

# CycleIE: Robust Document Information Extraction through Iterative Verification and Refinement

Zhengxuan Zhang<sup>1</sup>, Yin Wu<sup>1</sup>, Zhuowen Liang<sup>1</sup>, Haixun Wang<sup>3</sup>, Yuyu Luo<sup>1,2</sup>, Nan Tang<sup>1,2\*</sup>

<sup>1</sup>The Hong Kong University of Science and Technology (Guangzhou),

<sup>2</sup>The Hong Kong University of Science and Technology, <sup>3</sup>EvenUp

{zhang393, ywu450, zliang321}@connect.hkust-gz.edu.cn,

haixun@gmail.com, {yuyuluo, nantang}@hkust-gz.edu.cn

## Abstract

In document AI, reliable analytics require converting long, noisy (often multi-document) corpora into *heterogeneous structured data*—e.g., **tables** for numerical fields, **graphs** for relations, and **trees** for hierarchies. Yet, one-pass LLM extraction often yields incomplete or inconsistent structures because it lacks explicit verification and opportunities to revise earlier choices. We present CycleIE, an iterative information extraction (IE) framework that *closes the loop* between reasoning and acting by coupling ReAct with Monte Carlo Tree Search (MCTS). CycleIE employs a multi-agent workflow to iteratively retrieve, structure, extract, and refine content under verification guidance. By treating extraction as a feedback-driven search process, CycleIE systematically corrects omissions and inconsistencies that defeat one-pass methods, while remaining orthogonal to retrieval-augmented generation (RAG). Experiments on challenging document-based QA benchmarks demonstrate that CycleIE delivers **>10% relative improvements** over strong baselines, particularly in lengthy or multi-document contexts<sup>1</sup>.

## 1 Introduction

Large Language Models (LLMs) are powerful tools for document AI, yet reliable analytics over long and noisy corpora ultimately hinges on turning unstructured text into *heterogeneous structured data* that downstream methods can directly consume (e.g., tables, graphs, trees, and text chunks) (Wan et al., 2024; Zhu et al., 2025). Directly prompting LLMs to answer analytical queries without such structured extraction often yields vague, incomplete, or inaccurate outputs, especially on lengthy or multi-document contexts (Figure 1(a–b)). In particular, Figure 1(b) highlights superficial reasoning

driven by limited long-context handling (Li et al., 2025a; Wang et al., 2024b) and the inability to aggregate dispersed evidence (Zhang et al., 2024a; Ling et al., 2023). These observations motivate a structure-first approach that carefully extracts the right format for the task before reasoning.

In practice, the effectiveness of analytical applications fundamentally depends on accurate, **task-specific information extraction (IE)** from complex documents (Dagdelen et al., 2024; Appalaraju et al., 2021). This need spans finance (e.g., extracting key metrics from quarterly statements), law (e.g., compiling evidence across filings), and healthcare (e.g., summarizing patient records) (Kong et al., 2024; Nie et al., 2024; Ntinopoulos et al., 2025). Across these domains, the quality of downstream insights is largely determined by how effectively unstructured text is transformed into *the appropriate structured representation* for the target analysis (Li et al., 2025b; Fang et al., 2024).

**Challenges of Task-Specific IE.** Accurately extracting heterogeneous structures for specific analytics tasks requires synthesizing information dispersed over long and multi-source documents while choosing the right structure for the question. This often leads to incomplete extractions, inconsistent entities, or irrelevant content being included. These difficulties are amplified in the presence of complex layouts and long-range dependencies.

**Limitations of One-Pass IE.** State-of-the-art IE pipelines frequently adopt a **one-pass** strategy, directly prompting an LLM to produce the structured output in a single step (Liao et al., 2025; Wang et al., 2024a). As illustrated in Figure 1(c), even strong LLMs lack explicit verification and opportunities to revise earlier choices (structure selection, extraction scope, and values) within a single pass. Consequently, one-pass IE often yields incomplete or inaccurate structures, underscoring the need to move *beyond* direct, single-shot extraction (Xu et al., 2024; Madaan et al., 2023).

\* Nan Tang is the corresponding author

<sup>1</sup> The code is available: <https://github.com/HKUSTDial/CycleIE>

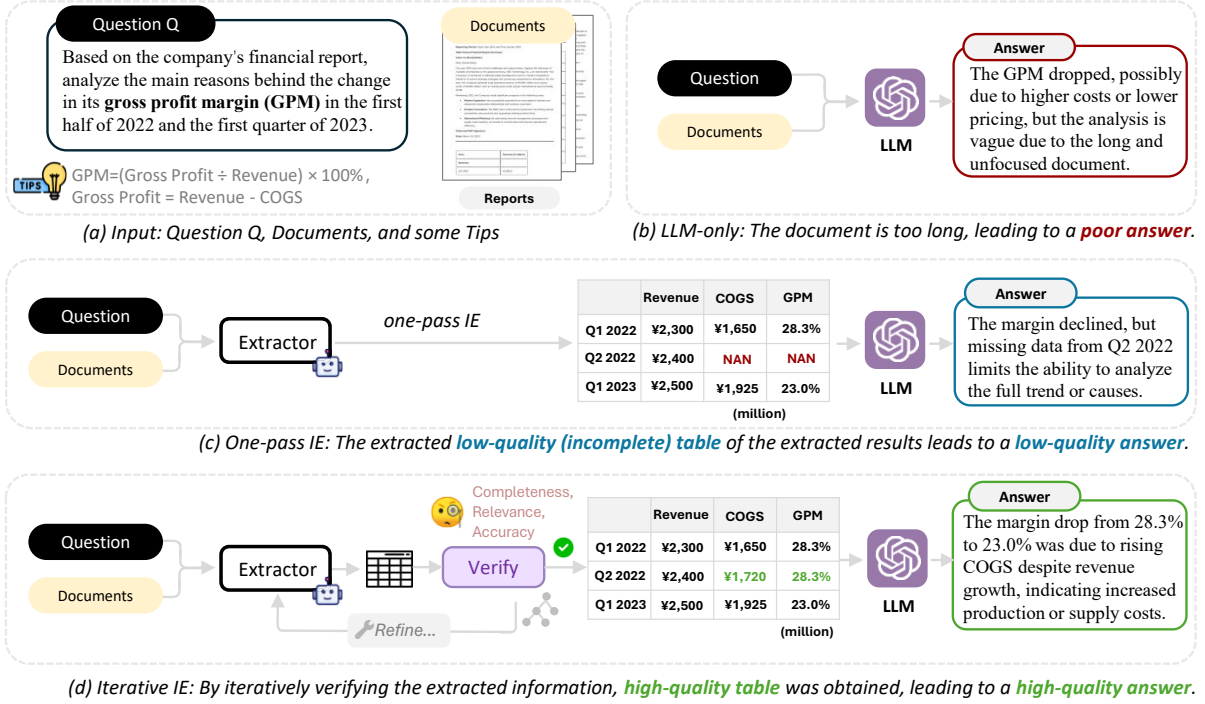


Figure 1: (a) Input question  $Q$  over given documents. (b) Without IE + LLMs for QA. (c) One-Pass IE + LLMs for QA. (d) Iterative IE + LLMs for QA.

**From One-Pass to Closed-Loop IE.** To address these limitations, we advocate a **closed-loop (iterative) IE** paradigm (Figure 1(d)): verification signals (completeness, relevance, accuracy) guide whether to re-retrieve, re-select structure, or re-extract, and the system repeats this loop until the structured output is auditable and sufficient for downstream reasoning. This perspective explicitly *decouples* structured extraction from reasoning, operates over user-provided documents (orthogonal to retrieval augmentation), and treats extraction as a verification-guided search over actions rather than a one-shot generation. We instantiate this paradigm with CycleIE, which couples ReAct with Monte Carlo Tree Search (MCTS) to select the next action under verification feedback and employs an anomaly-intervention mechanism to avoid verification deadlocks.

**Contributions.** Our contributions in this work are threefold:

(1) **Closed-Loop Iterative IE Framework.** We propose CycleIE, a novel multi-agent framework that shifts Information Extraction (IE) from single-pass generation to a verification-driven, iterative process. By decoupling extraction from reasoning, CycleIE reliably converts long, noisy documents into auditable structured data.

(2) **Verification-Guided Action Selection.** We

formulate extraction as a dynamic search over actions (retrieve, extract, verify, refine) by integrating ReAct with MCTS. Guided by explicit verification signals and supported by anomaly-intervention mechanisms, this approach effectively resolves deadlocks and enhances extraction fidelity.

(3) **Extensive Experiments and Findings.** Extensive evaluations on the *Loong* benchmark (up to 250K tokens) demonstrate CycleIE’s superiority. It surpasses StructRAG by up to 15.3 LLM-score points and 16 percentage points in Exact Match (EM). Compared to direct Qwen2-72B prompting, it achieves up to  $4.33\times$  higher EM with only a 15.3% performance drop at extreme context lengths.

## 2 Preliminary

**Task-Specific Information Extraction (IE).** In *document AI*, a data analytics task  $T$  operates over a user-provided collection of documents  $D = \{d_j\}$ . The goal is to extract the *heterogeneous structured data* needed to reliably solve  $T$  before performing any downstream reasoning.

**Unit Questions.** A complex task  $T$  can be decomposed (optionally by a planner) into unit questions  $\{Q_1, \dots, Q_n\}$ , where each  $Q_i$  targets a *single* structured form (e.g., a table of metrics, a relation graph, a hierarchy, or faithful text snippets). For

each  $Q_i$ , CycleIE iteratively (i) retrieves relevant segments  $R_i \subseteq D$ , (ii) selects an appropriate structure, (iii) extracts a candidate structure  $S_i$ , and (iv) verifies whether  $S_i$  is complete, relevant, and accurate enough to support downstream reasoning.

**Structured forms.** We consider four canonical structures that cover most analysis needs:

*Table* (■): Organizes data into tables for representing categorical or numerical information.

*Graph* (Ⓐ): Represents entities and their relations as nodes and directed/undirected edges.

*Tree* (Ⓐ): A hierarchical structure with a root and branches, suitable for nested relationships.

*Text Chunks* (≡): Coherent segments of text preserving semantic meaning for analysis.

**Evaluation protocol.** Because ground-truth structures are often unavailable, we evaluate the *end-to-end* effect of structured extraction by comparing *without IE*:  $\text{LLM}(T, D)$  (direct reasoning over raw documents) and *with IE*:  $\text{LLM}(T, E_{T,D})$  (reasoning over CycleIE-produced structures  $E_{T,D}$ ; if  $T$  is decomposed, CycleIE runs per  $Q_i$  and aggregates).

Final answers are scored against the **ground truth of  $T$** . This isolates the contribution of heterogeneous structured extraction to task performance.

**Remarks.** (a) *No document retrieval.* CycleIE operates on user-provided  $D$ ; it is orthogonal to RAG that retrieves external documents. (b) *End-to-end comparison.* For decomposed tasks  $\{Q_1, \dots, Q_n\}$  (e.g., via CoT or DeepSeek-R1), we compare final answers under the same reasoning model. (c) *Extensibility.* Additional formats (e.g., JSON, XML, domain schemas) can be supported without changing the loop.

### 3 The CycleIE Framework

CycleIE employs a multi-agent framework, where specialized **Agents** perform distinct **Actions** and communicate via structured **Signals** carrying feedback and intermediate results. These components are orchestrated through the **ReAct** paradigm, enabling adaptive and context-aware processing. To guide action selection, CycleIE integrates **Monte Carlo Tree Search (MCTS)** for evaluating and refining action sequences.

Given a complex analytical task  $T$ , CycleIE optionally decomposes it into unit questions  $\{Q_1, Q_2, \dots, Q_n\}$  via a **Planner** (an optional pre-processing module; see Appendix B), each triggering iterative information extraction and reasoning.

Extracted data is structured and fed into an LLM for sub-question answering, whose intermediate reasoning further informs the next steps. The following subsections detail the six core Agents (Section 3.1), ReAct-based orchestration (Section 3.2), and MCTS-based optimization (Section 3.3), for iterative IE of a unit question.

#### 3.1 The Agents of CycleIE

During the verification process, the system identifies potential causes of poor extraction quality. These may include incomplete information from retrieval that hinders reasoning, missing data detected during extraction, or unit questions that fail to retrieve relevant information and thus require refinement. We define these causes as **Signals**—informative cues that can be passed along in the reasoning loop to guide the optimization of retrieval results, improve extraction quality, or refine the unit questions themselves. Specifically, CycleIE employs six specialized **Agents** (illustrated in Figure 2(a)) to iteratively process each unit question  $Q_i$ :

**Retriever.** Initially identifies and retrieves relevant document segments  $R_i$  to address  $Q_i$ . It also receives explicit verification signals from the **Verifier**, prompting further retrieval iterations if initial segments are incomplete or inadequate.

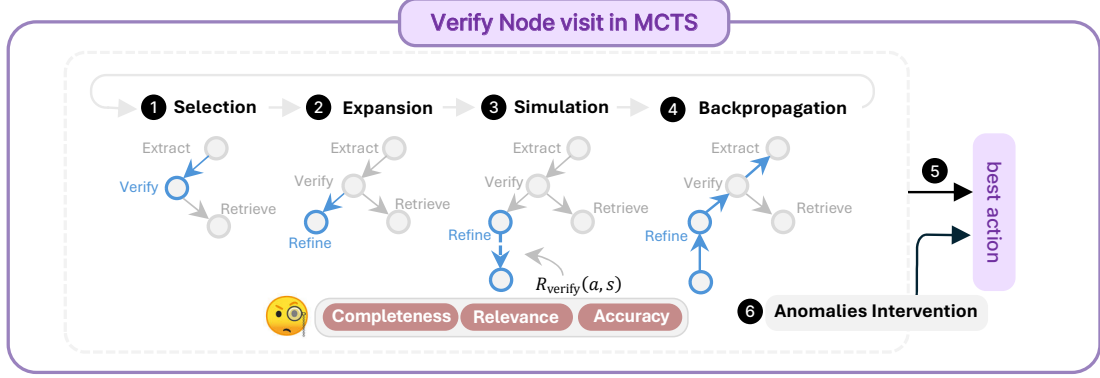
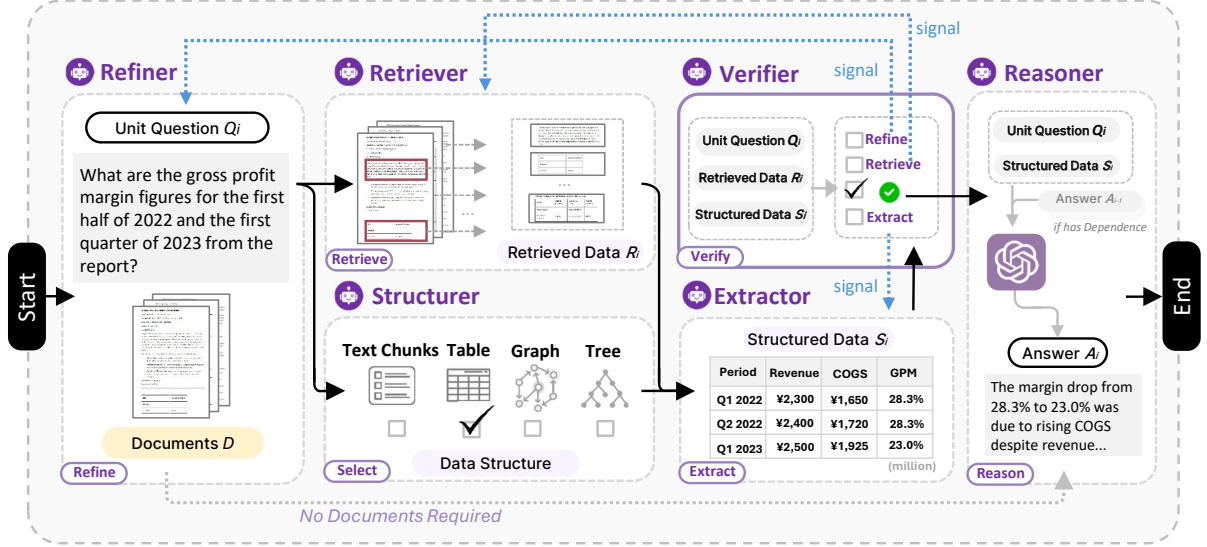
**Structurer.** Determines, in a single execution, the optimal structural representation (e.g., table, graph, tree, or text chunks) suitable for answering  $Q_i$ .

**Extractor.** Converts retrieved segments  $R_i$  into structured data  $S_i$ , adhering to the format selected by the **Structurer**. Upon receiving verification signals indicating issues, it iteratively adjusts extraction processes and refines the extracted data.

**Verifier.** Evaluates structured data  $S_i$  produced by the **Extractor** for completeness, relevance, and accuracy, generating explicit verification signals that guide iterative refinement by the **Retriever**, **Extractor**, and **Refiner**.

**Refiner.** Responds to verification signals from the **Verifier** to iteratively adjust queries, in order to improve extraction quality.

**Reasoner.** Executes once per unit question, utilizing the final verified structured data  $S_i$  to derive clear and interpretable intermediate answers  $A_i$ , guiding subsequent analytical processes  $Q_{i+1}$ .



(b) Illustration of MCTS decision-making when reaching the 'Verify' node.

Figure 2: Overview of CycleIE for one unit question.

For more detailed technical descriptions of each agent, see Appendix B.

### 3.2 Orchestrating Agents via ReAct

CycleIE orchestrates six specialized agents using the ReAct paradigm, enabling explicit inter-agent communication through **Actions** and **Signals** between agents. The reasoning state  $s$  maintains a unit question, retrieved segments, structured data, and verification feedback. Actions are executed by agents to advance extraction and reasoning, while Signals guide coordination and iterative refinement. Table 1 summarizes the six core actions and corresponding signals.

### 3.3 Action Selection with MCTS

CycleIE employs Monte Carlo Tree Search (MCTS) to optimize agent action sequences beyond the reactive bounds of ReAct. The decision process is framed as a tree-structured search prob-

Table 1: Core Actions and Verifier-Driven Signals in CycleIE

| Action                 | Agent       | Purpose / Trigger                        |
|------------------------|-------------|------------------------------------------|
| <b>refine</b>          | Refiner     | Improve subquestions                     |
| <b>retrieve</b>        | Retriever   | Rerun retrieval on bad segments          |
| <b>select</b>          | Structurer  | Choose best structure for question       |
| <b>extract</b>         | Extractor   | Extract structured content from segments |
| <b>verify</b>          | Verifier    | Check quality and emit signals           |
| <b>reason</b>          | Reasoner    | Final reasoning on structured data       |
| Signal                 | Flow        | Content                                  |
| <b>poor question</b>   | → Refiner   | Ask to reformulate or decompose          |
| <b>poor retrieval</b>  | → Retriever | Revise query and rerun                   |
| <b>poor extraction</b> | → Extractor | Improve extraction strategy              |

lem, where each **node**  $v$  represents an **action**  $a$  taken in a particular **state**  $s$ , *i.e.*, the current reasoning context as defined in Section 3.2. Each edge encodes a transition from one action-state pair  $(s, a)$  to another  $(s', a')$ , guided by transition rules  $\mathcal{T}(a)$  that define which actions are available after the current action—for example, the *verify* action is only valid after an *extract* action has populated relevant content. A **path** in the tree represents a full action sequence toward a terminal reasoning state. MCTS explores and simulates such paths, selecting the next action by balancing local verification signals and long-term reasoning rewards. To prevent reasoning deadlocks, we improve MCTS by introducing a new **intervention mechanism for anomaly handling**.

**Key Phases of MCTS.** MCTS-based decision-making in CycleIE consists of four key phases:

*1) Selection:* Starting from the root node  $(s, a)$ , CycleIE evaluates candidate nodes using the Upper Confidence Bound applied to Trees (UCT), which balances exploration and exploitation:

$$\text{UCT}(v) = \frac{Q(v)}{N(v)} + \alpha \sqrt{\frac{\ln(N(\text{parent}(v)))}{N(v)}} \quad (1)$$

where  $N(v)$  tracks the visit count of node  $v$  (initially 0), and  $Q(v)$  accumulates the total reward received by node  $v$ . By convention, if  $N(v) = 0$ , we set  $\text{UCT}(v) = +\infty$  so that every unvisited child is selected before any visited sibling. This allows informed trade-offs between deep analysis of already retrieved documents and broader searches for new information. The UCT score directs simulations toward promising nodes, thereby influencing the visit statistics used in final action selection.

*2) Expansion:* Once a promising node is selected, the system expands potential subsequent actions, filtered through a validity check conditioned on the current state-action pair  $(s, a)$ :

$$\mathcal{A}_{\text{valid}} = \{a' \mid a' \in \mathcal{T}(a) \wedge \text{Valid}_{\text{ReAct}}(a', s)\} \quad (2)$$

Here,  $\text{Valid}_{\text{ReAct}}(a', s)$  evaluates action plausibility within the ReAct framework, ensuring logical coherence (*e.g.*, prohibiting extraction actions before retrieval).

*3) Simulation:* From each newly expanded node, the system simulates action sequences until reaching a terminal state or a predefined simulation depth. Each simulated trajectory is scored by a comprehensive reward function:

$$R(a, s) = R_{\text{base}} + R_{\text{verify}} - R_{\text{cycle}} - R_{\text{penalty}} \quad (3)$$

The reward function combines intrinsic action value ( $R_{\text{base}}$ ), verification quality ( $R_{\text{verify}}$ , measuring completeness, relevance, and accuracy), penalties for repeated actions ( $R_{\text{cycle}}$ ), and task-specific operational costs ( $R_{\text{penalty}}$ ).

*4) Backpropagation:* The simulation reward  $R$  is used to update each node along the simulated path, propagating information upward:

$$N(v) \leftarrow N(v) + 1, \quad Q(v) \leftarrow Q(v) + R \quad (4)$$

**Best Action Selection.** After simulations, the system selects the optimal next action  $a_{\text{next}}$  based on node visit frequency:

$$a_{\text{next}} = \underset{a \in \mathcal{A}(s)}{\text{argmax}} N(s, a) \quad (5)$$

Here,  $\mathcal{A}(s)$  denotes the set of all possible actions from the current state  $s$ , and  $N(s, a)$  is the visit count of the node  $v$  that corresponds to taking action  $a$  in state  $s$  (*i.e.*,  $N(s, a) = N(v)$ ). The system executes the action with the highest visit count—shaped by prior UCT-guided simulations—thereby transitioning to a new state and informing subsequent steps.

**Intervention Mechanism for Anomalies.** While MCTS provides a principled framework for long-term planning, it may still fall into local loops—particularly verification deadlocks—where the system repeatedly cycles through *retrieve–extract–verify* without making progress. To mitigate this, we introduce an ‘Intervention Mechanism for Anomalies’ that dynamically overrides the action selected by standard MCTS when specific criteria are met:

$$a_{\text{next}} = \begin{cases} \text{reason}, & \text{if } C \wedge V_{\text{passed}} \\ \text{refine}, & \text{if } C \wedge N_{\text{total}} > \mu \wedge N_{\text{refine}} < \eta \\ \text{reason}, & \text{if } C \wedge N_{\text{total}} > \mu \wedge N_{\text{refine}} \geq \eta \\ a_{\text{MCTS}}, & \text{otherwise} \end{cases} \quad (6)$$

Here,  $C$  is a binary signal for cycle detection, triggered by repeated verification failures on similar content.  $V_{\text{passed}}$  indicates whether any verification has succeeded, while  $N_{\text{total}}$  and  $N_{\text{refine}}$  track the number of actions and refinements in the current episode. Thresholds  $\mu$  and  $\eta$  control the allowed action length and refinement depth. When  $C$  is active and  $N_{\text{total}} > \mu$ , the system prioritizes *refine* to avoid redundant attempts. If the refinement budget  $\eta$  is also exceeded, it defaults to *reasoning*.

with current content. This gating mechanism overrides the default MCTS flow when necessary, preventing verification loops and improving robustness in complex cases.

## 4 Experiment

This section evaluates CycleIE against competitive baselines and ablates its core components.

### 4.1 Experimental Setup

**Dataset.** This study evaluates various document-based data analysis tasks. We selected **four tasks** (spotlight localization, comparison, clustering, and reasoning chain) and **four document length settings** (ranging from 10K to 250K tokens) from the popular Loong benchmark (Wang et al., 2024b). As the document length increases, the information required to solve the tasks becomes more dispersed.

**Implementation.** All generation agents use Qwen2-72B-Instruct, while the long-context Retriever uses Qwen-Long, with other settings following the same configuration as Loong and StructRAG (Li et al., 2025b). Because the questions in the Loong dataset are complex and require decomposition into multiple steps for analysis, we enable the optional Planner preprocessing module (details in Appendix B) for question decomposition.

**Metrics.** We follow the standard settings of Loong (Wang et al., 2024b) and StructRAG, using their official codebases for evaluation. Two metrics are used: **LLM** (0–100), assessing output quality (accuracy, completeness, etc.); and **exact match rate (EM)**, which measures the proportion of perfect LLM scores, serving as an indicator of extraction quality, following the same protocol as Loong.

### 4.2 Baselines

**LLM-Only (no IE).** We evaluate five long-context LLMs: GPT-4o, Claude3.5-Sonnet, Kimi-Chat, Qwen2-72B-Instruct, and GLM4-9B.

**One-Pass IE Methods.** We repurpose four competitive RAG frameworks as one-pass extractors over the provided documents, omitting external retrieval: (1) **RAG** (Lewis et al., 2020) relies on standard chunk-based retrieval; (2) **RQ-RAG** (Chan et al., 2024) refines queries via LLM-based decomposition; (3) **GraphRAG** (Edge et al., 2024) supports reasoning via multi-layer knowledge graphs; and (4) **StructRAG** (Li et al., 2025b) routes diverse data formats for structured generation. Both GraphRAG and StructRAG use

Qwen2-72B-Instruct. We adopt LLM settings from Loong (Wang et al., 2024b) and RAG configurations from StructRAG.

### 4.3 Main Results

The main results are presented in Table 2, from which we derive the following key findings:

**1) Iterative IE vs. No IE.** Built upon the Qwen2-72B-Instruct model, CycleIE achieves substantial improvements by consistently extracting high-quality structured data across all document length groups, leading to significantly better analysis performance. On the short-document set (Set1), the overall LLM score improves from 60.11 to 78.71, and the EM metric rises from 0.29 to 0.48. The performance gains become even more pronounced as document length increases: 27.17 points on Set2, 36.39 on Set3, and 37.78 on Set4. These results show that high-quality structured data extraction can significantly enhance end-to-end reasoning accuracy, especially in longer contexts.

**2) Iterative IE vs. One-Pass IE.** CycleIE consistently outperforms RAG-based baselines across all document length groups. Compared to StructRAG, CycleIE achieves higher scores on Sets 1 through 4 (78.71, 72.88, 72.33, and 66.70, respectively), substantially surpassing StructRAG’s scores (69.43, 60.95, 57.92, and 51.42). The performance gap is even wider against RQ-RAG and GraphRAG—for example, on Set3, CycleIE exceeds RQ-RAG by 31.40 points and GraphRAG by 39.05 points. These results demonstrate that iterative information extraction is more effective than one-pass approaches, enabling better integration of retrieval and reasoning for document-level tasks.

**3) Performance across Different Task Types.** CycleIE shows strong performance across all task types. On the *Spotlight Locating* task, it consistently achieves high scores from Set1 to Set4 (91.75 to 83.19), significantly outperforming GPT-4o, which drops to 36.79 on Set4. In the *Comparison* task, CycleIE achieves 68.50 on Set4—12.88 points higher than StructRAG. For *Clustering*, it maintains a 12.45-point lead on Set1. Although GPT-4o slightly outperforms CycleIE on the *Chain of Reasoning* task for short documents, CycleIE surpasses it by 17.58 points on Set4. These results show that CycleIE achieves the best overall performance on all four document-length sets and leads on most task-set combinations.

**4) Robust Long-Document Handling.** CycleIE is robust against increasing document length. As

Table 2: Performance comparison across different models and methods on four sets of Document QA benchmarks. In particular, **Qwen2-72B-Instruct** serves as the base model in our method. We highlight the values corresponding to the **base** model, the **best** model, and the **2nd-best** model.

| Method             | Spotlight Locating |             | Comparison   |             | Clustering   |             | Chain of Reasoning |             | Overall      |             |
|--------------------|--------------------|-------------|--------------|-------------|--------------|-------------|--------------------|-------------|--------------|-------------|
|                    | LLM                | EM          | LLM          | EM          | LLM          | EM          | LLM                | EM          | LLM          | EM          |
| Set1 (10K-50K)     |                    |             |              |             |              |             |                    |             |              |             |
| GPT-4o             | 85.67              | <b>0.81</b> | 64.27        | 0.33        | 57.01        | 0.24        | <b>81.58</b>       | <b>0.55</b> | 70.40        | 0.44        |
| Claude3.5-Sonnet   | 60.85              | 0.55        | 69.07        | 0.47        | 58.63        | 0.13        | 68.57              | 0.50        | 63.69        | 0.37        |
| Qwen2-72B-Instruct | 68.49              | 0.55        | 60.60        | 0.37        | 47.08        | 0.08        | 70.39              | 0.36        | 60.11        | 0.29        |
| Kimi-Chat          | 81.11              | 0.74        | 46.70        | 0.20        | 47.84        | 0.07        | 53.77              | 0.17        | 55.02        | 0.24        |
| GLM4-9B-Chat       | 63.11              | 0.53        | 54.10        | 0.27        | 39.50        | 0.08        | 56.32              | 0.28        | 51.43        | 0.25        |
| RAG                | 51.08              | 0.35        | 44.53        | 0.27        | 37.96        | 0.05        | 53.95              | 0.35        | 46.11        | 0.23        |
| RQ-RAG             | 72.31              | 0.54        | 48.16        | 0.05        | 47.44        | 0.07        | 58.96              | 0.25        | 53.51        | 0.17        |
| GraphRAG           | 31.67              | 0.00        | 27.60        | 0.00        | 40.71        | 0.14        | 54.29              | 0.43        | 40.82        | 0.18        |
| StructRAG          | 74.53              | 0.47        | 75.58        | 0.47        | 65.13        | 0.23        | 67.84              | 0.34        | 69.43        | 0.35        |
| CycleIE (Ours)     | <b>91.75</b>       | 0.71        | <b>79.25</b> | <b>0.58</b> | <b>77.58</b> | <b>0.46</b> | 72.55              | 0.32        | <b>78.71</b> | <b>0.48</b> |
| Set2 (50K-100K)    |                    |             |              |             |              |             |                    |             |              |             |
| GPT-4o             | <b>86.76</b>       | <b>0.72</b> | 59.81        | 0.40        | 47.83        | 0.11        | 62.09              | <b>0.34</b> | 58.38        | 0.29        |
| Claude3.5-Sonnet   | 63.83              | 0.53        | 58.90        | 0.39        | 50.96        | 0.10        | 46.09              | 0.26        | 52.73        | 0.24        |
| Qwen2-72B-Instruct | 64.53              | 0.43        | 42.60        | 0.21        | 38.52        | 0.05        | 51.18              | 0.20        | 45.71        | 0.17        |
| Kimi-Chat          | 72.82              | 0.52        | 46.77        | 0.21        | 33.46        | 0.06        | 40.51              | 0.15        | 42.40        | 0.16        |
| GLM4-9B-Chat       | 65.04              | 0.54        | 41.80        | 0.23        | 30.72        | 0.02        | 42.34              | 0.17        | 40.19        | 0.17        |
| RAG                | 66.27              | 0.46        | 46.28        | 0.31        | 38.95        | 0.05        | 46.15              | 0.22        | 45.42        | 0.19        |
| RQ-RAG             | 57.35              | 0.35        | 50.83        | 0.16        | 42.85        | 0.03        | 47.60              | 0.10        | 47.09        | 0.10        |
| GraphRAG           | 24.80              | 0.00        | 14.29        | 0.00        | 37.86        | 0.00        | 46.25              | 0.12        | 33.06        | 0.03        |
| StructRAG          | 68.00              | 0.41        | 63.71        | 0.36        | 61.40        | 0.17        | 54.70              | 0.19        | 60.95        | 0.24        |
| CycleIE (Ours)     | 86.42              | 0.63        | <b>78.14</b> | <b>0.48</b> | <b>72.44</b> | <b>0.22</b> | <b>63.12</b>       | 0.19        | <b>72.88</b> | <b>0.32</b> |
| Set3 (100K-200K)   |                    |             |              |             |              |             |                    |             |              |             |
| GPT-4o             | 74.84              | <b>0.65</b> | 42.40        | 0.21        | 38.70        | 0.04        | 45.06              | 0.09        | 46.95        | 0.19        |
| Claude3.5-Sonnet   | 65.36              | 0.56        | 50.32        | 0.34        | 37.79        | 0.03        | 25.95              | 0.11        | 42.06        | 0.19        |
| Qwen2-72B-Instruct | 46.99              | 0.27        | 37.06        | 0.13        | 31.50        | 0.02        | 35.01              | 0.07        | 35.94        | 0.09        |
| Kimi-Chat          | 62.13              | 0.54        | 24.20        | 0.05        | 21.98        | 0.01        | 31.02              | 0.14        | 31.37        | 0.14        |
| GLM4-9B-Chat       | 69.19              | 0.56        | 37.99        | 0.18        | 26.63        | 0.01        | 32.30              | 0.09        | 37.36        | 0.16        |
| RAG                | 73.69              | 0.55        | 42.20        | 0.27        | 32.78        | 0.02        | 37.65              | 0.13        | 42.60        | 0.18        |
| RQ-RAG             | 50.50              | 0.13        | 44.62        | 0.00        | 36.98        | 0.00        | 36.79              | 0.07        | 40.93        | 0.05        |
| GraphRAG           | 15.83              | 0.00        | 27.40        | 0.00        | 42.50        | 0.00        | 43.33              | 0.17        | 33.28        | 0.04        |
| StructRAG          | 68.62              | 0.44        | 57.74        | 0.35        | 58.27        | 0.10        | 49.73              | 0.13        | 57.92        | 0.21        |
| CycleIE (Ours)     | <b>83.01</b>       | 0.50        | <b>75.48</b> | <b>0.41</b> | <b>72.01</b> | <b>0.25</b> | <b>62.59</b>       | <b>0.19</b> | <b>72.33</b> | <b>0.31</b> |
| Set4 (200K-250K)   |                    |             |              |             |              |             |                    |             |              |             |
| GPT-4o             | 36.79              | 0.19        | 23.97        | 0.08        | 30.40        | 0.00        | 32.89              | 0.07        | 31.11        | 0.07        |
| Claude3.5-Sonnet   | 36.91              | 0.24        | 28.82        | 0.05        | 28.68        | 0.00        | 28.77              | 0.08        | 30.51        | 0.08        |
| Qwen2-72B-Instruct | <u>33.18</u>       | 0.16        | 26.59        | 0.08        | <u>29.84</u> | 0.01        | <u>25.81</u>       | 0.04        | <u>28.92</u> | 0.06        |
| Kimi-Chat          | 20.17              | 0.12        | 9.17         | 0.00        | 5.65         | 0.00        | 22.61              | 0.11        | 13.50        | 0.05        |
| GLM4-9B-Chat       | 15.67              | 0.12        | 21.33        | 0.05        | 12.35        | 0.00        | 21.04              | 0.05        | 16.84        | 0.05        |
| RAG                | 52.17              | 0.24        | 24.60        | 0.10        | 26.78        | 0.00        | 17.79              | 0.00        | 29.29        | 0.07        |
| RQ-RAG             | 29.17              | 0.08        | 40.36        | 0.00        | 26.92        | 0.00        | 34.69              | 0.00        | 31.91        | 0.01        |
| GraphRAG           | 17.50              | 0.00        | 26.67        | 0.00        | 20.91        | 0.00        | 33.67              | <b>0.33</b> | 23.47        | 0.05        |
| StructRAG          | 56.87              | 0.19        | 55.62        | 0.25        | 56.59        | 0.00        | 35.71              | 0.05        | 51.42        | 0.10        |
| CycleIE (Ours)     | <b>83.19</b>       | <b>0.53</b> | <b>68.50</b> | <b>0.35</b> | <b>67.58</b> | <b>0.18</b> | <b>50.47</b>       | 0.09        | <b>66.70</b> | <b>0.26</b> |

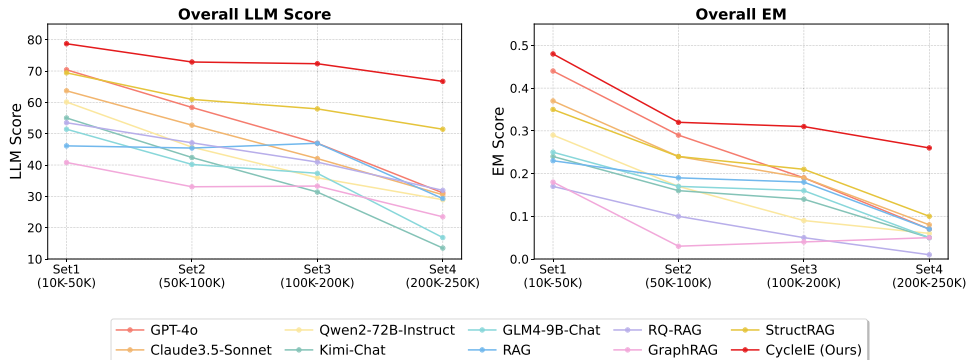


Figure 3: Visualization of overall LLM and EM scores of different methods across sets 1–4.

Table 3: Results of the ablation study.

|                    | Set 1        |             | Set 2        |             | Set 3        |             | Set 4        |             |
|--------------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|
|                    | LLM          | EM          | LLM          | EM          | LLM          | EM          | LLM          | EM          |
| CycleIE            | <b>78.71</b> | <b>0.48</b> | <b>72.88</b> | 0.32        | <b>72.33</b> | <b>0.31</b> | <b>66.70</b> | 0.26        |
| <i>w/o verify</i>  | 73.75        | 0.43        | 71.40        | <b>0.36</b> | 67.12        | 0.30        | 64.16        | <b>0.27</b> |
| <i>w/o extract</i> | 76.00        | 0.45        | 70.43        | 0.35        | 67.14        | 0.29        | 60.18        | 0.22        |
| LLM-only           | 60.11        | 0.29        | 45.71        | 0.17        | 35.94        | 0.09        | 28.92        | 0.06        |

shown in Figure 3, its overall score decreases by only 15.3% from Set1 (78.71) to Set4 (66.70), compared to much sharper drops for GPT-4o, Claude3.5-Sonnet, and Qwen2-72B-Instruct (51.9–55.8%), and 25.9% for StructRAG. On Set4, CycleIE outperforms StructRAG and GPT-4o by 15.28 and 35.59 points, respectively. These results underscore CycleIE’s particular strength in handling long-document inputs, maintaining high reasoning accuracy even as context length increases.

#### 4.4 Ablation Study

To illustrate how each component contributes to our multi-agent workflow, we present representative success and failure cases of CycleIE.

**Criticality of the Verification Module and Inter-Module Synergy.** The verification module is essential for the answer quality. On Set1, its removal (*w/o verify*, 73.75) caused a larger performance drop than removing the extraction module (*w/o extract*, 76.00), highlighting its role in filtering noisy information. Although ablated systems still outperform the base model (e.g., 67.12 by *w/o verify* on Set3 compared to 35.94 from Qwen2-72B), only the full system consistently achieves the highest LLM scores. On Set2, it reached 72.88, exceeding both *w/o verify* (71.40) and *w/o extract* (70.43), supporting their complementarity in terms of LLM score. The EM results are less consistent: *w/o verify* slightly exceeds the full model on Sets 2 and 4, suggesting a trade-off between average answer quality and perfect-score frequency.

**Amplified Benefits of Modularity in Long-Document Processing.** The benefits of modular design grow with document length. From Set1 to Set4, the performance gap between the full system and *w/o extract* widened from 2.71 to 6.52. The impact of removing verification varies across length settings and is largest on Set3, where the LLM score drops by 5.21 points. The larger gaps on Sets 3–4 for the extraction ablation suggest that extraction becomes particularly important in some long-context settings; the effect of verification is less monotonic.

## Question

Please read the financial statements of the companies above and answer the following questions based solely on the information provided... **What is the amount of Gree Electric's special reserves?**

Documents

## Standard Answer:

The amount of Gree Electric's special reserves is **26,924,166.37** yuan.

| Method        | Output                                                                                                                                                                                                                                                                                                                                                                       |         |                 |               |               |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|-----------------|---------------|---------------|
| w/o extract   | <p>Starting analysis for query...</p>  <p>The special reserve amount for Gree Electric as reported in its 2024 Q1 report is <b>687,042,264.52</b> yuan... ❌</p>                                                                                                                           |         |                 |               |               |
| w/o verify    | <p>Starting analysis for query...Identify the 'special reserve' value for Gree Electric...</p> <table border="1"> <thead> <tr> <th>Company</th><th>Special Reserve</th></tr> </thead> <tbody> <tr> <td>Gree Electric</td><td>Not mentioned</td></tr> </tbody> </table> <p>The special reserve value for Gree Electric is <b>unavailable</b> as it was not mentioned... ❌</p> | Company | Special Reserve | Gree Electric | Not mentioned |
| Company       | Special Reserve                                                                                                                                                                                                                                                                                                                                                              |         |                 |               |               |
| Gree Electric | Not mentioned                                                                                                                                                                                                                                                                                                                                                                |         |                 |               |               |
| CycleIE       | <p>Starting analysis for query... Identify the special reserve amount for Gree Electric...</p> <table border="1"> <thead> <tr> <th>Company</th><th>Special Reserve</th></tr> </thead> <tbody> <tr> <td>Gree Electric</td><td>26,924,166.37</td></tr> </tbody> </table> <p>The special reserve amount for Gree Electric is <b>26,924,166.37</b> yuan. ✅</p>                   | Company | Special Reserve | Gree Electric | 26,924,166.37 |
| Company       | Special Reserve                                                                                                                                                                                                                                                                                                                                                              |         |                 |               |               |
| Gree Electric | 26,924,166.37                                                                                                                                                                                                                                                                                                                                                                |         |                 |               |               |

Figure 4: A running example of CycleIE and its variants on the Loong dataset (see Appendix D).

#### 4.5 Case Study

Figure 4 shows why both targeted extraction and reasoning-aware verification are essential for financial QA. The *w/o extract* variant, forced to read raw tables, is overwhelmed by noise and picks an unrelated value, while the *w/o verify* variant finds partial evidence but cannot assess completeness and wrongly concludes the value is missing. In contrast, CycleIE merges both strengths: the extractor isolates focused candidate values, filtering distracting content, and the verifier checks their consistency with the query and context. This coordinated process lets CycleIE correctly identify Gree Electric’s reserve amount, showing that extraction or verification alone is insufficient without their interaction in noisy or irregular disclosures.

#### 4.6 Impact of Backbone Model Size

To assess whether CycleIE relies heavily on large foundation models, we replaced the default Qwen2-72B backbone with smaller variants (7B to 32B) across all document settings. As shown in Table 4, these experiments yield three key findings:

**Effectiveness at Small Scales.** CycleIE remains effective even with 7B models (e.g., Qwen2.5-7B achieves 55.00 LLM score on Set 1). This demonstrates that our iterative verification mechanism functions independently of model size and does not require a massive foundation model to operate.

**Scalability with Model Capacity.** CycleIE’s absolute performance generally improves with back-

Table 4: Performance comparison of CycleIE using different backbone model sizes across four document-length sets. Qwen2-72B is the default backbone used in our experiments.

| Backbone Model          | Set 1        |             | Set 2        |             | Set 3        |             | Set 4        |             |
|-------------------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|
|                         | LLM          | EM          | LLM          | EM          | LLM          | EM          | LLM          | EM          |
| Qwen1.5-7B              | 43.94        | 0.09        | 47.59        | 0.16        | 44.27        | 0.08        | 34.52        | 0.05        |
| Qwen2.5-7B              | 55.00        | 0.30        | 51.72        | 0.29        | 42.92        | 0.21        | 27.14        | 0.05        |
| Qwen2.5-14B             | 66.55        | 0.42        | 58.19        | 0.26        | 52.92        | 0.25        | 40.24        | 0.14        |
| Qwen2.5-32B             | 63.48        | 0.24        | 60.86        | 0.36        | 58.75        | 0.35        | 52.05        | 0.19        |
| <b>Qwen2-72B (Ours)</b> | <b>78.71</b> | <b>0.48</b> | <b>72.88</b> | <b>0.32</b> | <b>72.33</b> | <b>0.31</b> | <b>66.70</b> | <b>0.26</b> |

bone capacity (7B  $\rightarrow$  72B), especially on longer-document sets. The trend is not strictly monotonic for every set or metric (e.g., on Set1, Qwen2.5-14B scores 66.55 versus 63.48 for Qwen2.5-32B).

**Context Limitations of Small Models.** While smaller models perform well on short-to-medium documents, they exhibit significant performance drops on ultra-long contexts (Set 4). These results are consistent with possible context-capacity limitations, although architecture and model-generation differences may also contribute.

## 5 Related Work

**Document-based Data Analytics.** Document-based information extraction (IE) has evolved from early structured QA systems (Herzig et al., 2020; Eisenschlos et al., 2020) to multimodal pretraining models (Appalaraju et al., 2021; Huang et al., 2022) that capture both textual and layout information. Recent advances have further incorporated instruction tuning (Wu et al., 2023; Feng et al., 2023a), structure-aware modeling (Wang et al., 2024a; Masry and Hajian, 2024; Feng et al., 2023b), and structured reasoning over long or multimodal documents (Liang et al., 2026a; Wang et al., 2026). Beyond extracting isolated fields or tables, recent studies have increasingly emphasized preserving relational semantics, structural consistency, and verifiability when transforming documents into structured data (Zhang et al., 2026a,b; Liang et al., 2026b). This direction is further extended toward analytics over heterogeneous data workspaces, where information must be integrated and verified across documents and structured sources (Li et al., 2026).

Overall, these efforts reflect a broader shift from one-pass extraction toward structured and verifiable document analytics. Our work builds on this direction by enabling dynamic, iterative extraction and verification through specialized agents.

**Reasoning and Multi-Agent Workflows.** To ad-

dress complex tasks, reasoning frameworks have moved beyond simple retrieval-augmented generation (RAG) toward explicit intermediate structures, such as graphs and trees (Edge et al., 2024; Zhang et al., 2024b; Li et al., 2025b; Jeong et al., 2024; Besta et al., 2024), which facilitate structured evidence organization and multi-step reasoning. Related work has also explored grounding, verification, and interpretable reasoning over heterogeneous or multimodal evidence to improve the trustworthiness of analytical results (Tang et al., 2024; Wu et al., 2026). Meanwhile, methods such as ReAct (Yao et al., 2023) promote interleaved reasoning and tool use, while multi-agent frameworks (Hong et al., 2024; Xie et al., 2024) enable role-based planning and collaboration, alongside broader efforts toward adaptive and cost-aware execution (Luo et al., 2026).

Unlike prior work that primarily focuses on either task-specific document processing or generic agent orchestration, our method combines structured document analysis with a document-specialized multi-agent workflow for iterative extraction, verification, and refinement.

## 6 Conclusion

In this work, we introduce CycleIE, an iterative information extraction framework integrating ReAct and MCTS, to address the shortcomings of one-pass extraction in complex documents. Through iterative verification and refinement, CycleIE achieves over a 10% performance uplift on challenging benchmarks, offering a novel approach for obtaining high-quality structured information. Therefore, a key direction for future work is optimizing the search efficiency of CycleIE, potentially through more effective pruning strategies or heuristic-guided navigation.

**Limitations.** While CycleIE demonstrates strong performance with powerful LLMs, its effectiveness may decline when using smaller models as agents in zero-shot settings. In such cases, additional prompt tuning or lightweight fine-tuning may be necessary to maintain robustness and generalization across diverse domains.

**Ethics Statement.** While CycleIE improves extraction fidelity, its iterative architecture increases computational overhead. Deployments must ensure strict data privacy when processing sensitive documents. Furthermore, despite our verification mechanism, CycleIE remains an assistive tool requiring human-in-the-loop oversight for high-stakes decisions.

## References

- Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. 2021. DocFormer: End-to-end transformer for document understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 993–1003.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.
- Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. RQ-RAG: Learning to refine queries for retrieval augmented generation. In *First Conference on Language Modeling*.
- John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S. Rosen, Gerbrand Ceder, Kristin A. Persson, and Anubhav Jain. 2024. [Structured information extraction from scientific text with large language models](#). *Nature Communications*, 15(1):1418.
- Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitanaky, Robert Osazuwa Ness, and Jonathan Larson. 2024. [From local to global: A graph RAG approach to query-focused summarization](#). *arXiv preprint arXiv:2404.16130*.
- Julian Martin Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. [Understanding tables with intermediate pre-training](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296. Association for Computational Linguistics.
- Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. 2024. [Large language models \(LLMs\) on tabular data: Prediction, generation, and understanding—a survey](#). *arXiv preprint arXiv:2402.17944*.
- Hao Feng, Zijian Wang, Jingqun Tang, Jinghui Lu, Wengang Zhou, Houqiang Li, and Can Huang. 2023a. [UniDoc: A universal large multimodal model for simultaneous text detection, recognition, spotting and understanding](#). *arXiv preprint arXiv:2308.11592*.
- Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2023b. Layout-GPT: Compositional visual planning and generation with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 18225–18250.
- Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. [TaPas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333. Association for Computational Linguistics.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiaowu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.
- Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. [LayoutLMv3: Pre-training for document AI with unified text and image masking](#). In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4083–4091.
- Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. [Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050. Association for Computational Linguistics.
- Yaxuan Kong, Yuqi Nie, Xiaowen Dong, John M. Mulvey, H. Vincent Poor, Qingsong Wen, and Stefan Zohren. 2024. [Large language models for financial and investment management: Applications and benchmarks](#). *Journal of Portfolio Management*, 51(2):162–210.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020.

- Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.
- Boyan Li, Zhuowen Liang, Yupeng Xie, Xiaotian Lin, Tianqi Luo, Xinyu Liu, Yizhang Zhu, Zhangyang Peng, Yuan Li, Zhengxuan Zhang, Jiayi Zhang, Nan Tang, Guoliang Li, and Yuyu Luo. 2026. [DataSpace: Benchmarking data agents for verifiable analytics over heterogeneous workspaces](#). *arXiv preprint arXiv:2608.03451*.
- Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2025a. Long-context LLMs struggle with long in-context learning. *Transactions on Machine Learning Research*.
- Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. 2025b. StructRAG: Boosting knowledge intensive reasoning of LLMs via inference-time hybrid information structurization. In *The Thirteenth International Conference on Learning Representations*.
- Zhuowen Liang, Xiaotian Lin, Zhengxuan Zhang, Yuyu Luo, Haixun Wang, and Nan Tang. 2026a. Long-document QA with chain-of-structured-thought and fine-tuned SLMs. In *International Conference on Learning Representations*.
- Zhuowen Liang, Zhengxuan Zhang, Jiayang Wang, Jiazhao Chen, and Nan Tang. 2026b. [Beyond tables: Doc2DB-Bench for relationally faithful document-to-database construction](#). *arXiv preprint arXiv:2608.08459*.
- Wenhui Liao, Jiapeng Wang, Hongliang Li, Chengyu Wang, Jun Huang, and Lianwen Jin. 2025. DocLayLLM: An efficient multi-modal extension of large language models for text-rich document understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4038–4049.
- Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. In *Advances in Neural Information Processing Systems*, volume 36, pages 36407–36433.
- Yuxiang Luo, Chen Wang, and Nan Tang. 2026. [A risk decomposition framework for pre-hoc fine-tuning prediction](#). In *Forty-third International Conference on Machine Learning*.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594.
- Ahmed Masry and Amir Hajian. 2024. [LongFin: A multimodal document understanding model for long financial domain documents](#). *arXiv preprint arXiv:2401.15050*.
- Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M. Mulvey, H. Vincent Poor, Qingsong Wen, and Stefan Zohren. 2024. [A survey of large language models for financial applications: Progress, prospects and challenges](#). *arXiv preprint arXiv:2406.11903*.
- Vasileios Ntinopoulos, Hector Rodriguez Cetina Biefer, Igor Tudorache, Nestoras Papadopoulos, Dragan Odavic, Petar Risteski, Achim Haeussler, and Omer Dzemali. 2025. Large language models for data extraction from unstructured and semi-structured electronic health records: A multiple model performance evaluation. *BMJ Health & Care Informatics*, 32(1):e101139.
- Nan Tang, Chenyu Yang, Zhengxuan Zhang, Yuyu Luo, Ju Fan, Lei Cao, Sam Madden, and Alon Y. Halevy. 2024. Symphony: Towards trustworthy question answering and verification using RAG over multi-modal data lakes. *IEEE Data Engineering Bulletin*, 48(4):135–146.
- Mengting Wan, Tara Safavi, Sujay Kumar Jauhar, Yujin Kim, Scott Counts, Jennifer Neville, Siddharth Suri, Chirag Shah, Ryen W. White, Longqi Yang, Reid Andersen, Georg Buscher, Dhruv Joshi, and Nagu Rangan. 2024. [TnT-LLM: Text mining at scale with large language models](#). In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5836–5847.
- Dongsheng Wang, Natraj Raman, Mathieu Sibue, Zhiqiang Ma, Petr Babkin, Simerjot Kaur, Yulong Pei, Armineh Nourbakhsh, and Xiaomo Liu. 2024a. [DocLLM: A layout-aware generative language model for multimodal document understanding](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8529–8548. Association for Computational Linguistics.
- Liangwei Wang, Zhengxuan Zhang, Yi-Fan Cao, Fuguee Tsung, and Yuyu Luo. 2026. [TableTale: Reviving the narrative interplay between data tables and text in scientific papers](#). In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, pages 1–17. Association for Computing Machinery.
- Minzheng Wang, Longze Chen, Fu Cheng, Shengyi Liao, Xinghua Zhang, Bingli Wu, Haiyang Yu, Nan Xu, Lei Zhang, Run Luo, Yunshui Li, Min Yang, Fei Huang, and Yongbin Li. 2024b. [Leave no document behind: Benchmarking long-context LLMs with extended multi-doc QA](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5627–5646. Association for Computational Linguistics.
- Sijin Wu, Dan Zhang, Teng Hu, and Shikun Feng. 2023. [DocPrompt: Large-scale continue pretrain for](#)

zero-shot and few-shot document question answering. *arXiv preprint arXiv:2308.10959*.

Yin Wu, Zhengxuan Zhang, Jiayu Chen, Chang Xu, Yuyu Luo, Nan Tang, and Hui Xiong. 2026. [Explainable oracle bone script recognition via multi-modal pictographic reasoning](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 39460–39467.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, Leo Z. Liu, Yiheng Xu, Hongjin Su, Dongchan Shin, Caiming Xiong, and Tao Yu. 2024. OpenAgents: An open platform for language agents in the wild. In *First Conference on Language Modeling*.

Qiancheng Xu, Yongqi Li, Heming Xia, and Wenjie Li. 2024. [Enhancing tool retrieval with iterative feedback from large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9609–9619. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. 2024a. Chain of agents: Large language models collaborating on long-context tasks. In *Advances in Neural Information Processing Systems*, volume 37, pages 132208–132237.

Zhengxuan Zhang, Zhuowen Liang, Jiazhao Chen, Haixun Wang, and Nan Tang. 2026a. [Document-to-database: Extraction meets relational semantics](#). *Proceedings of the VLDB Endowment*, 19(9):2522–2535.

Zhengxuan Zhang, Zhuowen Liang, Haixun Wang, and Nan Tang. 2026b. DataMosaic: An interactive demonstration of constraint-driven document-to-database construction. *Proceedings of the VLDB Endowment*, 19(12):4570–4573.

Zhengxuan Zhang, Yin Wu, Yuyu Luo, and Nan Tang. 2024b. [MAR: Matching-augmented reasoning for enhancing visual-based entity question answering](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530. Association for Computational Linguistics.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2025. [Large language models for information retrieval: A survey](#). *ACM Transactions on Information Systems*, 44(1):1–54.

## A Efficiency and Latency Analysis

To assess the practical deployment feasibility of CycleIE, we analyze its runtime cost and API call frequency compared to direct prompting and human annotation. Table 5 summarizes the average time cost per question.

**Computational Cost vs. Performance.** Our system typically decomposes a complex query into approximately 2.5 unit questions, triggering a chain of retrieval, extraction, and reasoning actions. This process results in an average of 12 API calls to Qwen2-72B-Instruct per query. With an average latency of 4 seconds per call, the total end-to-end latency is roughly 48 seconds. While this introduces a significant overhead compared to the  $\sim 4$  seconds required for standard one-shot prompting, this additional compute is essential for the performance gains. As shown in Table 6, the extra inference time yields substantial improvements in answer quality, with EM scores improving by up to  $4.33\times$  on complex document sets (e.g., Set4).

**User Experience and Scalability.** To mitigate the perceived latency, CycleIE employs real-time streaming, displaying intermediate reasoning steps to the user instantly rather than blocking until completion. Furthermore, compared to human experts—who require 3–10 minutes per question depending on document length—CycleIE offers a highly scalable solution that is roughly  $4\text{--}13\times$  faster, striking a practical balance between automation efficiency and expert-level accuracy.

## B Agent Implementation Details

### B.1 Refiner

The Refiner agent optimizes the unit question.

**Implementation.** The agent uses the `step_optimize` function to improve the current step through rewording, keyword optimization, and reasoning adjustment. If refinement is insufficient, it can split the step into multiple sub-questions to ensure a clearer and more actionable process.

Table 5: Per-question average runtime.

| Setting         | Avg. API Calls | Avg. Time (s) | Notes                    |
|-----------------|----------------|---------------|--------------------------|
| CycleIE (ours)  | ~12            | ~48           | 4s per call to Qwen2-72B |
| Qwen2-72B       | 1              | ~4            | One-shot QA              |
| Human Annotator | –              | 180–600       | 3–10 mins                |

Table 6: Answer quality comparison between direct Qwen2-72B-Instruct (LLM-only) and CycleIE across document sets of Loong dataset.

| Set  | LLM Score |         | EM       |         | Improvement Ratio |       |
|------|-----------|---------|----------|---------|-------------------|-------|
|      | LLM-Only  | CycleIE | LLM-Only | CycleIE | LLM Score         | EM    |
| Set1 | 60.11     | 78.71   | 0.29     | 0.48    | 1.31×             | 1.66× |
| Set2 | 45.71     | 72.88   | 0.17     | 0.32    | 1.59×             | 1.88× |
| Set3 | 35.94     | 72.33   | 0.09     | 0.31    | 2.01×             | 3.44× |
| Set4 | 28.92     | 66.70   | 0.06     | 0.26    | 2.31×             | 4.33× |

### Refiner Prompt

You are a task optimization expert, assisting in refining the current step to better answer the question.

MAIN QUESTION: {original\_query}

CURRENT STEP BEING PROCESSED: {step\_description}

INFORMATION WE HAVE ALREADY EXTRACTED: {extracted\_text}

CONTENT WE HAVE RETRIEVED FROM DOCUMENTS: {retrieved\_text}

VERIFICATION RESULTS: {verification\_text}

First, analyze whether the current step is effective. If it is valid but can be improved, try to:

- Improve the step description to make it clearer and more specific.
- Optimize the search keywords to make them more targeted.
- Adjust the reasoning process and dependencies of the step.
- Only consider generating sub-questions if the current step cannot be optimized.

Please return your optimization results in the following JSON format:

```
{
  "steps": [
    {
      "step_number": <step_idx>,
      "description": "Revised step description",
      "requires_document": true/false,
      "search_keywords": ["keyword1", "keyword2", ...],
      "reasoning": "Justification for why this step needs to be optimized this way",
      "optimization_type": "refine_current_step" or "generate_sub_questions"
    }
  ]
}
```

If you choose to generate sub-questions, include 2 to 3 new sub-steps in the steps section that will help answer the main question.

complex queries into a series of executable steps. It analyzes the original query, considers available documents, and produces a structured plan with dependencies between steps.

### Planner Agent Prompt

As an expert research assistant, analyze and break down this query into clear executable steps:

ORIGINAL QUERY: "{query}" {document\_info}

For each step, provide: 1. A clear description of what needs to be done 2. Whether document retrieval is needed for this step 3. Specific keywords to search for (if document retrieval is needed) 4. Dependencies on previous steps (if any) 5. Use as few steps as possible; ideally, complete it in a single step. If it can be done in one step, there's no need for a second step to summarize the first, and just use the original query.

Important: For each step, consider if it depends on results from previous steps. If it does, list those dependencies.

CONSTRAINT: A step can ONLY depend on steps with LOWER step numbers (i.e., steps that come before it). For example, Step 3 can depend on Steps 1 and 2, but not on Steps 4, 5, etc.

The last step must be a synthesis step that integrates results from previous steps to provide the final answer. Also, the last step description should concat the ORIGINAL QUERY in the last sentence.

Output in JSON format with the following structure:

```
{
  "steps": [
    {
      "step_number": 1,
      "description": "...",
      "requires_document": true/false,
      "search_keywords": ["keyword1", "keyword2", ...],
      "reasoning": "...",
      "depends_on_steps": [list of step numbers depends on]
    },
    ...
  ]
}
```

## B.2 Planner Agent (Optional Preprocessing)

The Planner is an optional preprocessing module, separate from the six core agents in the iterative IE loop. It decomposes complex queries into executable unit questions (subtasks) and can replan when necessary.

**Implementation.** The Planner agent utilizes the `task_decomposer` function to break down

## B.3 Retriever Agent

The Retriever agent is responsible for retrieving relevant documents or document chunks based on the current task requirements. It features two retrieval modes. The first uses vector retrieval: user input documents are encoded into vectors and stored in

a vector database. The second employs Long Context LLM retrieval, leveraging prompts to guide the LLM. The second mode is the default.

**Implementation.** The Retriever agent employs the `context_retriever` function to search for and retrieve relevant document contexts. It leverages a document manager to perform semantic search based on keywords and task descriptions.

#### Retriever Agent Prompt

- Each fragment must: - Retain complete semantic meaning (e.g., full paragraphs, full tables, bullet lists, or code blocks). - Preserve structured content (do not cut off tables, item lists, or inline references). - Come from a document that clearly contains information relevant to the question.  
- Limitations: - Each fragment should be no longer than 500 tokens (truncate carefully if necessary, without breaking structure). - Return at most 3 fragments, prioritized by most relevant first. - If no content is relevant, return nothing.

Output format (one per line, sorted by relevance):

```
<filename>: <relevant content>
<filename>: <relevant content>
...
```

- Do not summarize, generate answers, or include explanations.  
- Only output the selected fragments in the format above.

Question: {search\_query}  
Documents: {retrieve\_content}

## B.4 Structurer Agent

The Structurer agent analyzes retrieved content to identify and determine appropriate structural representations for the information.

**Implementation.** This agent uses the `data_structure_selector` function to decide the most suitable data structure for representing information relevant to the current step (e.g., text, trees, tables, or graphs).

#### Structurer Agent Prompt

This is a data structure selection task. Based on the given 'question', choose the most suitable data structure to answer the question. You can choose from the following options:

- Text Chunk
- Tree
- Table
- Graph

Your answer should be concise and to the point. Return your answer in the following format directly: {answer: data structure}.  
The question is: {question}

## B.5 Verifier

The Verifier agent verifies the extracted data.

**Implementation.** The agent uses the `information_verifier` function to verify the information, including scoring and refinement suggestions.

#### Verify Scoring Prompt

Verify the following extracted information against the given criteria:

TASK: {step\_info['description']}

VERIFICATION CRITERIA: {criteria}

EXTRACTED INFORMATION: {extracted\_info}

RETRIEVED CONTEXT: {state.retrieved\_context.get(step\_num, "No context available")}

Evaluate the information based on these dimensions: - Completeness

(1-5): Does it provide all the necessary information for the task?

- Relevance (1-5): How relevant is the information to the specific task?

- Accuracy (1-5): Based on internal consistency, does the information seem accurate?

- Does the information need refinement? (Yes/No)

- Is the issue with the extracted information or with the retrieved context?

- What specific improvements are needed?

Respond in JSON format:

```
{  "verification_passed":
  true/false,
  "completeness": 1-5,
  "relevance": 1-5,
  "accuracy": 1-5,
  "needs_refinement":
  true/false,
  "issue_source": "extraction"
  or "retrieval" or "none",
  "refinement_suggestions":
  "Specific suggestions for
  improvement"
  ...}
```

## B.6 Extractor Agent

The Extractor agent extracts fine-grained information from documents and transforms raw text into structured data according to the selected structure.

**Implementation.** The Extractor uses the `information_extractor` function to analyze document content and extract relevant information based on the current task requirements and chosen data structure.

#### Extractor Agent Prompt

Extract the most relevant information from these document information and transform into the following structure:

STRUCTURE: {structure}

TASK: {step\_info['description']}

DOCUMENT INFORMATION: {docs}

EXTRACTION GUIDELINES: 1. Focus specifically on information that directly addresses the task 2. Extract key facts, figures, quotes, and findings 3. Maintain accuracy - don't add information not present in the documents 4. Note any contradictions or uncertainties in the documents 5. Organize the extracted information logically 6. If "STRUCTURE" is a "Graph" or a "Tree," return a tuple of two or three elements. EXTRACTED INFORMATION:

## B.7 Reasoner Agent

The Reasoner agent performs reasoning over extracted and structured information to synthesize answers or analytical insights.

**Implementation.** The Reasoner uses various reasoning patterns to draw conclusions from the extracted information.

### Reasoner Agent Prompt

Reason about the following information to address this specific step:  
STEP DESCRIPTION: {step\_description}  
{extracted\_content}  
{dependencies\_text}  
Based on this information, provide a clear, well-reasoned answer that addresses the step description. Focus specifically on what this step is asking for, while incorporating any relevant information from dependent steps. Also, the answer should be concise and to the point.  
ANSWER:

## C Experimental Setup Details

### C.1 Model and Data Setup

The experiments were conducted on a device equipped with eight NVIDIA 4090 GPUs. All generation agents (Planner, Structurer, Extractor, Verifier, Refiner, and Reasoner) use Qwen2-72B-Instruct. Retrieval follows a two-stage procedure. We initialize each query with StructRAG retrieval outputs (Li et al., 2025b). When the verifier’s average of completeness and relevance falls below threshold  $t = 2$  (on the 1–5 scale), Qwen-Long searches the original Loong documents (Wang et al., 2024b); otherwise the StructRAG retrieval is retained.

### C.2 Evaluation metrics

#### LLM Score Generation Prompt

[Gold Answer]  
{std\_ans}  
[The Start of Assistant’s Predicted Answer]  
{ans}  
[The End of Assistant’s Predicted Answer]  
[System] We would like to request your feedback on the performance of the AI assistant in response to the user question displayed above according to the gold answer. Please use the following listed aspects and their descriptions as evaluation criteria: - Accuracy and Hallucinations: The assistant’s answer is semantically consistent with the gold answer; The numerical value and order need to be accurate, and there should be no hallucinations. - Completeness: Referring to the reference answers, the assistant’s answer should contain all the key points needed to answer the user’s question; further elaboration on these key points can be omitted. Please rate whether this answer is suitable for the question. Please note that the gold answer can be considered as a correct answer to the question. The assistant receives an overall score on a scale of 0 to 100, where a higher score indicates better overall performance. Please note that if the assistant’s answer and the gold answer fully meet the above criteria, its overall rating should be the full marks (100). Please first provide a comprehensive explanation of your evaluation, avoiding any potential bias. Then, output a line indicating the score of the Assistant. PLEASE OUTPUT WITH THE FOLLOWING FORMAT, WHERE THE SCORE IS A SCALE OF 0 TO 100 BY STRICTLY FOLLOWING THIS FORMAT: "[[score]]", FOR EXAMPLE "Rating: [[100]]":  
<Start Output>  
Evaluation evidence: your evaluation explanation here, no more than 100 words Rating: [[score]]  
<End Output>  
Now, start your evaluation:

Following the approach of Loong (Wang et al., 2024b), we use GPT-4 as a judge to evaluate the model’s output by comparing it against the golden answer and the question’s requirements. The evaluation considers three aspects: Accuracy, Hallucina-

Table 7: Number of Loong test instances  $N_t$  per length set and task (Wang et al., 2024b). Overall is the instance-weighted average using these counts as weights.

| Task       | Set1 | Set2 | Set3 | Set4 |
|------------|------|------|------|------|
| Spotlight  | 53   | 70   | 80   | 47   |
| Comparison | 60   | 105  | 95   | 40   |
| Clustering | 113  | 246  | 194  | 88   |
| Chain      | 97   | 143  | 112  | 57   |
| Total      | 323  | 564  | 481  | 232  |

tions, and Completeness, with scores ranging from 0 to 100. Two key indicators are used: (1) LLM Scores: the score given by GPT-4, and (2) EM: the percentage of cases that received a score of 100 out of the total number of cases.

**Overall aggregation.** The Overall scores in Tables 2 and 6 are instance-weighted averages rather than unweighted macro-averages across the four task categories. Specifically,

$$\text{Overall} = \frac{\sum_t N_t s_t}{\sum_t N_t}, \quad (7)$$

where  $s_t$  denotes the mean LLM or EM score for task  $t$ , and  $N_t$  denotes the corresponding number of test instances. The task-specific values of  $N_t$  are reported in Table 7.

### C.3 Parameter Settings

All controller parameters were fixed across all datasets without tuning. The reward design follows Eq. (3): each rollout starts from a base reward of  $R_{\text{base}} = 0.5$ . The verifier returns completeness  $c$ , relevance  $r$ , and accuracy  $a$ , each on a 1–5 scale, which we combine with equal weights after max-normalization:

$$R_{\text{verify}} = \frac{c + r + a}{15}.$$

The UCT exploration constant  $\alpha = \sqrt{2}$  is used only in the exploration term of Eq. (1) and does not weight the three verifier dimensions. Penalties for expensive actions (0.2 for `refine`, 0.3 for `replan`) and a cycle penalty of 0.1 per extra attempt (capped at 0.5) are all bounded, ensuring stable exploration. Because verifier feedback is observed at every iteration, the controller automatically retries or reroutes when a step underperforms, making these constants insensitive and not dataset-dependent.

Trajectory and loop constraints serve as simple stability guards rather than tuned hyperparameters.

We allow at most  $\mu = 6$  retrieval/verification/replan attempts on the same step and at most  $\eta = 2$  high-cost refinements (refine/replan) within a single loop, preventing long or oscillatory reasoning paths.

Verifier thresholds are also fixed across datasets. If the average of completeness and relevance falls below 2 (i.e.,  $< 40\%$  of the scale), the controller triggers additional retrieval; otherwise it proceeds with refinement and extraction. This effectively requires context quality of at least “2” before the loop settles and worked robustly across all benchmarks.

### D Detailed Outputs of Case Study

The detailed outputs of Section 4.5 are as follows. Please note that the original dataset and outputs contain Chinese for ease of reader understanding. For presentation purposes, we have translated them into English without any modifications or polishing.

#### 1) w/o extract (but with retriever)

Starting analysis for query: {docs}  
Please read the financial reports of the multiple companies above and answer the following questions based only on the content seen above. For parts of the questions that are not covered in the financial reports, you can ignore those parts and only answer the other parts.  
Query: What is Gree Electric's special reserve?  
Loading and processing documents...  
Step 1: Identify the 'special reserve' value for Gree Electric in its financial statements. This involves searching through the provided documents for Gree Electric's specific financial data.  
[TEMP] Retrieving documents related to the query...  
[RETRIEVE] Query used: "Identify the 'special reserve' value for Gree Electric in its financial statements. [SEP] Gree Electric special reserve"  
[RETRIEVE] Retrieved content:  
"2024 Q1 Report": The value of the special reserve is 26,924,166.37 yuan.  

|                 |                |                 |
|-----------------|----------------|-----------------|
| Item            | Ending Balance | Opening Balance |
| ---             | ---            | ---             |
| Special Reserve | 26,924,166.37  | 26,969,643.44   |

  
"2024 First Quarter Report": The value of the special reserve is 11,770,107.42 yuan.  

|                 |                |                   |
|-----------------|----------------|-------------------|
| Item            | March 31, 2024 | December 31, 2023 |
| ---             | ---            | ---               |
| Special Reserve | 11,770,107.42  | 2,020,435.65      |

  
"2024 Q1 Report": The value of the special reserve is 687,042,264.52 yuan.  

|                 |                              |                  |
|-----------------|------------------------------|------------------|
| Item            | End of this reporting period | End of last year |
| ---             | ---                          | ---              |
| Special Reserve |                              |                  |

| 687,042,264.52 | 557,482,954.85 |  
[TEMP] Making inferences...  
Reasoning Result (Step 1): The 'special reserve' value for Gree Electric as of the end of the first quarter of 2024 is **687,042,264.52 yuan**, according to the "2024 Q1 Report".  
Step 2: Synthesize result from Step 1.  
[TEMP] Making inferences...  
Final Answer: The special reserve for Gree Electric in the 2024 Q1 Report is **687,042,264.52 yuan**.

#### 2) w/o verify

Starting analysis for query: {docs}  
Please read the financial reports and answer based only on the content seen above.  
Query: What is Gree Electric's special reserve?  
Loading and processing documents...  
Step 1: Locate 'special reserve' for Gree Electric from financial reports.  
[TEMP] Retrieving documents...  
[RETRIEVE] Query used: "Gree Electric Special Reserve 2024 Q1 Report"  
[RETRIEVE]  
"2024 Q1 Report": Value is 26,924,166.37 yuan. "2024 Q1 Report": Not mentioned for Gree Electric; Huamei Holdings has no record.  
[TEMP] Structuring content...  
Structure: Table  
[EXTRACT]  

|                 |                               |
|-----------------|-------------------------------|
| Company Name    | Special Reserve               |
| -----           | -----                         |
| Gree Electric   | Not mentioned                 |
| Huamei Holdings | No record of special reserves |

  
[TEMP] Making inferences...  
Reasoning Result (Step 1): 'special reserve' for Gree Electric was not mentioned in the 2024 Q1 report.  
Step 2: Synthesize the result.  
[TEMP] Making inferences...  
Final Answer: The special reserve value for Gree Electric is **unavailable** as it was not mentioned in the provided report.

#### 3) CycleIE

Starting analysis for query: {docs}  
Query: What is Gree Electric's special reserve?  
Step 1: Identify the special reserve amount from the 2024 Q1 report.  
[TEMP] Retrieving documents...  
[RETRIEVE] "2024 Q1 Report": Special reserve = 26,924,166.37 yuan.  
"2024 Q1 Report": Gree Electric's special reserve amount is 26,924,166.37 yuan.  
[TEMP] Structuring... Table selected.  
[EXTRACT]  

|               |                        |               |
|---------------|------------------------|---------------|
| Company       | Special Reserve Amount | Report Period |
| -----         | -----                  | -----         |
| Gree Electric | 26,924,166.37 yuan     | 2024 Q1       |

  
[VERIFY] Verification Passed: Completeness 5/5,

Table 8: Main Backend Components

| Component          | Role / Usage                                 |
|--------------------|----------------------------------------------|
| Flask              | Backend server and application interface     |
| FAISS              | Dense vector similarity search               |
| RoBERTa Embeddings | Generation of dense text representations     |
| Qwen2-72B-Instruct | Large language model for response generation |

Table 9: Vector Database Architecture

| Module          | Functionality                                              |
|-----------------|------------------------------------------------------------|
| FAISS           | Core engine for dense vector similarity search             |
| DocumentManager | Manages index creation, deletion, loading, and persistence |
| Document Index  | Independent vector index for each uploaded document        |
| Global Index    | Aggregated vector index across all documents               |

|                                                                                                                                                                                                                                                                                                                                          |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Relevance 5/5, Accuracy 5/5<br>[TEMP] Making inferences...<br><b>Reasoning Result (Step 1):</b> The special reserve is <b>26,924,166.37 yuan</b> .<br><b>Step 2:</b> Synthesize the final answer.<br>[TEMP] Making inferences...<br><b>Final Answer:</b> The special reserve for Gree Electric in 2024 Q1 is <b>26,924,166.37 yuan</b> . |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

E Technical Implementation of the Application

The application consists of a React-based frontend and a Flask-based backend. The backend integrates FAISS for dense vector retrieval, RoBERTa-based embeddings for text representation, and Qwen2-72B-Instruct for response generation. The main backend components are summarized in Table 8.

E.1 Vector Database Design

The retrieval component uses FAISS to maintain both document-level and global vector indices. Each uploaded document is associated with an independent index, allowing retrieval to be restricted to selected documents. A global index is also maintained to support retrieval across the complete document collection.

A DocumentManager component manages index creation, deletion, loading, and persistence. The vector database organization is summarized in Table 9.