you_rag

October 31, 2025

1 You.com Hackathon RAG Pipeline V2 Project

This RAG Pipeline is made for You.com Agentic Hackathon. It is focused on Finance, Accounting, or just legal documents. It is a scalabel pipeline which is multi model and uses multiple LLM agents to do the work. Key Features and Architecture of the RAG Pipeline:

1. LlamaParse enabled parsing of excel, word, pdf, images, and html documents. We used Llama Agentic AI based on GPT 4.1 mini for parsing 1000+ pages workflow and extracting them in markdown form.
2. Created three RAG Agents on you.com, all based on GPT 5 mini, and used Custom Agent API to make more than 1700+ API calls during the project.
3. Used RAG Agents to make logical document segemntation and document type tagging for metadata.
4. Used Recursive and Semantic Chunking for the Logical Documents
5. Created a FAISS Vector Storage for fast nearest neighbour similarlity retriever, and also because its scalable. Used Gemini 2.5 Flash for RAG LLM and BAAI/bge-large-en-v1.5 for embedding.
6. Created an advanced Query engine which includes Hybrid Retriver, Metadata Filtering and Query Routing, Query Expansion, Rerankers, and SubQuestion Query Engine to allow more complex question handling.
7. We then tested the RAG Pipeline against 15 Test queries

We obtained following results in our evaluation: 1. MRR: 85% 2. HitRate@10: 100% 3. Recall@10: 87% 4. HitRate@8: 93% 5. Recall@8: 84% 6. Relative Numarical Accuracy: 100% 7. Query Accuracy: 87% (13/15 correct answers) 8. Average Latency: 20.46 seconds

Overall, by applying you.com Agents we were able to create a competent scalable RAG pipeline which is able to process 1000+ pages worth of documents while focusing on high Relative Numerical Accuracy required for financial and legal tasks.

1.1 Document Parsing

In this part we applied LlamaParsee API to our documents. He had more than 30 different documents covering over 1000+ pages. They covered a variety of documents types like financial statements, excel workings, financial models, contracts, employee contracts, mortgage documents, ID, invoices, bank statement, PO agreement, and other different kinds of documents observed in day to day workings of a finance, accounting, or legal documents.

```
[1]: #importing essential llama index libraries and other libraries, we will import␣
     ↪others later when they are used
     from llama_index.core import SimpleDirectoryReader
     from llama_index.core import VectorStoreIndex, Document
     from llama_index.core.retrievers import VectorIndexRetriever
     from llama_index.core.query_engine import RetrieverQueryEngine
     from llama_index.core.settings import Settings
     from llama_index.llms.llama_cpp import LlamaCPP # available in case reader␣
     ↪wants to use local LLM
     from llama_index.embeddings.huggingface import HuggingFaceEmbedding
     from llama_index.readers.file import PDFReader
     from llama_cpp import Llama
     import requests
     import os, glob, asyncio, nest_asyncio
     from dotenv import load_dotenv
```

```
[6]: you_key = os.getenv('YOU_API_KEY')
     gemini_key = os.getenv('GOOGLE_API_KEY')
     llama_key = os.getenv('LLAMA_CLOUD_API_KEY')
```

```
[3]: from llama_parse import LlamaParse
```

```
[14]: parser = LlamaParse(
          # The parsing mode
          parse_mode="parse_page_with_agent",
          # The model to use
          model="openai-gpt-4-1-mini",
          # Whether to use high resolution OCR
          high_res_ocr=True,
          # Adaptive long table
          adaptive_long_table=True,
          outlined_table_extraction=True,
          output_tables_as_HTML=True,
          result_type= 'markdown'
      )
```

```
[6]: try:
         asyncio.get_running_loop().close()
     except RuntimeError:
         pass
     asyncio.set_event_loop(asyncio.new_event_loop())

     doc = parser.load_data('train/income-statement.xlsx')
```

```
2025-10-30 10:37:35,160 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 2274efdf-02db-4598-8cbe-0efab6f6f71e
```

```
2025-10-30 10:37:36,633 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/2274efdf-02db-4598-8cbe-0efab6f6f71e "HTTP/1.1 200 OK"
2025-10-30 10:37:39,193 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/2274efdf-02db-4598-8cbe-0efab6f6f71e "HTTP/1.1 200 OK"
2025-10-30 10:37:39,807 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/2274efdf-02db-4598-8cbe-0efab6f6f71e/result/markdown
"HTTP/1.1 200 OK"
```

[7]:
```
#test load
print(doc[0].text)
```

```
# SingleStep

|       | XYZ Limited                        |
|         | Income Statement                                  |
| ----- | --------------------------------- |
----------------------------------- | ------- |
----------------------------------------------------- |
|       |                                    |
|         | For the Years Ending \[Dec 31, 2020 and Dec 31, 2019] |
|       |                                    |
|         |                                                   |
|       | Revenue                            |
| 2020    | 2019                                              |
|       |                                    | Sales revenue
| 110,000 | 95,000                                            |
|       |                                    | (Less sales returns and
allowances) |         |                                                   |
|       |                                    | Service revenue
| 70,000  | 62,000                                            |
|       |                                    | Interest revenue
|         |                                                   |
|       |                                    | Other revenue
|         |                                                   |
|       | Total Revenues                     |
| 180,000 | 157,000                                           |
|       |                                    |
|         | \[42]                                             |
| \[42] | Expenses                           |
|         |                                                   |
|       |                                    | Advertising
| 1,000   | 1,000                                             |
|       |                                    | Bad debt
|         |                                                   |
|       |                                    | Commissions
|         |                                                   |
|       |                                    | Cost of goods sold
| 65,000  | 63,000                                            |
|       |                                  | Depreciation
|         |                                                       |
|       |                                  | Employee benefits
|         |                                                       |
|       |                                  | Furniture and equipment
|         | 8,000                                               |
|       |                                  | Insurance
|         |                                                       |
|       |                                  | Interest expense
| 4,200   | 5,200                                               |
|       |                                  | Maintenance and repairs
|         |                                                       |
|       |                                  | Office supplies
|         |                                                       |
|       |                                  | Payroll taxes
|         |                                                       |
|       |                                  | Rent
|         |                                                       |
|       |                                  | Research and development
|         |                                                       |
|       |                                  | Salaries and wages
| 55,000  | 55,000                                              |
|       |                                  | Software
|         |                                                       |
|       |                                  | Travel
|         |                                                       |
|       |                                  | Utilities
|         |                                                       |
|       |                                  | Web hosting and domains
|         |                                                       |
|       |                                  | Other
| 17,460  |                                                       |
|       | Total Expenses                   |
| 142,660 | 132,200                                             |
|       |                                  |
|         |                                                       |
|       |                                  | Net Income Before Taxes
| 37,340  | 24,800                                              |
|       |                                  | Income tax expense
| 14,936  | 9,920                                               |
|       |                                  |
|         |                                                       |
|       | Income from Continuing Operations |
| 22,404  | 14,880                                              |
|       |                                  | {42}
|         | \[42]                                               |
|       | Below-the-Line Items             |
|         |                                                       |
|       |                                   | Income from discontinued
operations |         |                                                         |
|       |                                   | Effect of accounting changes
|         |                                                     |
|       |                                   | Extraordinary items
|         |                                                     |
|       |                                   |
|         |                                                     |
|       | Net Income                        |
| 22,404  | 14,880                                              |
```

[8]:
```python
file_extractor = {
    ".pdf": parser,
    ".docx": parser,
    ".doc": parser,
    ".xlsx": parser,
    ".xls": parser,
    ".pptx": parser,
    ".jpg": parser,
    ".jpeg": parser,
    ".png": parser,
    ".html": parser,
}
```

[7]:
```python
lp = LlamaParse(
    parse_mode="parse_page_with_agent",
    model="openai-gpt-4-1-mini",
    high_res_ocr=True,
    adaptive_long_table=True,
    outlined_table_extraction=True,
    output_tables_as_HTML=True,
    result_type="markdown",
    verbose=True,
)

patterns = ["*.pdf","*.docx","*.doc","*.xlsx","*.xls","*.pptx","*.jpg","*.
 ↪jpeg","*.png","*.html"]
paths = []
for pat in patterns:
    paths.extend(glob.glob(os.path.join("train", "**", pat), recursive=True))

# Limit concurrency a bit to be polite and stable
sema = asyncio.Semaphore(5)

async def parse_one(path):
    async with sema:
        return await lp.aload_data(path)

results = await asyncio.gather(*(parse_one(p) for p in paths))
documents = [d for docs in results for d in docs]

print(f"Parsed {len(documents)} nodes")
print("Sample:", documents[0].text[:400])
```

```
2025-10-30 11:07:36,512 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 9915b0ec-1425-4355-9d82-94bda4d3c50e

2025-10-30 11:07:37,228 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 1fed80a5-74a9-46d2-a1bd-4975c036a793

2025-10-30 11:07:37,745 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"
2025-10-30 11:07:37,865 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9915b0ec-1425-4355-9d82-94bda4d3c50e "HTTP/1.1 200 OK"

Started parsing the file under job_id 9cf8d013-8df0-45c4-a690-672e792a83ae

2025-10-30 11:07:38,155 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id e2aa8ae7-9147-4650-9523-cc2f4e8d5372

2025-10-30 11:07:38,775 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"
2025-10-30 11:07:38,970 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1fed80a5-74a9-46d2-a1bd-4975c036a793 "HTTP/1.1 200 OK"

Started parsing the file under job_id 7e8a986b-6957-4ac2-ad96-2814bd134321

2025-10-30 11:07:39,276 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9cf8d013-8df0-45c4-a690-672e792a83ae "HTTP/1.1 200 OK"
2025-10-30 11:07:39,801 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/e2aa8ae7-9147-4650-9523-cc2f4e8d5372 "HTTP/1.1 200 OK"
2025-10-30 11:07:40,212 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7e8a986b-6957-4ac2-ad96-2814bd134321 "HTTP/1.1 200 OK"
2025-10-30 11:07:40,305 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9915b0ec-1425-4355-9d82-94bda4d3c50e "HTTP/1.1 200 OK"
2025-10-30 11:07:41,427 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1fed80a5-74a9-46d2-a1bd-4975c036a793 "HTTP/1.1 200 OK"
2025-10-30 11:07:41,637 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9cf8d013-8df0-45c4-a690-672e792a83ae "HTTP/1.1 200 OK"
2025-10-30 11:07:42,228 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/e2aa8ae7-9147-4650-9523-cc2f4e8d5372 "HTTP/1.1 200 OK"
```

```
2025-10-30 11:07:42,561 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7e8a986b-6957-4ac2-ad96-2814bd134321 "HTTP/1.1 200 OK"
2025-10-30 11:07:43,679 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9915b0ec-1425-4355-9d82-94bda4d3c50e "HTTP/1.1 200 OK"
2025-10-30 11:07:44,777 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1fed80a5-74a9-46d2-a1bd-4975c036a793 "HTTP/1.1 200 OK"
2025-10-30 11:07:45,225 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9cf8d013-8df0-45c4-a690-672e792a83ae "HTTP/1.1 200 OK"
2025-10-30 11:07:45,651 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/e2aa8ae7-9147-4650-9523-cc2f4e8d5372 "HTTP/1.1 200 OK"
2025-10-30 11:07:46,239 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7e8a986b-6957-4ac2-ad96-2814bd134321 "HTTP/1.1 200 OK"
2025-10-30 11:07:48,094 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9915b0ec-1425-4355-9d82-94bda4d3c50e "HTTP/1.1 200 OK"
2025-10-30 11:07:48,795 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9915b0ec-1425-4355-9d82-94bda4d3c50e/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:07:49,266 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1fed80a5-74a9-46d2-a1bd-4975c036a793 "HTTP/1.1 200 OK"
2025-10-30 11:07:50,132 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/e2aa8ae7-9147-4650-9523-cc2f4e8d5372 "HTTP/1.1 200 OK"
2025-10-30 11:07:50,228 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9cf8d013-8df0-45c4-a690-672e792a83ae "HTTP/1.1 200 OK"
2025-10-30 11:07:50,644 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7e8a986b-6957-4ac2-ad96-2814bd134321 "HTTP/1.1 200 OK"
2025-10-30 11:07:54,618 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1fed80a5-74a9-46d2-a1bd-4975c036a793 "HTTP/1.1 200 OK"
2025-10-30 11:07:55,531 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/e2aa8ae7-9147-4650-9523-cc2f4e8d5372 "HTTP/1.1 200 OK"
2025-10-30 11:07:56,076 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7e8a986b-6957-4ac2-ad96-2814bd134321 "HTTP/1.1 200 OK"
2025-10-30 11:07:56,279 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9cf8d013-8df0-45c4-a690-672e792a83ae "HTTP/1.1 200 OK"
2025-10-30 11:07:56,800 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/9cf8d013-8df0-45c4-a690-672e792a83ae/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:07:59,215 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 0267fe48-ac82-4c80-9c2e-10dc6ae43f66

2025-10-30 11:07:59,905 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 3d6cf921-db75-4946-bc6a-4f19bd122980

2025-10-30 11:08:00,636 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:08:00,895 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/e2aa8ae7-9147-4650-9523-cc2f4e8d5372 "HTTP/1.1 200 OK"
```

```
2025-10-30 11:08:01,364 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1fed80a5-74a9-46d2-a1bd-4975c036a793 "HTTP/1.1 200 OK"
2025-10-30 11:08:01,541 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7e8a986b-6957-4ac2-ad96-2814bd134321 "HTTP/1.1 200 OK"
2025-10-30 11:08:02,053 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:02,361 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7e8a986b-6957-4ac2-ad96-2814bd134321/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:08:03,546 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:08:04,476 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:06,517 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/e2aa8ae7-9147-4650-9523-cc2f4e8d5372 "HTTP/1.1 200 OK"
2025-10-30 11:08:06,926 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1fed80a5-74a9-46d2-a1bd-4975c036a793 "HTTP/1.1 200 OK"
2025-10-30 11:08:07,132 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:08:08,052 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:09,385 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id b15f5493-1582-4944-a9de-9de2fa3f3bc3

2025-10-30 11:08:10,722 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b15f5493-1582-4944-a9de-9de2fa3f3bc3 "HTTP/1.1 200 OK"
2025-10-30 11:08:11,642 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:08:12,046 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/e2aa8ae7-9147-4650-9523-cc2f4e8d5372 "HTTP/1.1 200 OK"
2025-10-30 11:08:12,353 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1fed80a5-74a9-46d2-a1bd-4975c036a793 "HTTP/1.1 200 OK"
2025-10-30 11:08:12,472 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:13,172 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b15f5493-1582-4944-a9de-9de2fa3f3bc3 "HTTP/1.1 200 OK"
2025-10-30 11:08:16,552 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b15f5493-1582-4944-a9de-9de2fa3f3bc3 "HTTP/1.1 200 OK"
2025-10-30 11:08:17,073 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:08:17,575 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/e2aa8ae7-9147-4650-9523-cc2f4e8d5372 "HTTP/1.1 200 OK"
2025-10-30 11:08:17,769 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1fed80a5-74a9-46d2-a1bd-4975c036a793 "HTTP/1.1 200 OK"
2025-10-30 11:08:18,087 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/e2aa8ae7-9147-4650-9523-cc2f4e8d5372/result/markdown
"HTTP/1.1 200 OK"
```

```
2025-10-30 11:08:18,303 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1fed80a5-74a9-46d2-a1bd-4975c036a793/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:08:18,490 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:21,069 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b15f5493-1582-4944-a9de-9de2fa3f3bc3 "HTTP/1.1 200 OK"
2025-10-30 11:08:22,910 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 91e7e30a-86d6-40d4-8bcc-69a156bbb967

2025-10-30 11:08:23,935 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:24,949 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967 "HTTP/1.1 200 OK"
2025-10-30 11:08:25,256 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:08:25,563 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 59fa2279-5718-4f60-84db-8c3ce2845027

2025-10-30 11:08:26,485 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b15f5493-1582-4944-a9de-9de2fa3f3bc3 "HTTP/1.1 200 OK"
2025-10-30 11:08:26,963 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/59fa2279-5718-4f60-84db-8c3ce2845027 "HTTP/1.1 200 OK"
2025-10-30 11:08:27,407 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967 "HTTP/1.1 200 OK"
2025-10-30 11:08:29,352 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:29,355 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/59fa2279-5718-4f60-84db-8c3ce2845027 "HTTP/1.1 200 OK"
2025-10-30 11:08:30,888 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:08:31,092 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967 "HTTP/1.1 200 OK"
2025-10-30 11:08:32,015 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b15f5493-1582-4944-a9de-9de2fa3f3bc3 "HTTP/1.1 200 OK"
2025-10-30 11:08:32,748 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/59fa2279-5718-4f60-84db-8c3ce2845027 "HTTP/1.1 200 OK"
2025-10-30 11:08:34,985 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:35,496 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967 "HTTP/1.1 200 OK"
2025-10-30 11:08:36,316 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:08:37,136 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/59fa2279-5718-4f60-84db-8c3ce2845027 "HTTP/1.1 200 OK"
2025-10-30 11:08:40,327 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
```

```
ai/api/parsing/job/b15f5493-1582-4944-a9de-9de2fa3f3bc3 "HTTP/1.1 200 OK"
2025-10-30 11:08:40,333 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:40,903 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967 "HTTP/1.1 200 OK"
2025-10-30 11:08:40,911 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b15f5493-1582-4944-a9de-9de2fa3f3bc3/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:08:42,076 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:08:43,022 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/59fa2279-5718-4f60-84db-8c3ce2845027 "HTTP/1.1 200 OK"
2025-10-30 11:08:43,286 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 53749373-672d-47ad-ad94-ff88b5811fba

2025-10-30 11:08:44,857 - INFO - HTTP Request: GET
https://api.cloud.llamaindex.ai/api/parsing/job/53749373-672d-47ad-
ad94-ff88b5811fba "HTTP/1.1 200 OK"
2025-10-30 11:08:45,841 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"

.

2025-10-30 11:08:46,557 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967 "HTTP/1.1 200 OK"
2025-10-30 11:08:47,204 - INFO - HTTP Request: GET
https://api.cloud.llamaindex.ai/api/parsing/job/53749373-672d-47ad-
ad94-ff88b5811fba "HTTP/1.1 200 OK"
2025-10-30 11:08:47,888 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"

.

2025-10-30 11:08:48,401 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/59fa2279-5718-4f60-84db-8c3ce2845027 "HTTP/1.1 200 OK"
2025-10-30 11:08:50,577 - INFO - HTTP Request: GET
https://api.cloud.llamaindex.ai/api/parsing/job/53749373-672d-47ad-
ad94-ff88b5811fba "HTTP/1.1 200 OK"
2025-10-30 11:08:51,217 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:52,189 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967 "HTTP/1.1 200 OK"
2025-10-30 11:08:53,519 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:08:54,749 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/59fa2279-5718-4f60-84db-8c3ce2845027 "HTTP/1.1 200 OK"
2025-10-30 11:08:54,955 - INFO - HTTP Request: GET
https://api.cloud.llamaindex.ai/api/parsing/job/53749373-672d-47ad-
ad94-ff88b5811fba "HTTP/1.1 200 OK"
```

```
2025-10-30 11:08:55,452 - INFO - HTTP Request: GET
https://api.cloud.llamaindex.ai/api/parsing/job/53749373-672d-47ad-
ad94-ff88b5811fba/result/markdown "HTTP/1.1 200 OK"
2025-10-30 11:08:56,592 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:08:57,526 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967 "HTTP/1.1 200 OK"
2025-10-30 11:08:58,837 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:09:00,309 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/59fa2279-5718-4f60-84db-8c3ce2845027 "HTTP/1.1 200 OK"
2025-10-30 11:09:00,810 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"
2025-10-30 11:09:00,910 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/59fa2279-5718-4f60-84db-8c3ce2845027/result/markdown
"HTTP/1.1 200 OK"

Started parsing the file under job_id b0cb2d7c-efa9-42b4-9b31-d5bc89d31e48

2025-10-30 11:09:02,125 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:09:02,690 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b0cb2d7c-efa9-42b4-9b31-d5bc89d31e48 "HTTP/1.1 200 OK"
2025-10-30 11:09:02,848 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967 "HTTP/1.1 200 OK"
2025-10-30 11:09:04,280 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:09:05,043 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b0cb2d7c-efa9-42b4-9b31-d5bc89d31e48 "HTTP/1.1 200 OK"
2025-10-30 11:09:07,671 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:09:08,131 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 1680729d-74c6-4d6b-8cc1-90e425a01cf1

2025-10-30 11:09:08,455 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967 "HTTP/1.1 200 OK"
2025-10-30 11:09:08,991 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b0cb2d7c-efa9-42b4-9b31-d5bc89d31e48 "HTTP/1.1 200 OK"
2025-10-30 11:09:09,054 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/91e7e30a-86d6-40d4-8bcc-69a156bbb967/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:09:09,460 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1680729d-74c6-4d6b-8cc1-90e425a01cf1 "HTTP/1.1 200 OK"
2025-10-30 11:09:09,759 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:09:11,809 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1680729d-74c6-4d6b-8cc1-90e425a01cf1 "HTTP/1.1 200 OK"
2025-10-30 11:09:12,791 - INFO - HTTP Request: POST
```

```
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id cd8ba7c7-5795-4aa3-be96-fe660c2ef5aa

2025-10-30 11:09:13,103 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:09:13,761 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b0cb2d7c-efa9-42b4-9b31-d5bc89d31e48 "HTTP/1.1 200 OK"
2025-10-30 11:09:14,121 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/cd8ba7c7-5795-4aa3-be96-fe660c2ef5aa "HTTP/1.1 200 OK"
2025-10-30 11:09:14,411 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b0cb2d7c-efa9-42b4-9b31-d5bc89d31e48/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:09:15,099 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:09:15,793 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1680729d-74c6-4d6b-8cc1-90e425a01cf1 "HTTP/1.1 200 OK"
2025-10-30 11:09:16,536 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/cd8ba7c7-5795-4aa3-be96-fe660c2ef5aa "HTTP/1.1 200 OK"
2025-10-30 11:09:18,547 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:09:19,485 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id fb5693fd-a2eb-4270-a409-b0008abdd3a9

2025-10-30 11:09:19,941 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/cd8ba7c7-5795-4aa3-be96-fe660c2ef5aa "HTTP/1.1 200 OK"
2025-10-30 11:09:20,248 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1680729d-74c6-4d6b-8cc1-90e425a01cf1 "HTTP/1.1 200 OK"
2025-10-30 11:09:20,478 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:09:20,865 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/fb5693fd-a2eb-4270-a409-b0008abdd3a9 "HTTP/1.1 200 OK"
2025-10-30 11:09:23,240 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/fb5693fd-a2eb-4270-a409-b0008abdd3a9 "HTTP/1.1 200 OK"
2025-10-30 11:09:23,935 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:09:26,255 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/cd8ba7c7-5795-4aa3-be96-fe660c2ef5aa "HTTP/1.1 200 OK"
2025-10-30 11:09:26,261 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1680729d-74c6-4d6b-8cc1-90e425a01cf1 "HTTP/1.1 200 OK"
2025-10-30 11:09:26,584 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:09:26,997 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/cd8ba7c7-5795-4aa3-be96-fe660c2ef5aa/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:09:27,314 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/fb5693fd-a2eb-4270-a409-b0008abdd3a9 "HTTP/1.1 200 OK"
2025-10-30 11:09:29,567 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
```

```
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:09:31,329 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"
```

```
Started parsing the file under job_id 0f0acdfa-bceb-48fb-a276-897505c07297
```

```
2025-10-30 11:09:31,723 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1680729d-74c6-4d6b-8cc1-90e425a01cf1 "HTTP/1.1 200 OK"
2025-10-30 11:09:31,725 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/fb5693fd-a2eb-4270-a409-b0008abdd3a9 "HTTP/1.1 200 OK"
2025-10-30 11:09:32,029 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:09:32,331 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/1680729d-74c6-4d6b-8cc1-90e425a01cf1/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:09:32,680 - INFO - HTTP Request: GET
https://api.cloud.llamaindex.ai/api/parsing/job/0f0acdfa-
bceb-48fb-a276-897505c07297 "HTTP/1.1 200 OK"
2025-10-30 11:09:35,301 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:09:35,404 - INFO - HTTP Request: GET
https://api.cloud.llamaindex.ai/api/parsing/job/0f0acdfa-
bceb-48fb-a276-897505c07297 "HTTP/1.1 200 OK"
2025-10-30 11:09:37,437 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:09:38,757 - INFO - HTTP Request: GET
https://api.cloud.llamaindex.ai/api/parsing/job/0f0acdfa-
bceb-48fb-a276-897505c07297 "HTTP/1.1 200 OK"
2025-10-30 11:09:39,138 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/fb5693fd-a2eb-4270-a409-b0008abdd3a9 "HTTP/1.1 200 OK"
2025-10-30 11:09:39,674 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/fb5693fd-a2eb-4270-a409-b0008abdd3a9/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:09:40,831 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
```

```
.
```

```
2025-10-30 11:09:41,548 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"
```

```
Started parsing the file under job_id 7cd8e685-deea-4afb-9b59-e78f9348fc09
```

```
2025-10-30 11:09:42,991 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66 "HTTP/1.1 200 OK"
2025-10-30 11:09:43,000 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7cd8e685-deea-4afb-9b59-e78f9348fc09 "HTTP/1.1 200 OK"
2025-10-30 11:09:43,471 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/0267fe48-ac82-4c80-9c2e-10dc6ae43f66/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:09:43,676 - INFO - HTTP Request: GET
```

```
https://api.cloud.llamaindex.ai/api/parsing/job/0f0acdfa-
bceb-48fb-a276-897505c07297 "HTTP/1.1 200 OK"
2025-10-30 11:09:46,214 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7cd8e685-deea-4afb-9b59-e78f9348fc09 "HTTP/1.1 200 OK"
2025-10-30 11:09:46,255 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:09:46,488 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id bdf465f1-10a9-439c-b7d9-c4d2c30a2032

2025-10-30 11:09:46,956 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 04333b76-4059-41a4-9a6f-59eb8fd741ee

2025-10-30 11:09:47,937 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/bdf465f1-10a9-439c-b7d9-c4d2c30a2032 "HTTP/1.1 200 OK"
2025-10-30 11:09:48,410 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/04333b76-4059-41a4-9a6f-59eb8fd741ee "HTTP/1.1 200 OK"
2025-10-30 11:09:49,655 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7cd8e685-deea-4afb-9b59-e78f9348fc09 "HTTP/1.1 200 OK"
2025-10-30 11:09:50,355 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/bdf465f1-10a9-439c-b7d9-c4d2c30a2032 "HTTP/1.1 200 OK"
2025-10-30 11:09:50,765 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/04333b76-4059-41a4-9a6f-59eb8fd741ee "HTTP/1.1 200 OK"
2025-10-30 11:09:51,272 - INFO - HTTP Request: GET
https://api.cloud.llamaindex.ai/api/parsing/job/0f0acdfa-
bceb-48fb-a276-897505c07297 "HTTP/1.1 200 OK"
2025-10-30 11:09:51,810 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:09:52,103 - INFO - HTTP Request: GET
https://api.cloud.llamaindex.ai/api/parsing/job/0f0acdfa-
bceb-48fb-a276-897505c07297/result/markdown "HTTP/1.1 200 OK"
2025-10-30 11:09:53,747 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/bdf465f1-10a9-439c-b7d9-c4d2c30a2032 "HTTP/1.1 200 OK"
2025-10-30 11:09:54,159 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/04333b76-4059-41a4-9a6f-59eb8fd741ee "HTTP/1.1 200 OK"
2025-10-30 11:09:54,606 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7cd8e685-deea-4afb-9b59-e78f9348fc09 "HTTP/1.1 200 OK"
2025-10-30 11:09:55,890 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id b9043b1b-2eea-4bf9-a317-f3eeb3ae8758

2025-10-30 11:09:57,187 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:09:57,266 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b9043b1b-2eea-4bf9-a317-f3eeb3ae8758 "HTTP/1.1 200 OK"
2025-10-30 11:09:58,136 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/bdf465f1-10a9-439c-b7d9-c4d2c30a2032 "HTTP/1.1 200 OK"
```

```
2025-10-30 11:09:58,753 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/04333b76-4059-41a4-9a6f-59eb8fd741ee "HTTP/1.1 200 OK"
2025-10-30 11:09:59,777 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b9043b1b-2eea-4bf9-a317-f3eeb3ae8758 "HTTP/1.1 200 OK"
2025-10-30 11:09:59,981 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7cd8e685-deea-4afb-9b59-e78f9348fc09 "HTTP/1.1 200 OK"
2025-10-30 11:10:02,643 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:10:03,344 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b9043b1b-2eea-4bf9-a317-f3eeb3ae8758 "HTTP/1.1 200 OK"
2025-10-30 11:10:03,622 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/bdf465f1-10a9-439c-b7d9-c4d2c30a2032 "HTTP/1.1 200 OK"
2025-10-30 11:10:04,098 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/04333b76-4059-41a4-9a6f-59eb8fd741ee "HTTP/1.1 200 OK"
2025-10-30 11:10:05,327 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7cd8e685-deea-4afb-9b59-e78f9348fc09 "HTTP/1.1 200 OK"
2025-10-30 11:10:07,744 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b9043b1b-2eea-4bf9-a317-f3eeb3ae8758 "HTTP/1.1 200 OK"
2025-10-30 11:10:08,147 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:10:09,088 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/bdf465f1-10a9-439c-b7d9-c4d2c30a2032 "HTTP/1.1 200 OK"
2025-10-30 11:10:09,712 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/04333b76-4059-41a4-9a6f-59eb8fd741ee "HTTP/1.1 200 OK"
2025-10-30 11:10:10,924 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7cd8e685-deea-4afb-9b59-e78f9348fc09 "HTTP/1.1 200 OK"
2025-10-30 11:10:13,090 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b9043b1b-2eea-4bf9-a317-f3eeb3ae8758 "HTTP/1.1 200 OK"
2025-10-30 11:10:14,251 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:10:15,167 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/04333b76-4059-41a4-9a6f-59eb8fd741ee "HTTP/1.1 200 OK"
2025-10-30 11:10:15,342 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/bdf465f1-10a9-439c-b7d9-c4d2c30a2032 "HTTP/1.1 200 OK"
2025-10-30 11:10:15,651 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/7cd8e685-deea-4afb-9b59-e78f9348fc09/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:10:15,854 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/bdf465f1-10a9-439c-b7d9-c4d2c30a2032/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:10:18,432 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b9043b1b-2eea-4bf9-a317-f3eeb3ae8758 "HTTP/1.1 200 OK"
2025-10-30 11:10:19,703 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:10:20,677 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/04333b76-4059-41a4-9a6f-59eb8fd741ee "HTTP/1.1 200 OK"
2025-10-30 11:10:20,873 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"
```

```
Started parsing the file under job_id 81ba3510-7473-4e7a-bdcd-e1e50fd7915f

2025-10-30 11:10:21,281 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/04333b76-4059-41a4-9a6f-59eb8fd741ee/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:10:22,233 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/81ba3510-7473-4e7a-bdcd-e1e50fd7915f "HTTP/1.1 200 OK"
2025-10-30 11:10:23,792 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b9043b1b-2eea-4bf9-a317-f3eeb3ae8758 "HTTP/1.1 200 OK"
2025-10-30 11:10:24,583 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/81ba3510-7473-4e7a-bdcd-e1e50fd7915f "HTTP/1.1 200 OK"
2025-10-30 11:10:24,592 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id 5c876ee7-df13-4f4b-8886-25ca5fbe7c7f

2025-10-30 11:10:25,070 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:10:26,197 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/5c876ee7-df13-4f4b-8886-25ca5fbe7c7f "HTTP/1.1 200 OK"
2025-10-30 11:10:27,734 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"
2025-10-30 11:10:27,938 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/81ba3510-7473-4e7a-bdcd-e1e50fd7915f "HTTP/1.1 200 OK"

Started parsing the file under job_id d6e9af96-a1cb-4b3a-9463-011ad68ffffb

2025-10-30 11:10:28,496 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/81ba3510-7473-4e7a-bdcd-e1e50fd7915f/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:10:28,654 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/5c876ee7-df13-4f4b-8886-25ca5fbe7c7f "HTTP/1.1 200 OK"
2025-10-30 11:10:29,167 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b9043b1b-2eea-4bf9-a317-f3eeb3ae8758 "HTTP/1.1 200 OK"
2025-10-30 11:10:29,170 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d6e9af96-a1cb-4b3a-9463-011ad68ffffb "HTTP/1.1 200 OK"
2025-10-30 11:10:29,679 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/b9043b1b-2eea-4bf9-a317-f3eeb3ae8758/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:10:30,549 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:10:31,587 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d6e9af96-a1cb-4b3a-9463-011ad68ffffb "HTTP/1.1 200 OK"
2025-10-30 11:10:31,664 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"

Started parsing the file under job_id afc0a617-253a-4b11-a261-c332f647a668

2025-10-30 11:10:31,985 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"
2025-10-30 11:10:32,082 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
```

```
ai/api/parsing/job/d6e9af96-a1cb-4b3a-9463-011ad68ffffb/result/markdown
"HTTP/1.1 200 OK"
```

```
Started parsing the file under job_id d9f1a998-33b7-44a4-844c-e018120195fd
```

```
2025-10-30 11:10:32,582 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/5c876ee7-df13-4f4b-8886-25ca5fbe7c7f "HTTP/1.1 200 OK"
2025-10-30 11:10:33,182 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/afc0a617-253a-4b11-a261-c332f647a668 "HTTP/1.1 200 OK"
2025-10-30 11:10:33,324 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d9f1a998-33b7-44a4-844c-e018120195fd "HTTP/1.1 200 OK"
2025-10-30 11:10:33,348 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/5c876ee7-df13-4f4b-8886-25ca5fbe7c7f/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:10:33,476 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"
```

```
Started parsing the file under job_id d18387c8-0704-4a35-8236-9cd2763ee184
```

```
2025-10-30 11:10:34,723 - INFO - HTTP Request: POST
https://api.cloud.llamaindex.ai/api/parsing/upload "HTTP/1.1 200 OK"
2025-10-30 11:10:34,826 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d18387c8-0704-4a35-8236-9cd2763ee184 "HTTP/1.1 200 OK"
```

```
Started parsing the file under job_id 5fae8a8b-cdd3-46e6-9471-7ed5eec60558
```

```
2025-10-30 11:10:35,618 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/afc0a617-253a-4b11-a261-c332f647a668 "HTTP/1.1 200 OK"
2025-10-30 11:10:35,729 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d9f1a998-33b7-44a4-844c-e018120195fd "HTTP/1.1 200 OK"
2025-10-30 11:10:35,889 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
```

```
.
```

```
2025-10-30 11:10:36,129 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/5fae8a8b-cdd3-46e6-9471-7ed5eec60558 "HTTP/1.1 200 OK"
2025-10-30 11:10:36,213 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/afc0a617-253a-4b11-a261-c332f647a668/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:10:37,257 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d18387c8-0704-4a35-8236-9cd2763ee184 "HTTP/1.1 200 OK"
2025-10-30 11:10:38,529 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/5fae8a8b-cdd3-46e6-9471-7ed5eec60558 "HTTP/1.1 200 OK"
2025-10-30 11:10:40,658 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d18387c8-0704-4a35-8236-9cd2763ee184 "HTTP/1.1 200 OK"
2025-10-30 11:10:41,352 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:10:42,889 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d9f1a998-33b7-44a4-844c-e018120195fd "HTTP/1.1 200 OK"
2025-10-30 11:10:43,195 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/5fae8a8b-cdd3-46e6-9471-7ed5eec60558 "HTTP/1.1 200 OK"
```

```
2025-10-30 11:10:43,811 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/5fae8a8b-cdd3-46e6-9471-7ed5eec60558/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:10:45,142 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d18387c8-0704-4a35-8236-9cd2763ee184 "HTTP/1.1 200 OK"
2025-10-30 11:10:46,780 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:10:47,293 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d9f1a998-33b7-44a4-844c-e018120195fd "HTTP/1.1 200 OK"
2025-10-30 11:10:51,593 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d18387c8-0704-4a35-8236-9cd2763ee184 "HTTP/1.1 200 OK"
2025-10-30 11:10:53,139 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d18387c8-0704-4a35-8236-9cd2763ee184/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:10:53,949 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:10:54,318 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d9f1a998-33b7-44a4-844c-e018120195fd "HTTP/1.1 200 OK"
2025-10-30 11:10:59,291 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:11:01,528 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d9f1a998-33b7-44a4-844c-e018120195fd "HTTP/1.1 200 OK"
2025-10-30 11:11:02,551 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/d9f1a998-33b7-44a4-844c-e018120195fd/result/markdown
"HTTP/1.1 200 OK"
2025-10-30 11:11:04,630 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:11:10,599 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:11:16,991 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:11:22,944 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:11:29,186 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:11:35,628 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
```

.

```
2025-10-30 11:11:41,585 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:11:47,704 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:11:53,959 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:12:04,243 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:12:10,326 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
```

```
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:12:16,283 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:12:23,451 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:12:29,596 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:12:35,939 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:12:42,089 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"

.

2025-10-30 11:12:48,233 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:12:54,242 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:13:00,318 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:13:06,666 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:13:12,748 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:13:18,822 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:13:24,784 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:13:30,937 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:13:37,384 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:13:43,430 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"

.

2025-10-30 11:13:49,574 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:13:56,147 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:14:02,134 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980 "HTTP/1.1 200 OK"
2025-10-30 11:14:06,527 - INFO - HTTP Request: GET https://api.cloud.llamaindex.
ai/api/parsing/job/3d6cf921-db75-4946-bc6a-4f19bd122980/result/markdown
"HTTP/1.1 200 OK"

Parsed 887 nodes
Sample:
# Form 1040
```

```
## U.S. Individual Income Tax Return 2016
Department of the Treasury-Internal Revenue Service
OMB No. 1545-0074
IRS Use Only-Do not write or staple in this space.

For the year Jan. 1-Dec. 31, 2016, or other tax year beginning ________, 2016,
ending ________, 20__
See separate instructions.

----

### Your first name and initial
Henrietta

### Last name
Homeowner
```

[8]:
```python
print("Sample2:", documents[-1].text[:400])
```

```
Sample2:
# DANI MARTINEZ
## COPYWRITING

Id card  : 123-456-7890
Email    : hello@reallygreatsite.com
Address  : 123 Anywhere St., Any City
Phone    : 123-456-7890

www.reallygreatsite.com

Studio Shodwe
```

[42]:
```python
def ext_to_source_type(ext: str) -> str:
    e = ext.lower()
    if e in {".pdf"}: return "pdf"
    if e in {".docx", ".doc"}: return "docx"
    if e in {".xlsx", ".xls"}: return "xlsx"
    if e in {".jpg", ".jpeg", ".png", ".tiff", ".tif"}: return "image"
    if e in {".pptx"}: return "pptx"
    if e in {".html", ".htm"}: return "html"
    return "other"

PARSE_CONFIG = {
    "parser": "LlamaParse",
    "parse_mode": "parse_page_with_agent",
    "model": "openai-gpt-4-1-mini",
    "high_res_ocr": True,
    "adaptive_long_table": True,
    "outlined_table_extraction": True,
    "output_tables_as_HTML": True,
    "result_type": "markdown",
}

documents_with_meta = []
by_source = {}

for path, docs in zip(paths, results):
    file_name = os.path.basename(path)
    source_type = ext_to_source_type(os.path.splitext(file_name)[1])

    # store grouping for later logical-doc work
    by_source[path] = docs

    for page_idx, d in enumerate(docs):
        md = d.metadata or {}
        md.update({
            "source_path": path,
            "file_name": file_name,
            "source_type": source_type,
            "page_index": page_idx,
            **PARSE_CONFIG,
        })
        d.metadata = md
        documents_with_meta.append(d)

# Replace your flat list if you want
documents = documents_with_meta

print(len(documents), documents[0].metadata)
```

```
887 {'source_path': 'train/blob_scanned_id_pay_return.pdf', 'file_name':
'blob_scanned_id_pay_return.pdf', 'source_type': 'pdf', 'split': 'train',
'page_index': 0, 'parser': 'LlamaParse', 'parse_mode': 'parse_page_with_agent',
'model': 'openai-gpt-4-1-mini', 'high_res_ocr': True, 'adaptive_long_table':
True, 'outlined_table_extraction': True, 'output_tables_as_HTML': True,
'result_type': 'markdown'}
```

[63]:
```
import pickle
with open('documents_train.pkl', 'wb') as file:
    pickle.dump(documents, file)
```

[187]:
```
def extract_agent_text(response):
    out = response.get("output") or []
    if isinstance(out, list) and out and isinstance(out[0], dict):
        txt = out[0].get("text")
        if isinstance(txt, str):
            return txt.strip()
    return str(response)

def rag_doctype_classifier(agent_id = '7c8cb2ad-7b1b-4396-892c-17a01807cf0b',␣
 ↪input_text = None, stream = False, timeout = 20):
    url = "https://api.you.com/v1/agents/runs"
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer {you_key}",
    }
    payload = {
        "agent": agent_id,
        "input": input_text,
        "stream": stream
    }
    resp = requests.post(url, json=payload, headers=headers, timeout=timeout)
    resp.raise_for_status()
    return extract_agent_text(resp.json())

def rag_page_continue(agent_id = 'c3123621-1eb5-478d-a671-52ab0980c9e7',␣
 ↪input_text = None, stream = False, timeout = 20):
    url = "https://api.you.com/v1/agents/runs"
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer {you_key}",
    }
    payload = {
        "agent": agent_id,
        "input": input_text,
        "stream": stream
    }
    resp = requests.post(url, json=payload, headers=headers, timeout=timeout)
    resp.raise_for_status()
    return extract_agent_text(resp.json())

def rag_query_router(agent_id = 'b2c1b36e-4da0-481f-a5ef-931b5cf60a1c',␣
 ↪input_text = None, stream = False, timeout = 20):
    url = "https://api.you.com/v1/agents/runs"
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer {you_key}",
    }
    payload = {
        "agent": agent_id,
        "input": input_text,
        "stream": stream
    }
    resp = requests.post(url, json=payload, headers=headers, timeout=timeout)
    resp.raise_for_status()
    return extract_agent_text(resp.json())
```

[188]:
```
rag_query_router(input_text= "how to make a cake?")
```

[188]:
```
'Other'
```

[183]:
```
rag_doctype_classifier(input_text=documents[2].text)
```

[183]:
```
'ID'
```

[97]:
```
documents[2].text
```

[97]:
```
'\n\n# KANSAS DRIVER\'S LICENSE\n\n**USA KS**\n\n| Field           | Information
|\n|-----------------|------------------------|\n| LIC. NO.        | K12-34-5678
|\n| DOB             | 01/11/1966             |\n| ISS             | 01/11/2017
|\n| EXP             | 01/11/2023             |\n| Name            | SAMPLE
|\n|                 | CARON ELIZABETH        |\n| Address         | 123 NORTH
STREET      |\n|                 | APT. 2                 |\n|                 |
TOPEKA, KS 66612-1234 |\n| CLASS          | A                      |\n| END
| NONE                  |\n| SEX            | F                      |\n| REST
| NONE                  |\n| HGT            | 5\'-06"                 |\n| WGT
| 140 lb                |\n| EYES           | BRO                    |\n| DD
| XX123XWMXX1           |\n| DONOR          | [x] DONOR              |\n|
Additional DD  | 23XWMX123XWM12        |\n\n**Signature:**  \n`Caron
Sample`\n\n\n'
```

[128]:
```
#logical document creation

metadata = []
current_doc_type = None
page_in_doc = 0
is_new_doc = True
logical_documents = []

for i, page in enumerate(documents):
    if i == 0:
        current_doc_type = rag_doctype_classifier(input_text = page.text)
        text = page.text
    else:
        prev_text = documents[i - 1].text

        output = rag_page_continue(input_text= f"""
        Page 1: {prev_text[:4000]}
        Doctype Page 1: {current_doc_type}
        Page 2: {page.text[:4000]}""")

        if output.startswith('y'):
            current_doc_type = rag_doctype_classifier(input_text = page.text)
            page_in_doc = 0
            text = page.text
            is_new_doc = True

        else:
            page_in_doc += 1
            text = text + "\n\n" + page.text
            is_new_doc = False
    if ((i+1) % 100) == 0:
        print(f'{i+1} Documents Processed')
    metadata.append({
        "page": i,
        'is_new_doc': is_new_doc,
        "doc_type": current_doc_type,
        'page_in_doc': page_in_doc,
        'file_name': page.metadata['file_name'],
    })

    if is_new_doc:
        logical_documents.append({
            'text': text,
            'doc_type': current_doc_type,
            'page_start': i,
            'page_end': i
        })
    else:
        logical_documents[-1]['page_end'] = i
        logical_documents[-1]['text'] = text
```

```
100 Documents Processed
200 Documents Processed
300 Documents Processed
400 Documents Processed
500 Documents Processed
600 Documents Processed
700 Documents Processed
800 Documents Processed
```

[133]:
```
metadata[:2]
```

```
[133]: [{'page': 0,
         'is_new_doc': True,
         'doc_type': 'TaxDocument',
         'page_in_doc': 0,
         'file_name': 'blob_scanned_id_pay_return.pdf'},
        {'page': 1,
         'is_new_doc': False,
         'doc_type': 'TaxDocument',
         'page_in_doc': 1,
         'file_name': 'blob_scanned_id_pay_return.pdf'}]
```

```
[129]: with open('metadata_train.pkl', 'wb') as file:
           pickle.dump(metadata, file)

       with open('logical_train.pkl', 'wb') as file:
           pickle.dump(logical_documents, file)

       print('Number of Logical Documents: ', len(logical_documents))
```

```
Number of Logical Documents:  630
```

1.2 RAG Pipeline

Now we have our 630 Logical Documents ready. We will use chunking on them before creating a FAISS vector store index and hybrid retriever powered by metadata filtering and query expansion. We then created many different engines based on the different doc types and a global fallback to make a subquestion query engine which is able to read in complex queries, break them apart and send them into different query engines, and then synthesize the results.

```
[130]: from llama_index.core import Settings, Document, VectorStoreIndex
       from llama_index.llms.gemini import Gemini
       from llama_index.embeddings.huggingface import HuggingFaceEmbedding

       # LLM: Gemini for RAG generation
       llm = Gemini(
           model = os.getenv("GEMINI_MODEL", "gemini-2.5-flash"),
           api_key = gemini_key,
           temperature = 0.1,
           max_tokens = 512,
       )
       Settings.llm = llm

       embed = HuggingFaceEmbedding(model_name="BAAI/bge-large-en-v1.5")

       Settings.embed_model = embed
```

```
/tmp/ipykernel_16361/2000535101.py:6: DeprecationWarning: Call to deprecated
class Gemini. (Should use `llama-index-llms-google-genai` instead, using
Google's latest unified SDK. See:
https://docs.llamaindex.ai/en/stable/examples/llm/google_genai/)
  llm = Gemini(
2025-10-30 23:57:16,316 - INFO - Load pretrained SentenceTransformer: BAAI/bge-
large-en-v1.5

modules.json:   0%|          | 0.00/349 [00:00<?, ?B/s]

config_sentence_transformers.json:   0%|          | 0.00/124 [00:00<?, ?B/s]

README.md: 0.00B [00:00, ?B/s]

sentence_bert_config.json:   0%|          | 0.00/52.0 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/779 [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/1.34G [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/366 [00:00<?, ?B/s]

vocab.txt: 0.00B [00:00, ?B/s]

tokenizer.json: 0.00B [00:00, ?B/s]

special_tokens_map.json:   0%|          | 0.00/125 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/191 [00:00<?, ?B/s]

2025-10-30 23:58:07,363 - INFO - 1 prompt is loaded, with the key: query
```

[141]:

```python
#recursive chunking

from langchain.text_splitter import RecursiveCharacterTextSplitter

splitter = RecursiveCharacterTextSplitter(chunk_size = 1600, chunk_overlap =␣
 ↪150)

chunked_documents = []

for idx, doc in enumerate(logical_documents):
    chunks = splitter.split_text(doc["text"])
    for chunk_idx, chunk in enumerate(chunks):
        chunked_documents.append(
            Document(
                text=chunk,
                metadata={
                    "doc_type": doc["doc_type"],
                    "chunk_index": chunk_idx,
                    "page_start": doc["page_start"],
                    "page_end": doc["page_end"],
                }
            )
        )
```

[166]:
```
chunked_documents[-1].metadata
```

[166]:
```
{'doc_type': 'FinancialModel',
 'chunk_index': 34,
 'page_start': 878,
 'page_end': 886}
```

[146]:
```
#semantic chunking

from llama_index.core.node_parser import SemanticSplitterNodeParser

semantic_splitter = SemanticSplitterNodeParser(
    buffer_size = 15,                      # keeps neighboring sentences in␣
 ↪context when deciding
    breakpoint_percentile_threshold = 90,  # higher = fewer, stronger splits
    embed_model = Settings.embed_model
)

# Convert coarse nodes back to Documents for the semantic splitter
semantic_chunks = semantic_splitter.get_nodes_from_documents(chunked_documents)
```

[147]:
```
print(len(semantic_chunks))
```

```
2558
```

[167]:
```
semantic_chunks[-1].metadata
```

[167]:
```
{'doc_type': 'FinancialModel',
 'chunk_index': 34,
 'page_start': 878,
 'page_end': 886}
```

[171]:
```
#scalable faiss store

from llama_index.vector_stores.faiss import FaissVectorStore
import faiss
from llama_index.vector_stores.faiss import FaissVectorStore
from llama_index.core import StorageContext

probe = Settings.embed_model.get_text_embedding("dimension probe")
dim = len(probe)

faiss_index = faiss.IndexFlatL2(dim)
vector_store = FaissVectorStore(faiss_index = faiss_index)

#Build a VectorStoreIndex over semantic chunks
storage_context = StorageContext.from_defaults(vector_store = vector_store)
```

```
index = VectorStoreIndex(semantic_chunks, storage_context=storage_context)
```

```
2025-10-31 00:35:41,581 - INFO - Loading faiss with AVX512 support.
2025-10-31 00:35:41,649 - INFO - Successfully loaded faiss with AVX512 support.
```

[189]:
```
dim
```

[189]: 1024

[193]:
```
#loading additonal libraries
from llama_index.core import Settings, get_response_synthesizer
from llama_index.core.retrievers import VectorIndexRetriever,␣
 ↪QueryFusionRetriever
from llama_index.retrievers.bm25 import BM25Retriever
from llama_index.core.query_engine import RetrieverQueryEngine,␣
 ↪SubQuestionQueryEngine
from llama_index.core.postprocessor import SentenceTransformerRerank
from llama_index.core.tools import QueryEngineTool
from llama_index.core.postprocessor.types import BaseNodePostprocessor
from collections import defaultdict
from llama_index.core.question_gen import LLMQuestionGenerator
```

[290]:
```
from typing import Optional
```

[321]:
```
all_nodes = list(index.docstore.docs.values())

#BM25 retriever
bm25_all = BM25Retriever.from_defaults(nodes=all_nodes, similarity_top_k=12)

class SoftDocTypeFilterPostprocessor(BaseNodePostprocessor):
    doc_type: Optional[str] = None
    keep_top_n: int = 8
    min_filtered: int = 2

    def _postprocess_nodes(self, nodes, query_bundle=None):
        if not nodes or not self.doc_type:
            return nodes[: self.keep_top_n]

        # preserve fused order
        filtered = [n for n in nodes if (n.node.metadata or {}).get("doc_type")␣
 ↪== self.doc_type]
        if len(filtered) >= self.min_filtered:
            return filtered[: self.keep_top_n]

        # Not enough filtered hits -> mix: filtered first, then best␣
 ↪off-doctype to fill
        taken_ids = {n.node.node_id for n in filtered}
        mixed = list(filtered)
        for n in nodes:
            if n.node.node_id in taken_ids:
                continue
            mixed.append(n)
            if len(mixed) >= self.keep_top_n:
                break
        return mixed


nodes_by_dt = defaultdict(list)
for n in all_nodes:
    dt = (n.metadata or {}).get("doc_type", "Unknown")
    nodes_by_dt[dt].append(n)

bm25_by_dt = {
    dt: BM25Retriever.from_defaults(nodes=nodes, similarity_top_k=12)
    for dt, nodes in nodes_by_dt.items() if nodes
}

#  (vector + BM25 + filter + reranker)
def make_engine_for_doctype(predicted_doc_type=None, k_per=12, final_top_n=5):

    vec = index.as_retriever(similarity_top_k=k_per)

    bm25 = bm25_by_dt.get(predicted_doc_type, bm25_all)

    bm25 = bm25_by_dt.get(predicted_doc_type, bm25_all)
    if predicted_doc_type in nodes_by_dt and␣
 ↪len(nodes_by_dt[predicted_doc_type]) < 100:
        bm25 = bm25_all

    # Fuse vector + BM25; set small num_queries for light LLM expansion
    fusion = QueryFusionRetriever(
        retrievers = [vec, bm25],
        llm = Settings.llm,
        similarity_top_k = k_per,
        num_queries = 2,
        mode = "reciprocal_rerank",
    )

    reranker = SentenceTransformerRerank(
        model = "cross-encoder/ms-marco-MiniLM-L-2-v2",
        top_n = final_top_n,
    )

    post = []
```

```python
    if predicted_doc_type and predicted_doc_type.lower() != "other":
        post.append(SoftDocTypeFilterPostprocessor(
            doc_type = predicted_doc_type,
            keep_top_n = final_top_n,
            min_filtered = 2,
        ))
    post.append(reranker)


    resp_synth = get_response_synthesizer(response_mode = "compact")

    return RetrieverQueryEngine.from_args(
        retriever = fusion,
        llm = Settings.llm,
        response_synthesizer = resp_synth,
        node_postprocessors = post,
        verbose = False,
    )

def get_engine_for_query(query):
    label = rag_query_router(input_text=query)
    predicted_doc_type = None
    if label and label.strip().lower() != "other":
        predicted_doc_type = label.strip()
    return make_engine_for_doctype(predicted_doc_type=predicted_doc_type)

tools = []
seen = set((n.metadata or {}).get("doc_type", "Unknown") for n in all_nodes)
for dt in sorted(seen):
    if not dt or dt.lower() == "unknown":
        continue
    qe_dt = make_engine_for_doctype(predicted_doc_type=dt)
    tools.append(
        QueryEngineTool(
            query_engine=qe_dt,
            metadata=ToolMetadata(
                name=f"engine_{dt}",
                description=f"Answer questions about {dt} documents",
            ),
        )
    )

# Global fallback tool (no doctype restriction)
qe_global = make_engine_for_doctype(predicted_doc_type=None)
tools.append(
    QueryEngineTool(
        query_engine=qe_global,
        metadata=ToolMetadata(
            name="engine_global",
            description="Use when the doctype is unclear or when␣
 ↪doctype-specific context seems missing which happens when metadata tagging␣
 ↪is wrong.",
        ),
    )
)

#main engine
question_gen = LLMQuestionGenerator.from_defaults(llm = Settings.llm)
qe_main = SubQuestionQueryEngine.from_defaults(
    query_engine_tools = tools,
    question_gen = question_gen,
    llm = Settings.llm,
    verbose = False,
    use_async = False,
)
```

```
2025-10-31 11:34:26,334 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,415 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,421 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,431 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,503 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,622 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,665 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,707 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,724 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,740 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,752 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,757 - DEBUG - Building index from IDs objects
2025-10-31 11:34:26,759 - WARNING - As bm25s.BM25 requires k less than or equal
to number of nodes added. Overriding the value of similarity_top_k to number of
nodes added.
2025-10-31 11:34:26,779 - DEBUG - Building index from IDs objects
```

1.3 Query Testing

In this part we will test 15 queries we made before training this pipeline. It is a range of queries mirroring the needs of finance/legal professionals in their day to day tasks as it cover field extraction, summarization, and complex questions centered around legal documents and excel workings.

[208]:
```
#test query
response = qe_main.query("What is the closing balance for John Doe's Bank␣
 ↪Statement?")
print(response)
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]
```

```
The closing balance for John Doe's bank statement is 17.03.
```

[228]:
```
#another test
import time
start_time = time.time()

query = "What are the main numerical figures I should know in the ACC Limited␣
↪revenue model and its workings? And Extract top 5 most important figures."
response = qe_main.query(query)

elapsed_time = time.time() - start_time

raw = str(response)

print('\nFinal Response:\n ---------------------- \n')
print(raw)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

The main numerical figures in the ACC Limited revenue model are:

*   **Cement Sales - Value in cr:**
    *   2015: 11,917
    *   2016: 11,112
    *   2017: 13,414
    *   2018 F: 14,441
    *   2019 F: 15,537
    *   2020 F: 16,707
    *   2021 F: 17,955
    *   2022 F: 19,287
*   **Cement Sales - Volume Million Tonnes:**
    *   2015: 23.62
    *   2016: 22.99
    *   2017: 26.21
    *   2018 F: 26.87
    *   2019 F: 27.54
    *   2020 F: 28.20
    *   2021 F: 28.86
    *   2022 F: 29.53
*   **Average price in crores per Million Tonnes:**
    *   2015: 505
    *   2016: 483
    *   2017: 512
    *   2018 F: 537
    *   2019 F: 564
    *   2020 F: 592
    *   2021 F: 622
    *   2022 F: 653
*   **Average price per tonne:**
    *   2015: 5,045
    *   2016: 4,833
    *   2017: 5,118
    *   2018 F: 5,374
    *   2019 F: 5,642
    *   2020 F: 5,924
    *   2021 F: 6,221
    *   2022 F: 6,532
*   **Average price per kg:**
    *   2015: 5.0
    *   2016: 4.8
    *   2017: 5.1
    *   2018 F: 5.4
    *   2019 F: 5.6
    *   2020 F: 5.9
    *   2021 F: 6.2
    *   2022 F: 6.5

Information regarding the main numerical figures in the ACC Limited revenue
model workings is not available.

The top 5 most important figures from the ACC Ltd revenue model are:

*   Cement Sales - Value in cr
*   Cement Sales - Volume Million Tonnes
*   Average price in crores per Million Tonnes
*   Capacity Million Tonnes
*   Capacity utilisation

The top 5 most important figures from the ACC Limited revenue model workings are
not available.

Query execution time: 43.139 seconds
```

```
[ ]:
```

[276]:
```python
#another test
import time
start_time = time.time()

query = "Extract driver licence details for Caron Elizabeth"
response = qe_main.query(query)

elapsed_time = time.time() - start_time

print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

Here are the driver's license details for Caron Elizabeth:

*   **License Number:** K12-34-5678
*   **Date of Birth (DOB):** 01/11/1966
*   **Issue Date (ISS):** 01/11/2017
*   **Expiration Date (EXP):** 01/11/2023
*   **Name:** SAMPLE CARON ELIZABETH
*   **Address:** 123 NORTH STREET, APT. 2, TOPEKA, KS 66612-1234
*   **Class:** A
*   **Endorsements (END):** NONE
*   **Sex:** F
*   **Restrictions (REST):** NONE
*   **Height (HGT):** 5'-06"
*   **Weight (WGT):** 140 lb
*   **Eyes:** BRO
*   **DD:** XX123XWMXX1
*   **Donor Status:** DONOR
*   **Additional DD:** 23XWMX123XWM12

Query execution time: 17.792 seconds
```

[292]:
```python
#another test, this time for a query not solved by RAG properly
import time
start_time = time.time()

query = "Extract revenue fields in the income statement for XYZ Limited."
response = qe_main.query(query)

elapsed_time = time.time() - start_time
```

```
print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

The revenue fields in the income statement for XYZ Limited are:
*   Fair value of milk produced
*   Gains arising from changes in fair value less estimated point-of-sale costs
of dairy livestock
*   Total Income

Query execution time: 15.610 seconds
```

[298]:
```
#another test, this time for a query not solved by RAG properly
import time
start_time = time.time()

query = "What are the top revenue fields and their values in March 2022 for the␣
 ↪Interglobe Aviation as per the Revenue/Financial Model?"
response = qe_main.query(query)

elapsed_time = time.time() - start_time

print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

In March 2022, the top revenue fields for Interglobe Aviation and their values
were:

*   **Passenger ticket revenue:** 450,243 Million INR
*   **Ancillary Revenue:** 56,280 Million INR

Query execution time: 17.992 seconds
```

[302]:
```python
#another test
import time
start_time = time.time()

query = "What is the total estimated monthly payment for the property mortgage␣
    ↪as per lender fees worksheet? Break down its key compnents too."
response = qe_main.query(query)

elapsed_time = time.time() - start_time

print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

The total estimated monthly payment for the property mortgage is $2,308.95. This
payment is comprised of the following key components:

*   Principal and interest: $1,869.37
*   Hazard insurance: $39.58
*   Real estate taxes: $400.00

Query execution time: 28.408 seconds
```

[303]:
```python
#another test
import time
start_time = time.time()

query = "Summarize Nvidia's financial results for second quarter of 2026."
response = qe_main.query(query)

elapsed_time = time.time() - start_time

print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------
```

```
NVIDIA reported second-quarter revenue of $601 million, an 18% increase from the
prior quarter and a 32% rise compared to the same period last year. Automotive
revenue specifically reached $586 million, showing a 3% sequential increase and
a 69% year-over-year growth. The company's net income for the quarter was
$25,783 million, resulting in diluted earnings per share of $1.05. When
excluding H20 related charges/releases, net, and their associated tax impact,
diluted earnings per share stood at $1.04.

Query execution time: 10.297 seconds
```

[315]:
```python
#another test
import time
start_time = time.time()

query = "What are the scope and requirements of ISA 220 Quality Management?"
response = qe_main.query(query)

elapsed_time = time.time() - start_time

print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

ISA 220 (Revised), *Quality Management for an Audit of Financial Statements*,
outlines the specific responsibilities of the auditor regarding quality
management at the engagement level for an audit of financial statements. It also
addresses the related responsibilities of the engagement partner. Furthermore,
this standard covers the auditor’s responsibilities concerning relevant ethical
requirements, including those pertaining to independence, specifically within
the context of accepting an audit engagement and for matters under the auditor's
control.

Query execution time: 11.595 seconds
```

[322]:
```python
#another test
import time
start_time = time.time()

query = "What are the earnings and Net Pay of James Bond as per his payslip?"
response = qe_main.query(query)
```

```
elapsed_time = time.time() - start_time

print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

James Bond's total earnings are 8800, which comprises a Basic Pay of 8000, an
Allowance of 500, and Overtime of 300. His Net Pay is 8000.

Query execution time: 24.044 seconds
```

[323]:
```
#another test
import time
start_time = time.time()

query = "What are the conditions for supply of services as per Purchase Order␣
 ↪Agreement of Great Ocean Road?"
response = qe_main.query(query)

elapsed_time = time.time() - start_time

print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

The supply of services is governed by the Terms and Conditions outlined in the
official Purchase Order issued by the Great Ocean Road Coast and Parks
Authority, which also details the applicable General Conditions.

Regarding the performance of these services, the provider is responsible for
supplying all necessary equipment. If the services are not delivered in
accordance with the agreement, the Authority is not obligated to make payment
until the services are correctly rendered. In such instances, the Authority may
require the supplier to remedy any default or re-perform the services within a
reasonable period. Should the default be irremediable, the services cannot be
```

```
re-performed, or if the supplier fails to act within the specified timeframe,
the Authority reserves the right to arrange for a third party to remedy or re-
perform the services, or to undertake the task itself. In these situations, the
supplier will be held responsible for any reasonable costs incurred.

Query execution time: 17.000 seconds
```

[334]:
```python
#another test
import time
start_time = time.time()

query = "Summarize  Representations, Warranties and Additional Covenants in␣
  ↪contractor service contract for Mercy Corps."
response = qe_main.query(query)

elapsed_time = time.time() - start_time

print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

The Contractor assures Mercy Corps that it has the complete rights and authority
to enter into and perform its contractual obligations, and that its performance
will not violate any agreements with third parties. Furthermore, the Contractor
guarantees that it possesses the required skills to execute the Services as
specified in the Statement of Work.

Query execution time: 9.280 seconds
```

[338]:
```python
#another test
import time
start_time = time.time()

query = "What is the total amount payable to CPB Software as per invoices in␣
  ↪2024?"
response = qe_main.query(query)

elapsed_time = time.time() - start_time

print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

The total amount payable to CPB Software for invoices in 2024 is 251.12 €. This
amount includes charges for user-account-1 totaling 154.30 € and user-account-2
totaling 96.82 € for the period 01.02.2024 to 29.02.2024.

Query execution time: 21.806 seconds
```

[345]:
```
#another test
import time
start_time = time.time()

query = "How many units were sold for Amarilla product in 2014? Give a total␣
  ↪amount, and then break down the total amount into units sold per segment for␣
  ↪Amarilla for the same period."
response = qe_main.query(query)

elapsed_time = time.time() - start_time

print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

A total of 49,117.5 units of the Amarilla product were sold in 2014.

The breakdown of units sold per segment for the Amarilla product in 2014 is as
follows:
*   **Government:** 22,124 units
*   **Small Business:** 4,467 units
*   **Midmarket:** 1,326 units
*   **Channel Partners:** 4,504 units
*   **Enterprise:** 7,523.5 units

Query execution time: 39.275 seconds
```

[347]:
```
#another test
import time
```

```
start_time = time.time()

query = "Explain Article 7 and 8 mentioned in the format of contract agreement."
response = qe_main.query(query)

elapsed_time = time.time() - start_time

print('\nFinal Response:\n ----------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 -----------------------

Article 7 establishes a limitation of personal liability for individuals
associated with the Owner. It specifies that no director, employee, consultant,
or agent of the Owner, or any person representing or acting on behalf of the
Owner in connection with the contract, shall have any personal liability to the
Contractor or any of its sub-contractors, agents, representatives, directors, or
employees. Furthermore, the Contractor, on its own behalf and on behalf of its
sub-contractors, directors, employees, agents, and representatives, waives and
disclaims any and all rights of action it or they may have, whether under tort
or contract or otherwise, against the Owner or any director, employee, agent,
consultant, or representative of the Owner for any act of omission or commission
done or omitted to be done.

Article 8 clarifies that any failure or delay by the Owner in enforcing a right,
remedy, obligation, or liability under the contract does not constitute a waiver
of that right, remedy, obligation, or liability. The Owner retains the
entitlement to enforce such provisions at any time, regardless of prior inaction
or delay.

Query execution time: 27.176 seconds
```

[351]:
```
#another test
import time
start_time = time.time()

query = "Summarize the IAS 16."
response = qe_main.query(query)

elapsed_time = time.time() - start_time
```

```
print('\nFinal Response:\n ---------------------- \n')
print(response)
print(f"\nQuery execution time: {elapsed_time:.3f} seconds")
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]


Final Response:
 ----------------------

IAS 16, effective from January 1, 1995, provides guidance on Property, Plant and
Equipment. It mandates that these assets should be recognized when it is
probable that future economic benefits will flow from them and their cost can be
reliably measured. Initial measurement of these assets is required to be at
cost.

The standard has undergone several amendments since its inception. It was
amended by IAS 1 in July 1997, and further revised in April and July 1998 to
ensure consistency with IAS 22, IAS 36, and IAS 37, with this revised version
becoming operative for periods beginning on or after July 1, 1999. Its scope was
also adjusted by IAS 40 in April 2000 and by IAS 41 in January 2001.

Additionally, two SIC Interpretations are relevant to IAS 16: SIC 14, which
addresses compensation for the impairment or loss of items, and SIC 23,
concerning major inspection or overhaul costs. Entities applying the cost model
for investment property under IAS 40 are required to furnish all the disclosure
requirements specified by IAS 16.

Query execution time: 13.898 seconds
```

[350]:
```
for i in range(len(all_nodes)):
    if 'ARTICLE 8' in all_nodes[i].text:
        print(all_nodes[i].get_content)
        print('\n')
```

```
<bound method TextNode.get_content of
TextNode(id_='db62d004-20d7-4214-8f7e-b51fba1dc4f8', embedding=None,
metadata={'doc_type': 'EmployeeContract', 'chunk_index': 4, 'page_start': 8,
'page_end': 10}, excluded_embed_metadata_keys=[], excluded_llm_metadata_keys=[],
relationships={<NodeRelationship.SOURCE: '1'>:
RelatedNodeInfo(node_id='f85a8d89-3f81-4ea1-af90-4be740fcc3ef', node_type='4',
metadata={'doc_type': 'EmployeeContract', 'chunk_index': 4, 'page_start': 8,
'page_end': 10},
hash='6c7dbc45cbc60e00df04db2f788962bdbb791651b909f32a9dc6e2b514511cff')},
metadata_template='{key}: {value}', metadata_separator='\n', text='### ARTICLE
6. Appendices\n\nThe Appendices listed in the attached list of Appendices shall
be deemed to form an integral part of this Contract  \nAgreement.  \nReference
in the Contract to any Appendix shall mean the Appendices attached hereto, and
the Contract shall be  \nread and construed accordingly.\n\n### ARTICLE 7. NO
LIABILITY ON DIRECTOR AND EMPLOYEE\n\nNo Director, employee, consultant or agent
of the OWNER or other person representing the  \nOWNER or acting on behalf of
the OWNER in or pursuant to the Contract or in the discharge of any obligation
to the  \nOWNER under the Contract or otherwise in relation to the Contract
shall have any personal liability to the  \nCONTRACTOR or any Sub-Contractor,
agent, representative, director or employee of the CONTRACTOR or to any  \nother
person acting for or on behalf of the CONTRACTOR and the CONTRACTOR on its own
behalf and on behalf of  \nits Sub Contractors, directors, employees, agents and
representatives hereby waives and disclaims any and all right  \nof action which
it or they may have whether under tort or Contract or otherwise against the
OWNER or any director,  \nemployee, agent, consultant or  \nrepresentative of
the OWNER for act of omission or commission done or omitted to be done.\n\n###
ARTICLE 8. WAIVER\n\nNo failure or delay by the OWNER in enforcing any right or
remedy of the OWNER in terms of the CONTRACT or any  \nobligation or liability
of the CONTRACTOR in terms thereof, shall be deemed to be a waiver of such
right, remedy,', mimetype='text/plain', start_char_idx=0, end_char_idx=1501,
metadata_seperator='\n', text_template='{metadata_str}\n\n{content}')>
```

1.4 Evaluation

[368]:
```python
gold_ids_by_query = {
    "What is the closing balance for John Doe's Bank Statement?": {
        "3c1f4f03-9757-493c-b0b8-c7d3650d6768",
    },
    "What are the main numerical figures I should know in the ACC Limited␣
↪revenue model and its workings? And Extract top 5 most important figures.": {
        "1f8dc041-1b55-4c40-bea2-170d7f9b97da",
    },
    "Extract driver licence details for Caron Elizabeth": {
        "4889f6a7-7ada-41da-961d-d781d0cf4946",
    },
    "What are the scope and requirements of ISA 220 Quality Management?": {
        "d09b3c01-33f9-4a7c-a08f-267d7282df47",
        "1a620326-d1a3-45d8-aedf-919638fc88af",
        "28b0ce01-6f42-4b5b-9939-ce8ff156b063",
    },
    "Extract revenue fields in the income statement for XYZ Limited.": {
        "7966b441-d0d2-4198-a426-a18fc6ea90dc",
    },
    "What are the top revenue fields and their values in March 2022 for the␣
↪Interglobe Aviation as per the Revenue/Financial Model?": {
        "80002318-d430-43f7-94d3-6ef5140ba3a1",
    },
    "What is the total estimated monthly payment for the property mortgage as␣
↪per lender fees worksheet? Break down its key compnents too.": {
        "fe2db0d8-ec2e-457c-86c6-a3dfe278f317",
        "4bf297dd-5c8b-4b51-bd48-e756809749c4",
    },
    "Summarize Nvidia's financial results for second quarter of 2026.": {
        "472095f3-d185-495a-86f6-fd5ac7493f01",
        "29e49d15-f434-4101-9c28-a2dd21d4a62d",
        "edd15543-65f0-41a6-8663-35ac8338f255",
    },
    "What are the earnings and Net Pay of James Bond as per his payslip?": {
        "83fd248a-3c02-4d20-a3c8-ed69db399c1d",
        "dec5a78a-9f08-43a9-ac5d-ca02085ac040",
    },
    "What are conditions for supply of services as per PO Agreement of Great␣
↪Ocean Road?": {
        "cdec08c4-118f-47c7-815d-c5ec58dc4446",
        "9223d137-7ad4-4373-aa2e-25b9dfeef6f9",
    },
    "Summarize  Representations, Warranties and Additional Covenants in␣
↪contractor service contract for Mercy Corps.": {
        "15edf483-c3d5-41be-83f2-047e99709169",
        "7dd5bd88-5e70-4832-a645-42b14fff94bd",
        "bb2c0831-8bd1-4aa0-9523-eca534d4983b",
    },
    "What is the total gross amount payable including taxes to CPB software as␣
↪per invoices in 2024?": {
        "9b2881ea-afb2-469a-ba81-40d2bba08e87",
        "142acf58-50c5-4541-9e21-497f6567c557",
    },
    "Summarize the IAS 16.": {
        "f75be588-6e68-4fb7-b979-d7b72ce1122c",
    },
    "How many units were sold for Amarilla product in 2014? Give a total␣
↪amount, and then break down the amount into units sold per segment for␣
↪Amarilla for the same period.": {
        "1708a705-95ed-4350-a1dc-24f577d48a9d",
        "3c202fc9-7d82-4449-a607-0c4cd27be31c",
        "6afe17d4-79ee-4509-94bd-ba0ee7991178",
        "5ba196d7-27a1-456a-b704-9f274b3721e9",
        "e6f0f5e2-4214-4f4c-84b1-b53d1b0dfb4a",
        "bcfc68ce-5482-4489-87c0-6a75f98073fc",
    },
    "Explain Article 7 and 8 in the format of contract agreement": {
        "db62d004-20d7-4214-8f7e-b51fba1dc4f8",
    },
}
```

```
# sanity: warn if any gold id not in current index
all_node_ids = set(getattr(index.docstore, "docs", {}).keys())
missing = [nid for ids in gold_ids_by_query.values() for nid in ids if nid not␣
 ↪in all_node_ids]
if missing:
    print("Warning: some gold node_ids not present in the index:", missing)
```

[369]:
```
all_nodes = list(index.docstore.docs.values())

# Global BM25 retriever (no filtering)
bm25_all = BM25Retriever.from_defaults(nodes=all_nodes, similarity_top_k=16)

def get_hits_no_meta(query: str, k_per: int = 16, final_top_n: int = 10,␣
 ↪num_queries: int = 2):
    """
    Metadata-agnostic retriever:
      - vector (global FAISS) + BM25 (global) -> fusion
      - optional small LLM-based expansion via num_queries
      - cross-encoder reranker
    """
    vec = index.as_retriever(similarity_top_k=k_per)

    fusion = QueryFusionRetriever(
        retrievers=[vec, bm25_all],
        llm=Settings.llm,                # Gemini
        similarity_top_k=k_per,
        num_queries=num_queries,
        mode="reciprocal_rerank",
    )

    # Retrieve
    hits = fusion.retrieve(query) or []

    # Rerank (no metadata filtering)
    reranker = SentenceTransformerRerank(
        model="cross-encoder/ms-marco-MiniLM-L-2-v2",
        top_n=final_top_n,
    )
    hits = reranker.postprocess_nodes(hits,␣
 ↪query_bundle=QueryBundle(query_str=query))
    return hits  # list[NodeWithScore]

def _node_id_of(hit):
    node = getattr(hit, "node", hit)
    return getattr(node, "node_id", None) or getattr(node, "id_", None)
```

```
def preview_no_meta(query: str, top_k: int = 10, k_per: int = 16, num_queries:␣
↪int = 2):
    hits = get_hits_no_meta(query, k_per=k_per, final_top_n=top_k,␣
↪num_queries=num_queries)
    for i, h in enumerate(hits, 1):
        node = h.node
        nid = _node_id_of(h)
        txt = (node.get_content() or "").replace("\n", " ")[:300]
        print(f"{i}. id={nid} score={getattr(h,'score',None)}")
        print(txt, "\n" + "-"*80)
```

```
2025-10-31 13:26:44,942 - DEBUG - Building index from IDs objects
```

[372]:
```
from collections import defaultdict

def eval_retrieval_no_meta(
    queries,
    gold_ids_by_query,
    Ks=(1,3,5,10),
    k_per: int = 16,
    final_top_n: int = 10,
    num_queries: int = 2,      # set 1 to avoid LLM expansion cost during eval
):
    Ks = sorted(set(Ks))
    maxK = max(Ks)
    final_top_n = max(final_top_n, maxK)

    recall_at = defaultdict(float)
    hit_at = defaultdict(float)
    mrr_sum = 0.0
    n_eval = 0

    for q in queries:
        gold = set(gold_ids_by_query.get(q, []))
        if not gold:
            continue
        n_eval += 1

        hits = get_hits_no_meta(q, k_per=k_per, final_top_n=final_top_n,␣
↪num_queries=num_queries)
        ranked_ids = [_node_id_of(h) for h in hits if _node_id_of(h)]

        # MRR
        rr = 0.0
        for rank, nid in enumerate(ranked_ids, start=1):
            if nid in gold:
                rr = 1.0 / rank
                break
        mrr_sum += rr

        # Recall@K and HitRate@K
        for K in Ks:
            topK = ranked_ids[:K]
            retrieved_rel = sum(1 for nid in topK if nid in gold)
            recall_at[K] += retrieved_rel / max(1, len(gold))
            hit_at[K]    += 1.0 if retrieved_rel > 0 else 0.0

    if n_eval == 0:
        return {"note": "no labeled queries"}

    out = {"n_eval": n_eval, "MRR": round(mrr_sum / n_eval, 4)}
    for K in Ks:
        out[f"Recall@{K}"] = round(recall_at[K] / n_eval, 4)
        out[f"HitRate@{K}"] = round(hit_at[K] / n_eval, 4)
    return out

eval_queries = list(gold_ids_by_query.keys())

metrics = eval_retrieval_no_meta(
    eval_queries,
    gold_ids_by_query,
    Ks = (1,3,5,8,10),
    k_per = 16,
    final_top_n = 10,
    num_queries = 2,
)
print(metrics)
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]

Batches:   0%|          | 0/1 [00:00<?, ?it/s]
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]
Batches:   0%|          | 0/1 [00:00<?, ?it/s]
Batches:   0%|          | 0/1 [00:00<?, ?it/s]
Batches:   0%|          | 0/1 [00:00<?, ?it/s]
{'n_eval': 15, 'MRR': 0.8541, 'Recall@1': 0.6, 'HitRate@1': 0.8, 'Recall@3':
0.6889, 'HitRate@3': 0.8667, 'Recall@5': 0.8111, 'HitRate@5': 0.9333,
'Recall@8': 0.8444, 'HitRate@8': 0.9333, 'Recall@10': 0.8667, 'HitRate@10': 1.0}
```

1.5 Conclusion

We obtained a high MRR of 0.85 which implies that gold nodes appear at top most of the time. HitRate is also quite good as among 10 top-k, it is 100%, implying that retriever is able to get nodes containing answers everytime if we have top k nodes to consider. Recall could be improved, but after more indepth error analysis we found that it is tied to RAG Agents we made with you.com, specially the RAG_PageContinue which got confused in similar content pages when they were from different documents, as we improve on them Recall could be improved and reach level of HitRate and MMR. We can also use screenshots in llamaparse and apply multiple Parsers suited specifically for specific file types and tune specifically for them.

With average latency of 20.5 seconds our RAG pipeline is able to answer questions on time, and while 2 out of 15 answers were incorrect, we had 100% relative numerical accuracy implying that Numerical Figures are being extracted as they appear in documents, with proper punctuation, decimals, and currency signs as needed.