SAGE: From Direct Answering to Evidence-Grounded Inference for Chinese Ancient Document Understanding

Anonymous ACL submission

Abstract

Chinese ancient document understanding demands complex visual, linguistic, and historical reasoning. Current Large Vision-Language Models (LVLMs) typically rely on an opaque, single-pass generation paradigm, often producing overconfident and weakly grounded responses. To address this, we propose **SAGE**, an evidence-grounded multi-agent framework that reformulates Chinese ancient document understanding as evidence-grounded inference rather than direct answer generation. SAGE coordinates specialized agents for task-aware planning, tool-mediated evidence acquisition, claim-level verification, and bounded replanning under a constrained shared-state runtime. This design supports bounded evidence seeking, answer revision, and abstention when grounding is insufficient. Experiments on the AncientDoc benchmark show that SAGE consistently outperforms matched direct-answering baselines across three LVLM backbones. Remarkably, SAGE with Qwen3.5-9B surpasses much larger monolithic LVLMs on most evaluated metrics, highlighting the importance of structured, evidence-grounded inference beyond model scaling.

1 Introduction

Chinese ancient documents preserve millennia of historical, literary, philosophical, and cultural knowledge (Wei et al., 2024; Zhao et al., 2025; Cao et al., 2024a). While extracting information from these page images is crucial for the digital humanities and cultural heritage preservation, the task introduces complexities far beyond modern document understanding (Mathew et al., 2020; Ma et al., 2024). These texts feature vertical layouts, traditional and variant characters, missing punctuation, dense annotations, and highly implicit historical-cultural semantics. As highlighted by recent benchmarks like AncientDoc (Yu et al., 2025), mastering this domain is inherently a multi-stage endeavor that requires models to seamlessly integrate visual reading, classical Chinese interpretation, and knowledge-grounded reasoning.

These compounding challenges severely restrict the ability of large vision-language models (LVLMs) to move beyond basic visual recognition toward faithful interpretation. Despite this, most existing LVLM-based systems still rely on a direct-answering paradigm (Fu et al., 2024; Ma et al., 2024): given an image and a query, the model generates a response in a single forward pass. This approach collapses reading, evidence acquisition, reasoning, and verification into one opaque generation step. Consequently, the resulting answers are frequently overconfident, weakly grounded, and notoriously difficult to diagnose when errors occur.

In authentic scholarly practice, however, even human experts are not omniscient. Historians answer complex questions by systematically reading the text, identifying visual clues, consulting external references, and verifying claims—crucially expressing uncertainty when evidence is lacking. This reality motivates a necessary paradigm shift in ancient document understanding: moving away from monolithic, direct LVLM generation toward an evidence-grounded, verification-aware, and abstention-capable inference process.

To this end, we propose **SAGE**, an evidence-grounded multi-agent framework for Chinese ancient document understanding. Moving away from the conventional single-step generation paradigm, **SAGE** decomposes evidence-grounded inference into three role-specialized agents: (1) a *Scholarly Planning Agent* that analyzes the query and formulates a task- and budget-aware evidence-seeking strategy; (2) a *Tool-Augmented Execution Agent* that invokes constrained tools—such as page reading, text normalization, term extraction, and local evidence retrieval—to gather evidence and synthesize a grounded response; and (3) a *Scholar Verifier Agent* that evaluates whether the gener-

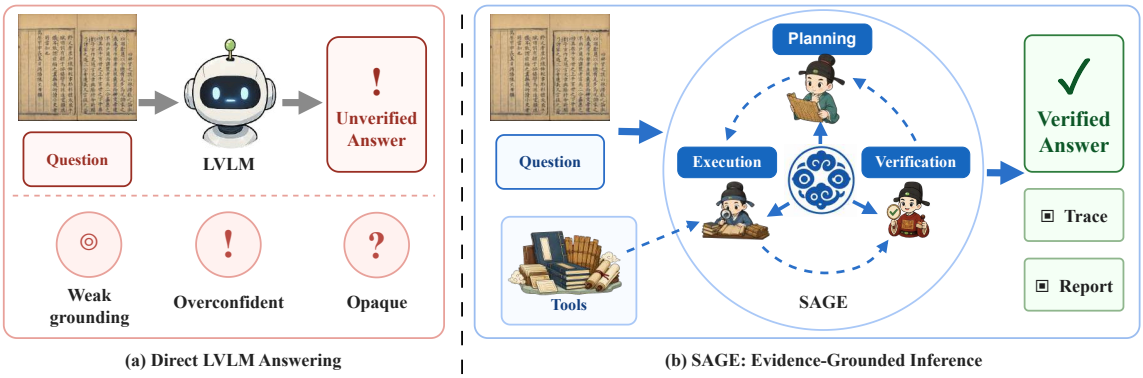

Figure 1: Direct LVLM answering vs. SAGE: Evidence-Grounded Inference.

ated claims are sufficiently supported by the acquired evidence. These agents operate under a constrained evidence-grounded runtime, which maintains a shared evidence state, regulates tool access and budget, records execution traces, and applies predefined control rules to stop, replan for additional evidence, or abstain when grounding is insufficient. Figure 1 illustrates the conceptual contrast between this evidence-grounded inference process and direct LVLM answering.

Extensive experiments on the AncientDoc benchmark across three LVLM backbones demonstrate that SAGE consistently surpasses matched direct-answering baselines on all evaluated metrics, yielding the most substantial gains on knowledge-intensive and linguistic tasks. Remarkably, SAGE equipped with the Qwen3.5-9B model achieves state-of-the-art performance on seven of the eight metrics, outperforming monolithic LVLMs that are orders of magnitude larger. This underscores a critical insight: evidence-grounded inference and rigorous verification can elevate ancient document understanding far more effectively than merely scaling base model parameters. Furthermore, trace-based analyses reveal that our framework adapts its tool usage to specific task demands, and that verifier-guided replanning measurably improves both objective evidence support and verifier confidence. These findings advocate for a paradigm shift: mastering ancient documents is not solely an LVLM scaling problem, but fundamentally an evidence-grounding and reliability challenge.

Our contributions are summarized as follows:

- We reformulate Chinese ancient document understanding as an evidence-grounded inference problem beyond direct LVLM answering, where systems expose supporting evidence, verification reports, abstention decisions, and execution traces in addition to final answers.
- We introduce SAGE, an evidence-grounded multi-agent framework that coordinates task-aware planning, tool-mediated evidence acquisition, claim-level verification, bounded replanning, and traceable diagnosis through role-specialized agents and a constrained shared-state runtime.
- We comprehensively evaluate SAGE on the AncientDoc benchmark across three backbone models. The results demonstrate consistent improvements over matched direct-answering baselines, while ablation and trace analyses validate the roles of planning, tool-mediated execution, and scholarly verification.

2 Related Work

2.1 Document and Ancient-Document Understanding Benchmarks

Document understanding has been extensively studied through document visual question answering, OCR-centric evaluation, and long-context multimodal benchmarks. Representative benchmarks such as DocVQA, TextVQA, OCRBench, OCRBench v2, and MMLongBench-Doc evaluate models' abilities to read, localize, and reason over textual or visual elements in document images and long documents (Mathew et al., 2020; Singh et al., 2019; Liu et al., 2024; Fu et al., 2024; Ma et al., 2024). These benchmarks have substantially advanced modern document understanding, but they mainly focus on contemporary documents, scene text, or general multimodal inputs.

Chinese ancient documents introduce a more specialized setting. They involve vertical layouts, traditional and variant characters, dense annotations,

missing punctuation, classical Chinese expressions, implicit semantics, and historical-cultural references. Recent benchmarks have begun to evaluate models on Classical Chinese and ancient-document tasks. Text-based benchmarks such as AC-EVAL, C^{3}Bench, Fùxì, and WenyanBENCH focus on ancient Chinese language understanding, translation, knowledge, and generation (Wei et al., 2024; Cao et al., 2024b; Zhao et al., 2025; Yao et al., 2025). Historical document resources and multimodal benchmarks such as HisDoc1B, BABM-LLM, and AncientDoc further extend evaluation to ancient document images and ancient books (Shi et al., 2025; Yu et al., 2025). Most directly related to our work, AncientDoc evaluates LVLMs on Chinese ancient document understanding across page-level OCR, vernacular translation, reasoning-based QA, knowledge-based QA, and linguistic variant QA.

These benchmarks demonstrate that ancient Chinese and ancient-document understanding require specialized evaluation beyond modern document tasks. However, they primarily evaluate final model outputs under a direct-answering paradigm. In contrast, SAGE focuses on the inference process itself: how an ancient-document system can acquire evidence, generate tentative answers, verify claims, abstain under insufficient evidence, and provide trace-based diagnostics during inference.

2.2 Ancient Chinese Models and Historical Document Processing

Another line of work develops domain-specific models and methods for ancient Chinese processing. Early pretrained models such as AnchiBERT adapt representation learning to ancient Chinese corpora, while recent LLM-based systems such as WenyanGPT and TongGu further explore continued pretraining, instruction tuning, retrieval augmentation, and knowledge-grounded Classical Chinese understanding (Tian et al., 2021; Yao et al., 2025; Cao et al., 2024a).

SAGE is complementary to these model-centric and translation-centric approaches. Rather than training a new ancient Chinese model, SAGE is designed as a model-agnostic inference-time framework for system-level evidence grounding. Rather than targeting text-only Classical Chinese understanding or translation alone, it addresses multi-task Chinese ancient document understanding from page images, where visual reading, text normalization, tool-mediated evidence acquisition, answer generation, and verification must be coordinated. Its core contribution is to operationalize ancient-document understanding as an evidence-grounded inference process, with explicit support for claim verification, bounded replanning, abstention, and trace-based component analysis.

3 SAGE Framework

3.1 Overview

As shown in Figure 2, SAGE is an evidence-grounded multi-agent framework that organizes Chinese ancient document understanding as a stateful evidence-seeking process. Given an ancient document image I and a natural-language question Q, the system maintains a shared evidence-grounded state and coordinates three role-specialized agent modules: a *Scholarly Planning Agent*, a *Tool-Augmented Execution Agent*, and a *Scholar Verifier Agent*. These modules are not executed as an unconstrained conversation or a fixed cascade. Instead, they operate under a constrained shared-state runtime, which regulates the available actions, tracks evidence and budget, records execution traces, and applies predefined control rules over the shared state to stop, replan, or abstain when necessary.

For each input, **SAGE** outputs

$$O = (A, E, R, z, \tau), \tag{1}$$

where A is the final answer, E is the collected evidence, R is an evidence report, z is a binary abstention indicator, and τ is the execution trace. These auxiliary outputs are not used to replace final-answer evaluation, but to make the answer-generation process more grounded, controllable, and diagnosable.

3.2 Scholarly Planning Agent

The Scholarly Planning Agent maps the input question to an executable evidence-seeking plan. Given the question Q and an initial budget b_0, the planner predicts the task type, identifies the required grounding sources, and selects a sequence of actions from the runtime-defined action space. Formally, the initial plan is:

$$P_0 = \mathrm{Plan}(Q, b_0) = (y, \mathbf{a}_0, \rho_0, b_0), \tag{2}$$

where y is the predicted task type, $\mathbf{a}_0 = (a_1, \ldots, a_m)$ is the planned action sequence, ρ_0 specifies the evidence requirement and verification criterion, and b_0 denotes the initial budget, such as

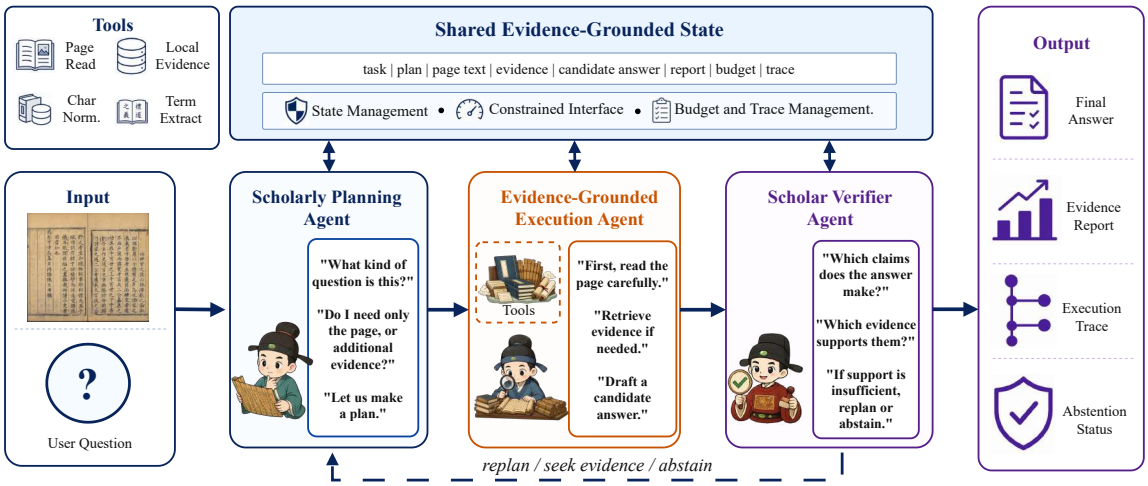

Figure 2: Overview of **SAGE**. SAGE reformulates Chinese ancient document understanding from direct answering into evidence-grounded inference by coordinating planning, tool-augmented execution, and verification under a constrained shared-state runtime. The runtime maintains an evidence-grounded state, exposes a constrained action interface, tracks budget, and records execution traces.

the maximum number of tool calls or replanning rounds.

The plan determines what evidence should be acquired before answer synthesis. For example, translation-oriented questions mainly require page reading and text normalization, knowledge-intensive questions require concept extraction and source-constrained retrieval. The planning agent therefore produces a strategy rather than a final answer, separating task decomposition from evidence acquisition and answer generation. Typical task-specific planning strategies are summarized in Table 1.

When the runtime later triggers a bounded replanning round, the same planner updates the action plan based on the current evidence state and remaining budget, allowing the system to adapt its evidence-seeking behavior rather than following a fixed pipeline.

Task	Evidence Need	Planned Strategy
OCR	Visual page evidence	Read page text and check recognition validity.
Translation	Page / normalized text	Read and normalize text, then generate a page-grounded vernacular translation.
Knowledge QA	Concept / historical evidence	Extract key concepts, retrieve source-constrained evidence, and verify factual claims.
Reasoning QA	Page-grounded evidence	Use recognized page text and question constraints to perform document-grounded reasoning.
Linguistic QA	Text + style evidence	Use page text with variant, rhetoric, style, or genre evidence for linguistic analysis.

Table 1: Task-aware planning strategies in **SAGE**. The planner selects evidence requirements and action strategies before answer synthesis.

3.3 Evidence-Grounded Execution Agent

The Evidence-Grounded Execution Agent instantiates the current plan into concrete tool-mediated actions. Given the document image I, question Q, current plan P_k, and shared state s_k, the agent executes only actions allowed by the constrained runtime, updates the evidence state, and produces a candidate answer for later verification.

The execution agent operates over a constrained action space: *read*, *normalize*, *extract*, *retrieve*, and *synthesize*. The *read* action applies OCR or page-reading tools to obtain textual evidence from the document image. The *normalize* action produces analyzable normalized forms while preserving the original recognized text, including traditional-to-simplified conversion and variant-character normalization when applicable. The *extract* action identifies candidate concepts, entities, book titles, names, places, or classical expressions from the question and page text. The *retrieve* action is invoked primarily for knowledge-intensive questions and uses the page context to search a frozen local knowledge evidence pool for relevant concept-level or historical-cultural evidence. The *synthesize* action generates a candidate answer conditioned on the current page text and evidence state. More details about the tool implementations and invocation protocols are provided in Appendix C.

Formally, the execution agent updates the page

text and evidence state through:

$$(X_k, E_k) = \mathrm{Exec}(I, Q, P_k, s_k), \tag{3}$$

where X_k denotes the recognized or normalized page text and E_k denotes the accumulated evidence state. The evidence state includes page-derived evidence for all tasks and may additionally include retrieved evidence for knowledge-intensive questions. Retrieved evidence is not treated as an answer source by itself; instead, it provides contextual support for answer synthesis and later verification.

Given the page text and accumulated evidence, the execution agent generates a candidate answer:

$$\hat{A}_k = \mathrm{Solve}(Q, X_k, E_k, P_k), \tag{4}$$

where $\hat{A}_k$ is treated as a hypothesis to be verified rather than the final output.

3.4 Scholar Verifier Agent

The Scholar Verifier Agent evaluates whether a candidate answer is sufficiently supported by the available page content and evidence state. Given the question Q, page text X_k, collected evidence E_k, and candidate answer $\hat{A}_k$, the verifier produces an evidence report R_k, which is stored in the shared state for runtime-level control.

For QA-oriented tasks, the verifier first decomposes the candidate answer into atomic claims:

$$C_k = \{c_1, c_2, \ldots, c_n\}. \tag{5}$$

It then estimates the support relation between each claim and the available grounding context:

$$v_i = \mathrm{Verify}(c_i, Q, X_k, E_k), \quad v_i \in \mathcal{Y}, \tag{6}$$

where $\mathcal{Y} = \{\textsc{Sup}, \textsc{Con}, \textsc{Ins}, \mathrm{N/A}\}$ denotes supported, contradicted, insufficient, and not applicable, respectively. The claim-level verification results are denoted as:

$$V_k = \{v_1, v_2, \ldots, v_n\}. \tag{7}$$

The verifier aggregates these judgments into an evidence report:

$$R_k = \mathrm{Aggregate}(C_k, V_k, X_k, E_k), \tag{8}$$

which summarizes evidence coverage, unsupported and contradicted claim rates, support confidence, and an abstention recommendation. If the main claims are sufficiently supported, the candidate answer becomes eligible for finalization by the runtime. Otherwise, the verifier records whether the current answer lacks support, misses necessary evidence, or should be rejected. The constrained shared-state runtime stores this feedback and may trigger replanning or abstention under the remaining budget.

The verifier applies task-sensitive criteria. Knowledge-intensive questions require stricter claim-evidence grounding and may trigger additional source-constrained retrieval or abstention. For translation, reasoning, and linguistic analysis, where the page text is often the primary grounding source, verification focuses on output validity, consistency with the document content, adequacy, and unsupported claims rather than exhaustive external support.

3.5 Constrained Shared-State Runtime

The constrained shared-state runtime provides the system-level infrastructure that organizes the three agent modules into a coherent evidence-grounded inference process. It is not an additional autonomous agent; rather, it defines the shared evidence state, constrained tool interface, budget limits, and trace schema used throughout inference. At iteration k, the runtime maintains:

$$s_k = (y, P_k, X_k, E_k, \hat{A}_k, R_k, b_k, \tau_k), \tag{9}$$

where y is the predicted task type, P_k is the current plan, X_k is the recognized or normalized page text, E_k is the accumulated evidence state, $\hat{A}_k$ is the current candidate answer, R_k is the evidence report, b_k is the remaining budget, and τ_k is the execution trace.

Unlike a fixed pipeline, the runtime does not require every input to follow the same action sequence. It executes the current plan under the constrained tool interface, updates the shared state after each tool call and verification step, and applies predefined control rules over the updated state:

$$d_k = \mathrm{Rule}(s_k), \quad d_k \in \mathcal{D}, \tag{10}$$

where

$$\mathcal{D} = \{\textsc{Stop}, \textsc{Replan}, \textsc{Abstain}\}. \tag{11}$$

In our implementation, STOP is selected when the evidence report indicates sufficient support or the task-specific adequacy check passes. REPLAN is selected when the current state suggests missing evidence and the remaining budget allows another evidence-seeking round. ABSTAIN is selected

Model / Backbone	Paradigm	Translation		Reasoning QA		Knowledge QA		Linguistic QA	
		CHRF++	BS-F1	CHRF++	BS-F1	CHRF++	BS-F1	CHRF++	BS-F1
DeepSeek-VL2 (Wu et al., 2024)	Direct	0.49	50.27	3.42	59.09	3.77	59.83	3.70	60.05
LLaVA-OneVision-72B (Li et al., 2024)	Direct	0.44	48.83	5.96	67.20	5.56	66.31	2.51	57.57
InternVL3-78B (Zhu et al., 2025)	Direct	4.45	62.40	4.95	65.99	5.25	65.79	2.38	57.78
Qwen2.5-VL-72B (Team, 2025b)	Direct	9.77	69.87	9.04	**71.40**	7.82	69.15	3.65	59.34
GPT-4o (Hurst et al., 2024)	Direct	3.02	58.86	7.99	70.52	8.02	70.01	4.16	64.58
Gemini2.5-Pro (Comanici et al., 2025)	Direct	11.41	72.50	8.68	69.33	8.88	68.94	5.22	62.06
Doubao-V2 (Team, 2025a)	Direct	0.56	52.39	7.15	68.78	8.75	69.15	3.32	57.70
Qwen-VL-Max (Bai et al., 2023)	Direct	12.30	71.03	8.60	71.30	7.58	68.67	3.31	58.77
InternVL3-8B (Zhu et al., 2025)	Direct	0.85	53.16	5.21	65.62	4.75	63.60	1.93	55.96
	SAGE	4.91	61.89	8.98	69.00	10.84	70.40	6.02	64.51
Qwen2.5-VL-7B (Team, 2025b)	Direct	7.04	65.59	7.34	69.96	5.87	66.75	2.65	57.48
	SAGE	10.16	67.70	8.04	70.99	8.45	69.92	4.05	62.18
Qwen3.5-9B (Qwen Team, 2026)	Direct	7.71	68.03	6.88	68.56	6.80	66.83	1.83	55.63
	SAGE	**16.01**	**75.29**	**9.35**	69.10	**11.81**	**71.12**	**7.72**	**65.35**

Table 2: Main results on AncientDoc. OCR is excluded because it primarily depends on external page-reading tools. We report CHRF++ and BERTScore F1 for four understanding tasks. Direct baselines use task-specific prompts with known task types, whereas gray rows denote SAGE runs without task-type labels. Best and second-best results are in bold and underline.

when evidence remains insufficient, the output fails verification, or the budget is exhausted.

The runtime follows three design principles.

Shared state management. The runtime maintains a unified state across planning, execution, and verification, including the predicted task type, current plan, page text, evidence, candidate answer, evidence report, remaining budget, and execution trace. This allows modules to reuse accumulated evidence and intermediate results instead of restarting from scratch.

Constrained action interface. The runtime exposes a fixed set of actions, including page reading, text normalization, concept extraction, local evidence retrieval, answer synthesis, and verification. Agent modules can only interact with the system through this interface, which prevents hallucinated tool use and keeps inference reproducible.

Budget and trace management. The runtime enforces limits on tool calls and replanning rounds, and records the execution trajectory, including planned actions, tool calls, retrieved evidence, candidate answers, evidence reports, replanning decisions, abstention decisions, and cost information when available. These traces support component-level diagnosis beyond final-answer accuracy.

In summary, the runtime turns planning, execution, and verification into a stateful and traceable evidence-grounded process. It provides shared memory, constrained tool access, budget control, and diagnostic logging, while keeping inference more adaptive than a fixed one-pass pipeline.

4 Experiments

We evaluate whether SAGE improves Chinese ancient document understanding beyond direct LVLM answering, and whether its evidence-grounded runtime produces more grounded and diagnosable outputs. Specifically, we study three questions: **(1)** Does SAGE improve final-answer quality over matched direct-answering baselines? **(2)** Are the gains consistent across different backbone models? **(3)** How do planning, tool-mediated evidence acquisition, scholarly verification, and bounded evidence seeking contribute to performance and grounding?

4.1 Experimental Setup

Benchmark and Metrics. We evaluate SAGE on AncientDoc (Yu et al., 2025), which contains five task types: OCR, translation, reasoning-based QA, knowledge-based QA, and linguistic QA. Our main evaluation focuses on the four understanding-oriented tasks: translation, reasoning-based QA, knowledge-based QA, and linguistic QA. We exclude OCR from the main comparison because OCR primarily measures page-reading or OCR tool quality, whereas our focus is on evidence acquisition, verification, and runtime control beyond raw text recognition. Following the AncientDoc evaluation protocol, we report CHRF++ and BERTScore

F1 (BS-F1).

Models and Settings. We compare SAGE with direct-answering LVLM baselines. For matched comparisons, we instantiate SAGE with InternVL3-8B (Zhu et al., 2025), Qwen2.5-VL-7B (Team, 2025b), and Qwen3.5-9B (Qwen Team, 2026), and compare each against its direct-answering counterpart. Direct baselines generate answers from the image-question input in a single pass using task-specific prompts, while SAGE receives the image-question input and performs task prediction, evidence acquisition, verification, and bounded replanning through the constrained shared-state runtime. Unless otherwise specified, all SAGE variants use the same constrained tool interface, frozen local evidence pool, verification protocol, and budget constraints. More details about baseline prompts, tool implementations, retrieval settings, and budgets are provided in Appendix A.

4.2 Main Results

Table 2 reports the main results on the four understanding-oriented AncientDoc tasks. The upper blocks list representative open-source and closed-source LVLMs under the direct-answering paradigm, while the lower block compares direct answering and SAGE under matched backbones: InternVL3-8B, Qwen2.5-VL-7B, and Qwen3.5-9B. For all tasks, we report CHRF++ and BERTScore F1 (BS-F1).

SAGE consistently improves over matched direct-answering baselines. Across three backbones and eight task metrics, SAGE improves every corresponding direct baseline, showing that the gains are not tied to a single model family. The improvements are especially large on Knowledge QA and Linguistic QA, where explicit evidence organization and verification are most beneficial. For example, with InternVL3-8B, SAGE improves Knowledge QA by +6.09 CHRF++ and +6.80 BS-F1, and Linguistic QA by +4.09 CHRF++ and +8.55 BS-F1.

Compared with much larger direct-answering LVLMs, SAGE also remains competitive. Qwen3.5-9B with SAGE achieves the best score on seven out of eight metrics and surpasses all direct LVLM baselines on Translation, Knowledge QA, and Linguistic QA. The only exception is Reasoning QA BS-F1, where Qwen2.5-VL-72B Direct is slightly higher. These results suggest that evidence-grounded inference and verification can improve ancient-document understanding beyond relying only on larger base LVLMs.

Variant	Reasoning QA		Knowledge QA	
	CHRF++	BS-F1	CHRF++	BS-F1
Direct LVLM	6.88	68.56	6.8	66.83
w/o Planner	9.27	67.06	9.34	67.8
w/o Page Reader	9.15	68.65	11.58	70.79
w/o Retrieval	-	-	11.78	70.89
w/o Verifier	9.13	68.98	11.23	70.08
SAGE	9.35	69.10	11.81	71.12

Table 3: Ablation study on reasoning-based QA and knowledge-based QA. We report CHRF++ and BERTScore F1 (BS-F1). Direct LVLM denotes single-pass answering with the same backbone. Other variants remove one replaceable component from SAGE while keeping the backbone, constrained runtime, execution agent, and budget settings fixed. Retrieval is ablated only on Knowledge QA, where the local evidence pool is used.

4.3 Ablation Study

We conduct ablation studies on reasoning-based QA and knowledge-based QA, which test page-grounded reasoning and knowledge-intensive evidence use. Since the constrained shared-state runtime and the Evidence-Grounded Execution Agent define the shared state and tool-mediated execution process, we keep them fixed and ablate replaceable components: planning, page reading, local retrieval, and verification. All ablations use Qwen3.5-9B as the backbone unless otherwise specified.

For *w/o Planner*, we replace the Scholarly Planning Agent with a fixed task-agnostic action sequence. For *w/o Page Reader*, we disable the explicit page-reading tool and rely on the backbone LVLM to read the document image directly. For *w/o Retrieval*, we disable retrieval from the frozen local evidence pool, which is only reported for Knowledge QA. For *w/o Verifier*, the candidate answer is returned without an evidence report or claim-level support judgments.

As shown in Table 3, the full SAGE achieves the best results on both tasks. All SAGE variants outperform the Direct LVLM baseline, indicating that the shared evidence-grounded runtime already provides substantial gains over single-pass answering. Removing the planner or verifier leads to consistent drops, especially in BS-F1, showing the importance of task-aware planning and scholarly verification. The effect of local retrieval is positive but modest on Knowledge QA, suggesting that SAGE's main

(a) Task-Adaptive Runtime Actions **(b) Verification-Guided Replanning**

Figure 3: Trace-based behavior analysis of SAGE. (a) *Task-Adaptive Runtime Actions*: action patterns across task types. Each cell reports the percentage of examples invoking the corresponding action, while the last column reports average final verifier confidence. (b) *Verification-Guided Replanning*: analysis on Knowledge QA examples where replanning is triggered. After replanning, both trace-derived support rate and verifier confidence increase.

gains come from page-grounded evidence organization, tool-mediated execution, and verification rather than from external knowledge alone.

4.4 Trace-based Behavior Analysis

Beyond final answer scores, SAGE records execution traces for each example, including planned actions, invoked tools, retrieved evidence, verification reports, and replanning decisions. We use these traces to examine whether SAGE produces task-adaptive execution patterns and whether verifier feedback leads to measurable changes in the inference process.

Figure 3(a) shows the action patterns across the four understanding-oriented tasks. Page reading, normalization, and verification are invoked for almost all examples, forming the common evidence-grounded runtime. In contrast, extraction and retrieval exhibit clear task specificity. Knowledge QA invokes extraction and local evidence retrieval in 90.39% of the examples, while reasoning QA invokes them in only 9.42% of the examples. Translation and linguistic QA rarely trigger retrieval, indicating that they mainly rely on page-level evidence rather than external knowledge. This supports that SAGE does not simply execute a fixed pipeline, but selects different evidence paths according to task semantics. The final confidence column further shows that the verifier maintains comparable confidence across tasks, suggesting that the adaptive use of evidence does not come at the cost of unstable verification behavior.

Figure 3(b) further analyzes the Knowledge QA examples where the verifier triggers replanning. After replanning, the trace-derived support rate increases from 69.23% to 74.45%, and verifier confidence increases from 67.99% to 71.59%. These results suggest that verifier feedback helps SAGE produce more strongly supported answers in cases initially judged as insufficiently grounded.

It should be noted that these statistics are derived from the system's own verification reports rather than human factuality annotations. Therefore, they are used as behavioral diagnostics of the runtime: the traces show when different tools are invoked, when replanning is triggered, and how the verifier's evidence-support assessment changes after an additional inference round.

5 Conclusion

We presented **SAGE**, an evidence-grounded multi-agent framework that moves Chinese ancient document understanding from direct LVLM answering toward evidence-grounded inference. SAGE coordinates role-specialized scholarly agents under a constrained shared-state runtime, supporting task-aware planning, tool-mediated evidence acquisition, scholarly verification, and bounded evidence seeking. By making evidence acquisition and verification explicit, SAGE provides a traceable basis for diagnosing unsupported or insufficiently grounded answers. Experiments show that SAGE improves over matched direct-answering baselines across multiple backbones, especially on knowledge-intensive and linguistic tasks. Ablation and trace-based analyses indicate that these gains arise from structured page-grounded evidence use, tool-mediated execution, and verification rather than larger backbones or external knowledge alone. Overall, ancient document understanding should be treated not only as a model-scaling problem, but also as an evidence-grounding and reliability challenge.

Limitations

SAGE is an initial exploration of evidence-grounded Chinese ancient document understanding. Although it improves over matched direct-answering baselines on AncientDoc, the current evaluation is limited to the tasks and data distribution covered by this benchmark, and its applicability to broader historical collections, different document layouts, and other ancient languages remains to be validated. In addition, the verifier and trace-based support statistics are system-generated diagnostics rather than human factuality annotations, so they should be interpreted as indicators of runtime behavior rather than definitive proof of answer correctness. Finally, SAGE introduces additional inference cost through planning, tool use, verification, and replanning. Future work may explore stronger evidence sources, human-grounded verification, and more efficient execution.

References

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*.

Jiahuan Cao, Dezhi Peng, Peirong Zhang, Yongxin Shi, Yang Liu, Kai Ding, and Lianwen Jin. 2024a. TongGu: Mastering classical Chinese understanding with knowledge-grounded large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4196–4210, Miami, Florida, USA. Association for Computational Linguistics.

Jiahuan Cao, Yongxin Shi, Dezhi Peng, Yang Liu, and Lianwen Jin. 2024b. C^3bench: A comprehensive classical chinese understanding benchmark for large language models. *Preprint*, arXiv:2405.17732.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Ling Fu, Zhebin Kuang, Jiajun Song, Mingxin Huang, Biao Yang, Yuzhe Li, Linghao Zhu, Qidi Luo, Xinyu Wang, Hao Lu, and 1 others. 2024. Ocrbench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning. *arXiv preprint arXiv:2501.00321*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and 1 others. 2024. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, Pan Zhang, Liangming Pan, Yu-Gang Jiang, Jiaqi Wang, Yixin Cao, and Aixin Sun. 2024. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations. In *Advances in Neural Information Processing Systems*, volume 37, pages 95963–96010. Curran Associates, Inc.

Minesh Mathew, Dimosthenis Karatzas, R Manmatha, and CV Jawahar. 2020. Docvqa: A dataset for vqa on document images. corr abs/2007.00398 (2020). *arXiv preprint arXiv:2007.00398*.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Yongxin Shi, Dezhi Peng, Yuyi Zhang, Jiahuan Cao, and Lianwen Jin. 2025. A large-scale dataset for chinese historical document recognition and analysis. *Scientific Data*, 12(1):169.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. *Preprint*, arXiv:1904.08920.

ByteDance Seed Team. 2025a. Seed1.5-vl technical report. *arXiv preprint arXiv:2505.07062*.

Qwen Team. 2025b. Qwen2.5-vl.

Huishuang Tian, Kexin Yang, Dayiheng Liu, and Jiancheng Lv. 2021. Anchibert: A pre-trained model for ancient chineselanguage understanding and generation. *Preprint*, arXiv:2009.11473.

Yuting Wei, Yuanxing Xu, Xinru Wei, Simin Yang, Yangfu Zhu, Yuqing Li, Di Liu, and Bin Wu. 2024. AC-EVAL: Evaluating Ancient Chinese language understanding in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1600–1617, Miami, Florida, USA. Association for Computational Linguistics.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, and 1 others. 2024. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*.

Xinyu Yao, Mengdi Wang, Bo Chen, and Xiaobing Zhao. 2025. Wenyangpt: A large language model for classical chinese tasks. *Preprint*, arXiv:2504.20609.

Haiyang Yu, Yuchuan Wu, Fan Shi, Lei Liao, Jinghui Lu, Xiaodong Ge, Han Wang, Minghan Zhuo, Xuecheng Wu, Xiang Fei, Hao Feng, Guozhi Tang, An-Lan Wang, Hanshen Zhu, Yangfan He, Quanhuan Liang, Liyuan Meng, Chao Feng, Can Huang, and 2 others. 2025. Benchmarking vision-language models on chinese ancient documents: From ocr to knowledge reasoning. *Preprint*, arXiv:2509.09731.

Shangqing Zhao, Yuhao Zhou, Yupei Ren, Zhe Chen, Chenghao Jia, Fang Zhe, Zhaogaung Long, Shu Liu, and Man Lan. 2025. Fùxì: A benchmark for evaluating language models on ancient chinese text understanding and generation. *Preprint*, arXiv:2503.15837.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, and 1 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*.

A Implementation Details

This section provides implementation details for the experimental setup in the main paper, including model backbones, direct-answering baselines, SAGE runtime configuration, decoding settings, and execution budgets.

A.1 Backbone Models

We instantiate SAGE with three LVLM backbones: InternVL3-8B, Qwen2.5-VL-7B, and Qwen3.5-9B. For each backbone, we compare SAGE with a matched direct-answering baseline using the same underlying model. This matched comparison isolates the effect of the evidence-grounded runtime from the effect of backbone choice. Unless otherwise specified, all backbone parameters are kept fixed during evaluation; SAGE changes only the inference procedure by adding planning, tool-mediated evidence acquisition, verification, and bounded replanning.

A.2 Direct-Answering Baselines

The direct-answering baseline receives the document image and the user question as input and generates the final answer in a single pass. We use task-specific prompts for the four understanding-oriented tasks evaluated in the main paper: translation, reasoning-based QA, knowledge-based QA, and linguistic QA. The baseline does not receive retrieved evidence, intermediate verification reports, or replanning feedback. This setting represents the standard LVLM usage pattern where the model is expected to read, reason, and answer directly from the image-question input.

A.3 SAGE Runtime Configuration

SAGE uses the same input as the direct-answering baseline but executes an evidence-grounded runtime before producing the final answer. The runtime consists of four main steps: (i) task prediction, which identifies the type of ancient-document understanding task; (ii) evidence acquisition, which invokes tools such as page reading, normalization, extraction, and local evidence retrieval when applicable; (iii) answer generation, which produces a candidate answer grounded in the collected evidence; and (iv) scholarly verification, which checks whether the answer is supported by the available evidence and may trigger bounded replanning.

All SAGE variants use the same constrained tool interface, frozen local evidence pool, verification protocol, and budget constraints. Ablation variants remove or replace a specific component while keeping the remaining runtime configuration fixed.

A.4 Decoding Settings

For fair comparison, we use the same decoding configuration for direct answering and SAGE answer generation under the same backbone. We use deterministic or low-temperature decoding to reduce variance in final answers. The maximum generation length is set to cover the longest expected outputs in translation and knowledge-intensive QA tasks. The same decoding settings are also used for planning and verification unless otherwise specified.

A.5 Execution Budgets

SAGE operates under bounded inference budgets to prevent unbounded tool use or indefinite replanning. We limit the maximum number of tool calls, retrieved evidence snippets, verification passes, and replanning rounds for each example. The same budgets are used across all backbone models and tasks. Replanning is triggered only when the verifier judges that the current answer has insufficient

evidence support or low confidence. If the budget is exhausted, the system returns the best supported answer produced so far.

A.6 Reproducibility Notes

All experiments are conducted under the official AncientDoc evaluation protocol. We report CHRF++ and BERTScore F1 for the four understanding-oriented tasks: translation, reasoning-based QA, knowledge-based QA, and linguistic QA. OCR is excluded from the main comparison because the focus of SAGE is evidence acquisition, verification, and runtime control beyond raw page reading. We keep the evidence pool frozen during evaluation and apply the same retrieval and verification settings to all matched SAGE variants.

B Prompts

This appendix reports the core prompt templates used by SAGE. The implementation prompts are written in Chinese because the target documents and tasks are Chinese ancient-document understanding tasks. For readability, we present lightly translated templates while keeping the task labels, tool names, and runtime placeholders unchanged.

B.1 Prompt Organization

SAGE uses a small family of prompts rather than a single monolithic instruction. The task classifier first maps the user request to one of five task types. The scholarly planning prompt then produces a structured tool plan under the current budget and evidence state. Task-specific answer prompts synthesize candidate outputs from page text and, when needed, local evidence. Finally, the verifier prompt checks whether the candidate answer is supported by the available evidence and returns a structured report used by the runtime to stop, replan, or abstain.

B.2 Task Classification Prompt

Purpose. The classifier normalizes heterogeneous user questions into the fixed task ontology used by the runtime.

> You are the task classifier for SAGE. Given the user question, decide which of the following five task types it belongs to: `ocr`, `translation`, `reasoning_qa`, `knowledge_qa`, or `linguistic_qa`.
>
> `ocr`: the user asks to extract, recognize, transcribe, or copy text from an image.
>
> `translation`: the user asks to translate ancient text into modern Chinese or vernacular Chinese.
>
> `reasoning_qa`: the user asks a question requiring causal, semantic, ideological, or implicit inference based on page content.
>
> `knowledge_qa`: the user asks about people, places, allusions, terms, historical facts, or cultural knowledge.
>
> `linguistic_qa`: the user asks about rhetoric, genre, language style, period style, or literary features.
>
> Output exactly one label and do not explain.
>
> User question: `{question}`

B.3 Scholarly Planning Prompt

Purpose. The planner does not answer the question directly. It produces a JSON strategy that can only call tools exposed by the constrained runtime.

> You are the Scholarly Planning Agent of SAGE. You do not directly answer the user question. Your only responsibility is to generate a structured JSON strategy plan, which will be executed by the constrained SAGE runtime.
>
> Inputs: planning mode `{planning_mode}`, round index `{round_index}`, task type `{task}`, user question `{question}`, tool registry `{tool_registry}`, task budgets `{budgets}`, current state summary `{current_state_summary}`, previous strategy `{previous_strategy}`, verification feedback `{verification_feedback}`, successful strategy memory `{strategy_memory}`, and failure memory `{failure_memory}`.
>
> Planning principles: Only use tools listed in the tool registry. Do not exceed the current task budget. Do not generate or execute Python code. At the initial planning stage. `translation`, `reasoning_qa`, `knowledge_qa`, and `linguistic_qa` usually require `normalize_text`. `knowledge_qa` can use `retrieve_evidence` over the local knowledge pool. After `solve_task`, always call `verify_answer`. During replanning, usually do not repeat OCR; for `knowledge_qa`, retrieve additional evidence if the verifier requests it. If evidence remains insufficient, plan a conservative verification-ending path that allows the verifier to trigger abstention.
>
> Recommended initial workflows: `ocr`: `ocr_image` → `verify_answer`. `translation`: `ocr_image` → `normalize_text` → `solve_task` → `verify_answer`. `reasoning_qa`: `ocr_image` → `normalize_text` → `solve_task` → `verify_answer`. `knowledge_qa`: `ocr_image` → `normalize_text` → `extract_terms` → `retrieve_evidence` → `solve_task` → `verify_answer`. `linguistic_qa`: `ocr_image` → `normalize_text` → `solve_task` → `verify_answer`.
>
> Return JSON only. Do not explain and do not use Markdown. The JSON must contain the task label, maximum tool calls, an ordered list of steps, a fallback list, and a task-specific output rule.

B.4 Page Reading Prompt

Purpose. The page-reading prompt enforces document-layout assumptions for Chinese ancient books and prevents explanatory text from entering the OCR state.

> You are an OCR model specialized in Chinese ancient-book vertical pages. Strictly recognize the text in the image. Read columns from right to left, and read each column from top to bottom. Output only the OCR result; do not output explanations, prefixes, suffixes, Markdown, numbering, or prompting text. Preserve original characters and original wording. Do not convert between simplified and traditional Chinese and do not modernize the text. Do not guess missing characters from meaning. Main text, interlinear notes, and small annotations should be output according to their reading position if they belong to the textual content. Ignore column lines, borders, stains, decorative patterns, seals, and non-textual page numbers. The final output must contain only recognized text.

B.5 Task-Specific Answer Prompts

Purpose. The answer prompts share a common style constraint: answer directly, avoid chain-of-thought or evidence narration, preserve key names and terms, and abstain only when the available information is genuinely insufficient.

Translation.

> Translate the ancient OCR text into modern vernacular Chinese. Inputs are `{ocr_text}` and normalized reference text `{normalized_text}`. Output only the translation. Do not explain the translation process and do not start with phrases such as "this passage means". Preserve the main information in the source. If the OCR contains missing-character marks, handle them cautiously without excessive inference. Perform semantic translation rather than character conversion; rewrite classical syntax into natural modern Chinese, adding omitted subjects, objects, or logical relations when necessary. Do not fabricate information absent from the OCR text.

Reasoning QA.

> Answer reasoning questions from the page text. Inputs are the question `{question}`, ancient OCR or normalized text `{text}`, and available evidence `{evidence}`. Answer the question directly without saying "according to the page". For causes, motives, effects, or functions, use natural formulations such as "because ..." or "the reason is ...". For ideas, cultural meaning, or institutional features, summarize with expressions such as "this reflects ...". Preserve key terms, names, book titles, institutions, numbers, and proper expressions from the question and page. Necessary semantic inference is allowed, but the inference must be integrated into the answer rather than shown as reasoning steps. Usually answer in one to three sentences.

Knowledge QA.

> Answer knowledge questions about people, places, allusions, terms, historical facts, or cultural knowledge using page text `{text}` and local knowledge-base evidence `{evidence}`. Give a concise answer similar to a dictionary-style explanation. Do not separate page information from knowledge-base evidence and do not cite the retrieval process. For terms, allusions, people, places, books, or institutions, prefer forms such as "X refers to ...", "X is ...", or "namely ...". Include only the definition, source, function, influence, or cultural meaning most relevant to the question. If evidence is limited but the question concerns common knowledge, provide a cautious brief answer; if it is impossible to judge, answer "cannot be determined".

Linguistic QA.

> Answer questions about language style, rhetoric, genre, or literary features using page text `{text}` and, when necessary, model knowledge. Do not use web search. Provide a clear judgment. If asked about rhetoric, first state the rhetorical category; if asked about genre, first state the genre; if asked about language style, first summarize the style. Point to concrete textual evidence such as words, sentence patterns, parallel structures, or imagery. Avoid generic literary comments detached from the page text. If the page text is insufficient, say that the textual basis is insufficient.

B.6 Verifier Prompt

Purpose. The verifier checks candidate answers without rewriting the task from scratch. It returns a structured report used by the runtime for stopping, replanning, or abstention.

> You are the Scholar Verifier Agent of SAGE. Your task is not to answer the question again, but to perform a lightweight quality check on the candidate answer. Adjust strictness by task type: knowledge QA and reasoning QA require evidence consistency; translation and OCR only require checks for format, completeness, and obvious errors.
>
> Inputs: task type `{task}`, user question `{question}`, candidate answer `{answer}`, rule-split claims `{rule_claims}`, and available evidence `{evidence}`.
>
> Use the labels `supported`, `contradicted`, `insufficient`, and `not_applicable`. For `knowledge_qa`, prioritize retrieved evidence and do not support claims without evidence. For `reasoning_qa`, allow cautious page-grounded inference. For `linguistic_qa`, require concrete textual or stylistic evidence. For `translation`, do not demand external evidence; check whether the output is modern Chinese, preserves the main information, and avoids irrelevant explanations or disclaimers. For OCR, check whether usable OCR output exists.
>
> Return JSON only. The JSON contains a cleaned final answer, claim list, claim-level verifications, an evidence report, an abstention flag, a confidence score, a `needs_more_evidence` flag, and an issue list.

Tool	Input	Output
Page Reader	Document image	Page-level OCR text and source metadata
Char Normalizer	Raw OCR text	Normalized text and character-level changes
Term Extractor	Question and page text	Candidate terms and retrieval queries
Local Evidence	Question, terms, and page context	Page evidence and local KB snippets

Table 4: Evidence-acquisition tool interface used by SAGE. Answer synthesis and verification are handled by the agent modules rather than listed as tools in this table.

C Tool Interface, Evidence Pool, and Retrieval

This section describes the constrained evidence-acquisition interface used by **SAGE**. We focus on four tools: page reading, character normalization, term extraction, and local evidence retrieval. Answer synthesis is performed by the Evidence-Grounded Execution Agent, and verification is handled by the Scholar Verifier Agent; they are not counted as evidence-acquisition tools in this section.

C.1 Tool Registry

SAGE exposes a small fixed registry of evidence-acquisition tools. Each tool has a predefined input-output schema, and the runtime validates the planned action sequence before execution. This prevents unconstrained tool use and makes the evidence path traceable.

C.2 Page Reader

The Page Reader obtains page-grounded textual evidence from the document image. In our experiments, it is implemented with Qwen-VL-OCR following the page-reading prompt in Appendix B.4. Given a document image, the tool returns page-level OCR text together with basic source metadata. The recognized text is used as the primary page-level evidence for all task types and is passed to downstream normalization, extraction, answer synthesis, and verification modules.

C.3 Character Normalization

The Char Normalizer converts raw OCR text into a more consistent form for downstream reasoning while preserving the original OCR output. In our implementation, this tool is based on the OpenCC Python library for traditional-to-simplified Chinese conversion. The normalized text is used by downstream extraction, retrieval, answer synthesis, and verification modules, while the original OCR text remains available as page-level evidence.

C.4 Term Extraction

The Term Extractor identifies candidate terms from the question and page text. It is implemented as a lightweight rule-based component rather than a separate neural entity recognizer. The extractor collects quoted expressions, term-like spans, and Chinese phrases from the question and page text, then deduplicates them to form a candidate term list. These terms are used to build retrieval queries for knowledge-intensive questions, especially Knowledge QA.

C.5 Local Evidence Retriever

The Local Evidence Retriever searches a frozen offline evidence pool, constructed before evaluation for ancient-document QA. To build this pool, we first collect candidate evidence through web search using terms derived from the AncientDoc knowledge-QA setting, and then filter, clean, and consolidate the returned content into short evidence snippets. Each entry is stored as a key-value record, where the key corresponds to a term, entity, concept, book title, or historical-cultural expression, and the value contains a concise explanatory snippet with source metadata when available.

During inference, the evidence pool is fixed and never updated with test examples or model outputs. This offline design serves two purposes. First, it prevents uncontrolled external access during evaluation, making the evidence path reproducible and inspectable. Second, it ensures that the main experiments and ablation studies use the same external knowledge source, so performance differences reflect the effect of the SAGE runtime rather than changes in available evidence.

The retriever receives the question, extracted terms, and page context. It always preserves the current page text as page-level evidence. For Knowledge QA, it additionally retrieves up to eight local evidence snippets from the offline pool. Retrieval uses transparent matching rules, including exact key matching, substring matching, year-equivalent matching, key occurrence in the question, and Chinese character overlap. Returned snip-

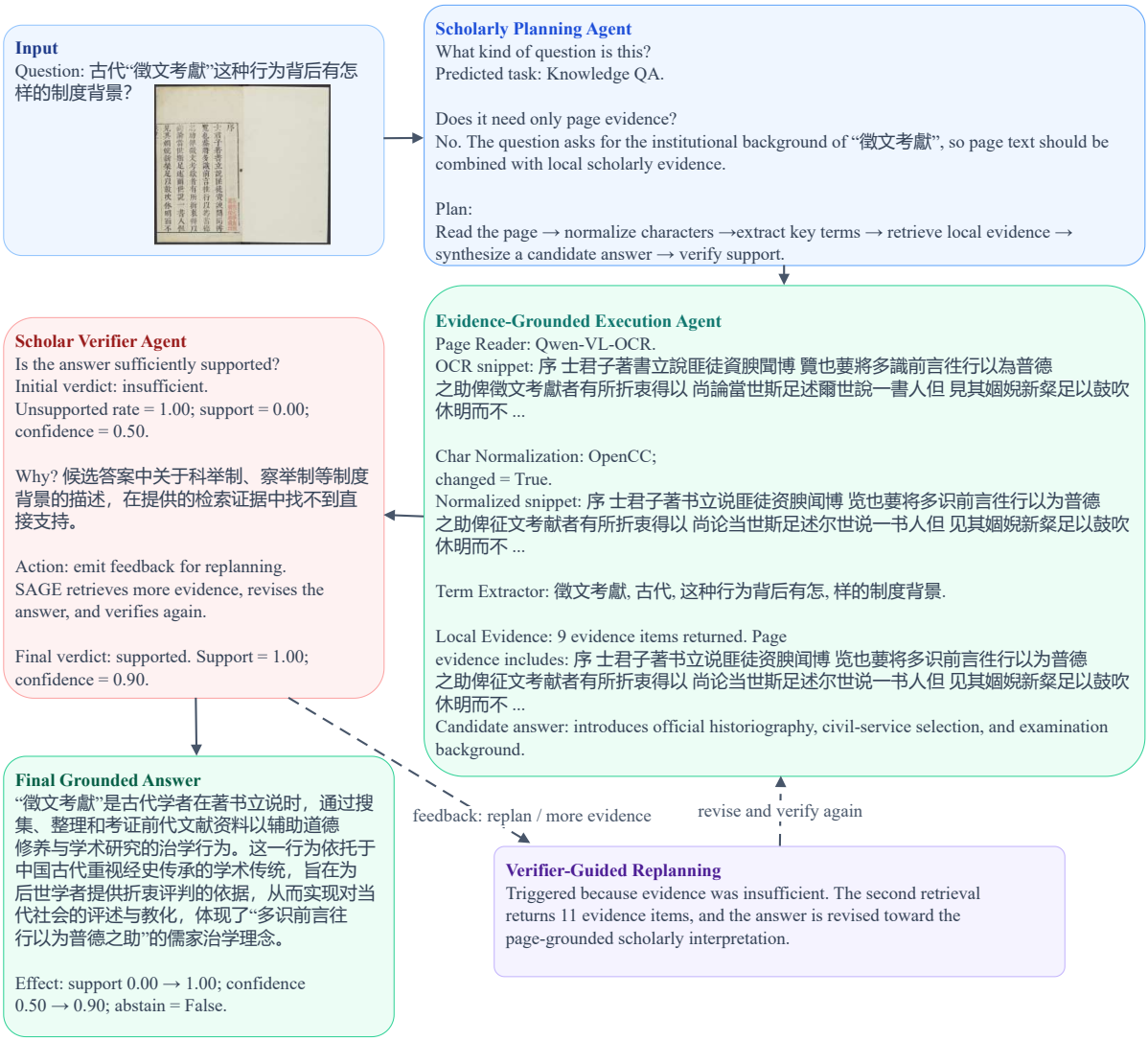

Figure 4: Qualitative trace of a Knowledge QA example. The verifier identifies unsupported institutional claims in the initial candidate answer, triggers replanning, and leads to a revised answer that is better grounded in the page evidence and local retrieval results.

pets are ranked, deduplicated, and recorded with their evidence ids, matched queries, match basis, scores, and source metadata.

C.6 Budgets and Traceability

The runtime constrains tool use with task-specific budgets. The default maximum number of tool calls is 3 for OCR, 5 for translation, 5 for reasoning QA, 7 for Knowledge QA, and 6 for linguistic QA. The runtime rejects planned strategies that exceed the budget or contain unregistered tools. Retrieval is also bounded: each local evidence call returns at most eight KB snippets in addition to the page context.

All tool invocations are recorded in the execution trace, including tool names, inputs implied by the shared state, outputs, retrieved evidence, retrieval statistics, and errors if any. These traces support the behavior analysis in the main paper and allow us to diagnose how SAGE acquires evidence before answer synthesis and verification.

D Qualitative Case Study

Figure 4 presents a representative Knowledge QA example from the execution traces. The input question asks about the institutional background behind the practice of “*zhengwen kaoxian*” (*collecting texts and examining documents*). The Scholarly Planning Agent first classifies the instance as Knowledge QA and plans an evidence path that combines page reading, character normalization, term extraction, local evidence retrieval, answer synthesis, and verification.

The Evidence-Grounded Execution Agent then reads the page using Qwen-VL-OCR, normalizes the recognized text, extracts key terms, and retrieves local evidence together with the page context. The initial candidate answer explains the practice by referring to official historiography, civil-service selection, and examination systems. However, the Scholar Verifier Agent finds that these institutional claims are not directly supported by the retrieved evidence, yielding an insufficient verdict with support rate 0.00 and confidence 0.50.

Based on this feedback, SAGE triggers a replanning round. The revised answer removes the unsupported institutional expansion and instead grounds the explanation in the page evidence: the practice is framed as a scholarly activity of collecting, organizing, and examining earlier documents to support moral cultivation and textual judgment. After replanning, the verifier marks the answer as supported, with support rate increasing from 0.00 to 1.00 and confidence increasing from 0.50 to 0.90. This case illustrates how **SAGE** uses verifier feedback to move from an over-extended candidate answer toward a more evidence-grounded final response.