semantic Search:** "Tell me about the risk factors of the highest-performing rideshare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

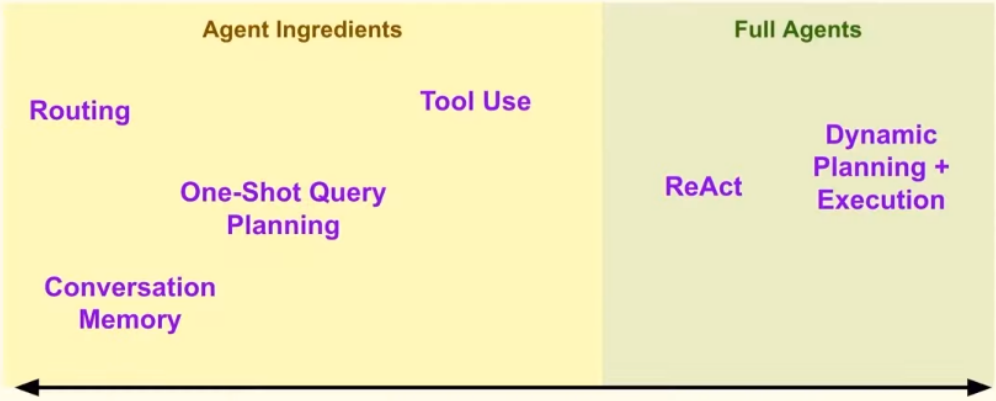
⚠️ Single-shot
⚠️ No query understanding/planning
⚠️ No tool use
⚠️ No reflection, error correction
⚠️ No memory (stateless)

From RAG to Agents

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



From Simple to Advanced Agents



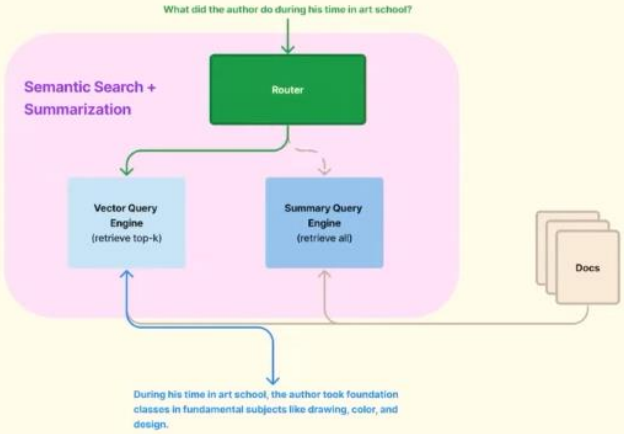
Simple
Lower Cost
Lower Latency

Advanced
Higher Cost
Higher Latency

Routing

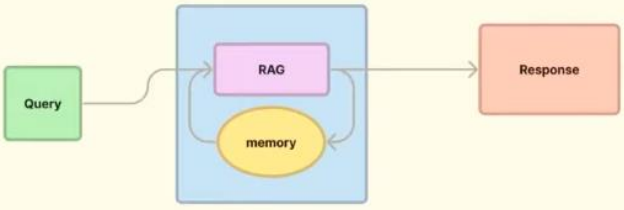
Simplest form of agentic reasoning.

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



Conversation Memory

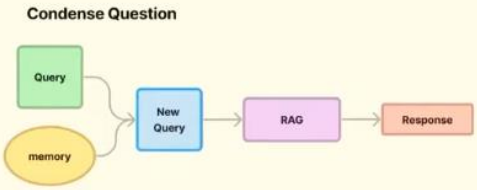
In addition to current query, take into account **conversation history as input to your RAG pipeline.**

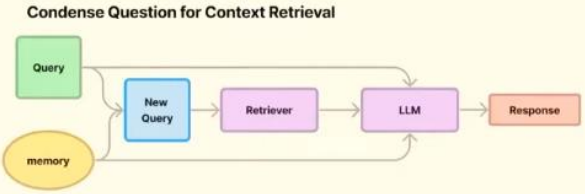


Conversation Memory

How to account for conversation history in a RAG pipeline?

- Condense question
- Condense question + context

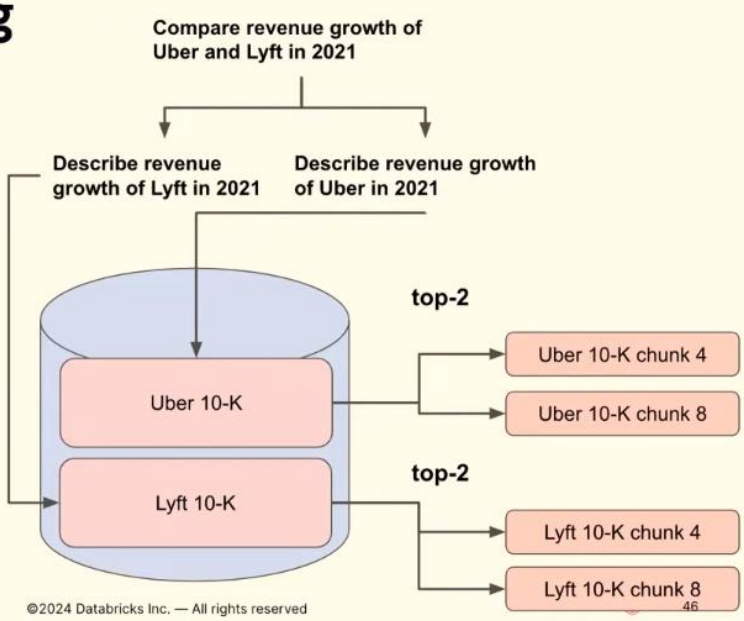




Query Planning

Break down query into parallelizable sub-queries.

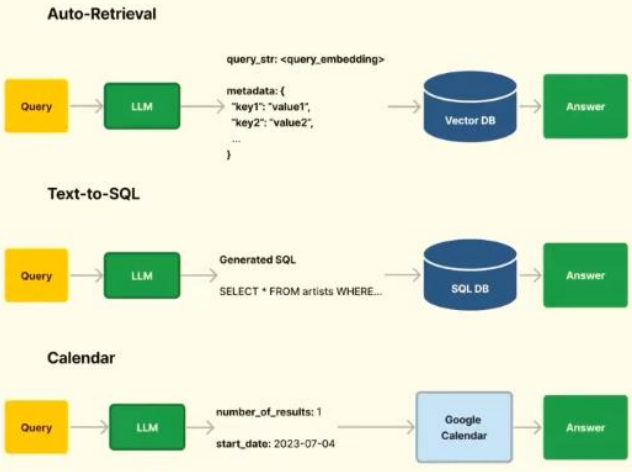
Each sub-query can be executed against any set of RAG pipelines



Tool Use

Use an LLM to call an API

Infer the parameters of that API

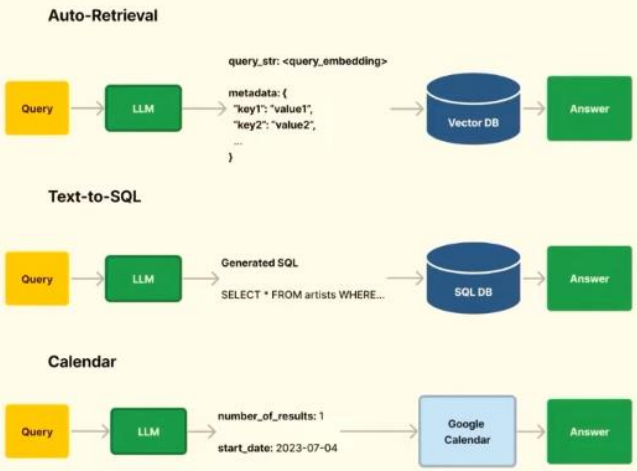


Tool Use

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?

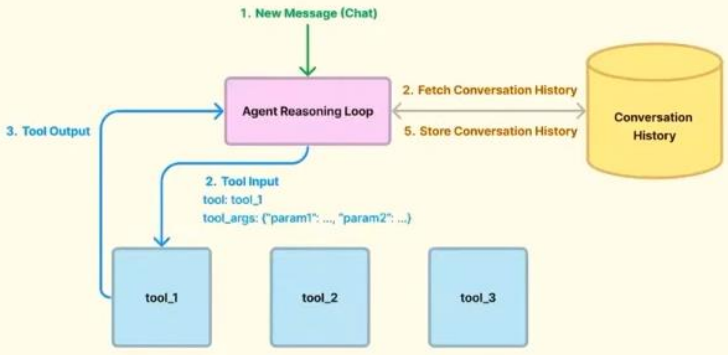
A key capability in many QA use cases (auto-retrieval, text-to-SQL, and more)



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)

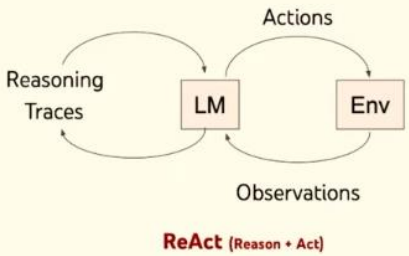
DAG-based planning (deterministic): Generate a deterministic DAG of steps. Replan if steps don't reach desired state.

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.

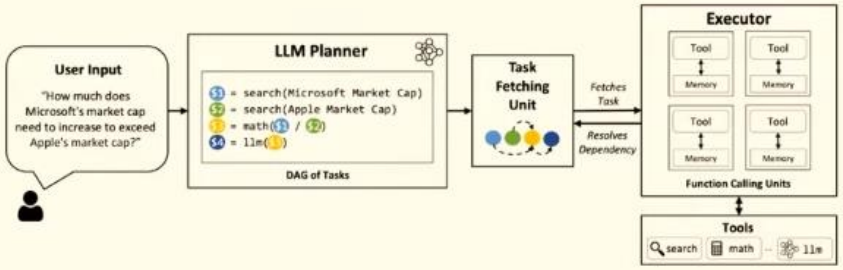


ReAct + RAG Guide

Function Calling Anthropic Agent

Agent Reasoning: DAG-based Planning

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



LLMCompiler Agent

Structured Planner Agent

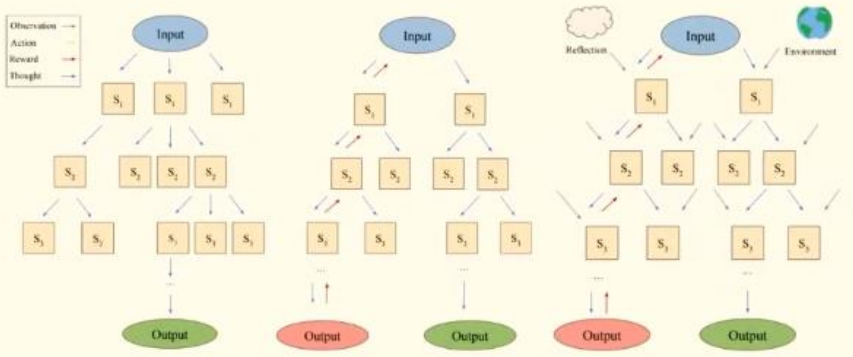
Agent Reasoning: Tree-based Planning

Tree of Thoughts (Yao et al. 2023)

Reasoning via Planning (Hao et al. 2023)

Language Agent Tree Search (Zhou et al. 2023)

LATS Guide



a) Tree-of-Thoughts b) Reasoning via Planning c) Language Agent Tree Search

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.

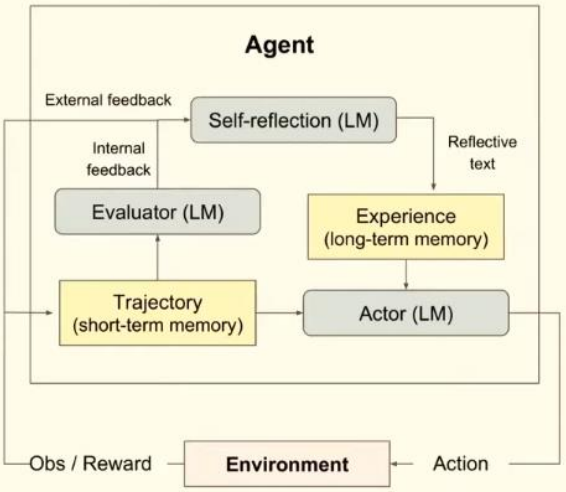
Self-Reflection

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

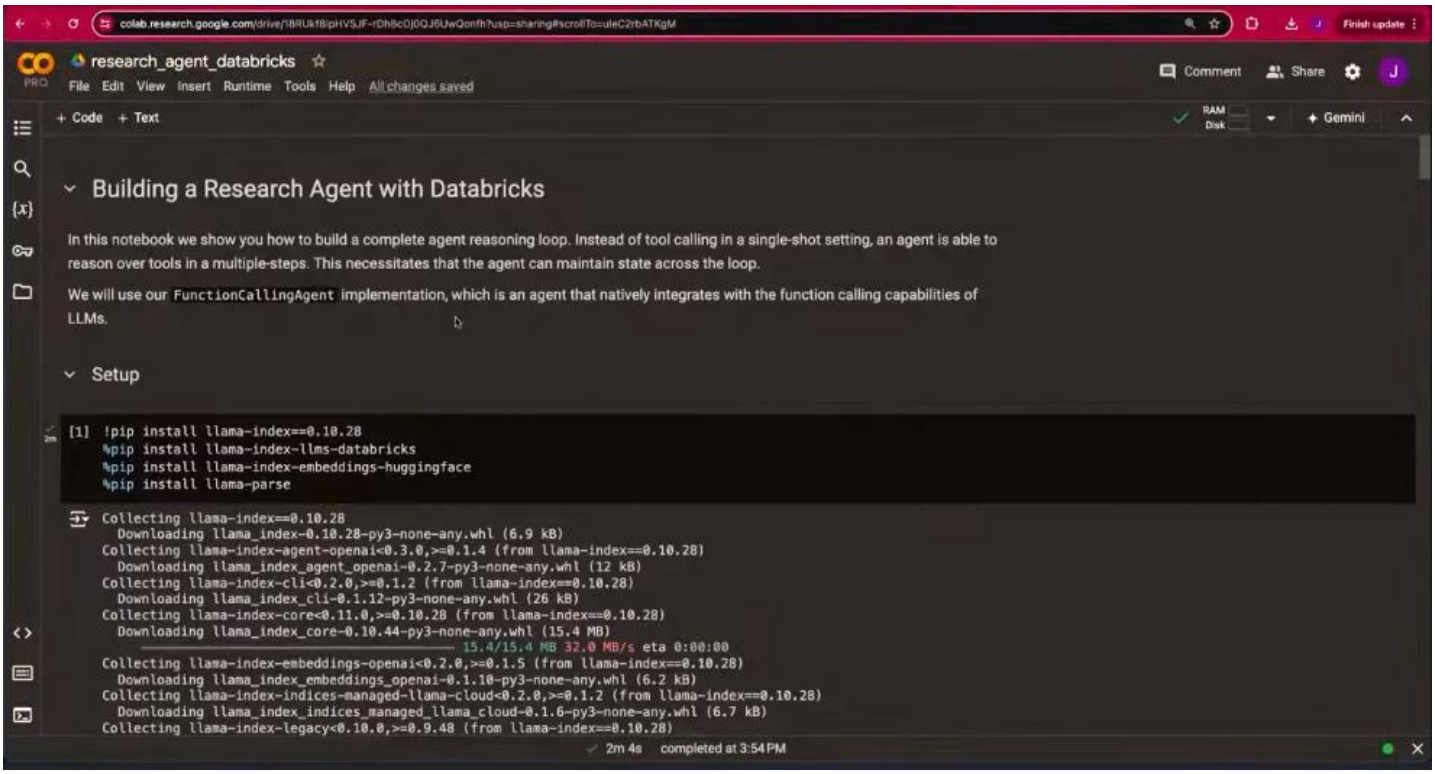
LlamaCloud for Advanced RAG

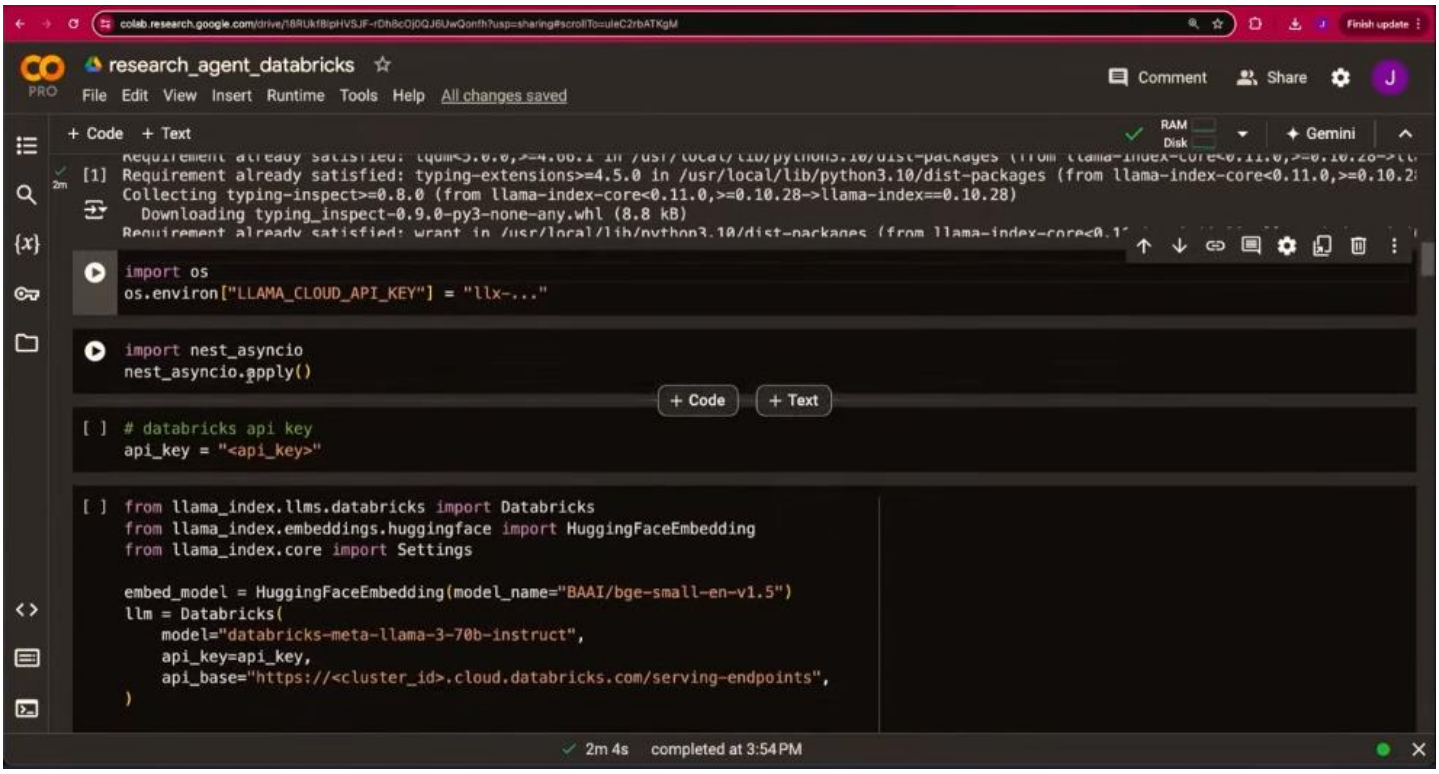
Building RAG/agents in an enterprise setting? We'd love to chat!

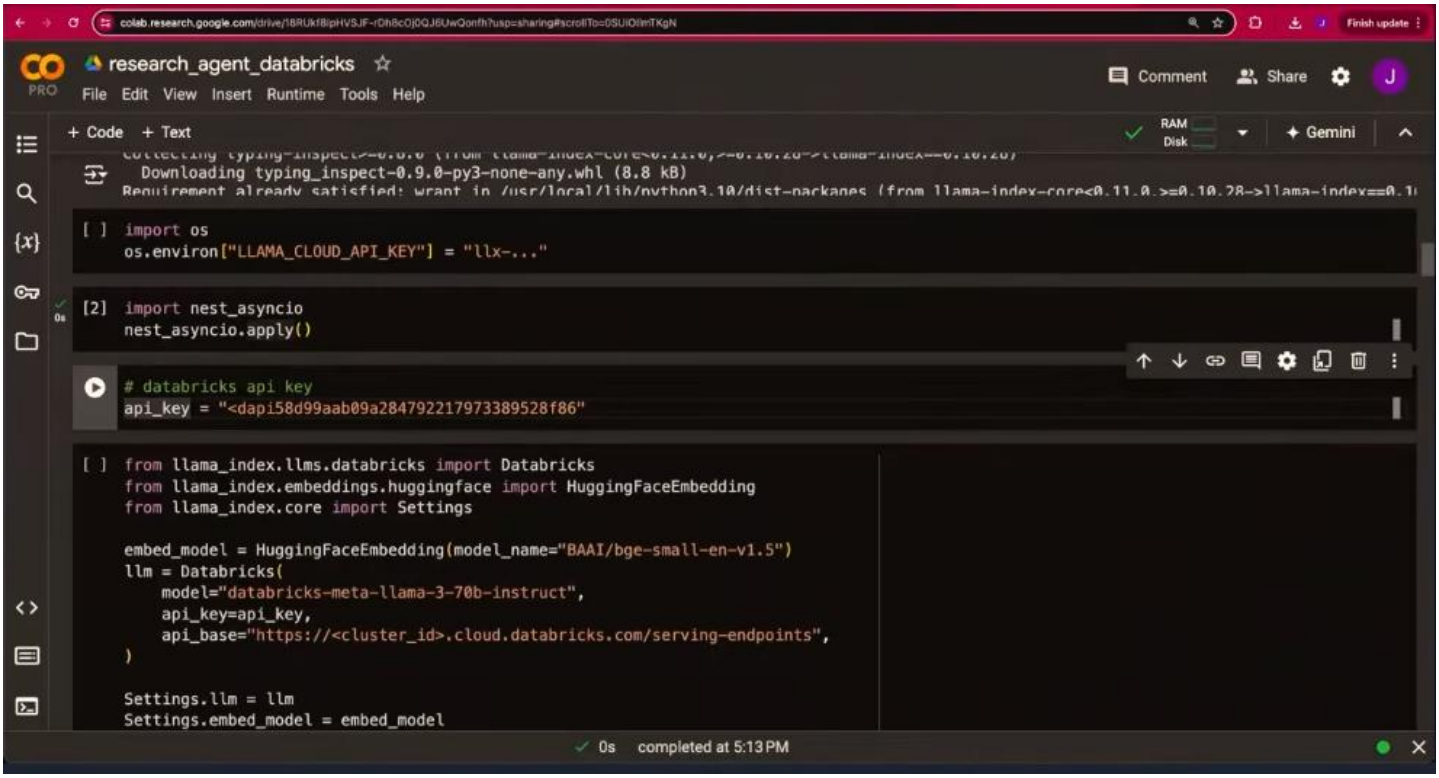
Workshop

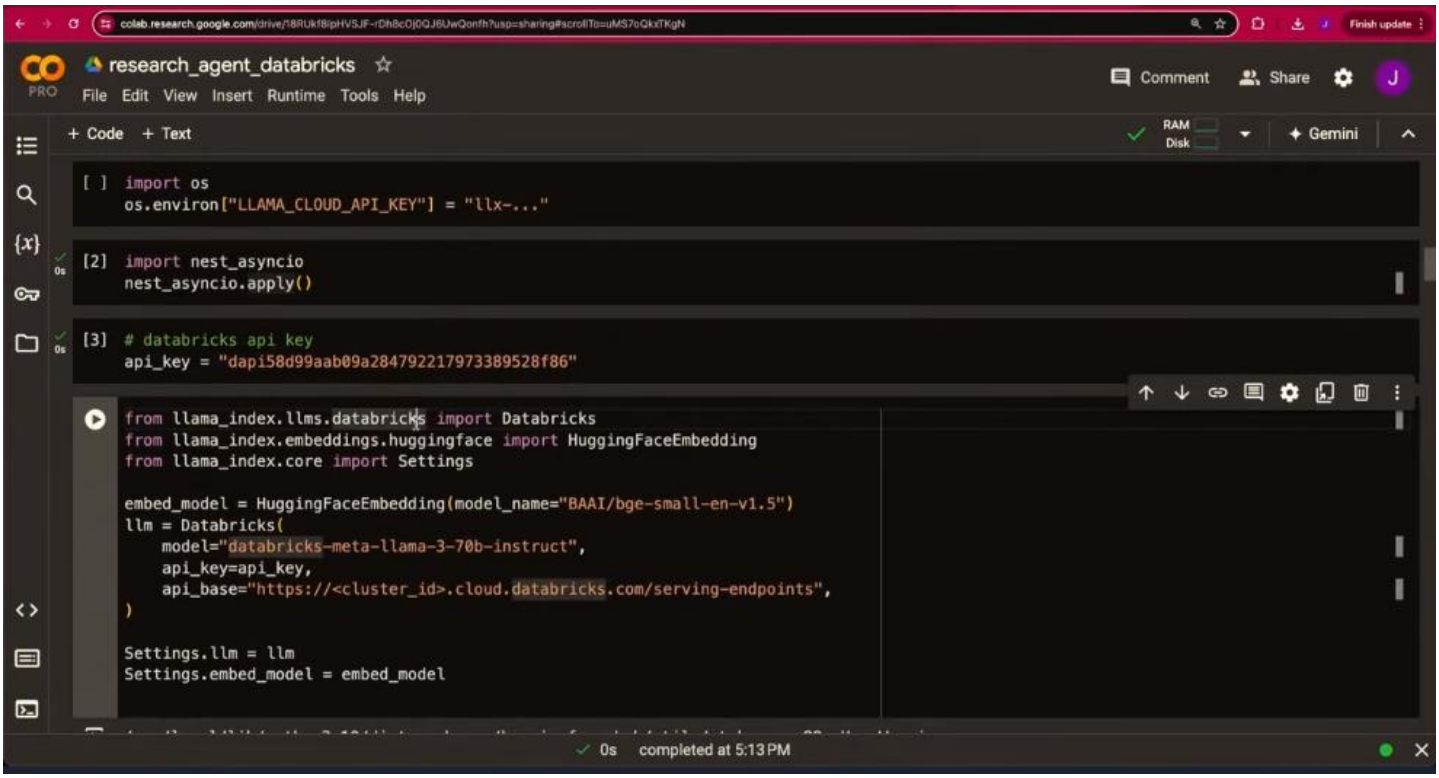
Let's extend our RAG pipeline into an agent!

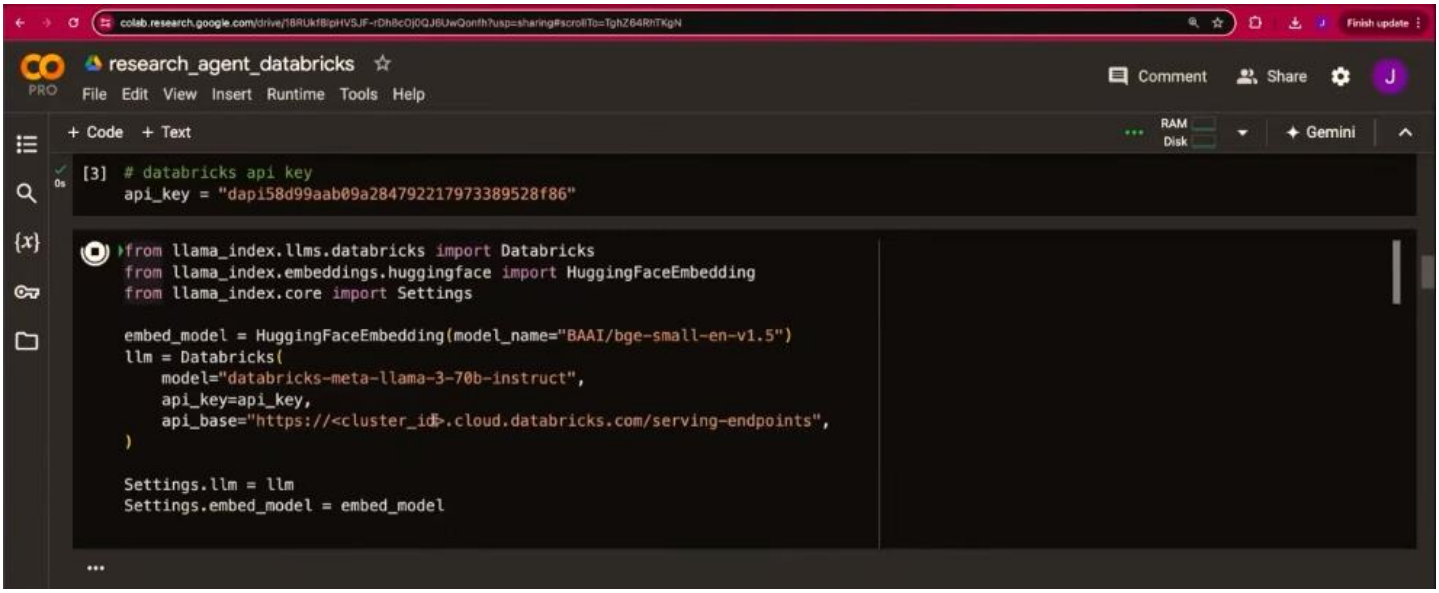
https://colab.research.google.com/drive/18RUkf8IpHVSJF-rDh8cOj0QJ6UwQonfh?usp=sharing

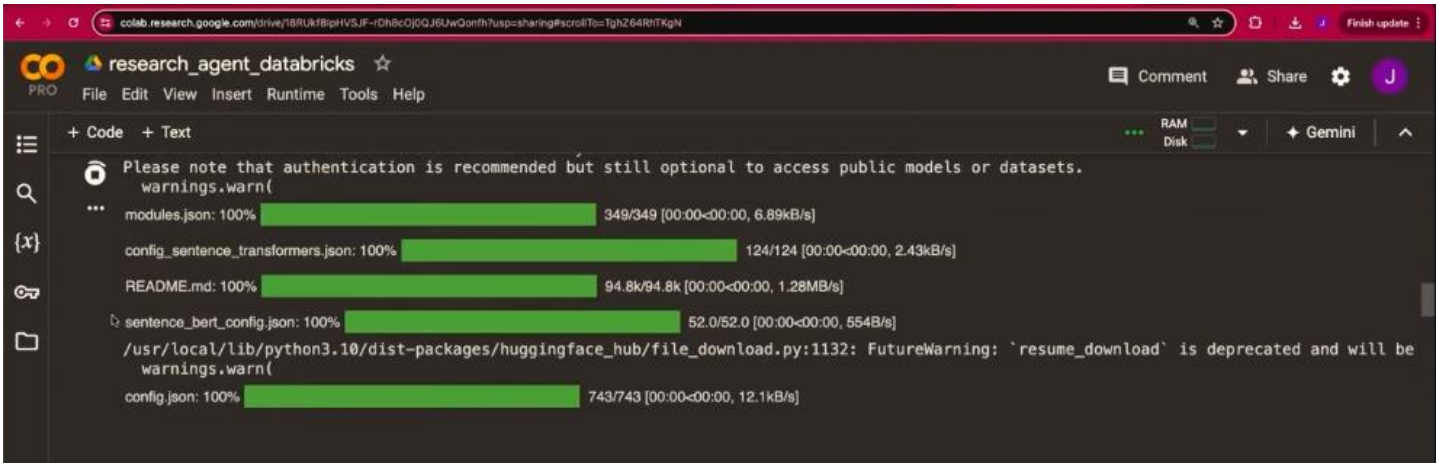


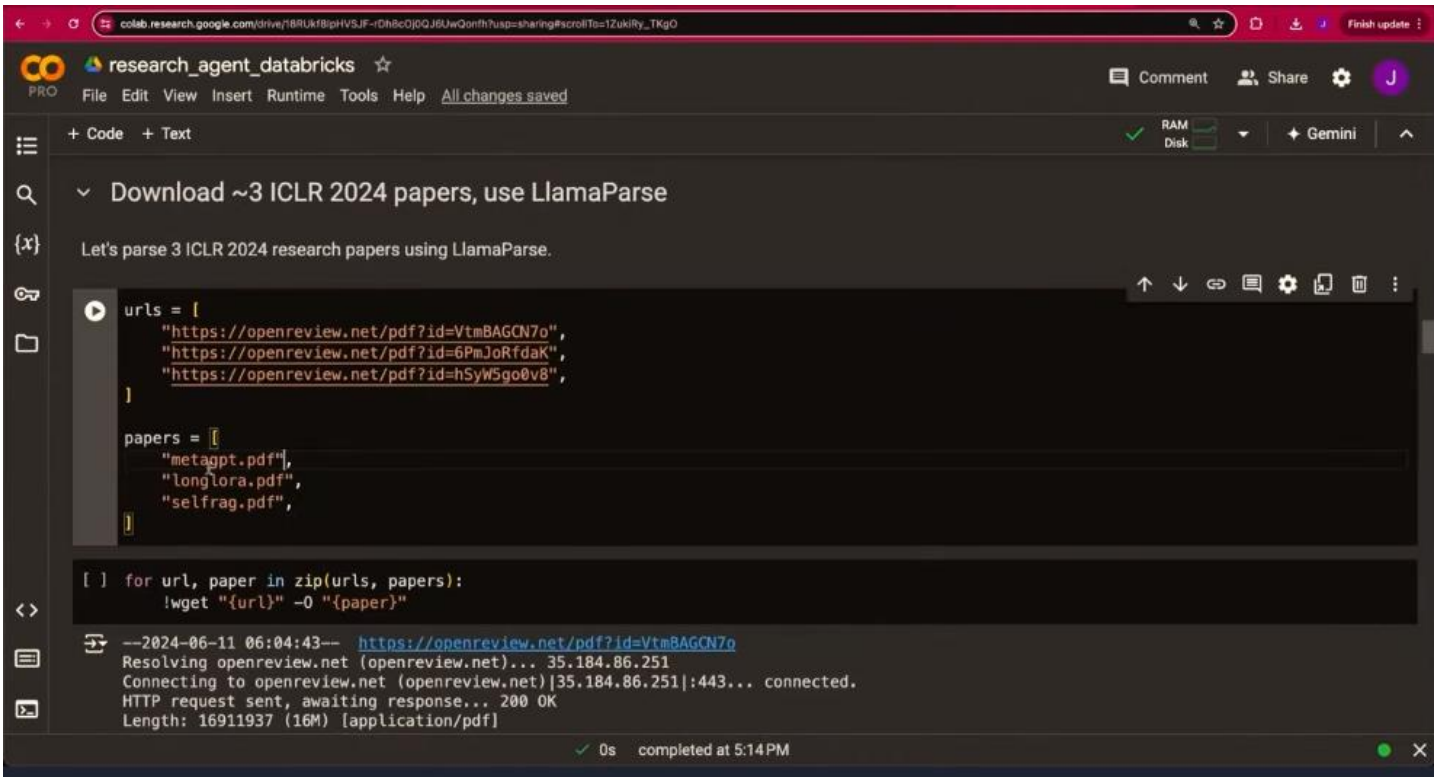

research_agent_databricks
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Gemini
[1] Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.10/dist-packages (from llama-index-core<0.11.0,>=0.10.2
Collecting typing-inspect>=0.8.0 (from llama-index-core<0.11.0,>=0.10.28->llama-index==0.10.28)
Downloading typing_inspect-0.9.0-py3-none-any.whl (8.8 kB)
import os
os.environ["LLAMA_CLOUD_API_KEY"] = "llx-..."
import nest_asyncio
nest_asyncio.apply()
+ Code
+ Text
# databricks api key
api_key = "<api_key>"
from llama_index.llms.databricks import Databricks
from llama_index.embeddings.huggingface import HuggingFaceEmbedding
from llama_index.core import Settings
embed_model = HuggingFaceEmbedding(model_name="BAAI/bge-small-en-v1.5")
llm = Databricks(
model="databricks-meta-llama-3-70b-instruct",
api_key=api_key,
api_base="https://<cluster_id>.cloud.databricks.com/serving-endpoints",
)
2m 4s completed at 3:54 PM

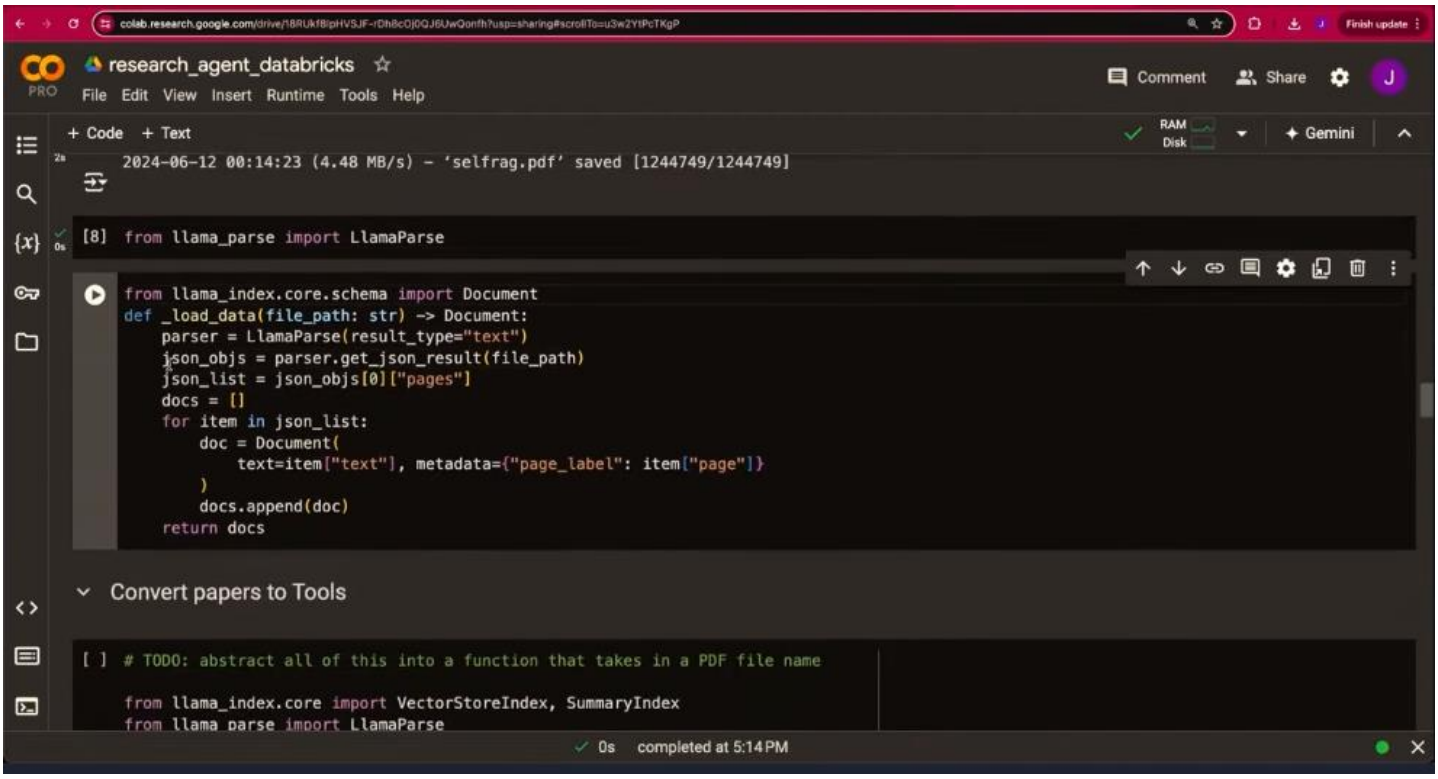

research_agent_databricks
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Gemini
Downloading typing_inspect-0.9.0-py3-none-any.whl (8.8 kB)
import os
os.environ["LLAMA_CLOUD_API_KEY"] = "llx-..."
[2] import nest_asyncio
nest_asyncio.apply()
# databricks api key
api_key = "<dapi58d99aab09a284792217973389528f86"
from llama_index.llms.databricks import Databricks
from llama_index.embeddings.huggingface import HuggingFaceEmbedding
from llama_index.core import Settings
embed_model = HuggingFaceEmbedding(model_name="BAAI/bge-small-en-v1.5")
llm = Databricks(
model="databricks-meta-llama-3-70b-instruct",
api_key=api_key,
api_base="https://<cluster_id>.cloud.databricks.com/serving-endpoints",
)
Settings.llm = llm
Settings.embed_model = embed_model
0s completed at 5:13 PM

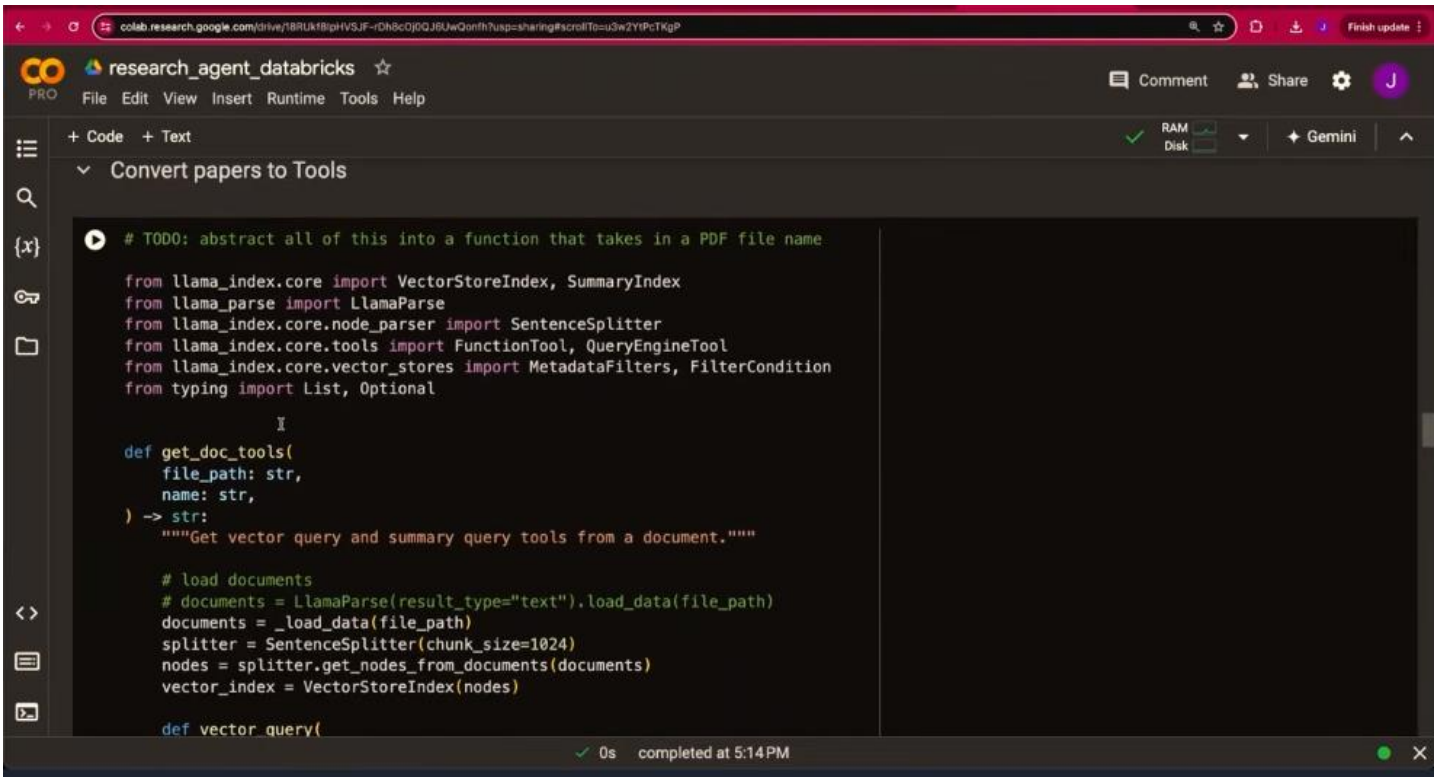

research_agent_databricks
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Gemini
[ ] import os
os.environ["LLAMA_CLOUD_API_KEY"] = "llx-..."
[2] import nest_asyncio
nest_asyncio.apply()
[3] # databricks api key
api_key = "dapi58d99aab09a28479221797338952 8f86"
from llama_index.llms.databricks import Databricks
from llama_index.embeddings.huggingface import HuggingFaceEmbedding
from llama_index.core import Settings
embed_model = HuggingFaceEmbedding(model_name="BAAI/bge-small-en-v1.5")
llm = Databricks(
model="databricks-meta-llama-3-70b-instruct",
api_key=api_key,
api_base="https://<cluster_id>.cloud.databricks.com/serving-endpoints",
)
Settings.llm = llm
Settings.embed_model = embed_model
0s completed at 5:13 PM

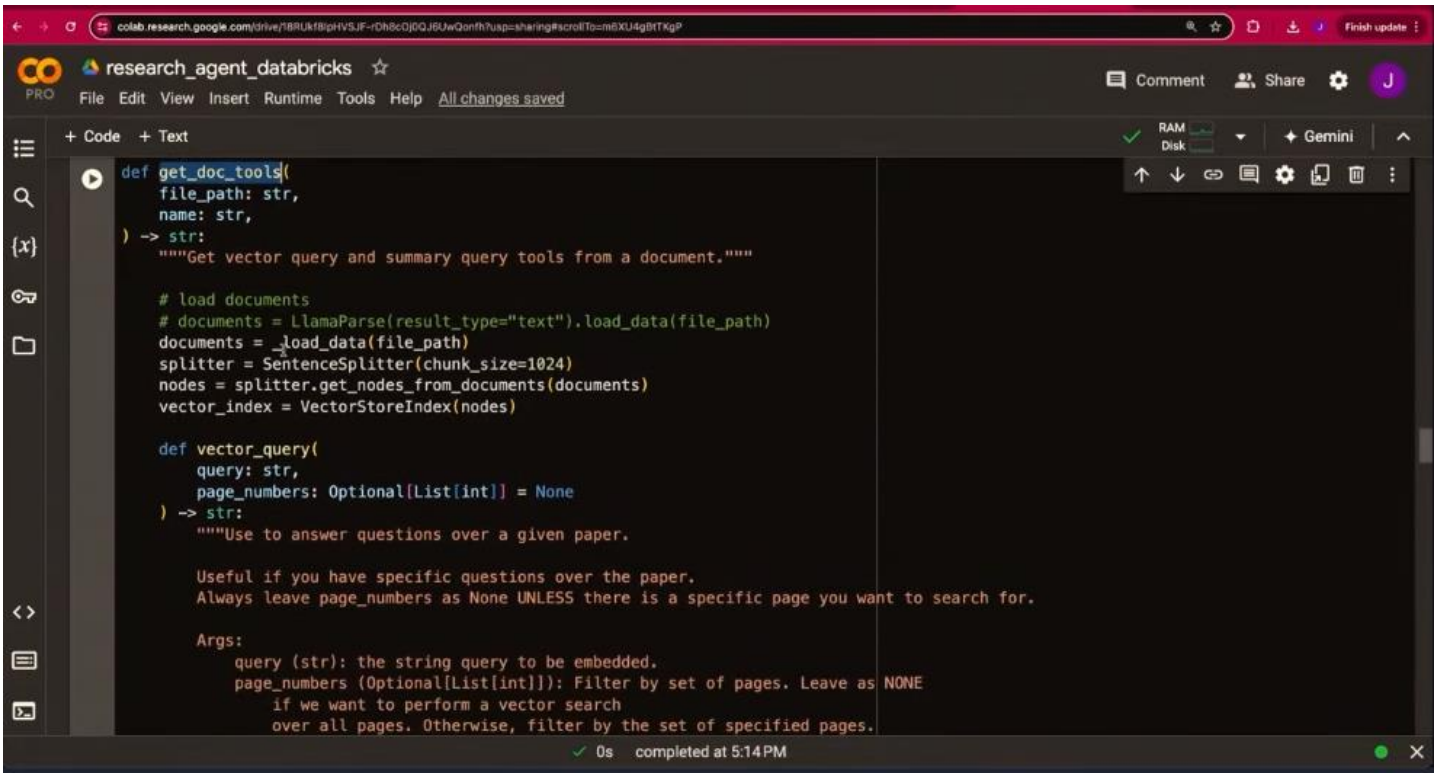

research_agent_databricks
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Gemini
[3] # databricks api key
api_key = "dapi58d99aab09a28479221797338952 8f86"
from llama_index.llms.databricks import Databricks
from llama_index.embeddings.huggingface import HuggingFaceEmbedding
from llama_index.core import Settings
embed_model = HuggingFaceEmbedding(model_name="BAAI/bge-small-en-v1.5")
llm = Databricks(
model="databricks-meta-llama-3-70b-instruct",
api_key=api_key,
api_base="https://<cluster_id>.cloud.databricks.com/serving-endpoints",
)
Settings.llm = llm
Settings.embed_model = embed_model

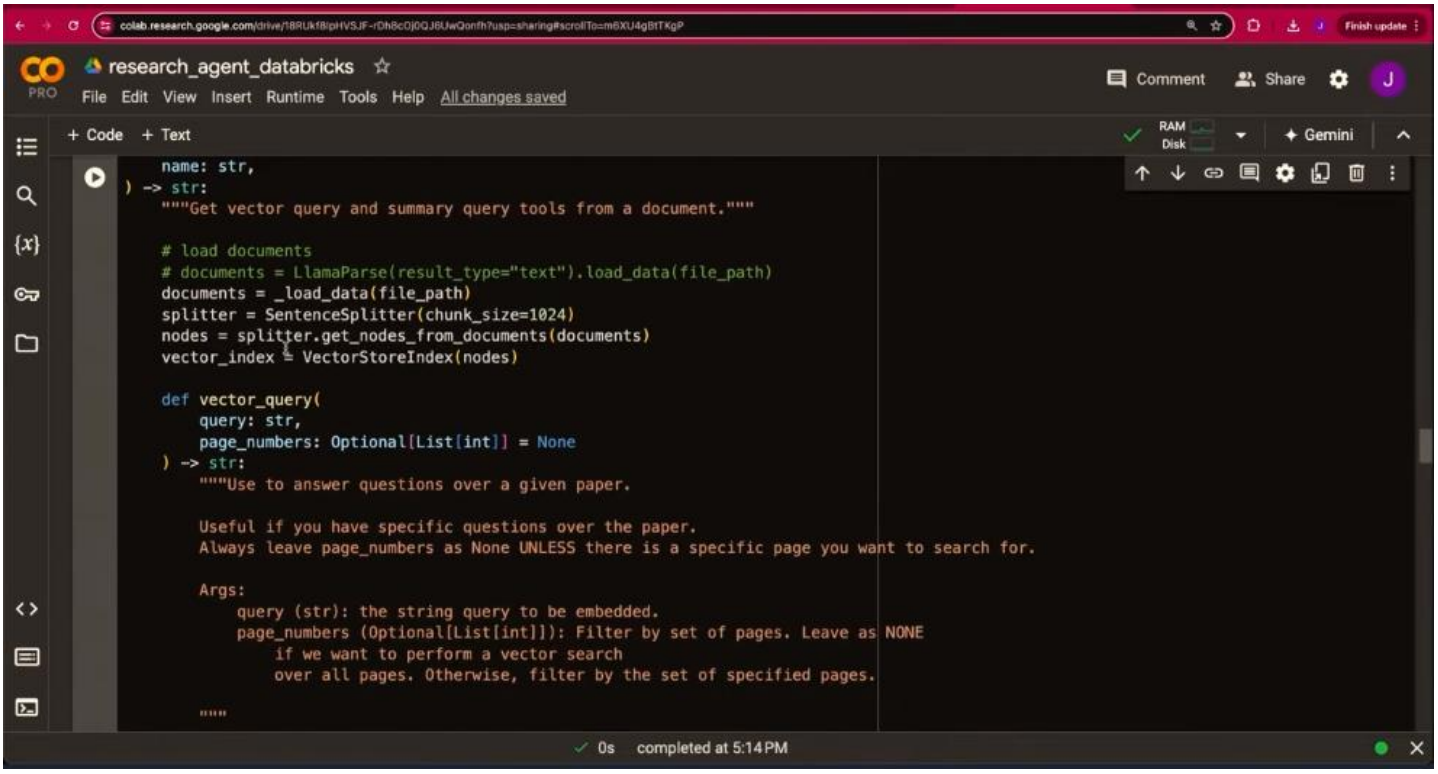

research_agent_databricks
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Gemini
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(
modules.json: 100% 349/349 [00:00<00:00, 6.89kB/s]
config_sentence_transformers.json: 100% 124/124 [00:00<00:00, 2.43kB/s]
README.md: 100% 94.8k/94.8k [00:00<00:00, 1.28MB/s]
sentence_bert_config.json: 100% 52.0/52.0 [00:00<00:00, 554B/s]
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be
warnings.warn(
config.json: 100% 743/743 [00:00<00:00, 12.1kB/s]

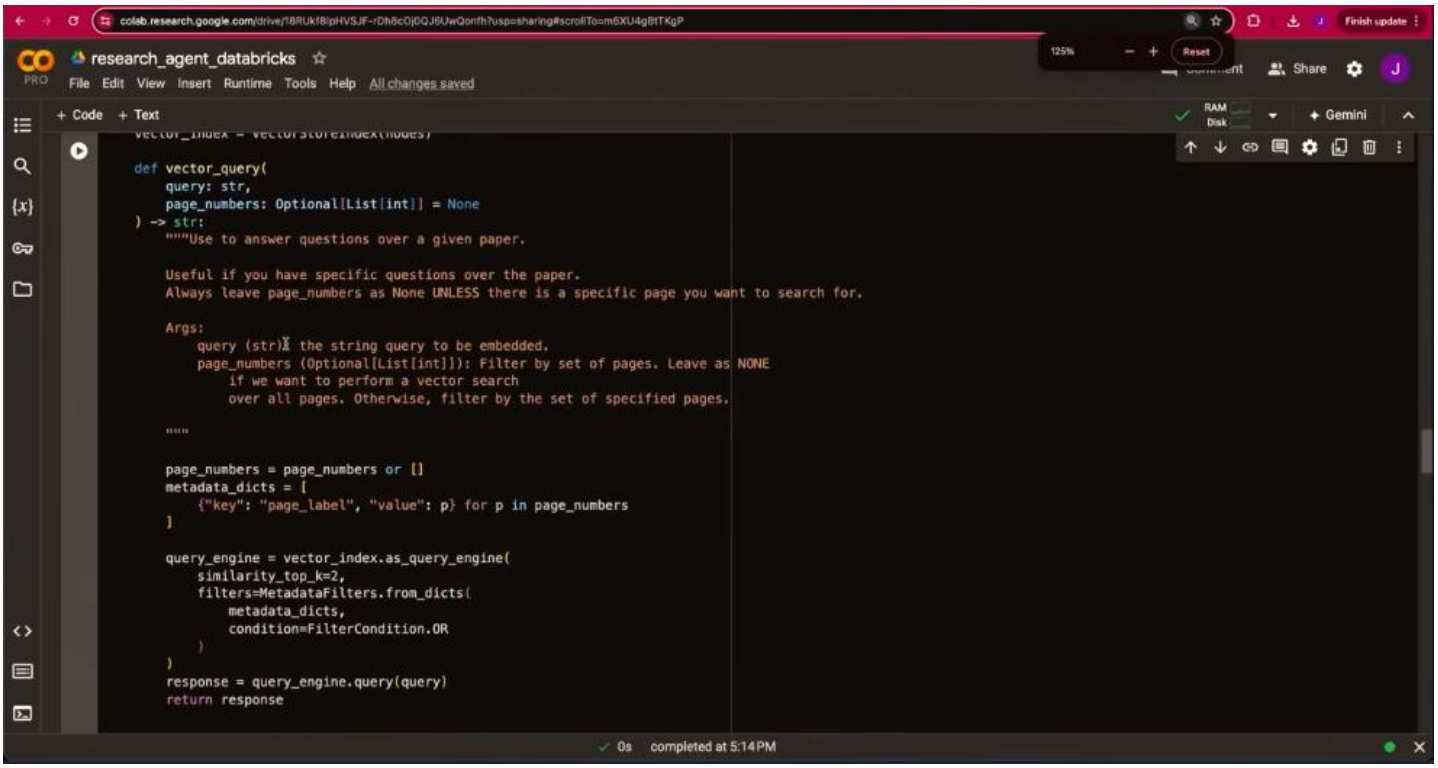

research_agent_databricks
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Gemini
Download ~3 ICLR 2024 papers, use LlamaParse
Let's parse 3 ICLR 2024 research papers using LlamaParse.
urls = [
"https://openreview.net/pdf?id=VtmBAGCN7o",
"https://openreview.net/pdf?id=6PmJoRfdaK",
"https://openreview.net/pdf?id=hSyW5go0v8",
]
papers = [
"metagpt.pdf",
"longlora.pdf",
"selfrag.pdf",
]
[ ] for url, paper in zip(urls, papers):
!wget "{url}" -O "{paper}"
--2024-06-11 06:04:43-- https://openreview.net/pdf?id=VtmBAGCN7o
Resolving openreview.net (openreview.net)... 35.184.86.251
Connecting to openreview.net (openreview.net)|35.184.86.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 16911937 (16M) [application/pdf]
0s completed at 5:14 PM

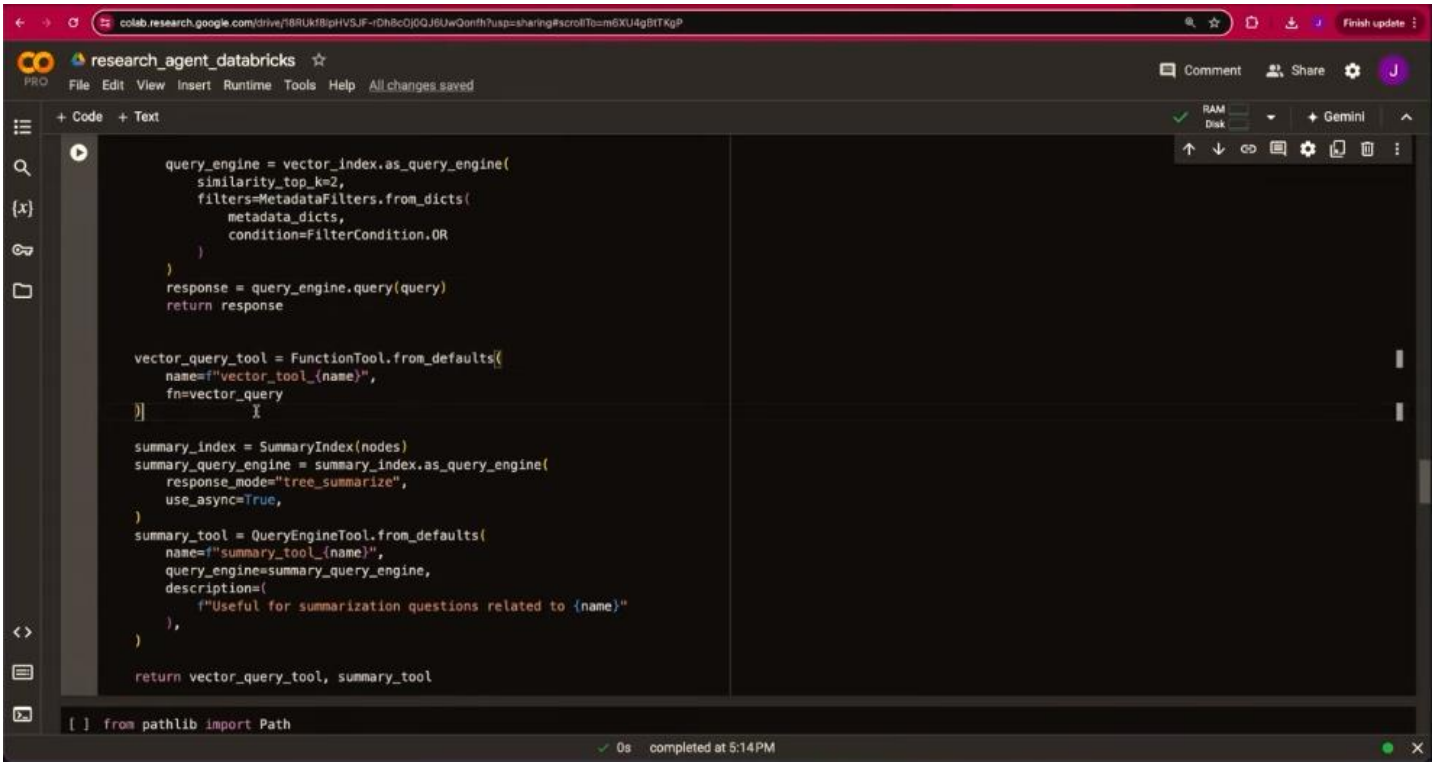

research_agent_databricks
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Gemini
2024-06-12 00:14:23 (4.48 MB/s) - 'selfrag.pdf' saved [1244749/1244749]
[8] from llama_parse import LlamaParse
from llama_index.core.schema import Document
def _load_data(file_path: str) -> Document:
parser = LlamaParse(result_type="text")
json_objs = parser.get_json_result(file_path)
json_list = json_objs[0]["pages"]
docs = []
for item in json_list:
doc = Document(
text=item["text"], metadata={"page_label": item["page"]}
)
docs.append(doc)
return docs
Convert papers to Tools
[ ] # TODO: abstract all of this into a function that takes in a PDF file name
from llama_index.core import VectorStoreIndex, SummaryIndex
from llama_parse import LlamaParse
0s completed at 5:14 PM


research_agent_databricks
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Gemini
Convert papers to Tools
# TODO: abstract all of this into a function that takes in a PDF file name
from llama_index.core import VectorStoreIndex, SummaryIndex
from llama_parse import LlamaParse
from llama_index.core.node_parser import SentenceSplitter
from llama_index.core.tools import FunctionTool, QueryEngineTool
from llama_index.core.vector_stores import MetadataFilters, FilterCondition
from typing import List, Optional
def get_doc_tools(
file_path: str,
name: str,
) -> str:
"""Get vector query and summary query tools from a document."""
# load documents
# documents = LlamaParse(result_type="text").load_data(file_path)
documents = _load_data(file_path)
splitter = SentenceSplitter(chunk_size=1024)
nodes = splitter.get_nodes_from_documents(documents)
vector_index = VectorStoreIndex(nodes)
def vector_query(
0s completed at 5:14 PM


research_agent_databricks
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Gemini
def get_doc_tools(
file_path: str,
name: str,
) -> str:
"""Get vector query and summary query tools from a document."""
# load documents
# documents = LlamaParse(result_type="text").load_data(file_path)
documents = _load_data(file_path)
splitter = SentenceSplitter(chunk_size=1024)
nodes = splitter.get_nodes_from_documents(documents)
vector_index = VectorStoreIndex(nodes)
def vector_query(
query: str,
page_numbers: Optional[List[int]] = None
) -> str:
"""Use to answer questions over a given paper.
Useful if you have specific questions over the paper.
Always leave page_numbers as None UNLESS there is a specific page you want to search for.
Args:
query (str): the string query to be embedded.
page_numbers (Optional[List[int]]): Filter by set of pages. Leave as NONE
if we want to perform a vector search
over all pages. Otherwise, filter by the set of specified pages.
0s completed at 5:14 PM


research_agent_databricks
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Gemini
name: str,
) -> str:
"""Get vector query and summary query tools from a document."""
# load documents
# documents = LlamaParse(result_type="text").load_data(file_path)
documents = _load_data(file_path)
splitter = SentenceSplitter(chunk_size=1024)
nodes = splitter.get_nodes_from_documents(documents)
vector_index = VectorStoreIndex(nodes)
def vector_query(
query: str,
page_numbers: Optional[List[int]] = None
) -> str:
"""Use to answer questions over a given paper.
Useful if you have specific questions over the paper.
Always leave page_numbers as None UNLESS there is a specific page you want to search for.
Args:
query (str): the string query to be embedded.
page_numbers (Optional[List[int]]): Filter by set of pages. Leave as NONE
if we want to perform a vector search
over all pages. Otherwise, filter by the set of specified pages.
"""
0s completed at 5:14 PM

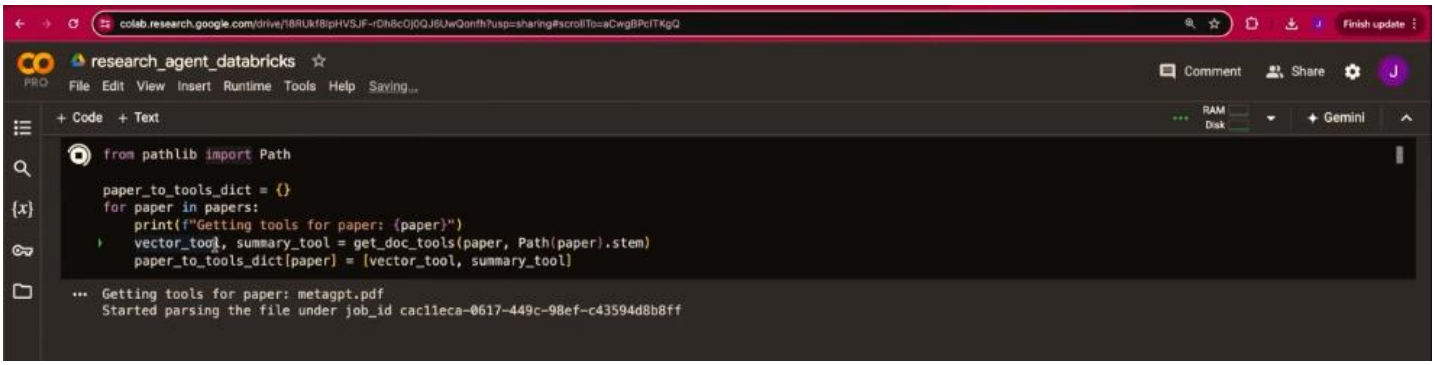

research_agent_databricks
File Edit View Insert Runtime Tools Help All changes saved
125%
Reset
Share
+ Code + Text
RAM
Disk
Gemini
def vector_query(
query: str,
page_numbers: Optional[List[int]] = None
) -> str:
"""Use to answer questions over a given paper.
Useful if you have specific questions over the paper.
Always leave page_numbers as None UNLESS there is a specific page you want to search for.
Args:
query (str): the string query to be embedded.
page_numbers (Optional[List[int]]): Filter by set of pages. Leave as NONE
if we want to perform a vector search
over all pages. Otherwise, filter by the set of specified pages.
"""
page_numbers = page_numbers or []
metadata_dicts = [
{"key": "page_label", "value": p} for p in page_numbers
]
query_engine = vector_index.as_query_engine(
similarity_top_k=2,
filters=MetadataFilters.from_dicts(
metadata_dicts,
condition=FilterCondition.OR
)
)
response = query_engine.query(query)
return response
0s completed at 5:14 PM



```
        query_engine = vector_index.as_query_engine(
            similarity_top_k=2,
            filters=MetadataFilters.from_dicts(
                metadata_dicts,
                condition=FilterCondition.OR
            )
        )
        response = query_engine.query(query)
        return response


    vector_query_tool = FunctionTool.from_defaults(
        name=f"vector_tool_{name}",
        fn=vector_query
    )

    summary_index = SummaryIndex(nodes)
    summary_query_engine = summary_index.as_query_engine(
        response_mode="tree_summarize",
        use_async=True,
    )
    summary_tool = QueryEngineTool.from_defaults(
        name=f"summary_tool_{name}",
        query_engine=summary_query_engine,
        description=(
            f"Useful for summarization questions related to {name}"
        ),
    )

    return vector_query_tool, summary_tool
```

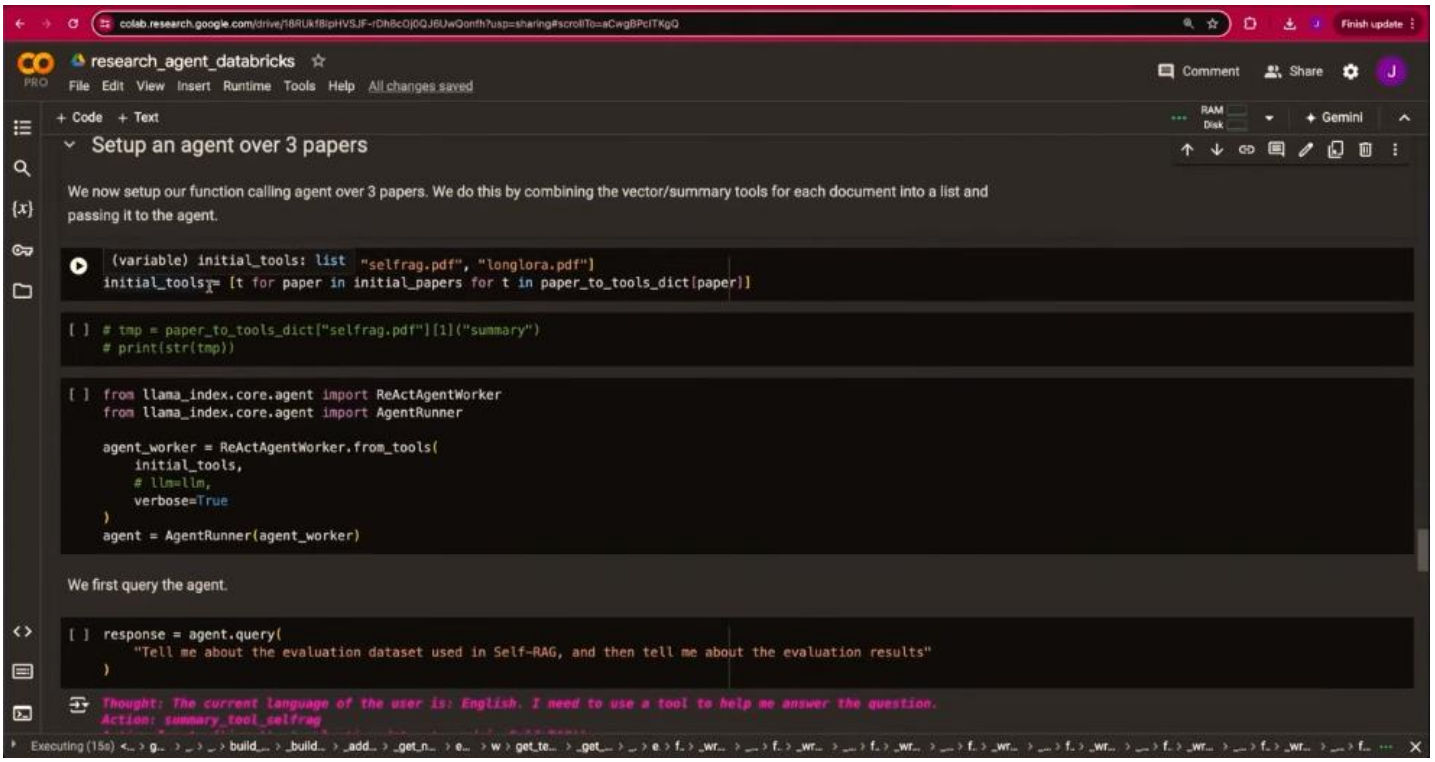
```
[ ] from pathlib import Path
```

0s completed at 5:14 PM



```
from pathlib import Path

paper_to_tools_dict = {}
for paper in papers:
    print(f"Getting tools for paper: {paper}")
    vector_tool, summary_tool = get_doc_tools(paper, Path(paper).stem)
    paper_to_tools_dict[paper] = [vector_tool, summary_tool]
```

```
Getting tools for paper: metagpt.pdf
Started parsing the file under job_id cac11eca-0617-449c-98ef-c43594d8b8ff
```



Setup an agent over 3 papers

We now setup our function calling agent over 3 papers. We do this by combining the vector/summary tools for each document into a list and passing it to the agent.

```
(variable) initial_tools: list
initial_tools = [t for paper in initial_papers for t in paper_to_tools_dict[paper]]
```

```
[ ] # tmp = paper_to_tools_dict["selfrag.pdf"][1]("summary")
    # print(str(tmp))
```

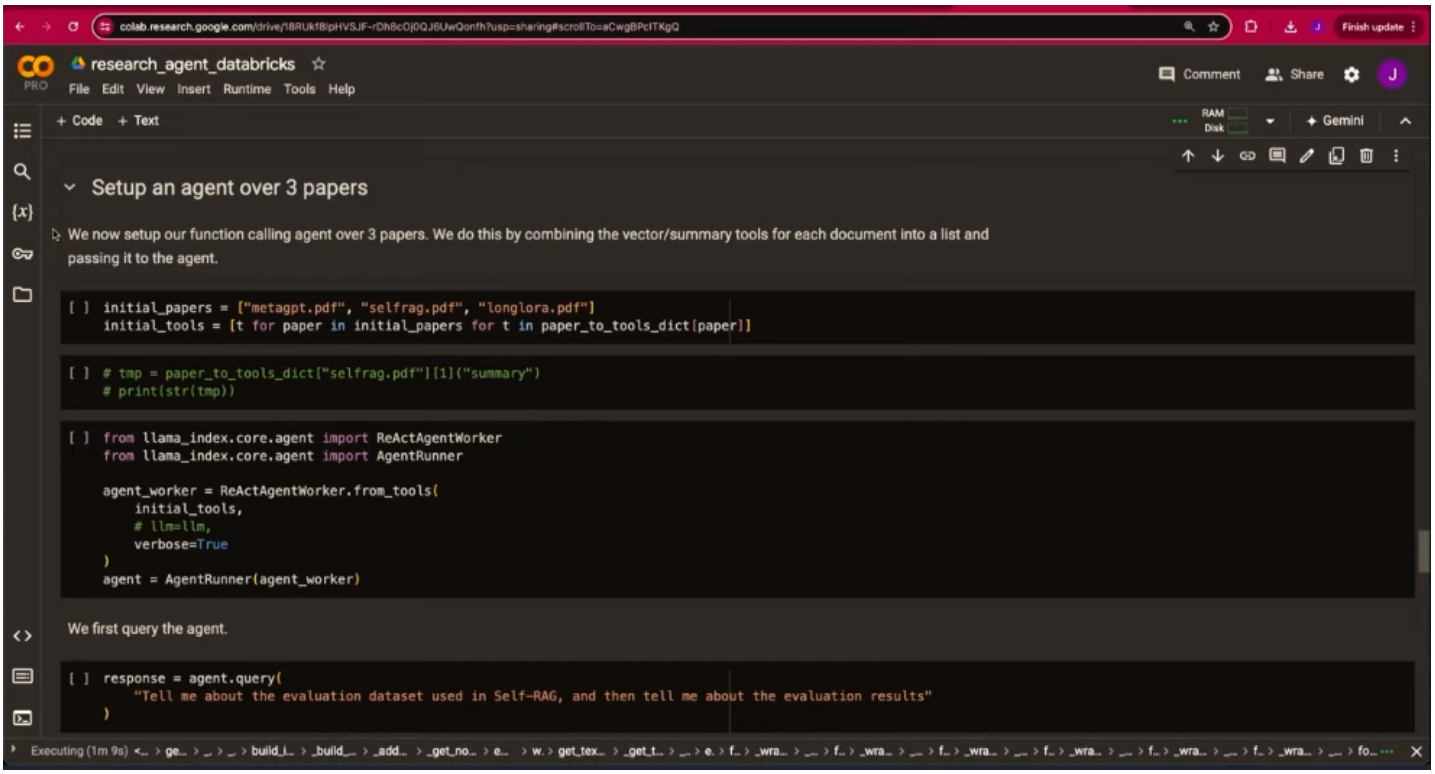
```
[ ] from llama_index.core.agent import ReActAgentWorker
    from llama_index.core.agent import AgentRunner

    agent_worker = ReActAgentWorker.from_tools(
        initial_tools,
        # llm=llm,
        verbose=True
    )
    agent = AgentRunner(agent_worker)
```

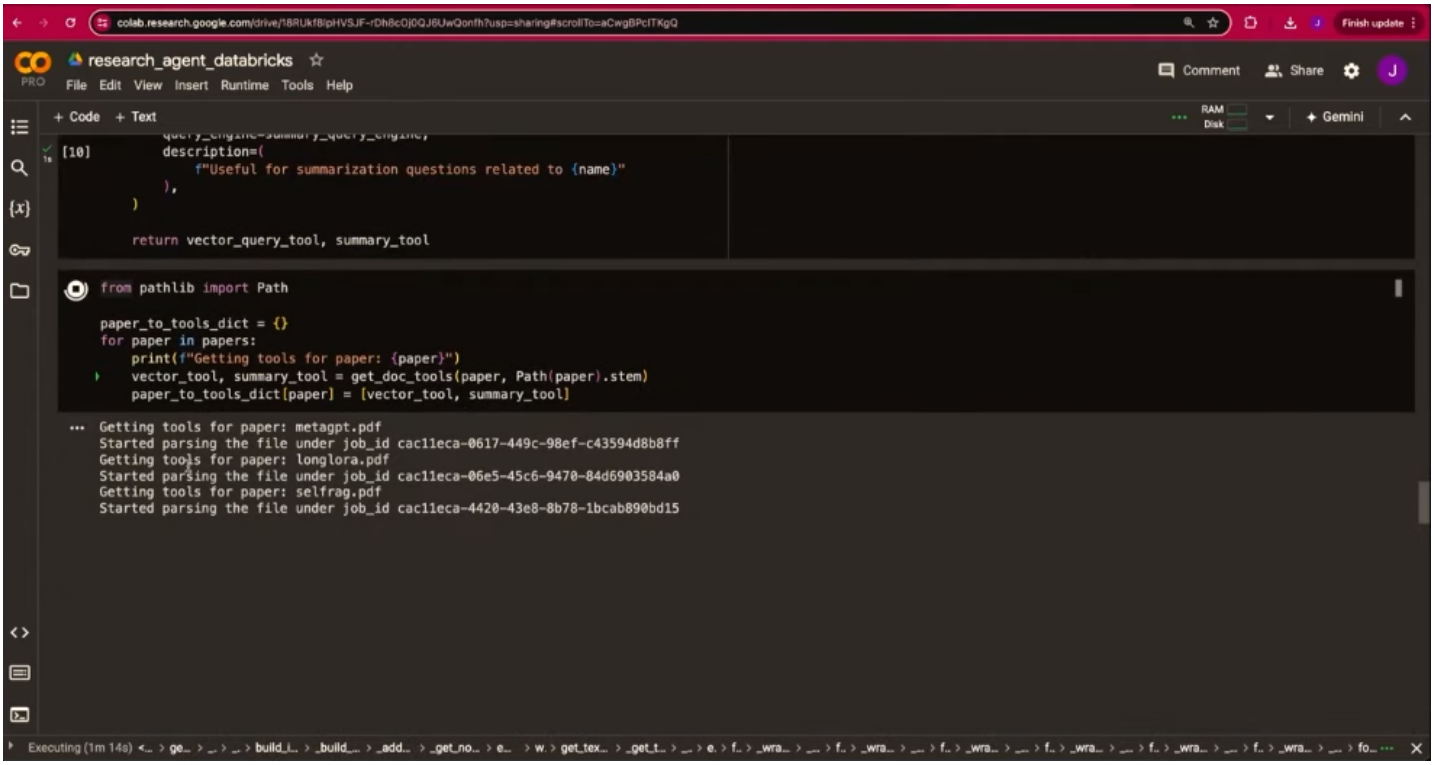
We first query the agent.

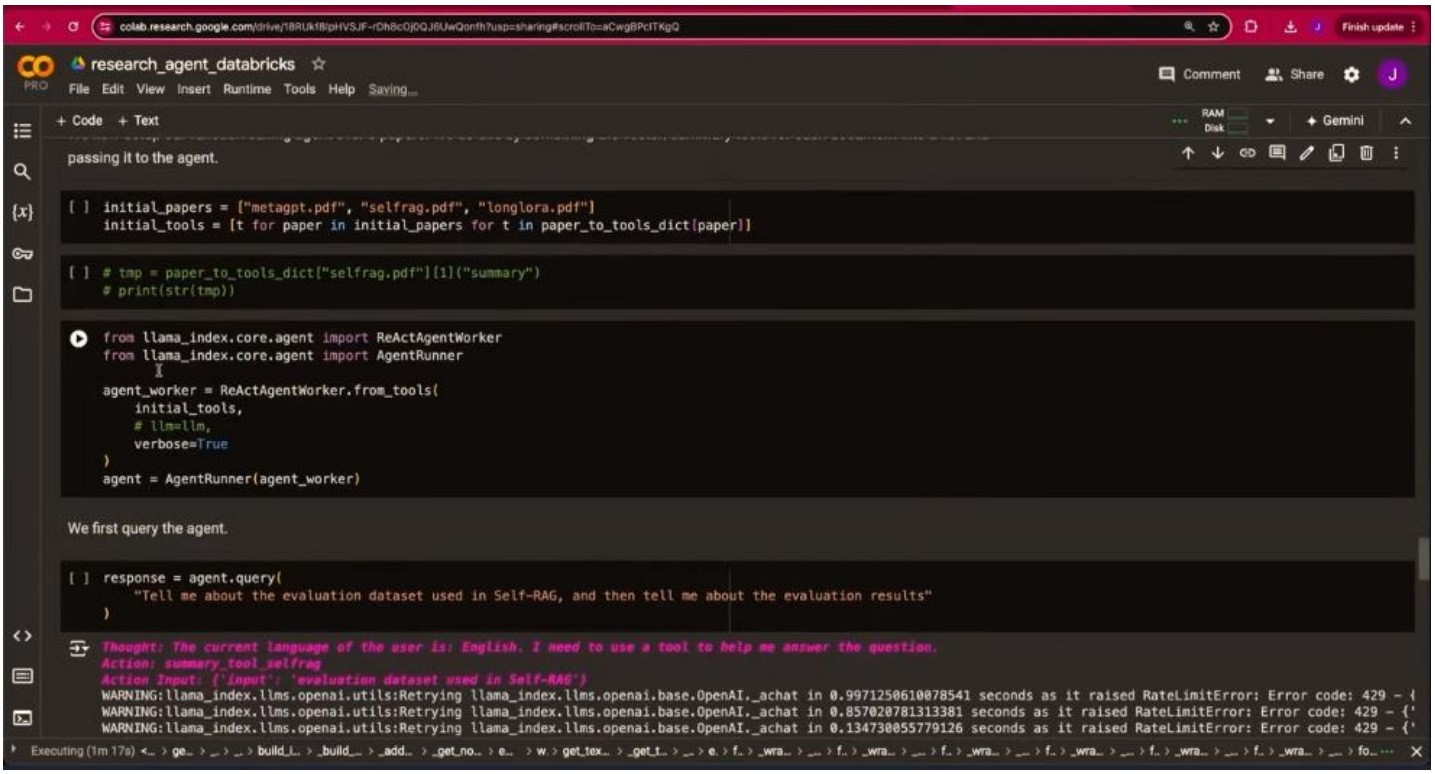
```
[ ] response = agent.query(
        "Tell me about the evaluation dataset used in Self-RAG, and then tell me about the evaluation results"
    )
```

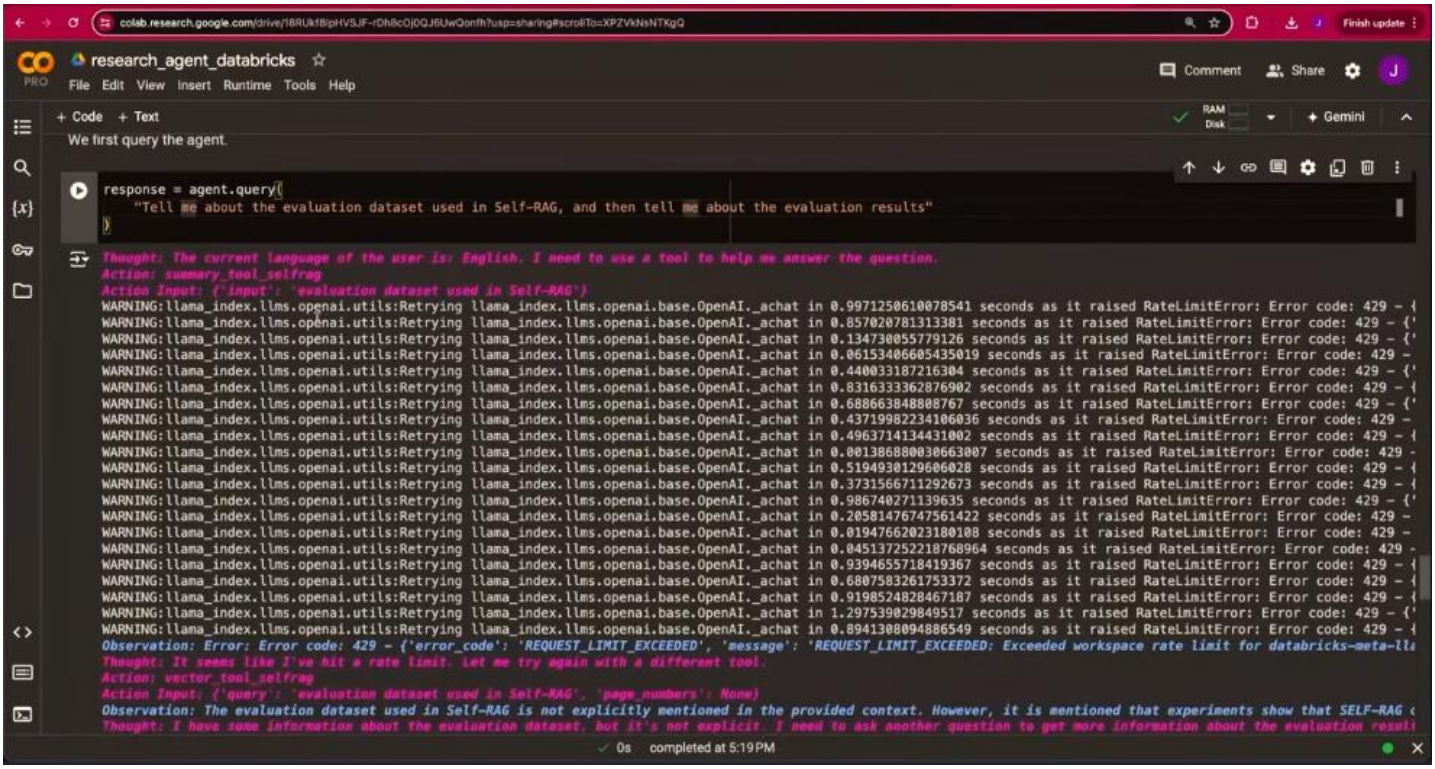
```
Thought: The current language of the user is: English. I need to use a tool to help me answer the question.
Action: summary_tool_selfrag
```

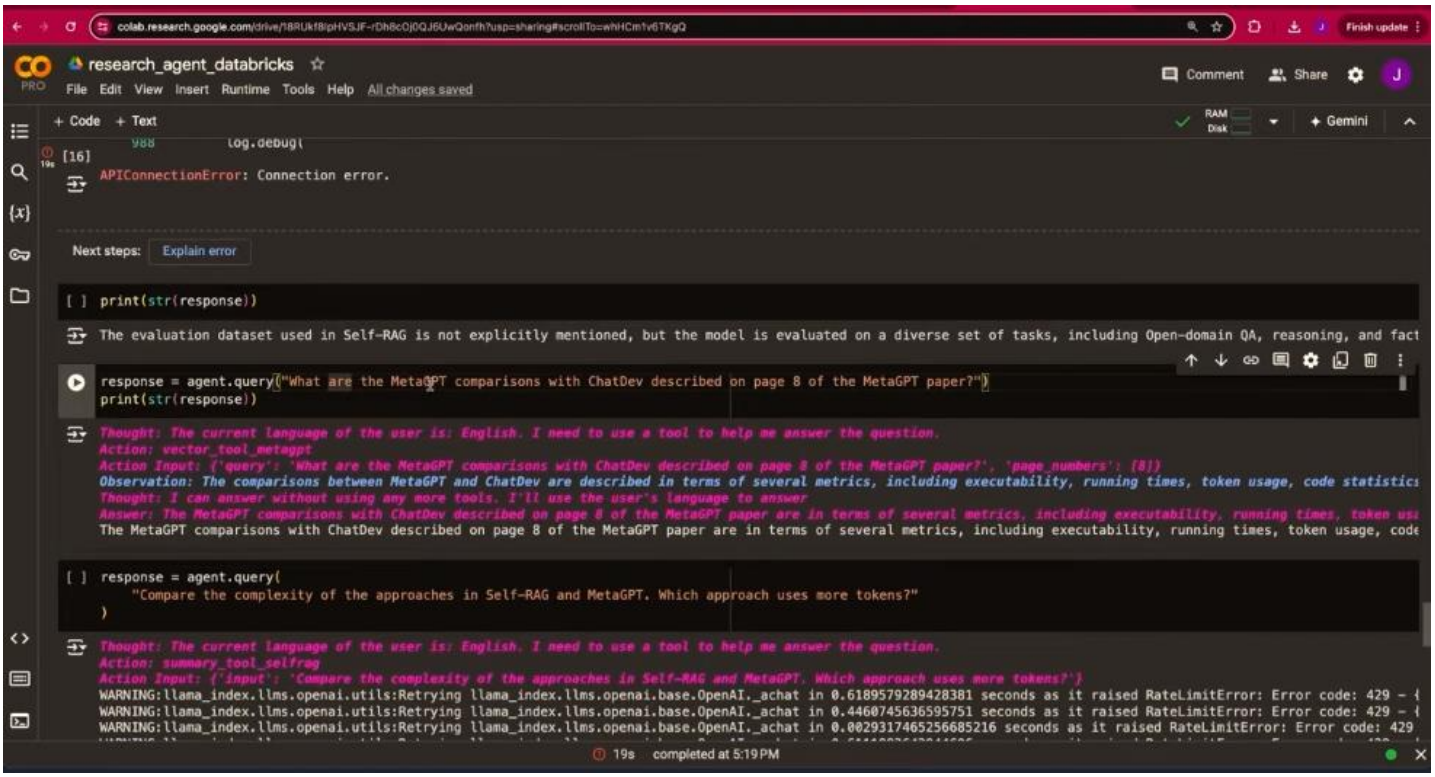
Executing (15s) <... > g... > _ > _ > build_... > _build... > _add... > _get_n... > e... > w > get_te... > _get_... > _ > e > f.. > _wr... > _... > f. > _wr... > _... > f. > _wr... > _... > f. > _wr... > _... > f. > _wr... > _... > f. > _wr... > _... > f...

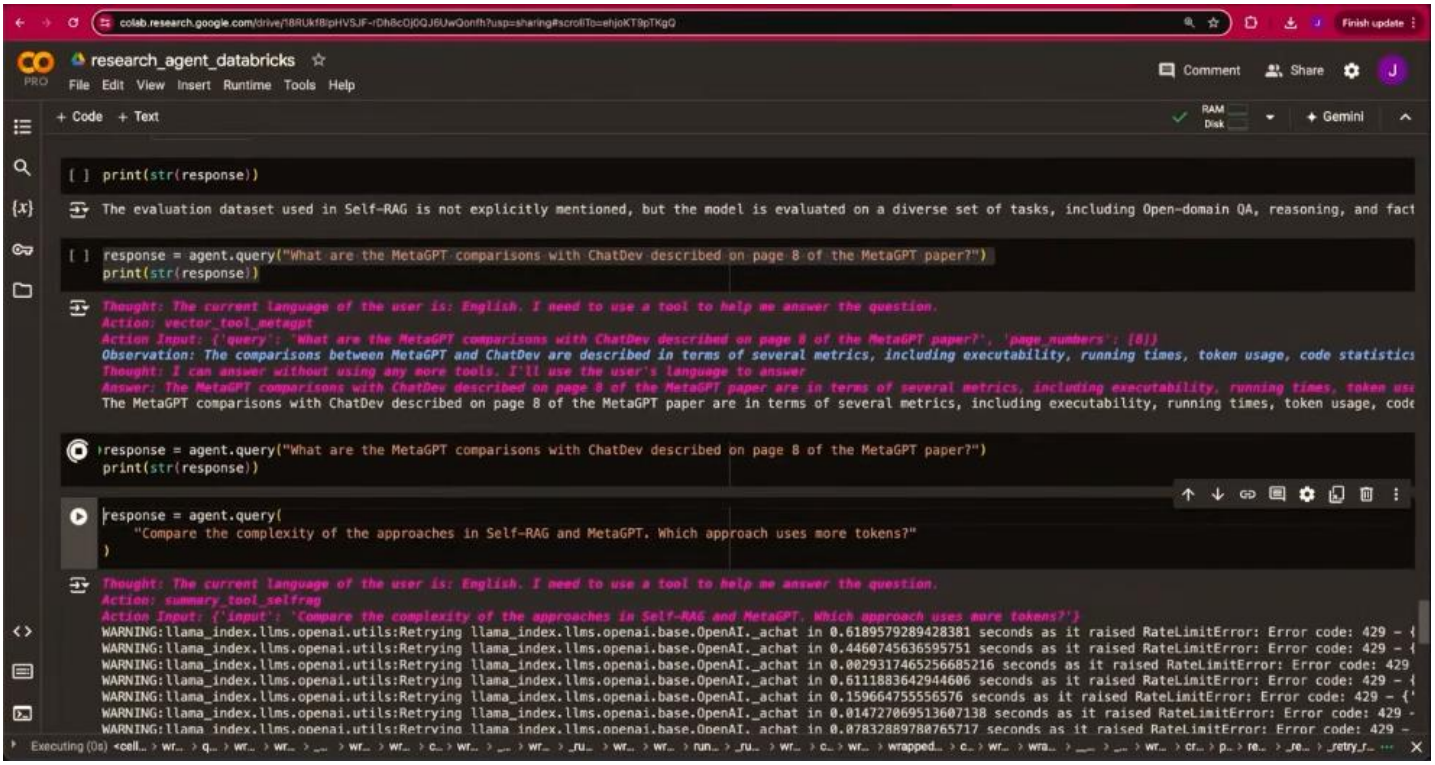

research_agent_databricks
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
Gemini
Setup an agent over 3 papers
We now setup our function calling agent over 3 papers. We do this by combining the vector/summary tools for each document into a list and passing it to the agent.
initial_papers = ["metagpt.pdf", "selfrag.pdf", "longlora.pdf"]
initial_tools = [t for paper in initial_papers for t in paper_to_tools_dict[paper]]
# tmp = paper_to_tools_dict["selfrag.pdf"][1]("summary")
# print(str(tmp))
from llama_index.core.agent import ReActAgentWorker
from llama_index.core.agent import AgentRunner
agent_worker = ReActAgentWorker.from_tools(
initial_tools,
# llm=llm,
verbose=True
)
agent = AgentRunner(agent_worker)
We first query the agent.
response = agent.query(
"Tell me about the evaluation dataset used in Self-RAG, and then tell me about the evaluation results"
)

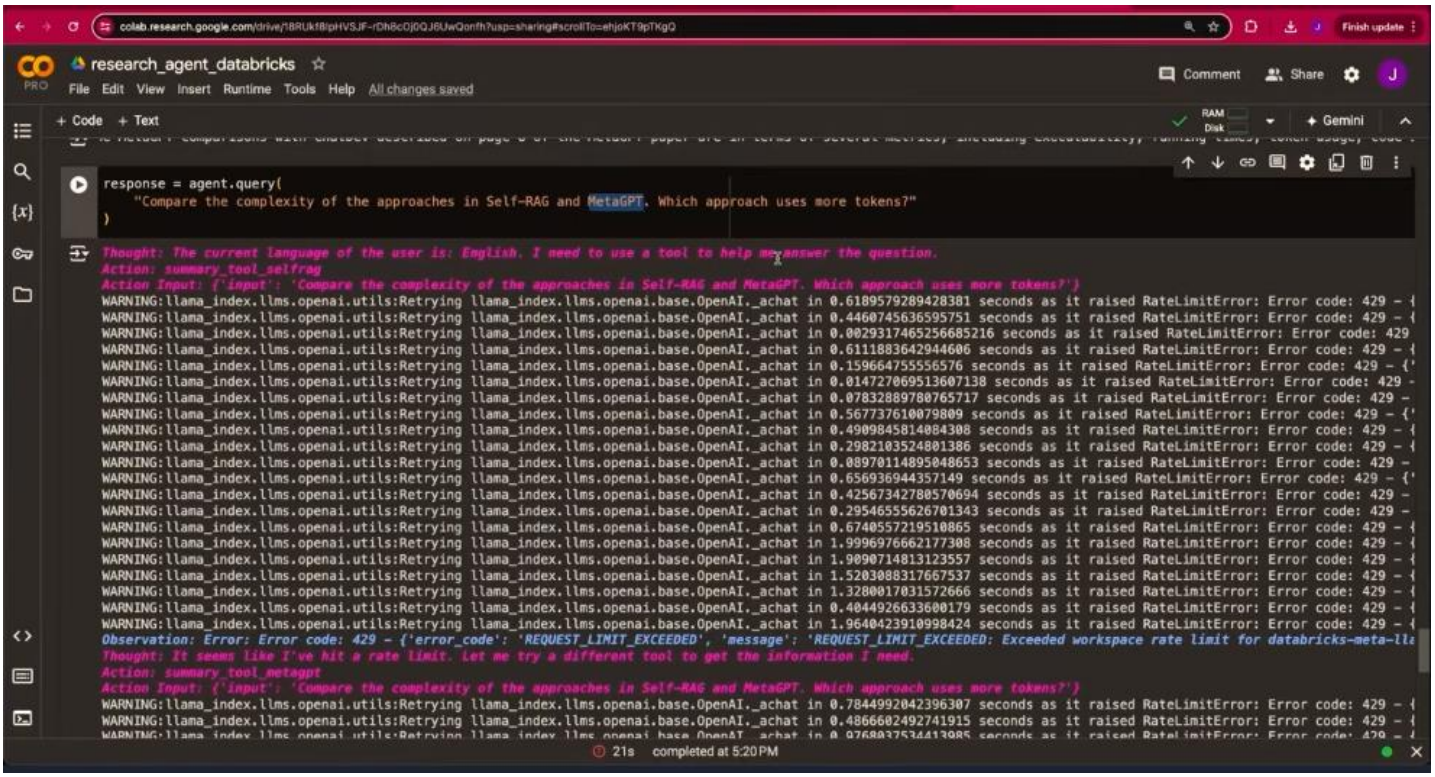

research_agent_databricks
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
Gemini
description=(
f"Useful for summarization questions related to {name}"
),
)
return vector_query_tool, summary_tool
from pathlib import Path
paper_to_tools_dict = {}
for paper in papers:
print(f"Getting tools for paper: {paper}")
vector_tool, summary_tool = get_doc_tools(paper, Path(paper).stem)
paper_to_tools_dict[paper] = [vector_tool, summary_tool]
Getting tools for paper: metagpt.pdf
Started parsing the file under job_id cac11eca-0617-449c-98ef-c43594d8b8ff
Getting tools for paper: longlora.pdf
Started parsing the file under job_id cac11eca-06e5-45c6-9470-84d6903584a0
Getting tools for paper: selfrag.pdf
Started parsing the file under job_id cac11eca-4420-43e8-8b78-1bcab890bd15


research_agent_databricks
File Edit View Insert Runtime Tools Help Saving...
Comment Share
+ Code + Text
Gemini
passing it to the agent.
initial_papers = ["metagpt.pdf", "selfrag.pdf", "longlora.pdf"]
initial_tools = [t for paper in initial_papers for t in paper_to_tools_dict[paper]]
# tmp = paper_to_tools_dict["selfrag.pdf"][1]("summary")
# print(str(tmp))
from llama_index.core.agent import ReActAgentWorker
from llama_index.core.agent import AgentRunner
agent_worker = ReActAgentWorker.from_tools(
initial_tools,
# llm=llm,
verbose=True
)
agent = AgentRunner(agent_worker)
We first query the agent.
response = agent.query(
"Tell me about the evaluation dataset used in Self-RAG, and then tell me about the evaluation results"
)
Thought: The current language of the user is: English. I need to use a tool to help me answer the question.
Action: summary_tool_selfrag
Action Input: {'input': 'evaluation dataset used in Self-RAG'}
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.9971250610078541 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.857020781313381 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.134730055779126 seconds as it raised RateLimitError: Error code: 429 - {'
Executing (1m 17s)


research_agent_databricks
File Edit View Insert Runtime Tools Help
Comment Share
+ Code + Text
Gemini
We first query the agent.
response = agent.query(
"Tell me about the evaluation dataset used in Self-RAG, and then tell me about the evaluation results"
)
Thought: The current language of the user is: English. I need to use a tool to help me answer the question.
Action: summary_tool_selfrag
Action Input: {'input': 'evaluation dataset used in Self-RAG'}
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.9971250610078541 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.857020781313381 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.134730055779126 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.06153406605435019 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.440033187216304 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.8316333362876902 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.688663848808767 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.43719982234106036 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.4963714134431002 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.0013868800306630 07 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.5194930129606028 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.3731566711292673 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.986740271139635 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.20581476747561422 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.01947662023180108 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.045137252218768964 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.9394655718419367 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.6807583261753372 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.9198524828467187 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 1.297539029849517 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.8941308094886549 seconds as it raised RateLimitError: Error code: 429 - {
Observation: Error: Error code: 429 - {'error_code': 'REQUEST_LIMIT_EXCEEDED', 'message': 'REQUEST_LIMIT_EXCEEDED: Exceeded workspace rate limit for databricks-meta-lla
Thought: It seems like I've hit a rate limit. Let me try again with a different tool.
Action: vector_tool_selfrag
Action Input: {'query': 'evaluation dataset used in Self-RAG', 'page_numbers': None}
Observation: The evaluation dataset used in Self-RAG is not explicitly mentioned in the provided context. However, it is mentioned that experiments show that SELF-RAG
Thought: I have some information about the evaluation dataset, but it's not explicit. I need to ask another question to get more information about the evaluation resul
0s completed at 5:19 PM


research_agent_databricks
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Gemini
[16]
APIConnectionError: Connection error.
Next steps: Explain error
print(str(response))
The evaluation dataset used in Self-RAG is not explicitly mentioned, but the model is evaluated on a diverse set of tasks, including Open-domain QA, reasoning, and fact
response = agent.query("What are the MetaGPT comparisons with ChatDev described on page 8 of the MetaGPT paper?")
print(str(response))
Thought: The current language of the user is: English. I need to use a tool to help me answer the question.
Action: vector_tool_metagpt
Action Input: {'query': 'What are the MetaGPT comparisons with ChatDev described on page 8 of the MetaGPT paper?', 'page_numbers': [8]}
Observation: The comparisons between MetaGPT and ChatDev are described in terms of several metrics, including executability, running times, token usage, code statistics
Thought: I can answer without using any more tools. I'll use the user's language to answer
Answer: The MetaGPT comparisons with ChatDev described on page 8 of the MetaGPT paper are in terms of several metrics, including executability, running times, token usa
The MetaGPT comparisons with ChatDev described on page 8 of the MetaGPT paper are in terms of several metrics, including executability, running times, token usage, code
response = agent.query(
"Compare the complexity of the approaches in Self-RAG and MetaGPT. Which approach uses more tokens?"
)
Thought: The current language of the user is: English. I need to use a tool to help me answer the question.
Action: summary_tool_selfrag
Action Input: {'input': 'Compare the complexity of the approaches in Self-RAG and MetaGPT. Which approach uses more tokens?'}
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.6189579289428381 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.4460745636595751 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.0029317465256685216 seconds as it raised RateLimitError: Error code: 429
19s completed at 5:19 PM


research_agent_databricks
File Edit View Insert Runtime Tools Help
Comment Share
+ Code + Text
Gemini
print(str(response))
The evaluation dataset used in Self-RAG is not explicitly mentioned, but the model is evaluated on a diverse set of tasks, including Open-domain QA, reasoning, and fact
response = agent.query("What are the MetaGPT comparisons with ChatDev described on page 8 of the MetaGPT paper?")
print(str(response))
Thought: The current language of the user is: English. I need to use a tool to help me answer the question.
Action: vector_tool_metagpt
Action Input: {'query': 'What are the MetaGPT comparisons with ChatDev described on page 8 of the MetaGPT paper?', 'page_numbers': [8]}
Observation: The comparisons between MetaGPT and ChatDev are described in terms of several metrics, including executability, running times, token usage, code statistics
Thought: I can answer without using any more tools. I'll use the user's language to answer
Answer: The MetaGPT comparisons with ChatDev described on page 8 of the MetaGPT paper are in terms of several metrics, including executability, running times, token usa
The MetaGPT comparisons with ChatDev described on page 8 of the MetaGPT paper are in terms of several metrics, including executability, running times, token usage, code
response = agent.query("What are the MetaGPT comparisons with ChatDev described on page 8 of the MetaGPT paper?")
print(str(response))
response = agent.query(
"Compare the complexity of the approaches in Self-RAG and MetaGPT. Which approach uses more tokens?"
)
Thought: The current language of the user is: English. I need to use a tool to help me answer the question.
Action: summary_tool_selfrag
Action Input: {'input': 'Compare the complexity of the approaches in Self-RAG and MetaGPT. Which approach uses more tokens?'}
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.6189579289428381 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.4460745636595751 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.0029317465256685216 seconds as it raised RateLimitError: Error code: 429
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.6111883642944606 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.159664755556576 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.014727069513607138 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.07832889780765717 seconds as it raised RateLimitError: Error code: 429 -
Executing (0s)


colab.research.google.com/drive/1BRUkfBipHVSJF-rDh8cOj0QJ6UwQonfh?usp=sharing#scrollTo=ehjoKT9pTKgQ
Finish update
research_agent_databricks
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Gemini
response = agent.query(
"Compare the complexity of the approaches in Self-RAG and MetaGPT. Which approach uses more tokens?"
)
Thought: The current language of the user is: English. I need to use a tool to help me answer the question.
Action: summary_tool_selfrag
Action Input: {'input': 'Compare the complexity of the approaches in Self-RAG and MetaGPT. Which approach uses more tokens?'}
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.6189579289428381 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.4460745636595751 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.0029317465256685216 seconds as it raised RateLimitError: Error code: 429
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.6111883642944606 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.1596647555556576 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.014727069513607138 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.07832889780765717 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.567737610079809 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.490984581408430 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.2982103524801386 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.08970114895048653 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.656936944357149 seconds as it raised RateLimitError: Error code: 429 - {'
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.42567342780570694 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.29546555626701343 seconds as it raised RateLimitError: Error code: 429 -
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.6740557219510865 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 1.9996976662177308 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 1.9090714813123557 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 1.5203088317667537 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 1.3280017031572666 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.4044926633600179 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 1.9640423910998424 seconds as it raised RateLimitError: Error code: 429 - {
Observation: Error: Error code: 429 - {'error_code': 'REQUEST_LIMIT_EXCEEDED', 'message': 'REQUEST_LIMIT_EXCEEDED: Exceeded workspace rate limit for databricks-meta-lla
Thought: It seems like I've hit a rate limit. Let me try a different tool to get the information I need.
Action: summary_tool_metagpt
Action Input: {'input': 'Compare the complexity of the approaches in Self-RAG and MetaGPT. Which approach uses more tokens?'}
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.7844992042396307 seconds as it raised RateLimitError: Error code: 429 - {
WARNING:llama_index.llms.openai.utils:Retrying llama_index.llms.openai.base.OpenAI._achat in 0.4866602492741915 seconds as it raised RateLimitError: Error code: 429 - {
21s completed at 5:20 PM