Navigating Large-Scale Document Collections: MuDABench for Multi-Document Analytical QA

Zhanli Li[1,3] **Yixuan Cao**[1,2*] **Lvzhou Luo**[1,2] **Ping Luo**[1,2]
[1]State Key Laboratory of AI Safety, Institute of Computing Technology,
Chinese Academy of Sciences (CAS), Beijing 100190, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]Wenlan School of Business, Zhongnan University of Economics and Law, Wuhan 430073, China
lizhanli@stu.zuel.edu.cn {caoyixuan, luolvzhou23s, luop}@ict.ac.cn

Abstract

This paper introduces the task of analytical question answering over large, semi-structured document collections. We present MuDABench, a benchmark for multi-document analytical QA, where questions require extracting and synthesizing information across numerous documents to perform quantitative analysis. Unlike existing multi-document QA benchmarks that typically require information from only a few documents with limited cross-document reasoning, MuDABench demands extensive inter-document analysis and aggregation. Constructed via distant supervision by leveraging document-level metadata and annotated financial databases, MuDABench comprises over 80,000 pages and 332 analytical QA instances. We also propose an evaluation protocol that measures final answer accuracy and uses intermediate-fact coverage as an auxiliary diagnostic signal for the reasoning process. Experiments reveal that standard RAG systems, which treat all documents as a flat retrieval pool, perform poorly. To address these limitations, we propose a multi-agent workflow that orchestrates planning, extraction, and code generation modules. While this approach substantially improves both process and outcome metrics, a significant gap remains compared to human expert performance. Our analysis identifies two primary bottlenecks: single-document information extraction accuracy and insufficient domain-specific knowledge in current systems. MuDABench is available at https://github.com/Zhanli-Li/MuDABench.

1 Introduction

Large language models (LLMs) combined with retrieval-augmented generation (RAG) are now the dominant paradigm for question answering over unstructured content such as the web, enterprise knowledge bases, and document repositories (Gao et al., 2023). In most settings, these systems treat documents as loosely-related snippets: the goal is to retrieve a small set of passages that fit into a single context window and then answer the query in one or a few model calls. Wikipedia-style multi-hop datasets such as HotpotQA and its successors (Yang et al., 2018b; Ho et al., 2020; Trivedi et al., 2022; Zhu et al., 2024; Levy et al., 2025) instantiate this view, and recent work on long-context benchmarks extends it to longer inputs without changing the underlying interaction pattern.

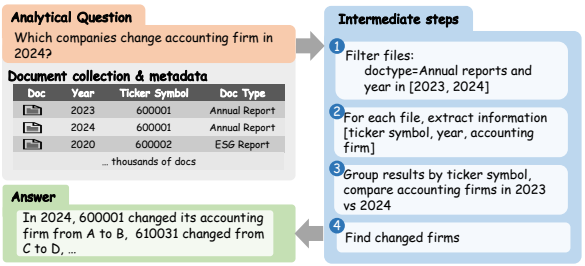

Figure 1: An example of multi-doc analytical QA. The collection of documents behind a question is organized into a semi-structured database through metadata, and answering the question involves first identifying which documents are useful and then targeting the information extraction for the final aggregated answer.

However, another class of real-world document QA applications, namely, analytical QA over multi-document collections, has received limited research attention. Here, a document collection behaves like a semi-structured database: documents are complementary along dimensions such as entity, year, or document type, and answering a question requires aggregating information across dozens of filings. For example, financial regulators analyze annual reports, ESG disclosures, and corporate announcements of listed companies to detect abnormal changes in accounting firms or risk indicators; researchers survey hundreds of papers to construct

*Corresponding Author: Yixuan Cao.
This work was done during Zhanli Li's internship at the Institute of Computing Technology, Chinese Academy of Sciences.

Findings of the Association for Computational Linguistics: ACL 2026, pages 6877–6898
July 2-7, 2026

Table 1: Benchmark Comparison. Compared to Wikipedia-type benchmarks and benchmarks with long contexts, our benchmark has advantages in the number and size of documents as well as document structuring.

Dataset	Doc / Q	Pages / Doc	Multihop	Metadata	Multilingual	Realistic	Publicly released
HotpotQA (Yang et al., 2018b)	∼ 3 webs	-	✓	✗	✗	✓	✓
2WikiMultiHopQA (Ho et al., 2020)	∼ 3 webs	-	✓	✗	✗	✓	✓
MuSiQue (Trivedi et al., 2022)	∼ 2-4 webs	-	✓	✗	✗	✗	✓
FanOutQA (Zhu et al., 2024)	∼ 5-7 webs	-	✓	✗	✗	✓	✓
MoreDocsSameLen (Levy et al., 2025)	∼ 2 webs	-	✓	✗	✗	✗	✓
Financebench (Islam et al., 2023)	1 pdf	∼ 10-200 pages	✗	✓	✗	✓	✓
Aryn (Anderson et al., 2024)	-	4-7 pages	✓	✗	✗	✓	✗
DocETL (Shankar et al., 2024)	-	-	✓	✗	✗	✓	✗
LongBench (Bai et al., 2024)	-	-	✓	✗	✓	✓	✓
RULER (Hsieh et al., 2024)	-	-	✓	✗	✗	✗	✓
Loong (Wang et al., 2024a)	∼ 11 pdf	∼ 30 pages	✓	✗	✓	✓	✓
LongDocURL (Deng et al., 2024)	1 pdf	85.6 pages	✓	✗	✗	✗	✓
M3DocVQA (Cho et al., 2025)	∼1.4 pdf	∼12 pages	✓	✓	✗	✓	✓
FinAgentBench (Choi et al., 2025)	1 pdf	∼ 100 pages	✗	✓	✗	✓	✗
MuDABench (Ours)	14.8 pdf	149.7 pages	✓	✓	✓	✓	✓

performance tables over datasets and tasks; and public-sector agencies aggregate heterogeneous reports to audit policy outcomes. In these settings, missing one relevant document or misinterpreting one table can invalidate the final conclusion.

Figure 1 illustrates an example of analytical QA that is of critical concern to financial regulators. The underlying data consists of annual reports from multiple companies over several years, and the question asks which companies changed their accounting firms in 2024, as this may signal significant financial changes. To answer this question, the required steps include: filtering all company annual reports from 2023 and 2024, extracting information tuples (year, company, accounting firm) from each report, then aggregating the information into records of the form (company, 2023 accounting firm, 2024 accounting firm, whether changed), and finally outputting the list of companies that made changes. Regulators can then focus on examining the financial status of these companies to detect problems early.

The key challenge of this problem is the large number of documents involved in the analysis. More specifically, not only is the document collection large, but the number of documents requiring actual data extraction is also substantial, potentially thousands of documents, which stands in stark contrast to datasets like HotPotQA (Yang et al., 2018b). Therefore, traditional approaches that directly perform retrieval over all documents or rely on long-context methods both fail, necessitating further research tailored to this problem.

Current benchmarks do not cover this research problem. Wikipedia-based multi-hop datasets capture compositional reasoning but operate over short, homogeneous pages and small numbers of documents per question. Long-context benchmarks such as LongBench, RULER, and LongDocURL probe context-length limits, but they typically assume that all relevant content fits into a single context window. While recent efforts like M3DocVQA (Cho et al., 2025) extend multimodal understanding to multiple documents, they operate on relatively small scales (∼12 pages) compared to real-world repositories. In the financial domain, FinanceBench evaluates single-document QA (Islam et al., 2023), and FinAgentBench (Choi et al., 2025) introduces "agentic retrieval" to precisely locate document types and chunks. However, these benchmarks focus on *retrieval precision* rather than the downstream *aggregation and analysis* of content from massive collections. System papers such as Aryn and DocETL propose multi-step workflows but do not release large-scale public benchmarks (Anderson et al., 2024; Shankar et al., 2024). Table 1 summarizes these trends.

This paper introduces **MuDABench**, a benchmark for *multi-document analytical QA* over large collections of financial filings. Built from annual reports, ESG reports, and corporate announcements of Chinese and U.S. listed companies, MuDABench spans over 80,000 pages, 332 questions. For each question, we construct a document set with 15 documents on average. This quantity is sufficiently large such that the combined length exceeds the context window of current long-context LLMs, yet remains manageable to control the cost of LLM API calls during evaluation. We provide metadata information for these documents and an-

notate an intermediate data point set that captures the essential per-document facts required to answer the question.

For evaluation, we primarily focus on *final* answer correctness, while also introducing a process-oriented diagnostic metric based on intermediate results. Enabled by the intermediate data point set in our dataset, this auxiliary signal is evaluated with task-specific LLM-as-judge protocols, including double-check fact-coverage estimation for standard RAG and cell-wise evaluation for document-grounded workflows.

We propose a metadata-aware multi-agent workflow that plans sub-queries, performs single-document extraction, normalizes answers into flat JSON, and aggregates them with generated analysis code. Experimental results show that ordinary RAG frameworks achieve low accuracy even with large retrieval budgets, and our workflow substantially improves final-answer accuracy while also yielding more complete intermediate extraction patterns in many cases. But all methods remain significantly below human performance. We conduct a detailed analysis and identify the key challenges of this task, including the requirement that a large number of single-document information extractions must all be correct (resulting in low overall success rates), as well as the insufficient domain-specific knowledge required for effective planning.

2 Related Work

There are a number of works for QA on documents, including QA on pages, images, and tables within a document, QA on a single document, and QA on multiple documents. We describe each of these works below.

QA on Document Elements Complex document elements, such as tables and images, pose distinct challenges for LLMs owing to their structured and visual characteristics. In the realm of document image QA, Kahou et al. (2017) pioneered the FigureQA dataset, which comprises synthetic scientific-style figures, including line plots, dot-line plots, vertical and horizontal bar graphs, and pie charts. Complementing this, Mathew et al. (2021) introduced DocVQA, a dataset encompassing over 12,000 document images paired with questions. Recent advancements in optical character recognition (OCR) and multimodal LLMs have facilitated effective performance by open-source models on these tasks. In the realm of Table QA, Pang et al. (2024) developed the TabIS benchmark, employing single-choice questions to assess LLMs, while Wu et al. (2025) created TableBench, a comprehensive dataset sourced from industry. These investigations underscore a performance disparity, with open-source models trailing behind proprietary counterparts, such as the GPT, which exhibit near-human performance in table-based reasoning.

QA on Single Document Single-document QA involves a user specifying a document and using its information to answer questions. The development of long-context models and RAG systems has led to significant improvements in single-document QA results. This is particularly evident in specialized domains and multimodal contexts. For instance, in the financial domain, which requires specialized knowledge, the correctness rate of FinanceBench (Islam et al., 2023) has increased from VectifyAI (2024) to 98.7%. But there are still challenges here, in single-document multimodal QA, Deng et al. (2024) introduced LongDocURL, highlighting the challenges that document layout poses for LLMs.

QA on Multi-Document Multi-document QA broadly involves utilizing both the web and specific document repositories as data sources. This task presents heightened complexity, necessitating that LLMs synthesize information and reason across disparate documents. Existing multi-document benchmarks, often primarily sourcing data from Wikipedia, frequently emphasize multi-hop reasoning problems involving multiple entities. These benchmarks highlight persistent deficiencies in LLMs' capabilities for robust multi-hop reasoning (Ho et al., 2020; Trivedi et al., 2022; Zhu et al., 2024; Levy et al., 2025). In the financial domain, FinAgentBench (Choi et al., 2025) targets the retrieval stage, evaluating whether agents can identify the correct document types and passages. Similarly, M3DocVQA (Cho et al., 2025) addresses the challenge of visual reasoning across multiple documents.

However, a common characteristic of much prior work is its treatment of multiple documents primarily as data sources from which relevant snippets are retrieved and aggregated into a single context for an LLM. Studies introducing benchmarks like Loong (Wang et al., 2024a) and RULER (Hsieh et al., 2024) reveal significant limitations in current long-context LLMs specifically for multi-document QA. However, existing research has predominantly focused on addressing multi-document

questions through a single LLM call, without explicitly distinguishing among individual documents within the query set.

Crucially, analytical queries require comprehensive multi-step analysis across documents. While prior work has proposed frameworks for such multi-step, multi-document QA systems (Anderson et al., 2024; Shankar et al., 2024), standardized benchmarks for evaluating this capability remain publicly unavailable, and their documents are very short. To address this gap, we present MuDABench, a novel benchmark for Multi-Document Analysis and targeting scenarios involving document sets exceeding the context window of a single long-context LLM.

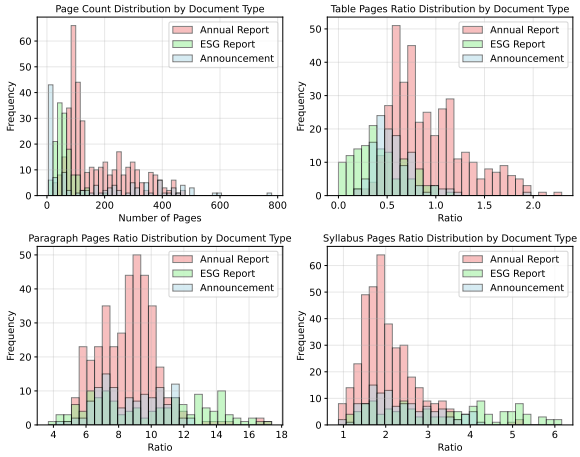

Figure 2: Document Elements Distribution. The distribution is divided into three sections: annual report, ESG report and announcement

3 Benchmark

We collected 589 documents from US and Chinese listed companies with explicit metadata. Second, we set up about 5-38 PDF documents after each question, which is more than all existing work in terms of document pages and far exceeds the maximum LLM context. In the following, we will introduce our document types and annotation process in turn, and finally introduce our evaluation metrics.

3.1 Document Source

Our document collection constitutes the most comprehensive repository of financial documents among available benchmarks. We systematically crawled annual reports, corporate announcements, and ESG report documents from two primary sources: cninfo[1] and SEC[2].

Annual reports contain comprehensive disclosures of listed companies' operational status, published annually. These documents feature extensive structured tabular data.

Announcements represent ad-hoc disclosures by listed companies, with significant proportions of scanned documents.

ESG reports disclose corporate performance in environmental, social responsibility, and governance. These documents are characterized by complex visual layouts, including richly colored backgrounds and extensive pictorial elements (Li and Yang, 2025; Zhang et al., 2025; Li et al., 2026).

This heterogeneous document format distribution ensures benchmark diversity and fits Hui et al. (2024)'s emphasis on the importance of parsing for document QA. Figure 2 illustrates the format distribution following parsing through an advanced commercial PDF processing tool provided by ChatDOC (2025)[3].

3.2 Metadata Annotation

Each document is accompanied by metadata, such as subject category and author's name in academic contexts, or date and coverage area in news domains. In our benchmark, each document is annotated with three metadata fields:

Ticker symbol: Identifies the company associated with the document.

Fiscal year: Indicates the period the document covers, distinct from its publication year.

Document type: Classified as annual reports (US or CN), ESG reports, meeting of shareholders announcement, or profit distribution equity announcements.

This metadata can be used to filter or prioritize documents without accessing their specific content.

3.3 Question Annotation

We employ a distant supervision annotation strategy (Yang et al., 2018a) combined with expert curation to construct our benchmark. First, we leverage authoritative financial databases to curate a comprehensive repository of structured data points, encompassing metrics such as revenue, executive details, dividends, and social responsibility indicators. These data are systematically organized into a master spreadsheet where each row corresponds to a document D_i, indexed by metadata fields M_i.

[1] cninfo: https://www.cninfo.com.cn/
[2] SEC: https://www.sec.gov/
[3] ChatDOC: https://chatdoc.com/

To generate the questions Q_j, financial domain experts designed natural language question templates targeting specific analytical tasks (e.g., trend analysis, peer comparison, see more in Appendix A.8). These templates are instantiated using the structured data to produce diverse and realistic queries. Crucially, to facilitate robust evaluation via an LLM-as-a-judge, experts manually transcribed the specific structured indicators required for each question into natural language statements. These descriptive statements constitute the intermediate information set $\mathcal{S}_j$, serving as the fine-grained ground truth for verifying whether the model has correctly extracted the necessary facts from the documents.

Formally, for each question Q_j, we sample a set of k relevant documents to form the collection $\mathcal{D}_j = \{D_{j1}, D_{j2}, \ldots, D_{jk}\}$. The final dataset is formalized as:

$$\mathcal{X} = \{(Q_j, \mathcal{D}_j, \mathcal{M}_j, \mathcal{S}_j) \mid j = 1, 2, \ldots, n\}, \quad (1)$$

where $\mathcal{M}_j$ represents the metadata set for documents in $\mathcal{D}_j$, $\mathcal{S}_j$ denotes the set of natural language fact descriptions derived from the structured data, and n is the total number of samples.

3.4 Question Grouping

Roughly speaking, we categorize the benchmarks into **Simple** and **Complex** problems, based on three key dimensions: the volume of single-document information required, the complexity of numerical computation involved, and the depth of logical reasoning demanded to derive the answer.

For instance, a typical simple problem is formulated as: *Please calculate the variance of the company's total cost in 2021 based on your knowledge base.* In contrast, a more complex version of the same problem would be: *Please calculate the variance of total costs for companies audited by Big 4 accounting firms in your knowledge base for the year 2021.* The increased difficulty here is reflected in multiple layers of reasoning: first, identifying all companies in the dataset that meet the "audited by Big 4" criterion (which may require cross-referencing multiple documents or verifying implicit attributes); second, extracting total cost figures for only those filtered entities; and third, performing the variance calculation on this subset of data. Such a problem thus demands both conditional filtering of information and multi-step logical integration, distinguishing it from the straightforward extraction-computation pipeline of simple questions. More cases are in the Appendix A.8.

3.5 Annotation Verification

Despite the existence of specialized databases as a reference, manual labeling may also contain errors. Therefore, we adopt a multi-document problem Q, use DeepSeek R1 to generate a single document query q_i for every single document, and then input it into an RAG system, to get an answer a_i on the document, and if there is any contradiction with S_i, then the problem is manually re-labeled or the question description needs to be modified. However, since ChatDOC is unable to adjust the number of recalled chunks, we did not include it in our subsequent experiments.

3.6 Evaluation Metrics

We evaluate each system with three metrics: **process accuracy**, **final-answer accuracy**, and **full accuracy**. Among them, **final-answer accuracy** is our primary end-task metric, while **process accuracy** is mainly used as a diagnostic signal for intermediate extraction quality. We note that process coverage can be less reliable when equivalent evidence can be expressed in multiple non-atomic fact forms.

Final-answer accuracy. For each question $Q_i \in \mathcal{Q}$, let A_i be the gold final answer and $\hat{A}_i$ be the model prediction. Let $T_i \in \{0, 1\}$ denote whether $\hat{A}_i$ is semantically equivalent to A_i (judged by an LLM):

$$\text{Accuracy}_{\text{final}} = \frac{1}{|\mathcal{Q}|} \sum_i T_i. \quad (2)$$

Process accuracy. Let $\mathcal{S}_i$ denote the gold set of minimal supporting facts for Q_i, and $\mathcal{I}_i$ denote the facts extracted by the system.

(a) Standard RAG (question-level). For standard RAG systems (single retrieved context, no explicit document alignment), we estimate fact coverage by judging how many gold supporting facts in $\mathcal{S}_i$ are semantically supported by the extracted information $\mathcal{I}_i$. Formally, we write

$$C_i = \frac{|\mathcal{I}_i \cap \mathcal{S}_i|}{|\mathcal{S}_i|}, \quad (3)$$

where the intersection denotes judge-determined semantic matches rather than exact string identity. Because a single judge may overestimate coverage,

we apply a double-check judge that estimates the error/missing ratio:

$$E_i = \frac{\#\text{ incorrect or missing facts in } \mathcal{S}_i}{|\mathcal{S}_i|}. \quad (4)$$

We then use conservative coverage

$$\tilde{C}_i = \min\big(C_i,\ 1 - E_i\big). \quad (5)$$

(Manual verification: agreement improves from $16/30$ to $26/30$.)

(b) Document-grounded workflow (cell-wise on aligned rows). Since MuDABench is derived from remotely annotated structured data, it is natural to evaluate how well the required table content can be reconstructed when answering such questions. We therefore evaluate process quality in a cell-wise manner on aligned rows. Let $\mathcal{C}_i$ be the set of required gold metric cells across all aligned rows for question Q_i, and $\hat{\mathcal{C}}_i$ be the subset of correctly extracted cells:

$$C_i^{\text{cell}} = \frac{|\hat{\mathcal{C}}_i|}{|\mathcal{C}_i|}. \quad (6)$$

To assess the reliability of this cell-wise judge, we manually audited 30 cases and compared the automatic cell-level decisions against human verification. The resulting cell-level agreement was 81.93%. To unify both settings, define the per-question process score as

$$P_i = \begin{cases} \tilde{C}_i, & \text{standard RAG,} \\ C_i^{\text{cell}}, & \text{document-grounded workflow.} \end{cases} \quad (7)$$

Then process accuracy is defined as

$$\text{Accuracy}_{\text{process}} = \frac{1}{|\mathcal{Q}|} \sum_i P_i. \quad (8)$$

Full accuracy. Finally, we report a strict joint metric: a sample is counted as correct only if process is fully correct and final answer is correct. Let $m_i = \mathbf{1}[P_i = 1]$. Then

$$\text{Accuracy}_{\text{full}} = \frac{1}{|\mathcal{Q}|} \sum_i m_i\, T_i. \quad (9)$$

4 Methodology

MuDABench presents a unique challenge where document collections exceed the context window of current LLMs, rendering single-pass ingestion infeasible (Huang et al., 2023; Levy et al., 2025).

Algorithm 1 Metadata-Aware Multi-Agent Analytic QA Workflow

Input: Query Q, document collection $\mathcal{D} = \{D_1, \ldots, D_n\}$, metadata $\mathcal{M} = \{M_1, \ldots, M_n\}$, batch size B
Output: Final answer A

// Phase 1: Planning
$\mathcal{T} \leftarrow \textsc{PlanAgent}(Q, \mathcal{M}_{\text{schema}})$ ▷ Generate sub-query templates with optional metadata restrictions
// Phase 2: Metadata-Guided Extraction
$\mathcal{Q}_{\text{pairs}} \leftarrow \emptyset$
for $i = 1$ **to** n **do**
  for each $T_j \in \mathcal{T}$ **do**
    if $\textsc{SatisfyRestriction}(M_i, T_j)$ **then**
      $q_{i,j} \leftarrow \textsc{FillTemplate}(T_j, M_i)$ ▷ Instantiate the template using document metadata
      $a_{i,j} \leftarrow \textsc{RAGSystem}(D_i, q_{i,j})$ ▷ Single-document targeted extraction
      $\mathcal{Q}_{\text{pairs}} \leftarrow \mathcal{Q}_{\text{pairs}} \cup \{(M_i, q_{i,j}, a_{i,j})\}$
    end if
  end for
end for
// Phase 3: Schema Definition and Batch Normalization
$S_{\text{json}} \leftarrow \textsc{DefineSchema}(\textsc{Sample}(\mathcal{Q}_{\text{pairs}}), Q)$
$\mathcal{J} \leftarrow \emptyset$
$K \leftarrow \lceil |\mathcal{Q}_{\text{pairs}}| / B \rceil$
for $k = 1$ **to** K **do**
  $\mathcal{B}_k \leftarrow \textsc{GetBatch}(\mathcal{Q}_{\text{pairs}}, k, B)$
  $\mathcal{J}_k \leftarrow \textsc{NormAgent}(\mathcal{B}_k, S_{\text{json}})$
  $\mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}_k$
end for
// Phase 4: Programmatic Analysis
$p_{\mathcal{J}} \leftarrow \textsc{SaveJSON}(\mathcal{J})$ ▷ Save full structured records to an external file
$\mathcal{J}_{\text{demo}} \leftarrow \textsc{Sample}(\mathcal{J})$ ▷ Provide only examples to the code agent
$C_{\text{code}} \leftarrow \textsc{CodeAgent}(Q, \mathcal{J}_{\text{demo}}, S_{\text{json}}, p_{\mathcal{J}})$
$R_{\text{exec}} \leftarrow \textsc{Execute}(C_{\text{code}}, p_{\mathcal{J}})$
// Phase 5: Final Synthesis
$A \leftarrow \textsc{FinalAgent}(Q, R_{\text{exec}}, \mathcal{J}_{\text{demo}})$
return A

To address this, we propose a scalable **Multi-Agent Analytic QA Workflow** that explicitly orchestrates multi-step reasoning over large-scale repositories. A key feature of this approach is its ability to scale to processing hundreds or thousands of documents. The procedure is detailed in Algorithm 1 and visualized in Figure 3. The workflow consists of four specialized components:

Scalable Planning Agent: Instead of retrieving documents immediately, this agent decomposes the global query Q into question templates to be asked on each document. The template can be filled with metadata of documents. This abstraction minimizes planning errors and ensures the approach scales to collections of arbitrary size.

Document-Level Information Extractor: We perform targeted extraction by instantiating the query templates for each document D_i using its specific metadata M_i. A standard document RAG system then processes these instantiated queries in

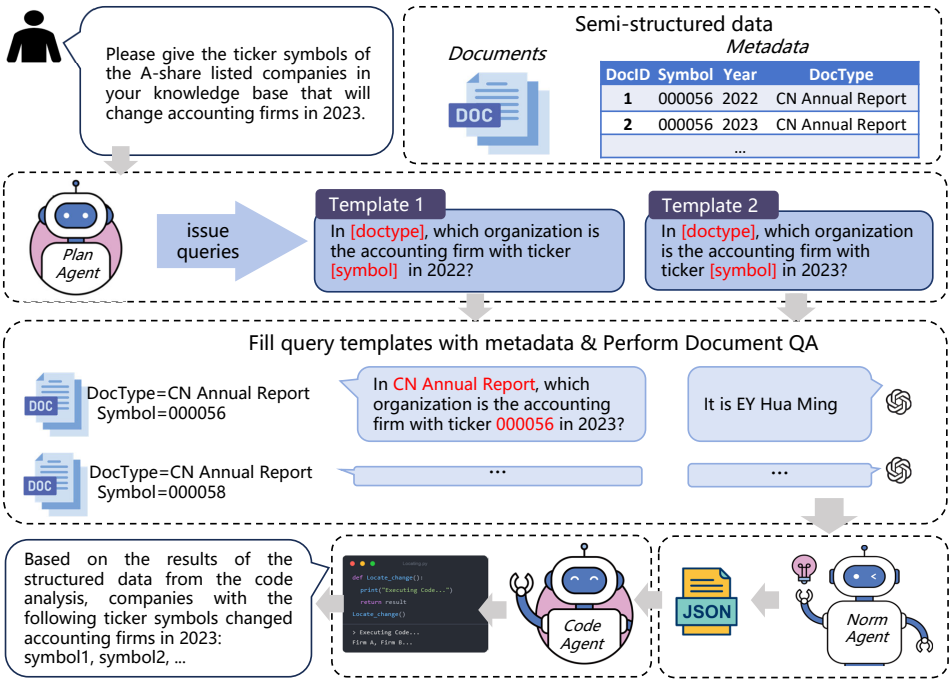

Figure 3: Document agentic workflow. A planning agent uses a metadata schema to generate sub-queries, an end-to-end RAG system answers single-document queries, then responses are normalized to JSON and analyzed by code to obtain the final answer.

parallel, producing intermediate textual evidence that captures local facts.

Scalable Norm Agent: To enable downstream programmatic reasoning, this agent converts unstructured extraction transcripts into structured JSON records. Crucially, to avoid context overflow when processing thousands of documents, we adopt a *batch-iterative* strategy: a schema is defined from a small sample, and subsequent records are normalized in batches under this unified schema.

Scalable Code Agent: Rather than feeding the entire extracted information into the LLM, we provide the agent with the schema and some examples. The agent synthesizes a program to perform analysis over the full structured dataset $\mathcal{J}$ (Wang et al., 2024b), yielding the final answer A.

5 Experiment

5.1 Experimental Setup

The experiments are conducted on MuDABench. As a natural baseline, we employ a RAG system (Lewis et al., 2020) over the multi-document corpus. We consider two prompt variants: one that omits document metadata and one that injects all metadata into the prompt (detailed in Appendix A.7). Both use OpenAI's File Search as the retrieval layer with `GPT-4o-2024-11-20` as the reader (**?**). To study the effect of recall, we set the number of retrieved chunks to $|\mathcal{D}|$ and then increase it to $1.5 \times |\mathcal{D}|$, $2 \times |\mathcal{D}|$, and $2.5 \times |\mathcal{D}|$.[4] All other hyperparameters follow OpenAI defaults.

We also evaluate our proposed agentic workflow. We use `DeepSeek-R1-0528` (Guo et al., 2025) for the planning and code agent, `DeepSeek-Chat-V3-0324` (Liu et al., 2024) for the normalization agent, and OpenAI file search for single document QA. To control cost, we use `gpt-4o-2024-11-20` for one high-budget chunk of the workflow (approximately 30,000 tokens per document) and `gpt-4.1-mini-2025-04-14` (OpenAI, 2025) for the remaining workflow experiments. To study robustness under noisy contexts, we additionally inject $0.5 \times |\mathcal{D}|$ irrelevant documents in the 5-chunk workflow setting, e.g., adding 2023 filings to questions about 2021–2022.

Given the task complexity, we adopt an LLM-as-judge protocol (Gu et al., 2024). Evaluation has two components: (1) assessing information extraction to obtain C_i, and (2) verifying answer correctness, from which we derive the three metrics defined in the previous Section 3.6. For RAG, since it lacks explicit intermediate outputs, we evaluate the retrieval recall by checking if the retrieved

[4] OpenAI caps the number of chunks at 50; larger multipliers would truncate some examples and bias the evaluation.

Model	Simple			Complex		
	$Acc_{process}$	Acc_{final}	Acc_{full}	$Acc_{process}$	Acc_{final}	Acc_{full}
GPT 4o + Chunk $= 1\,\lvert\mathcal{D}\rvert$	0.1572	0.0663	0.0241	0.1459	**0.0482**	0.0181
GPT 4o + Chunk $= 1.5\,\lvert\mathcal{D}\rvert$	0.1761	0.0964	0.0301	0.1801	**0.0482**	**0.0241**
GPT 4o + Chunk $= 2\,\lvert\mathcal{D}\rvert$	0.1793	**0.1265**	**0.0422**	0.2212	0.0361	0.0181
GPT 4o + Chunk $= 2.5\,\lvert\mathcal{D}\rvert$	**0.2163**	0.1084	0.0301	**0.2623**	**0.0482**	0.0120
GPT 4o + Chunk $= 1\,\lvert\mathcal{D}\rvert$ + Metadata	0.1338	0.1084	0.0422	0.1398	0.0301	**0.0181**
GPT 4o + Chunk $= 1.5\,\lvert\mathcal{D}\rvert$ + Metadata	0.1620	0.1145	0.0301	0.1773	0.0181	0.0120
GPT 4o + Chunk $= 2\,\lvert\mathcal{D}\rvert$ + Metadata	0.1978	**0.1386**	0.0422	0.2232	0.0361	**0.0181**
GPT 4o + Chunk $= 2.5\,\lvert\mathcal{D}\rvert$ + Metadata	**0.2514**	0.1325	**0.0542**	**0.2522**	**0.0422**	0.0120
WF w/ GPT 4o + Chunk $= 1$	0.4179	0.0667	0.0000	0.4021	0.0667	0.0095
WF w/ GPT 4.1 mini + Chunk $= 3$	0.5803	**0.2430**	0.0654	0.5338	0.0865	0.0673
WF w/ GPT 4.1 mini + Chunk $= 5$	0.5888	0.2243	**0.0748**	**0.5749**	**0.1619**	**0.1143**
Noise WF w/ GPT 4.1 mini + Chunk $= 5$	**0.5961**	0.1636	0.0727	0.5680	0.1238	0.0762
Human Performance	0.8940	0.8334	0.7334	0.8120	0.7334	0.6667

Table 2: Experiment Results. Bolded labels indicate the optimal setup within each model group. The $Acc_{process}$ metric has been adjusted to reflect the full evaluation set.

Document Category	Avg. Tokens / Doc	Chunk = 1		Chunk = 3		Chunk = 5	
		Simple	Complex	Simple	Complex	Simple	Complex
A-share Annual Report (CN)	499k	0.4696	0.4555	0.6537	0.6674	0.6447	0.6570
A-share ESG Report (CN)	72k	0.3998	0.3898	0.6067	0.4813	0.5865	0.4992
A-share Announcement (CN)	144k	0.3903	0.3786	0.5222	0.4976	0.5542	0.5575
US Stock Annual Report (EN)	120k	0.4472	0.3955	0.3167	0.5374	0.4643	0.7375

Table 3: Impact of Chunk Number on $Acc_{process}$ across Document Categories, including average document length.

Metric	Simple	Complex	Avg
Independent accuracy			
Planning	86.7%	93.3%	90.0%
Extraction	40.0%	20.0%	30.0%
Normalization	100.0%	100.0%	100.0%
Code	93.3%	93.3%	93.3%
Accuracy given all previous steps correct			
Planning	86.7%	93.3%	90.0%
Extraction	38.5%	14.3%	25.9%
Normalization	100.0%	100.0%	100.0%
Code	80.0%	100.0%	85.7%

Table 4: The percentage of correct in each step

chunks cover the gold facts $\mathcal{S}_i$. Specific prompts are provided in Appendix A.7. All LLMs are run with temperature 0 for reproducibility. Two volunteers answer a subset of the benchmark to estimate human performance.

5.2 Main Result

The experimental results, summarized in Table 2, highlight the significant challenges posed by MuDABench and the distinct behaviors of different system architectures.

Standard RAG pipelines struggle with multi-document aggregation, and metadata injection only provides limited gains. Even when powered by GPT-4o, commercial RAG systems exhibit clear structural limitations on MuDABench. Increasing the number of retrieved chunks generally improves evidence coverage, as reflected by higher process accuracy, but this gain does not translate reliably into better final answers: on simple questions, final-answer accuracy improves only up to a point and then fluctuates, while on complex questions it remains consistently low despite better coverage. This pattern suggests that the main bottleneck is not merely retrieval recall, but the model's ability to synthesize fragmented evidence into a correct aggregated conclusion. Explicitly incorporating document metadata offers only partial mitigation. Although metadata can provide a coarse global structure and sometimes improves performance, the overall gains remain limited, indicating that neither larger retrieval budgets nor metadata cues are sufficient without a more structured reasoning workflow.

Agentic workflows significantly improve end-to-end answer quality. The proposed agentic workflow substantially outperforms direct RAG in final-answer accuracy, demonstrating the value of decomposing extraction and reasoning into modular stages. Although the workflow also tends to

achieve higher process-coverage scores, we treat this metric as a diagnostic signal rather than the primary basis for comparison, because equivalent evidence can be represented in flexible and sometimes non-atomic ways. Under stronger chunk budgets, the workflow delivers markedly better end-to-end performance than standard RAG. Furthermore, robustness analysis reveals that injecting noise (irrelevant documents) causes a noticeable drop in final-answer accuracy, especially on complex questions, indicating that downstream aggregation and reasoning remain important bottlenecks.

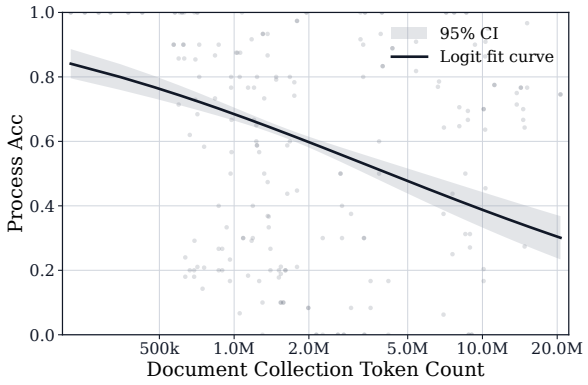

Figure 4: The Impact of Document Collection Token Count on Document Information Extraction

5.3 Fine-Grained Error Analysis

We conduct a fine-grained diagnostic study on 30 randomly selected examples under the 5-chunk workflow setting, with results summarized in Table 4. Since final-answer accuracy is our primary metric, we use process-oriented analysis mainly to identify bottlenecks rather than as a fully reliable standalone measure. Process coverage can be noisy because equivalent evidence may be expressed in non-atomic forms: for example, "an increase of 6% from 2021 to 2022" may correspond to two separate facts such as "100k in 2021" and "106k in 2022", while the final answer may only require the growth rate. With this caveat, the results still indicate that document-level information extraction is the main bottleneck. We also find planning errors caused by insufficient financial domain knowledge (Appendix A.2.1), and coding errors mainly due to encoding or JSON path-reading issues.

To further explore why document-level extraction remains difficult, we estimated a logit model with fixed question type effects. The resulting curve, shown in Figure 4, suggests that information extraction becomes more difficult as document length increases.

This length-dependent performance degradation is further corroborated by the category-wise breakdown in Table 3. In general, shorter document categories tend to be easier to process, although the pattern is not uniform across all settings. For example, A-share ESG reports, which have a relatively small average token count, achieve competitive extraction accuracy, but they are not consistently the best-performing category under every chunk budget; in several settings, A-share annual reports or U.S. annual reports perform better. Overall, Table 3 suggests that document length is an important factor, but extraction difficulty also depends on document type and language.

6 Conclusion

We present MuDABench, the first large-scale benchmark designed for complex, metadata-driven analysis over semi-structured document collections. It requires navigating over 80,000 pages of financial documents. Evaluations reveal that standard RAG systems struggle at this scale. While our metadata-aware agentic workflow significantly improves final-answer accuracy over the baselines, it still trails human performance. We hope MuDABench serves as a rigorous testbed for future scalable document analysis systems.

Limitations

Our work is restricted to the financial domain due to the scarcity of dense semi-structured data elsewhere. We also limited the dataset size. Although distant supervision can increase the size easily, the testing is costly and does not yield new discoveries. Finally, because financial atomic facts involve numerous issues related to granularity and equivalence, they must be handled with care when evaluating different QA systems.

Ethical Considerations

The documents curated in MuDABench are collected exclusively from publicly available financial disclosures (e.g., filings from the SEC and cninfo), ensuring that no private or non-public personal information is compromised. While this benchmark utilizes real-world financial figures, it is intended solely for the research and evaluation of NLP systems. We used AI assistants for minor language polishing; they did not contribute to ideas, experiments, or results.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (No. 62206265, 62076231).

References

Eric Anderson, Jonathan Fritz, Austin Lee, Bohou Li, Mark Lindblad, Henry Lindeman, Alex Meyer, Parth Parmar, Tanvi Ranade, Mehul A Shah, and 1 others. 2024. The design of an llm-powered unstructured analytics system. *arXiv preprint arXiv:2409.00847*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

ChatDOC. 2025. Bridge your PDFs to RAG-Ready data. Accessed: 2025-08-02.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. 2025. M3docvqa: Multi-modal multi-page multi-document understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 6237–6247.

Chanyeol Choi, Chaewoon Kim, Jaeseon Ha, Jihoon Kwon, Minjae Kim, Hojun Choi, Yongjin Kim, Alejandro Lopez-Lira, Juneha Hwang, Suyeol Yun, and Yongjae Lee. 2025. Finagentbench: A benchmark dataset for agentic retrieval in financial question answering. *arXiv preprint arXiv:2508.14052*.

Chao Deng, Jiale Yuan, Pi Bu, Peijie Wang, Zhong-Zhi Li, Jian Xu, Xiao-Hui Li, Yuan Gao, Jun Song, Bo Zheng, and 1 others. 2024. Longdocurl: a comprehensive multimodal long document benchmark integrating understanding, reasoning, and locating. *arXiv preprint arXiv:2412.18424*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

Yunpeng Huang, Jingwei Xu, Junyu Lai, Zixu Jiang, Taolue Chen, Zenan Li, Yuan Yao, Xiaoxing Ma, Lijuan Yang, Hao Chen, and 1 others. 2023. Advancing transformer architecture in long-context large language models: A comprehensive survey. *arXiv preprint arXiv:2311.12351*.

Yulong Hui, Yao Lu, and Huanchen Zhang. 2024. Uda: A benchmark suite for retrieval augmented generation in real-world document analysis. *Advances in Neural Information Processing Systems*, 37:67200–67217.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. Financebench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. 2017. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*.

Shahar Levy, Nir Mazor, Lihi Shalmon, Michael Hassid, and Gabriel Stanovsky. 2025. More documents, same length: Isolating the challenge of multiple documents in rag. *arXiv preprint arXiv:2503.04388*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Zhanli Li, Huiwen Tian, Lvzhou Luo, Yixuan Cao, and Ping Luo. 2026. Deepread: Document structure-aware reasoning to enhance agentic search. *arXiv preprint arXiv:2602.05014*.

Zhanli Li and Zichao Yang. 2025. Esg rating disagreement and corporate total factor productivity: Inference and prediction. *Finance Research Letters*, 78:107127.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.

OpenAI. 2025. GPT-4.1 mini — OpenAI API Documentation. https://developers.openai.com/api/docs/models/gpt-4.1-mini. Accessed: April 2026.

Chaoxu Pang, Yixuan Cao, Chunhao Yang, and Ping Luo. 2024. Uncovering limitations of large language models in information seeking from tables. *arXiv preprint arXiv:2406.04113*.

Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G Parameswaran, and Eugene Wu. 2024. Docetl: Agentic query rewriting and evaluation for complex document processing. *arXiv preprint arXiv:2410.12189*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

VectifyAI. 2024. "mafin2.5-financebench: Finance benchmark evaluation". https://github.com/VectifyAI/Mafin2.5-FinanceBench. Accessed: June 2024.

Minzheng Wang, Longze Chen, Cheng Fu, Shengyi Liao, Xinghua Zhang, Bingli Wu, Haiyang Yu, Nan Xu, Lei Zhang, Run Luo, and 1 others. 2024a. Leave no document behind: Benchmarking long-context llms with extended multi-doc qa. *arXiv preprint arXiv:2406.17419*.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024b. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, and 1 others. 2025. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25497–25506.

Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018a. Dcfee: A document-level chinese financial event extraction system based on automatically labeled training data. In *Proceedings of ACL 2018, System Demonstrations*, pages 50–55.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018b. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Lei Zhang, Xin Zhou, Chaoyue He, Di Wang, Yi Wu, Hong Xu, Wei Liu, and Chunyan Miao. 2025. Benchmarking multimodal understanding and complex reasoning for esg tasks. *arXiv preprint arXiv:2507.18932*.

Andrew Zhu, Alyssa Hwang, Liam Dugan, and Chris Callison-Burch. 2024. Fanoutqa: A multi-hop, multi-document question answering benchmark for large language models. *arXiv preprint arXiv:2402.14116*.

A Appendix

A.1 Comparison on the Same Selected Subset

To provide a fair comparison between human performance and agentic workflows under the same evaluation scope, we report results on the same selected subset used in our human perfermance. This subset is divided into **Simple** and **Complex** cases. Table 5 shows that, although stronger workflow configurations improve both process and final-answer performance on this subset, all agentic systems still remain substantially below human performance.

Table 5: Performance comparison on the subset of Human Performance.

Model	**Simple**			**Complex**		
	$Acc_{process}$	Acc_{final}	Acc_{full}	$Acc_{process}$	Acc_{final}	Acc_{full}
WF w/ GPT 4o + Chunk = 1	0.5936	0.1429	0.0000	0.5396	**0.3333**	0.0667
WF w/ GPT 4.1 mini + Chunk = 3	0.6983	**0.2667**	**0.2000**	0.6978	**0.3333**	**0.2667**
WF w/ GPT 4.1 mini + Chunk = 5	0.6897	0.2000	**0.2000**	**0.7230**	**0.3333**	0.2000
Noise WF w/ GPT 4.1 mini + Chunk = 5	**0.7328**	**0.2667**	0.1333	0.6906	0.2667	0.1333
Human Performance	0.8940	0.8334	0.7334	0.8120	0.7334	0.6667

A.2 Case Study

A.2.1 Planning Errors Prior to Information Extraction

In complex financial question answering, a substantial fraction of failures arise already in the planning phase, before any document-level extraction is performed. These errors are typically rooted in insufficient domain knowledge about market conventions and disclosure practices, which leads the agent to design sub-queries that are structurally misaligned with the underlying task.

Figure 5 illustrates a representative planning error driven by an incorrect mental model of corporate event frequencies. The user query explicitly requests the companies with the *highest number* of extraordinary general meetings. However, the Plan Agent generates a sub-query that only verifies the *existence* of such a meeting (i.e., whether at least one was convened). This effectively reduces a counting problem to a binary classification problem, and ignores the fact that listed firms may hold multiple extraordinary general meetings within a single fiscal year. As a result, the downstream pipeline never triggers the aggregation logic necessary to rank companies by meeting counts.

Figure 6 shows a second class of planning failure related to annual report disclosure protocols. The task requires identifying changes in accounting firms between 2021 and 2022. Instead of decomposing the task into two extraction steps—retrieving the engaged accounting firm in 2021 and in 2022, and then comparing them—the agent attempts to locate an explicit textual description of the "change" event within a single document. This strategy contradicts standard reporting practices, where annual reports usually disclose only the currently engaged firm for that specific fiscal year rather than the full transition history. Because the plan does not incorporate this protocol knowledge, the system fails to construct the necessary multi-hop reasoning chain and the retrieval stage subsequently breaks down.

A.2.2 Errors After Information Extraction

Even when the planning stage is successful, errors can still emerge in the information extraction and normalization stages. Figure 7 illustrates a failure caused by coupled extraction and normalization issues. Financial reports routinely present data for both the current and previous fiscal years within the same table or paragraph. Models often struggle to disambiguate which subset of these values corresponds to the target reporting period, leading to extractions that conflate multi-year information. When such ambiguous entries are later merged with strictly single-year values from other documents, the resulting heterogeneity in temporal scope severely complicates normalization and downstream analysis.

Figure 8 depicts a related failure mode at the schema alignment stage. Here, the extracted JSON records deviate from the predefined schema, for example by introducing inconsistent field names or missing mandatory keys. Although these deviations may appear minor at the textual level, they cause runtime

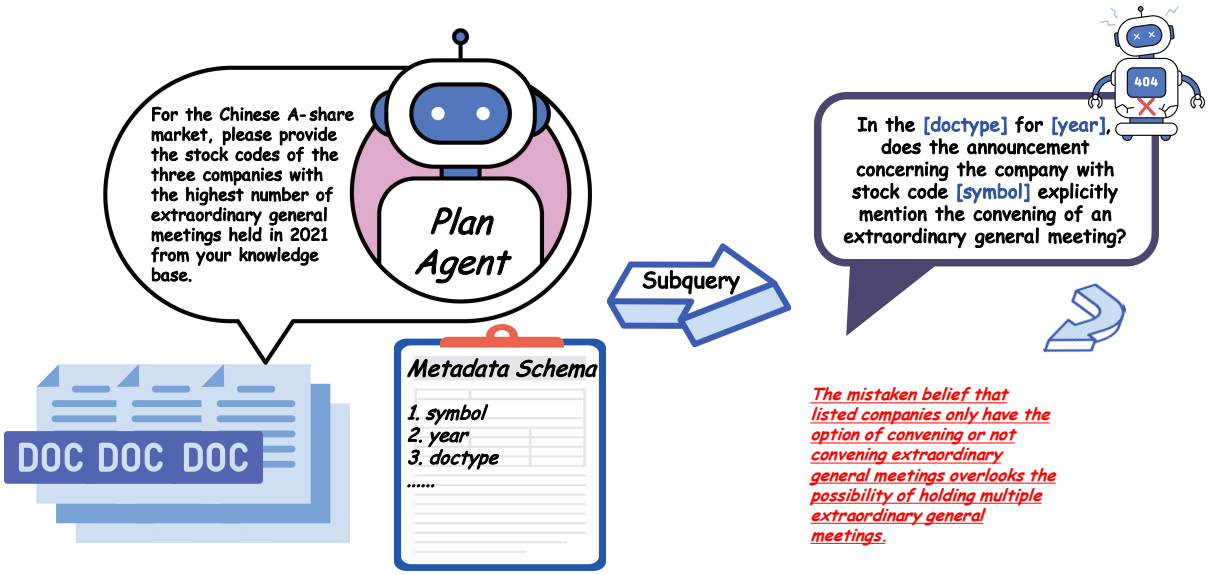

Figure 5: Case Study: Planning errors stemming from a lack of knowledge regarding shareholder meetings

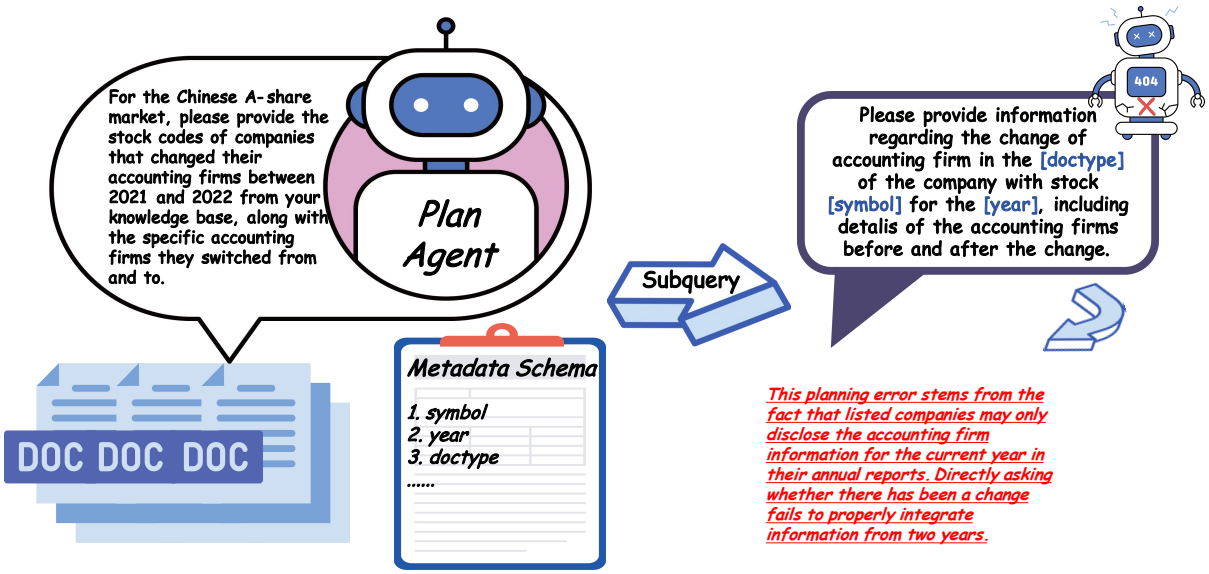

Figure 6: Case Study: Planning errors stemming from a lack of established annual report preparation protocols

exceptions in the code analysis agent and prevent the execution of otherwise valid analytical programs. This highlights that robust large-scale multi-document analysis requires not only accurate extraction but also strict adherence to a stable schema across all stages of the workflow.

A.3 Data Source Details

To construct our dataset, we sourced data from several authoritative financial databases widely used by researchers. These sources are detailed in Table 6.

A.4 Benchmark Construction Detail

Overall, our annotation is semi-automated by means of distant supervision. First we download some structured data in `.csv` format from CSMAR, which stores financial and non-financial metrics of listed companies, and then we match each document with the data in this `.csv`, and then our annotators just need to convert each metric into natural language descriptions and define a series of questions, through the permutations and combinations our annotation strategy can be easily scaled to generate large amounts of data.

Question
Based on US stock market, Which three companies achieved the highest operating profit in 2021 in your knowledge base? Provide their stock symbols.
Branch
Subtask: According to the {doctype} for stock symbol {symbol} in {year}, what was the company's operating profit? restriction: year = [2021] doctype = [US Stock Annual Report]
Docset = Docset.fliter(restriction) for metadata in Docset: Querys.append(Subtask.format(metadata)) Answer.append(llm.chat(Query)) # append the operating profit of 2020 and 2021 w/o year information
{Querys,Answers} -> [JSON, schema] # Incorrectly normalizing 2020 data to 2021 and dismiss data of 2021 **JSON**: [{ symbol: year: operating_profit: }] **Correct Schema**: JSON match with the Schema
Correct Code: Logically correct for multi-document problems.
Wrong Answer: Wrong, because there a wrong data in the JSON

Figure 7: Case Study: Ambiguous Information Extraction and Normalization Failure

Data	Source	Description
CN Annual Report	https://www.cninfo.com.cn/	Annual financial disclosures of Chinese listed companies
US Annual Report	https://www.sec.gov/	Annual financial disclosures (e.g., 10-K) of US listed companies
CN ESG Report	https://www.cninfo.com.cn/	Environmental, Social, and Governance reports of Chinese companies
CN Announcement	https://www.wind.com.cn/	Ad-hoc corporate announcements of Chinese listed companies
CN Financial Data	https://data.csmar.com/	Structured financial metrics and market data for China
US Financial Data	https://data.csmar.com/	Structured financial metrics and market data for the US

Table 6: Data Source. Among them, cninfo and SEC are free public data, and wind and CSMAR are commercial databases.

A.4.1 Announcement of merger strategy

For the announcement category, since there is no standardization of announcements made by different companies, but it can be confirmed that the disclosure of information has a lag, so we use a MERGE strategy, that is, the same category of announcements of 2021 and 2022 as the announcement of the fiscal year 2021. This ensures that the information in the document includes all the information of the fiscal year 2021.

A.4.2 Cross-year Problem

It's worth mentioning that for the cross-year problem, we generate an intermediate information collection $\mathcal{S}$ that is half the size of the document collection $\mathcal{D}$, and we merge two years of information for the same company. An intuitive example would be, *Which are the three companies with the highest revenue growth rates from 2022–2023 among U.S. publicly traded companies in your knowledge base*, and the S_i for this question would look like this: *Company A's revenue in 2022 was xx, and in 2023 it was xx, an increase of xx*, and in this way we encourage the QA system to do some simple computational reasoning in the information extraction. This design doesn't affect the design of the evaluation metrics in our paper.

<table>
<tr><th colspan="2">Question</th></tr>
<tr><td colspan="2">For China's A-share market, analyze the donation efficiency (donation efficiency = social donationssocial contribution value per share) based on the data in your knowledge base for the year 2023, giving the ticker symbols, the amount of social donations, and the donation efficiency of the three highest companies.</td></tr>
<tr><th>Branch 1</th><th>Branch 2</th></tr>
<tr><td>Subtask: What is the specific value of social donations paid in the {year} year {doctype} in which the ticker symbol is {symbol}?
restriction:
year = [2021]
doctype = [CN annual report, CN ESG resport, CN shareholders' meeting type of announcement, CN profit distribution class announcement]</td><td>Subtask: What is the specific value of the social contribution value per share for the {year} year {doctype} with the ticker symbol {symbol}?
restriction:
year = [2021]
doctype = [CN annual report, CN ESG resport, CN shareholders' meeting type of announcement, CN profit distribution class announcement]</td></tr>
<tr><td>Docset = Docset.fliter(restriction)
for metadata in Docset:
Querys.append(Subtask.format(metadata))
Answers.append(llm.chat(Query))</td><td>Docset = Docset.fliter(restriction)
for metadata in Docset:
Querys.append(Subtask.format(metadata))
Answers.append(llm.chat(Query))</td></tr>
<tr><td>{Querys,Answers} -> [JSON, schema]
JSON: [{
symbol:
year:
social_donations:
}]
Correct Schema:
JSON match with the Schema</td><td>{Querys,Answers} -> [JSON, schema]
JSON: [{
symbol:
year:
social_contribution_value_per_share:
}]
Wrong Schema:
didn't cover columns' true name</td></tr>
<tr><td colspan="2">Wrong Code: can not read wrong column</td></tr>
<tr><td colspan="2">Wrong Answer: Sorry I can't give the answer, because the code return errors so that I can't get enough information to answer this question.</td></tr>
</table>

Figure 8: Case Study: Schema Alignment Error in Code Execution

Model	Source	Usage
ChatDOC	https://chatdoc.com/	PDF parsing
OpenAI GPT 4o	https://platform.openai.com/docs/models/gpt-4o?snapshot=gpt-4o-2024-11-20	LLM for RAG
OpenAI GPT 4.1 mini	https://platform.openai.com/docs/models/gpt-4.1-mini	LLM for RAG
OpenAI File Search	https://platform.openai.com/docs/guides/tools-file-search	Retrieve
DeepSeek R1	https://openrouter.ai/deepseek/deepseek-r1-0528	Plan and Code Agent
DeepSeek V3	https://openrouter.ai/deepseek/deepseek-chat-v3-0324	Norm Agent
DeepSeek V3.2	https://openrouter.ai/deepseek/deepseek-v3.2	Cell-wise Rejudge Judge
Kimi k2	https://openrouter.ai/moonshotai/kimi-k2	Other Judge Tasks

Table 7: Model Source. All of the above except DeepSeek and Kimi are closed-source models, and all of them call commercial API services in the experiments of this paper.

A.5 Model Details

In this paper we use a range of expensive commercial modeling services, from pdf parsing and model measurement, as shown in Table 7.

A.6 Judge Details

We use different judge models for different evaluation components. In particular, `DeepSeek-V3.2` is used for the cell-wise rejudge setting on aligned rows, which requires fine-grained field-level matching between extracted evidence and aligned gold rows, while `Kimi K2` is used for the remaining correctness judgments. All judge models are run at temperature 0.

A.7 Prompt Template

A.7.1 Prompt Template of adding metadata to RAG

Prompt: adding metadata to RAG

The list of document metadata you can query is as follows: **[document_info]**
You need to answer the question: **[question]**

A.7.2 Prompt Template of Plan Agent

Prompt: Plan Agent

- **Initial Prompt**: You are a multi-document problem decomposition and query assistant. The user is provided with a metadata description of a complex task and multiple documents. Each document has a unique metadata identifier, and instead of answering the final question directly, you need to design specialized query templates for each document so that you can extract information from it that is relevant to the overall task and organize these subqueries in a reasonable order. Also you need to generate the query templates based on using the same language as the language of user's task. For example, if the user speaks Chinese, you generate the query template in Chinese.
- **Question template specification**
  1. **Single Document Oriented**: Each sub-question template must focus on a single document to ensure that the required information can be located within that document.
  2. **Semantic Mutual Exclusivity**: different templates should be independent of each other in meaning and not duplicated; populated to be able to ask questions naturally and smoothly and logically self-consistent.
  3. **Metadata Placement**: all available metadata fields are placed by {} in at least one template, while all metadata fields must use naming consistent with the metadata field description when placed in templates.
  4. **Metadata constraints (optional)**: for each sub-question template, metadata can be constrained. If you think that answering a multi-document question requires that this sub-question template restricts a certain (some) metadata, please use the name of that metadata as a label, with all possible values listed in list format within the label. For example: `"restriction": {"year": ["2021", "2022"]}`. If there is no restriction, the field can be left out.
- **User-supplied information**:
  - Multi-document task description: **[task]**
  - Available metadata fields: **[metadata_description]**
- **Output format** (JSON only; no extra text)

```
[
   {
      "subtask": "Format template for subquestion",
      "restriction": {"year": ["2021","2022"]}
   },
   {
      "subtask": "Format template for subquestion"
   }
]
```

A.7.3 Prompt Template of Norm Agent (Stage 1)

Prompt: Norm Agent (schema)

- **Initial Prompt**: You are an expert in structured data extraction, extract structured data related to complex tasks directly from multiple sub-question answered conversations and transform it into json format, the final output json data should be easily scalable when transform similar conversations to json. Also you need to generate schema based on using the same language as the users's task language. For example, if the user speaks Chinese, you generate the query template in Chinese.
- **Input information**:
  1. Complex problem description: **[task]**
  2. Complete dialog record of sub-question answers: **[multi_conversation]**
- **Processing requirements**:
  - extract all specific data values (numbers, options, measurements, etc.) related to the complex task and standardize the units of measurement.
  - identify variable types and add metadata:
    - **Classification variables**: list the values that actually occur
    - **Ordinal variables**: Preserve sequential relationships
    - **Quantitative variables**: specify the units used
  - For information not provided by the user, keep it as null.
  - Naming of variables should reflect the meaning of the variable and the unit of measure.
  - ideal json structure must just have one level, no nested structure and can be easily analysis by python code.
- **Output format**: Please output data surrounded by `<json>...</json>` (must be a JSON `list[dict]`), and add a brief schema with `<des>...</des>`.

A.7.4 Prompt Template of Norm Agent (Stage 2)

Prompt: Norm Agent (continuation)

- **Initial Prompt**: You are a json continuation helper that converts new conversation records to json data based on the original conversation record converted json data (`list[dict]`) format provided by the user, which is easy to merge with the original data.
- **Input information**:
  1. json data of the original conversation record: **[json]**
  2. new conversation record: **[new_conversation]**
- **Processing requirements**:
  1. Convert the new conversation record to json format, making sure it is consistent with the original data structure.
  2. Maintain consistency in variable naming and units of measure.
  3. Make sure the new json data can be seamlessly connected to the original data.
- **Output Format**: Please enclose your extended section with `<json>...</json>` (a `list[dict]` that can be concatenated to the original data with + in Python).

A.7.5 Prompt Template of Code Agent

Prompt: Code Agent

- **Initial Prompt**: You are a question answering expert, the user will provide a complex task and multiple copies of related json data and their paths, you need to write code based on this data to get the data needed to answer the question. Please note for each available, the user only provides the few shot of original json data (`list[dict]`) for you to write code for this task, and the user will provide the path to the json data, you need to read the json data from the user-provided path, execute the code will directly solve the task ideally.
- **Input information**:
  1. task description: **[task]**
  2. available json data (few shot): **[json_data]**
  3. path to the json data: **[json_path]**
  4. schema of the json data: **[json_schema]**
- **Processing requirements**
  1. Analyze the task description to identify key issues and data requirements.
  2. Based on the json data provided, write executable python code to read the json data from the user-provided path and extract the required information.
  3. The code output should be readable, ideally the code output should answer the task directly.
- **Output format**: Wrap your code in `<execute>...</execute>` and you can add necessary explanations outside the tags.

A.7.6 Prompt of Final Answer

Prompt: Final Answer

The user is provided with a complex task, JSON data description, Python code, and run results. Produce the final concise answer (in the user's language).
Inputs: **[task]**, **[data]**, **[code]**, **[code_resp]**.

A.7.7 Prompt Template of judging Information Extraction (cell-wise on aligned rows)

Prompt: Judge — Information Extraction

The user will provide:

1. Multi-document question: **[question]**
2. Source table headers: **[source_headers]**
3. One gold source row (`row_index = {row_index}`): **[source_row]**
4. Aligned target document metadata (already matched by the dataset, do not judge metadata again): **[aligned_doc_meta]**
5. Metric columns to judge from this row (`metric_total = {metric_total}`): **[metric_columns]**
6. Complete dialog record of all sub-question answers on this document: **[agent_conversation]**

Evaluation Criteria:

1. Judge only against this single gold row and this single document dialog.
2. This gold row has already been aligned to the correct document by the dataset metadata. Treat the document identity as given.

3. Do not score metadata fields in this step. Judge only the metric columns in this row one by one.
4. A metric column counts as correct only if the document dialog contains the same core fact under the correct symbol and year context.
5. For numeric columns, treat the extraction as correct if the integer digits and the first decimal place are correct, unless the task clearly requires exact discrete identifiers.
6. Extra information in the dialog does not hurt correctness.
7. Count each column at most once.

TASK:

1. Return only the metric fields that are correctly supported by the dialog.

Constraints:

1. `correct_metric_fields` must contain only field names from the provided metric columns.
2. If none are correct, return an empty list.

Return JSON only:

```
{
  "correct_metric_fields": ["<metric field name>"]
}
```

A.7.8 Prompt Template of Judging RAG Information Extraction (The correct side)

Prompt: Judge — RAG Information Extraction (The correct side)

The user will provide:

1. Information (`list`) needed to answer the question (total_required = {`len_source`}): **[source_answer]**
2. Information extracted by the model (each part separated by `<next chunk>` is independent of each other): **[info]**

Evaluation Criteria:

1. If the information extracted by the model contains the key information with true entity from the reference, the correct extraction is added by one.
2. If the information does not match the entity or does not provide the entity information, it is judged incorrect and the number of correct extractions remains unchanged.
3. If the key information is missing or does not match the reference, it is judged incorrect and the number of correct extractions remains unchanged.
4. If the extraction is missing or does not match the reference information, it is judged incorrect and the number of correct extractions remains unchanged.
5. If the extracted information involves numerical values, the model is considered correct as long as it correctly extracts integer digits and the first decimal place.

Note: If the extraction of the model contains information other than the reference information or uses a different language, this does not affect the determination. Multiple correct extractions of the same information are counted only once.

TASK: Calculate the number of intersections between the information extracted by the model and the information that needs to be extracted.

Return JSON:

```
{
   "correct_extractions": <number of correct entries>,
   "total_required": len_source,
   "explanation": "..."
}
```

A.7.9 Prompt Template of Judging RAG Information Extraction (The incorrect side)

Prompt: Judge — RAG Information Extraction (The incorrect side)

The user will provide: The user will provide:

1. Information (`list`) needed to answer the question (total_required = {`len_source`}): **[source_answer]**
2. Information extracted by the model (each part separated by `<next chunk>` is independent of each other): **[info]**

Evaluation Criteria:

1. Treat the reference information list as the gold standard.
2. For each required entry in the reference list, check all extracted chunks:
3. If at least one chunk correctly contains the key information with the true entity, this entry is counted as correctly extracted.
4. If the key information is missing, conflicts with the reference (wrong entity/value), or is otherwise incorrect, this entry is counted as an error.
5. If the extracted information involves numerical values, the model is considered correct as long as it correctly extracts integer digits and the first decimal place.
6. Extra information that is not in the reference list, or uses a different language, does not affect the judgment. Only the correctness of the required entries is considered.
7. Multiple correct extractions of the same reference entry are counted only once.
8. error_extractions is the number of required entries that are incorrect or missing (i.e., entries in the reference list that do not have a correct extraction).

TASK:You are asked to consider each part of the model output separated by ¡next chunk¿ as a piece of information extracted by the model, and compute:

1. error_extractions = number of incorrect or missing entries among the required information.
2. total_required = len_source.

Return JSON only:

```
{
   "error_extractions": <number of incorrect or missing entries>,
   "total_required": {len_source},
   "explanation": "your simple explanation"
}
```

A.7.10 Prompt Template of Judging Final Answer

Prompt: Judge — Final Answer

The user will provide:

1. Multi-document question: **[question]**
2. Multi-document reference answer: **[final_answer]**
3. Model's answer: **[model_answer]**

Evaluation Criteria:

1. Model's answer will be judged as correct if it contains the key information in the reference answer.
2. If the key information is missing or the answer does not match the reference answer, the answer will be judged as incorrect.
3. If the answer is missing or does not match the reference answer, the answer is judged as incorrect.
4. If the answer involves a numerical value, it is considered correct as long as the model answers the whole number of digits and the first decimal place correctly.

Note: If a model answer contains additional information beyond the reference answer or use different language, this does not affect the judgment as correct.
TASK: Determine whether the model's final answer is correct with respect to the reference answer.
Return JSON:

```
{"is_correct": true/false, "explanation": "..."}
```

A.8 Benchmark Examples

Table 8: Some Examples of Our Benchmark

Simple	
Question	**Source Answer**
For China's A-share market, please provide the stock codes of the three companies with the highest capital adequacy ratio in 2021 from your knowledge base.	Stock code [symbol] had a capital adequacy ratio of xx in [year].
Please calculate the range of total operating costs for A-share companies in 2021 from your knowledge base.	Based on [year] data, the maximum total operating cost for A-share companies was xx, the minimum was xx, and the range was xx.
Based on the US stock market, which three companies had the highest proportion of other business revenue in 2021 in your knowledge base? List their stock symbols.	Company [symbol] had main business revenue of $xx million and other business revenue of $xx million in [year], with other revenue representing xx% of total revenue.
Based on the US stock market, which three companies had the highest proportion of other business revenue in 2022 in your knowledge base? List their stock symbols.	Company [symbol] had main business revenue of $xx million and other business revenue of $xx million in [year], with other revenue representing xx% of total revenue.
For China's A-share market, which companies in your knowledge base had their equity registration date in 2023 advanced by more than two weeks compared to 2022? Provide their stock codes.	Stock code [symbol] had an equity registration date of xx in [year] and xx in [year+1].
Complex	
Question	**Source Answer**
Based on the US stock market, report the top three companies by total assets in 2022, including their stock symbols and exact asset values.	Stock code [symbol] had total assets of $xx in [year].
For China's A-share market, please calculate the amount of change in the total cost of doing business in your knowledge base from 2021 to 2023 in terms of extreme variance.	Based on data from [year1] to [year2], the maximum change in total operating cost for A-share companies was xx, the minimum change was xx, and the range of changes was xx.
Please provide the stock codes of the three companies in your knowledge base with the highest pre-tax dividend per share growth rates in the 2022–2023 annual distribution and their specific growth rates.	Stock code [symbol] from year to year+1, the pre-tax dividend per share changes from xx to xx, with a growth rate of xx.
Based on the data in your knowledge base for the 2023 annual distribution, analyze the intervals between the share registration date and the ex-dividend date, and provide the ticker symbols of the three companies with the shortest intervals and the number of days between them.	Stock code [symbol] in [year] yearly distribution had a share registration date of xx, ex-rights and ex-dividend date of xx, with an interval of xx days.
Please provide the stock codes of the outlier firms whose rate of change in capital adequacy, total operating costs, or net income exceeds the knowledge base mean $\pm$ twice the standard deviation over the period 2022–2023.	Stock code [symbol] ratio of change in capital adequacy xx (knowledge base mean xx, standard deviation xx), ratio of change in total operating costs xx (knowledge base mean xx, standard deviation xx), and ratio of change in net income xx (knowledge base mean xx, standard deviation xx) over the period from [year] to [year+1].