DeALOG: Decentralized Multi-Agents Log-Mediated Reasoning Framework

Abhijit Chakraborty[*1] **Ashish Raj Shekhar**[1†] **Shiven Agarwal**[1†] **Vivek Gupta**[*1]
[1]Arizona State University
{achakr40,vgupt140}@asu.edu

Abstract

Complex multi-hop question answering over text, tables, and images requires integrating heterogeneous evidence with inspectable reasoning. Single-LLM chain-of-thought is opaque, while planner–executor pipelines concentrate faults in a central plan, propagating early errors. We introduce DeALOG, a decentralized, central-planner-free multi-agent framework whose specialized agents, i.e., Table, Context, Visual, Summarizing, and Verification, coordinate only by reading and appending to a shared, typed natural-language log as persistent memory, with a lightweight scheduler rather than a planner. Shared memory itself is not new: blackboard architectures and recent LLM shared-memory systems all use one, but each retains a *semantic* control unit (a manager, judge, or LLM agent selector). Our contribution is the control design and the finding it isolates: DeALOG's scheduler is *non-semantic*, and under matched agents, retriever, prompts, and corruption seeds, semantic central control is the robustness bottleneck. Rather than competing on raw accuracy, DeALOG targets what a central planner forfeits: robustness, since no single plan concentrates errors and evidence stays globally visible, and transparency, since the append-only log yields a provenance-anchored, human-readable reasoning trace. At 30% evidence corruption DeALOG loses 6.4% relative EM versus 18–31% for blackboard and planner systems under the same harness, and swapping in an LLM controller alone degrades 2.5× more.

1 Introduction

Multi-hop question answering (QA) over complex data, such as tables, textual passages, and images, requires compositional reasoning and the integration of heterogeneous evidence. For example (Fig. 1), answering *"What is the birth year of the oldest American author who is older than the author of Eat Pray Love?"* involves identifying entities from a table, extracting attributes from text, and reasoning jointly over nationality and birth years. To address such multi-hop reasoning, prior work has explored three main directions: (i) *Table QA* approaches based on semantic parsing and neural modeling (e.g., WIKITABLEQUESTIONS and successors such as TAPEX, TAPAS, TACUBE, OMNITAB, and LEVER) (Pasupat and Liang, 2015; Liu et al., 2021; Herzig et al., 2020; Zhou et al., 2022; Jiang et al., 2022; Ni et al., 2023); (ii) *reasoning-by-prompting* methods that rely on chain-of-thought or structured prompting (e.g., REACTABLE, Trees/-Graphs of Thought) (Wei et al., 2022; Zhang et al., 2024; Besta et al., 2023); and (iii) *multi-agent or tool-based pipelines*, typically organized as planner–executor architectures (e.g., HUGGINGGPT, BINDER, TABSQLIFY, REWOO, AUTOTQA, MACT) (Shen et al., 2023; Cheng et al., 2022; Nahid and Rafiei, 2024; Xu et al., 2023; Zhu et al., 2024b; Zhou et al., 2025). Despite their differences, these approaches largely occupy two ends of a design spectrum. **Single-LLM methods** based on chain-of-thought prompting (Wei et al., 2022) perform reasoning implicitly within a single context window; while effective for short or shallow queries, they struggle to maintain explicit intermediate state, verify partial results, and remain stable under long or branching reasoning chains. At the other extreme, **planner-based pipelines** externalize reasoning through explicit plans and tool invocations, enabling stepwise execution and re-planning based on intermediate outputs. However, this explicit structure introduces a complementary limitation: errors made early in the plan tend to propagate through downstream steps, concentrating

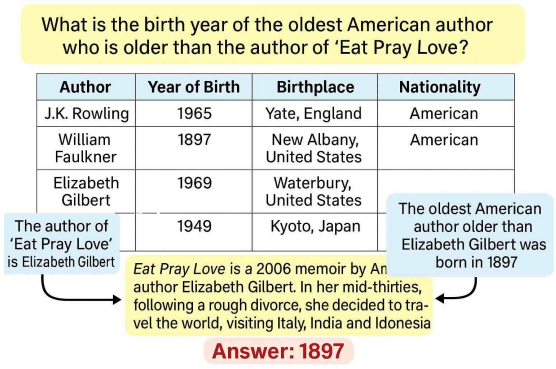


Figure 1: Example of multi-hop table question answering. The system identifies the author of *"Eat Pray Love"*, extracts metadata from a passage and a table, reasons over birth years and nationalities to find the correct answer.

[*]Corresponding author
[†]These authors contributed equally to this work.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 19470–19493
October 24-29, 2026 ©2026 Association for Computational Linguistics

uncertainty in the central planner and limiting robustness and transparency. *Together, these limitations call for a framework that exposes intermediate structure and evidence dependencies without committing to a single, fixed plan that can amplify early errors.* We introduce DEALOG, a decentralized, central-planner-free framework in which specialized agents like Table, Context, Visual, Summarizing, and Verification, never communicate directly, but read and append *typed* entries to a shared natural-language log that serves at once as persistent memory and a human-readable reasoning trace. A **SummarizingAgent** drafts an answer and a **VerificationAgent** cross-checks it against the log, while a lightweight scheduler manages turn-taking and termination but prescribes no plan.

What is new, and what is not. A shared memory among specialists is a classical idea: blackboard systems (Erman et al., 1980; Schwartz, 1995a) and their LLM-era reboots such as METAGPT (Hong et al., 2023), the LLM-blackboard of LBMAS (Han and Zhang, 2025), and BLACKBOARD-DS (Salemi et al., 2025) all coordinate through one. We do not claim the substrate. Every prior shared-memory system retains a *semantic* control unit (a manager, judge, or LLM that reads the question and selects the next agent); DEALOG's scheduler is *non-semantic* and never inspects question content, assigns subtasks, or re-plans (Table 1). The contribution is thus an isolated *finding*: with agents, retriever, prompts, and corruption seeds fixed and only the controller swapped, semantic central control is the robustness bottleneck. Blackboard systems under our harness degrade 2.8–4.8× more than DEALOG at 30% corruption, and an LLM controller on DEALOG's own agents degrades 2.5× more (§3.2). Because raw accuracy under a matched retrieval-and-extraction harness offers limited signal about the role of coordination, we center our claims on the two axes a central planner forfeits. *Robustness:* with no single plan to concentrate errors and all evidence globally visible, corrupted intermediate evidence is outweighed by independently grounded peer entries rather than propagated downstream, with the verifier contributing a safe fallback that prevents committing a faulty correction. *Transparency:* because every step is recorded as a typed, provenance-anchored entry, the reasoning trace is provenance-recorded by construction. We characterize both axes through controlled comparisons against a centralized-plan Planner variant and a Plan→Log hybrid under corrupted inputs and long-horizon reasoning, and against an external planner under corruption, motivating the removal of the central planner in favor of decentralized, log-mediated coordination.

Our framework offers the following key contributions:

1. **Non-semantic control as a design principle:** A decentralized design in which specialized agents communicate only through a typed natural-language log under a deterministic, question-agnostic scheduler with verifier-triggered re-engagement; we formalize the scheduler–planner boundary across nine dimensions (Table 1), report the scheduler's measured control-flow statistics (§3.2), and isolate the controller empirically (Table 6).
2. **Semantic central control is the robustness bottleneck:** A controlled robustness study comparing DEALOG against internal centralized-plan variants (Planner, Plan→Log, Planner++), external planners (AutoTQA, REWoO, HuggingGPT), and shared-memory blackboard systems (LBMAS, BLACKBOARD-DS, METAGPT) under synthetic, realistic, and *naturally occurring* evidence failures and long-horizon reasoning, with an attribution ablation locating the gain in global evidence visibility rather than the verifier.
3. **Transparency via a provenance-anchored log:** The typed, append-only log is a provenance-anchored reasoning trace in which each step is tied to its source; we introduce a log-groundedness metric and a counterfactual *necessity* test showing that cited entries are causally used (ablating one flips the answer 60–70% of the time). (We claim grounding and provenance only, not auditability or a human-usability result.)
4. **Matched-condition evaluation and ablations:** We evaluate DEALOG on six datasets, i.e., FinQA (Chen et al., 2021b), TAT-QA (Zhu et al., 2021), WikiTableQuestions (Pasupat and Liang, 2015), FeTaQA (Nan et al., 2021), CRT-QA (Zhang et al., 2023), and MultiModalQA (Talmor et al., 2021), against 11 baselines on three backbones under a shared retriever and extraction harness, with 95% bootstrap CIs, and ablate the shared log, agent specialization, verification, deduplication, and context-window sensitivity.

2 DEALOG Framework

Our approach draws from decentralized team architectures and *shared memory* among agents. Traditional *blackboard systems* enabled expert modules to solve problems via a common blackboard (Erman et al., 1980; Schwartz, 1995a,b,c), but always under a *control component* that inspected the blackboard's content to pick the next knowledge source; their LLM-era successors keep that role as an LLM controller (Han and Zhang, 2025; Salemi et al., 2025) or a role/SOP orchestrator (Hong et al., 2023). *Generative agents* have used shared memory for simulations (Park et al., 2023). While GRAPH-OF-THOUGHTS and AMAR explored multi-agent reasoning, they either integrate reasoning within one model or distribute tasks without communication (Besta et al., 2023; Sami et al., 2025). This work establishes a *log-mediated* multi-agent reasoning framework for factual QA with five LLM agents: *Table*, *Context*, *Visual*, *Summarizing*, and *Verification*, communicating through a shared log (Fig. 2), and removes the semantic control component entirely (App. F.21 tabulates the differences from prior shared-memory systems). Details are in App. A.

Design goals and assumptions. Our design empha-

Capability	DEALOG Sched.	Central Planner
Inspects question semantics	✗	✓
Generates task graphs	✗	✓
Assigns subtasks	✗	✓
Re-plans on failure	✗	✓
Controls reasoning strategy	✗	✓
Semantic awareness	None	Full
Single point of failure	✗	✓
Learned gating role	Stop/cont.	Absent
Analogy	OS sched.	Project mgr

Table 1: Scheduler vs. planner distinction. The scheduler manages *when* agents run, not *what* they do.

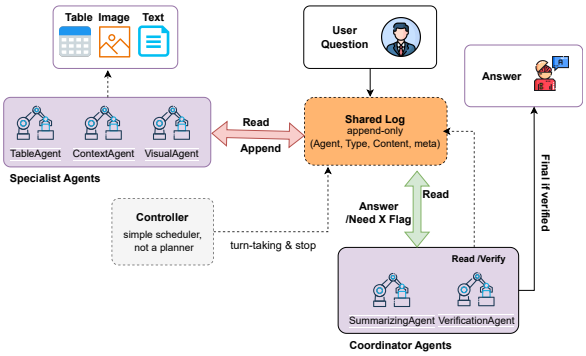


Figure 2: DEALOG: central-planner-free, log-mediated QA. Specialized agents read from and append typed entries to a shared append-only log; the Summarizer synthesizes a candidate answer and the Verifier cross-checks it. The scheduler is *non-semantic*: it cycles agents in fixed order (Table → Context → Visual → Summarizing → Verification), gating each turn by the agent's `should_act` precondition rather than by task content, and halts on a Verifier OK or at the round budget R. A Verifier FLAG triggers exactly one re-engagement round in which upstream agents re-read the log and may append corrections.

sizes (i) *modularity*, enabling plug-in specialists without retraining; (ii) *transparency*, via a provenance-aware shared log as persistent memory; and (iii) *robustness*, by keeping early errors from concentrating in a central plan and propagating—with the verifier as a safe fallback rather than an active repair mechanism. We assume access to tabular corpora, text, and images; agents operate zero-/few-shot. Instead of a central planner, a lightweight *scheduler* manages turn-taking, guardrails, and termination without prescribing plans. Its fixed agent order is question-agnostic (a structural prior, not a task plan), so DEALOG is *central-planner-free* rather than free of all ordering.

Scheduler vs. planner: a formal distinction. A key design choice is replacing a central planner with a lightweight scheduler. Table 1 formalizes this across nine dimensions. The DEALOG controller operates at the *resource-management* level with no semantic awareness, like an OS process scheduler, while central planners (Shen et al., 2023; Zhu et al., 2024b; Xu et al., 2023) operate at the *reasoning-control* level. Empirically, under 30% corruption the planner drops 24.7% EM and DEALOG 6.4% (Table 5). The learned gate is a binary continue/stop classifier, not a planning component.

Agent roles and I/O contracts. The **TableAgent** parses tables and posts cell-level facts. The **ContextAgent** retrieves passages. The **VisualAgent** converts charts to text. The **SummarizingAgent** composes answers or summaries. The **VerificationAgent** checks calculations and factual support. Each agent implements `should_act(log)` (trigger heuristic) and `act(log)` (LLM call for typed entry). The **VerificationAgent** checks consistency, recalculates, verifies units, and ensures answers are supported by log entries. On detecting inconsistencies, it re-engages. Unlike frameworks using triples or code, the verifier operates on unstructured logs without curated error taxonomies. DEALOG's faithfulness stems from multi-agent consensus grounded in the shared log.

Shared log and types. Each entry is a tuple (`Agent`, `Type`, `Content`, `meta`) where `meta` stores step index, time, and provenance. We use a fixed vocabulary of `Type`: LOOKUP, QUOTE, VISUAL, SUMMARY, ANSWER, and FLAG/OK. All agents have *global visibility*; ablations show restricting visibility markedly hurts performance. We deduplicate near-duplicate entries using ROUGE-L similarity (threshold > 0.85) combined with per-agent caps, saving 13.3% of tokens with negligible accuracy impact (Δ EM $= -0.008$; Table 16). Pseudocode for the controller loop is given in Algorithm 1 (App. F).

Controller loop. Each round, the scheduler invites the Table, Context, and Visual agents to contribute; when new evidence appears, the Summarizing Agent drafts an answer, which the Verification Agent checks before stopping. A FLAG triggers one re-engagement round to fix the issue. Most questions resolve in 3–4 agent calls; we cap rounds at $R = 6$ and apply per-agent and duplicate-entry limits as safety guardrails.

Gating. A small learned gate decides when to stop retrieving. It is trained on WikiTQ traces disjoint from the evaluation split, then applied unchanged to all six datasets while saving 9–18% of tokens on tabular benchmarks with negligible accuracy cost, and (correctly) never firing on CRT-QA, where longer reasoning matters (Table 14). Crucially, *headline accuracy in Tables 2 and 3 is reported without the gate*, so the accuracy claim cannot depend on the classifier's quality.

Memory. Older log entries are summarized with citations to fit the context window; doubling the window from 4k to 8k tokens yields only +0.013 EM on average (Table 15), confirming that reasoning difficulty, not context length, is the main bottleneck.

We provide full prompts, log types, regex patterns, no-progress criteria, and truncation rules in App. A and the public code release.[1] The repository provides the complete code, prompts, corruption generators, shared

[1] https://github.com/CoRAL-ASU/Agentic_ADAF.git

seeds, and per-example noise keys needed to reproduce Table 5, together with representative log traces for audit.

3 Experiments

Setup: We evaluate DeALOG and baselines on six multi-hop question answering datasets: FinQA, TAT-QA, WikiTableQuestions (WikiTQ), FeTaQA, CRT-QA, and MMQA. These datasets cover numerical reasoning, compositional queries, and open-domain retrieval. All datasets use matched backbone configurations and identical test sets, adhering to default splits and evaluation protocols. Dataset-specific statistics, such as the number of questions and modality composition, are in Table 32 (App. F.23).

Input Filtering and Retrieval: We use a consistent BM25+miniLM (Robertson and Zaragoza, 2009; Wang et al., 2020) retriever for input filtering and retrieval across all methods. This retriever extracts relevant passages, table segments, or image captions as input context. No task-specific filtering or preprocessing is applied to maintain comparability.

Metrics and Normalization. Table 31 (App. F.22) specifies the exact metric per dataset (EM, or the official scorer; TAT-QA reports $(\text{EM} + \text{F1})/2$; FeTaQA EM* applies a uniform `Answer:`-suffix extraction rule to every method). All normalization rules (numeric parsing, unit canonicalization, floating-point tolerance ± 0.01) are implemented in a shared function used identically across all methods.

Per-dataset evaluation protocol. Every dataset uses its official scorer on its default test split. The pipeline is closed-book i.e. no Wikipedia, web search, or knowledge graphs. Retrieval only filters the evidence shipped with each example, never an external corpus. Every method gets the same evidence pool and the same scorer (see Tables 2 and 3).

Evaluation Approach: Accuracy is the primary metric, computed either as exact match or via each dataset's official scoring rules (Table 31). We define a catastrophic error (Figure 5) as a prediction that is empty, off by more than 50% on a numeric question, or that mentions entities not present in the evidence, these are the failures we most want to avoid. An operator chain (Tables 5, 18) measures the minimum number of retrieval, arithmetic, comparison, or lookup steps needed to reach the correct answer; we read this from the released reasoning programs where available, and count distinct evidence hops otherwise. All results are averaged over five random trials, reported with 95% bootstrap confidence intervals (1,000 resamples) (Zrimšek and Štrumbelj, 2024) in Tables 2 and 3. Where we report a per-column best, we treat overlapping 95% CIs over the shared seeds as statistically indistinguishable rather than as a win. Efficiency is reported as LLM calls per query, total tokens used, and per-query latency (mean and P90). Corruption degradation is reported as a *relative* drop, $(\text{clean} - \text{corrupted})/\text{clean}$, except in the attribution ablation (Table 6a), which reports absolute EM points on the dev split.

LLM Models: We test three open backbones (LLaMA-3 8B, Mistral 7B, Qwen-3 8B) on identical footing. Summarizing and Verification agents run at temperature 0; Table and Context agents at 0.3. BLIP-2 (Li et al., 2023) and PaddleOCR (Cui et al., 2025) process images inline, so reported latencies (Tables 12, 13, 20) include this cost. Models run on A100s via vLLM.

Tool fairness. Tables 2 and 3 use a *tool-restricted* setting: every method receives identical multimodal tools (BLIP-2, PaddleOCR) but no Python interpreter or calculator, isolating reasoning quality from code-execution engineering. Natively code-executing methods (ReAcTable, Codex, Program-of-Thoughts) are additionally evaluated with their interpreter enabled in App. F.1; DeALOG runs without code execution in both settings.

Baseline Methods: We compare our proposed approach against a comprehensive set of baselines including multi-hop reasoning methods such as Chain-of-Thought prompting (Wei et al., 2022), REWoO (Xu et al., 2023), Chameleon (Lu et al., 2023), Fire-Act (Chen et al., 2023), Lumos (Yin et al., 2024), and HUSKY (Kim et al., 2024); table question answering models like Codex (Chen et al., 2021a) and TableCritic (Yu et al., 2025); and multi-agent systems including AutoTQA (Zhu et al., 2024b), ReAcTable (Zhang et al., 2024) and Dater (Ye et al., 2023). We additionally compare against Chain-of-Table (Wang et al., 2024b), Program-of-Thoughts (Chen et al., 2022), and TabSQLify (Nahid and Rafiei, 2024) on the datasets where their public implementations apply (discussed in §3). All share comparable LLM backbones and run zero-shot under the same harness.

3.1 Results and Analysis

Accuracy is equalized under matched conditions. Under a shared BM25+miniLM retriever, a uniform `Answer:`-suffix extraction rule, and no code execution, agent and prompting strategies cluster tightly: across Tables 2 and 3, most baselines fall within a few EM points of one another on each dataset, and DeALOG is competitive rather than categorically stronger. On FinQA DeALOG attains 76.5% (LLaMA-3 8B) and 76.4% (Qwen-3 8B), and on FeTaQA 75.8% and 76.1%, ahead of multi-agent baselines such as AutoTQA, ReAcTable, and TableCritic but by margins that frequently overlap at 95% CIs. The pattern is not uniform: TableCritic leads TAT-QA across all three backbones, and CoT spikes on Mistral 7B. We read this clustering as the central methodological point, i.e., *when retrieval and answer extraction are fixed and code execution is disallowed, raw accuracy reflects the shared harness more than the coordination strategy, and is a weak axis for comparing agent architectures*. We therefore treat accuracy as a competitiveness check and locate DeALOG's contribution in robustness and transparency, the two axes the harness does not equalize.

These comparisons confirm that, under matched conditions, no coordination strategy dominates on accuracy;

Method	FeTaQA	FinQA	TAT-QA
CoT	$54.9_{\pm1.9}$ / **79.7**$_{\pm1.3}$ / $54.6_{\pm1.9}$	$56.8_{\pm1.9}$ / **79.4**$_{\pm1.3}$ / $61.7_{\pm1.7}$	$54.5_{\pm1.9}$ / $61.2_{\pm1.7}$ / $55.6_{\pm1.9}$
REWoO	$68.5_{\pm1.5}$ / $70.4_{\pm1.5}$ / $69.7_{\pm1.5}$	$70.0_{\pm1.5}$ / $70.3_{\pm1.5}$ / $69.7_{\pm1.5}$	$68.5_{\pm1.5}$ / $69.7_{\pm1.5}$ / $70.0_{\pm1.5}$
Chameleon	$70.2_{\pm1.5}$ / $71.8_{\pm1.4}$ / $70.5_{\pm1.4}$	$70.7_{\pm1.5}$ / $71.9_{\pm1.4}$ / $70.1_{\pm1.4}$	$70.2_{\pm1.5}$ / $71.4_{\pm1.4}$ / $70.1_{\pm1.4}$
FireAct	$72.3_{\pm1.4}$ / $72.4_{\pm1.3}$ / $72.1_{\pm1.4}$	$72.7_{\pm1.4}$ / $72.8_{\pm1.3}$ / $72.4_{\pm1.4}$	$72.3_{\pm1.4}$ / $72.2_{\pm1.3}$ / $72.4_{\pm1.4}$
Lumos	$73.3_{\pm1.4}$ / $73.2_{\pm1.3}$ / $73.9_{\pm1.3}$	$73.2_{\pm1.4}$ / $73.0_{\pm1.3}$ / $73.2_{\pm1.3}$	$73.0_{\pm1.4}$ / $73.9_{\pm1.3}$ / $73.8_{\pm1.3}$
HUSKY	$73.3_{\pm1.3}$ / $74.0_{\pm1.3}$ / $73.8_{\pm1.3}$	$73.4_{\pm1.3}$ / $73.1_{\pm1.3}$ / $73.9_{\pm1.3}$	$73.6_{\pm1.3}$ / $73.2_{\pm1.3}$ / $74.0_{\pm1.3}$
AutoTQA	$69.4_{\pm1.5}$ / $70.2_{\pm1.5}$ / $69.5_{\pm1.5}$	$69.8_{\pm1.5}$ / $70.1_{\pm1.5}$ / $69.4_{\pm1.5}$	$69.3_{\pm1.5}$ / $69.9_{\pm1.5}$ / $70.0_{\pm1.5}$
Dater	$68.5_{\pm1.6}$ / $69.2_{\pm1.5}$ / $68.7_{\pm1.6}$	$68.9_{\pm1.6}$ / $69.3_{\pm1.5}$ / $68.2_{\pm1.6}$	$68.7_{\pm1.6}$ / $68.1_{\pm1.5}$ / $68.8_{\pm1.6}$
ReAcTable	$69.0_{\pm1.5}$ / $70.2_{\pm1.4}$ / $72.7_{\pm1.3}$	$69.4_{\pm1.5}$ / $70.1_{\pm1.4}$ / $72.8_{\pm1.3}$	$69.2_{\pm1.5}$ / $69.5_{\pm1.4}$ / $74.3_{\pm1.3}$
Codex	$68.5_{\pm1.6}$ / $69.4_{\pm1.5}$ / $74.0_{\pm1.3}$	$68.2_{\pm1.6}$ / $69.1_{\pm1.5}$ / $73.3_{\pm1.3}$	$68.1_{\pm1.6}$ / $68.7_{\pm1.5}$ / $73.3_{\pm1.3}$
TableCritic	$73.1_{\pm1.3}$ / $74.2_{\pm1.3}$ / $75.2_{\pm1.2}$	$73.4_{\pm1.3}$ / $74.5_{\pm1.3}$ / $75.2_{\pm1.2}$	**75.4**$_{\pm1.3}$ / **75.7**$_{\pm1.3}$ / **74.8**$_{\pm1.2}$
DeALOG	**75.8**$_{\pm1.1}$ / $76.9_{\pm1.1}$ / **76.1**$_{\pm1.0}$	**76.5**$_{\pm1.1}$ / $75.2_{\pm1.1}$ / **76.4**$_{\pm1.0}$	$72.4_{\pm1.2}$ / $73.1_{\pm1.2}$ / $71.2_{\pm1.2}$

Table 2: Results (%) on FeTaQA (EM*), FinQA (EM), TAT-QA (mean of the official EM and F1, $(\text{EM} + \text{F1})/2$; Table 31) with $\pm$95% bootstrap CIs (1,000 resamples, 5 seeds). Format: LLaMA-3 8B / Mistral 7B / Qwen-3 8B. Bold = small differences within overlapping CIs.

Method	MMQA (full)	MMQA (text/table)	WikiTQ
CoT	$56.4_{\pm1.9}$ / $58.2_{\pm1.3}$ / $68.9_{\pm1.6}$	$56.5_{\pm1.8}$ / $55.2_{\pm1.2}$ / $56.1_{\pm1.8}$	$53.1_{\pm1.9}$ / $58.7_{\pm1.3}$ / $64.4_{\pm1.6}$
REWoO	$70.0_{\pm1.5}$ / $67.8_{\pm1.5}$ / $69.5_{\pm1.5}$	$69.9_{\pm1.5}$ / $69.6_{\pm1.5}$ / $70.4_{\pm1.5}$	$68.6_{\pm1.5}$ / $70.5_{\pm1.5}$ / $71.0_{\pm1.5}$
Chameleon	$70.4_{\pm1.5}$ / $72.5_{\pm1.4}$ / $70.9_{\pm1.4}$	$71.4_{\pm1.4}$ / $72.7_{\pm1.4}$ / $70.3_{\pm1.4}$	$69.4_{\pm1.5}$ / $71.8_{\pm1.4}$ / $70.1_{\pm1.4}$
FireAct	$72.7_{\pm1.4}$ / $72.8_{\pm1.3}$ / $71.2_{\pm1.4}$	$72.5_{\pm1.4}$ / $73.5_{\pm1.3}$ / $71.8_{\pm1.4}$	$71.0_{\pm1.4}$ / $72.4_{\pm1.3}$ / $71.9_{\pm1.4}$
Lumos	$73.6_{\pm1.4}$ / $73.4_{\pm1.3}$ / $73.7_{\pm1.3}$	$72.5_{\pm1.3}$ / $73.1_{\pm1.3}$ / $72.5_{\pm1.3}$	$71.1_{\pm1.4}$ / $74.7_{\pm1.3}$ / $73.7_{\pm1.3}$
HUSKY	$73.6_{\pm1.3}$ / $74.9_{\pm1.3}$ / $73.6_{\pm1.3}$	$72.8_{\pm1.3}$ / $73.2_{\pm1.3}$ / $73.7_{\pm1.3}$	$72.8_{\pm1.3}$ / $73.5_{\pm1.3}$ / $74.0_{\pm1.3}$
AutoTQA	$69.7_{\pm1.5}$ / $70.4_{\pm1.5}$ / $69.0_{\pm1.5}$	$69.7_{\pm1.5}$ / $70.3_{\pm1.5}$ / $68.8_{\pm1.5}$	$68.0_{\pm1.5}$ / $71.9_{\pm1.4}$ / $70.0_{\pm1.5}$
Dater	$68.9_{\pm1.6}$ / $69.3_{\pm1.5}$ / $68.4_{\pm1.6}$	$68.7_{\pm1.6}$ / $69.7_{\pm1.5}$ / $67.4_{\pm1.6}$	$68.1_{\pm1.6}$ / $69.2_{\pm1.5}$ / $68.7_{\pm1.6}$
ReAcTable	$69.0_{\pm1.5}$ / $70.3_{\pm1.4}$ / $73.4_{\pm1.3}$	$68.7_{\pm1.5}$ / $70.2_{\pm1.4}$ / $73.6_{\pm1.3}$	$68.5_{\pm1.5}$ / $70.9_{\pm1.4}$ / $72.5_{\pm1.3}$
Codex	$69.0_{\pm1.6}$ / $69.9_{\pm1.5}$ / $74.9_{\pm1.3}$	$67.4_{\pm1.6}$ / $69.1_{\pm1.5}$ / $73.6_{\pm1.3}$	$66.4_{\pm1.6}$ / $70.3_{\pm1.5}$ / $73.5_{\pm1.3}$
TableCritic	$73.2_{\pm1.3}$ / $73.4_{\pm1.3}$ / $75.4_{\pm1.2}$	$73.8_{\pm1.3}$ / $74.0_{\pm1.3}$ / $75.4_{\pm1.2}$	$72.7_{\pm1.3}$ / $74.4_{\pm1.3}$ / **76.6**$_{\pm1.2}$
DeALOG	**76.7**$_{\pm1.0}$ / **76.8**$_{\pm1.1}$ / **76.2**$_{\pm1.0}$	**74.2**$_{\pm1.1}$ / **76.0**$_{\pm1.1}$ / **75.5**$_{\pm1.0}$	**75.2**$_{\pm1.1}$ / **76.5**$_{\pm1.1}$ / $75.3_{\pm1.0}$

Table 3: Accuracy (%) on MMQA full, MMQA text/table, and WikiTQ with $\pm$95% bootstrap CIs (1,000 resamples, 5 seeds). Format: LLaMA-3 8B / Mistral 7B / Qwen-3 8B. Bold = small differences within overlapping CIs. DeALOG attains the per-column best on every column except WikiTQ / Qwen-3, where TableCritic reports 76.6% (CI overlaps with DeALOG's 75.3%).

specialized systems and DeALOG alike sit within a narrow band, including on MMQA and WikiTQ (Table 3); we make no claim of a clean-accuracy advantage. This makes accuracy *alone* insufficient, rather than uninformative, for our central question of what decentralization buys: a few cells still separate (e.g. TableCritic on TAT-QA, DeALOG on MMQA/WikiTQ), but the spreads are narrow under matched conditions, which motivates the robustness and transparency analyses that follow.

Tool-enabled comparison. DeALOG runs without a code interpreter in both settings of Tables 10 and 11 (App. F.1; PoT rationale in §3). Tool gains on FinQA average +2.0 EM across baselines, yet DeALOG without code execution still wins on 2 of 3 backbones and is within 0.4 EM on the third; and on MMQA and WikiTQ (Table 11, dev split) DeALOG without tools exceeds even the *tool-enabled* baselines on all three backbones, indicating its advantage there is not attributable to tool access.

Efficiency and Resource Utilization: DeALOG uses fewer LLM calls than most multi-agent baselines (3.1 vs. 3.4–3.8; Table 12) but the highest *sequential* latency (0.95s); the parallel variant Par-R6 cuts this to 0.68s at no accuracy cost and is the production operating point (Pareto sweep in Table 20). Learned gating reduces tokens by up to 18% and latency by up to 2.18× at ≤ 0.01 EM cost (Table 14), because the gate fires preferentially on the most expensive examples.

Ablations and log behavior. Doubling the context window (4k→8k) yields only +0.013 average EM (Table 15, App. F)which is a reasoning difficulty, not context length, is the bottleneck. Long traces degrade gracefully: accuracy stays high through 5–6 steps and 0.88 at 7–8 on MMQA-dev slices (sample-size caveat, Table 18), settling at ∼0.70 on the larger CRT-QA test set. Removing ROUGE-L deduplication adds 13.3% tokens at ΔEM$=0.000$ (Table 16; per-agent rates in Table 17, App. F); with gating it gives ∼25% token reduction. Per-method latency is in Figure 4 (App. F).

Corruption protocol. For the robustness experiments in Tables 5 and 23, we inject controlled corruption into both (i) retrieved text spans returned by the BM25+miniLM retriever and (ii) table cell values from the source tables, under two families. *Structural/nu-*

meric noise samples one of three operations with equal probability per corrupted item: *numeric perturbation* (replace a numeric cell or quoted number with a value drawn uniformly from a $\pm 10\%$ band around the original), *row swap* (exchange the contents of two random rows within the same table), and *deletion* (remove the span or replace the cell with an empty token). *Semantic* noise samples between *paraphrase substitution* (replace a span with a fluent but content-altering LLM paraphrase) and *entity swap* (replace a named entity or quantity with a plausible same-type distractor drawn from the example's own evidence). Corruption is applied at two stages: *before any agent reads the evidence* (so retrieval and table-parsing agents see corrupted inputs), and additionally to log entries posted by `TableAgent` and `ContextAgent` before downstream agents consume them (the same in-log injection point used for the fault-injection ablation in Table 23). The corruption *rate* (0%, 10%, 20%, 30% in Table 5; 10%, 20%, 30% in Table 23) is the fraction of evidence items perturbed in a given run. We run the protocol on MMQA and, as a second table-centric dataset, TAT-QA. The corruption draws are keyed by (seed, example) and are therefore shared across all nine systems, i.e., Planner, Plan→Log, Planner++, AutoTQA, REWoO, the three shared-memory systems LBMAS, BLACKBOARD-DS, and METAGPT, and DEALOG (Planner++ and the shared-memory systems on the panels where they are run), so every column of Table 5 is evaluated on identical corrupted inputs (paired comparisons over five seeds). *Calibration.* The synthetic rates are not arbitrary: on the clean MMQA pipeline we measured the organic frequency of each upstream failure, i.e., retrieval miss 14%, OCR misread 13%, caption error 10%, and table-parse error 9% (App. F.18), so the 10–30% band brackets what the deployed pipeline actually produces.

Planner and Plan→Log: controlled architectural variants. The Planner and Plan→Log comparators (Table 5, Table 23, Figure 5) are deliberately *minimal internal variants* of DEALOG that isolate one variable: whether the system commits to a centralized natural-language plan before reasoning begins. They are not SoTA external planners, those (REWoO, AutoTQA, Chameleon, Lumos, FireAct) are evaluated on clean inputs in Tables 2–3. All three share DEALOG's backbone, retriever, prompt budget, and guardrails; only the planning component varies.

Planner is a single LLM call that plans, solves, and answers; *Plan→Log* seeds a heuristic plan into the normal DEALOG coordinator and never revises it, so the gap to DEALOG measures the *cost of pre-committing to a plan under noise* with LLM, retriever, and stopping fixed (pseudocode and details in App. B).

External validity under real planners. Two external multi-agent planners, AutoTQA (Zhu et al., 2024b) and REWoO (Xu et al., 2023), run under the identical protocol and seeds degrade far more than DEALOG under structural noise (21.4% and 21.1% relative vs. 6.4%), tracking the internal Planner (24.7%), and the ordering holds under semantic noise (13.6% vs. 19.2%). The internal Planner is thus not a strawman: real planners share its robustness class regardless of sophistication. A third planner, HuggingGPT (Shen et al., 2023), on HybridQA reproduces the ordering (App. F.26).

Robustness to evidence corruption. DEALOG's advantage is on robustness, not clean accuracy. Under structural MMQA noise, DEALOG retains 73.0% EM at 30% corruption, versus 55.0 (internal Planner), 66.0 (Plan→Log), 64.0 (the budget-matched Planner++), 55.0 (AutoTQA), and 52.7 (REWoO)—a 6.4% relative drop against 24.7% (Planner), 21.4% (AutoTQA), and 21.1% (REWoO) (Table 5, Figure 5); the gap to AutoTQA widens from 8.0 points on clean inputs to 18.0 points at 30%. The pattern persists under *semantic* noise (67.4% vs. 61.4% for the strongest comparator at 30%) and on the second dataset, TAT-QA (Table 25, App. F: 62.8 vs. 58.9 at 30%), and every DEALOG lead survives a paired bootstrap over the shared seeds. Because no central plan inscribes an early error and all evidence stays globally visible, a corrupted entry is outweighed by independently grounded peer entries rather than propagated. This is not the verifier, which catches only a minority of corruptions (below); the mechanism is located in §3.2. Conclusions hold across retriever (BM25→dense) and depth ($k \in \{3, 5, 10\}$): DEALOG's relative drop stays 0.06–0.07 (App. F.27).

Shared-memory systems share the planners' robustness class. The novelty question is whether the gain comes from shared memory (a known idea) or from removing semantic control. Table 5 answers it directly: LBMAS, BLACKBOARD-DS, and METAGPT all coordinate through a shared memory, yet at 30% corruption they drop 18.2%, 21.4%, and 30.9%, i.e., 2.8–$4.8\times$ DEALOG's 6.4%, tracking the planners rather than DEALOG. The ordering holds on TAT-QA (DEALOG 16.0% vs. LBMAS 19.1% and BLACKBOARD-DS 21.9%; Table 25). What these systems retain, and DEALOG removes, is an LLM control unit; §3.2 swaps only that component.

Naturally occurring failures. Synthetic corruption could in principle overstate the effect, so we also slice the *clean* MMQA run by organic upstream failures, i.e., examples where the retriever missed gold evidence, PaddleOCR misread a cell, or the BLIP-2 caption was wrong, with no injection at all (Table 4, top). DEALOG leads the Planner by $+10.0$, $+9.0$, and $+7.0$ EM on the three slices, and under *realistic* injected noise at 20% (OCR confusions, header drift, table cropping; Table 4, bottom) it degrades less on every error type. The synthetic, realistic, and organic views agree, so the robustness claim does not rest on the corruption generator.

Long-horizon reasoning. As chains lengthen, the centralized Planner collapses (78.0% at 2–3 operators to 50.0% at 8+), while DEALOG stays substantially flatter (80.0 to 72–77; Table 5). We emphasize this *relative stability* causes the widening gap to the Planner, rather

(a) Natural-failure slices of the clean MMQA run (no injection)

Slice	n	DeALOG EM	Planner EM	Δ
Retriever missed gold	180	**54.0**	44.0	+10.0
PaddleOCR misread	120	**49.0**	40.0	+9.0
BLIP-2 caption wrong	90	**46.0**	39.0	+7.0

(b) Realistic injected noise at 20% (MMQA dev; clean ref. 80.1 / 75.2)

Error type	DeALOG Δ	Planner Δ	Adv.
OCR confusion	−7.3	−9.1	+1.8
Header drift	−10.2	−12.5	+2.3
Table cropping	−12.7	−15.4	+2.7

Table 4: Robustness beyond synthetic corruption. (a) Organic failure slices carved from the clean run by inspecting the log for missing gold evidence, OCR, or caption errors; (b) per-type realistic noise, each applied in isolation (absolute EMs in Table 26, App. F.16). Five shared seeds throughout.

than the absolute value at 8+ operators, where the slice is small and the point estimate should be read with that caveat. App. 19 stratifies the dominant error source by chain length: retrieval misses dominate short chains (46%) but cede to summarization errors on the longest (34% at 8+ vs. 20% at 2–3), consistent with log clutter as the long-horizon bottleneck (Limitations).

3.2 Where the Robustness Comes From

Reviewers reasonably ask *which* component buys the robustness. We answer with two controlled swaps on the same agents, retriever, prompts, backbone (LLaMA-3 8B), and paired corruption draws (Table 6).

(a) Attribution ablation. Disabling one mechanism at a time, removing *global evidence visibility* inflates the clean→30% drop most (−12.7 → −17.9), the verifier fallback less (−15.6), while deduplication and `should_act` gating are near-neutral (−13.3, −13.6). Robustness is thus structural, i.e., every agent re-reads all evidence, and not an artifact of the verifier or the efficiency gates.

(b) Controller swap. DeALOG-LLMCtrl keeps everything and adds one LLM call per round that reads the log and selects the next agent, the control-unit pattern of LbMAS. The clean gap is 0.02 EM, but it widens monotonically with corruption to 0.09 at 30% (4.5×): LLM-Ctrl drops 15.8% against DeALOG's 6.4%, at 1.41× the tokens and +0.32 s latency. A semantic selector that reads noisy entries becomes a single point of failure; a scheduler that never reads them cannot. Together, (a) and (b) isolate the paper's central claim: the gain is from *removing semantic central control*, not from shared memory, which every system in Table 5 already has.

What the scheduler actually does. Because DeALOG still uses `should_act` heuristics, a learned stop gate, and verifier-triggered re-engagement, one may ask whether control has merely moved rather than been removed. Measured over all six datasets (LLaMA-3 8B, 5 seeds), every trigger conditions on *log state* (is there a table? is an image mentioned? did the verifier flag?), never on question semantics; the VisualAgent fires only on image/OCR questions, the learned gate never fires on CRT-QA, and only 5% of final answers change after re-engagement, so re-engagement is a rare, bounded correction rather than a hidden re-planning loop.

Transparency and faithfulness. The shared log is not only a coordination substrate but a provenance-anchored trace. On FeTaQA (mean over five seeds), log-groundedness, the fraction of answer *tokens* traceable to a log entry is 0.72, with 76/100 and 73/100 answers rated "supported" by two independent LLM judges (Table 21). Token-level grounding is conservative: it scores faithful rewording as ungrounded, so the lexical 0.72 sitting below the semantic 76% reflects paraphrase, not unsupported content which is confirmed by BERTScore F1 0.90, QAGS in $[0.63, 0.68]$, and an entailment-based grounding of 0.78. Program-level coverage on FinQA corroborates this beyond lexical matching: 0.64 of DeALOG's answers reproduce the gold reasoning program's executed output under the official ±0.01 tolerance (Table 22). Fault injection (Table 23) makes the verifier's role precise: it catches a minority of corruptions (0.09/0.21/0.31 at 10/20/30%) and at 30% repairs essentially none, yet EM erodes only mildly (0.77 → 0.73) because the system falls back to its pre-flag answer rather than emit a faulty repair. The verifier is thus a *safe fallback*, not self-correction: at 30%, 61% of fallbacks preserve a correct answer and only 13% forgo a recoverable one (Table 24). A taxonomy of 155 incorrect MMQA cases (Table 36) shows the asymmetry where arithmetic/unit errors caught at 88%, visual/OCR at only 17.1%, thus motivating the VLM re-verification below.

Groundedness is necessary but not sufficient; a necessity test. Groundedness shows an answer *can* be traced to log entries, not that they were *used*. We therefore delete the single entry the Summarizer cites and re-run it, on 100 correct answers per dataset: the answer flips in 64% (FeTaQA), 70% (FinQA), and 60% (MMQA) of cases (Table 28, App. F.19), so cited evidence is causally load-bearing; most non-flips have redundant support from a second entry.

Multimodal claim, stated at the right size. Splitting MMQA by what the question needs (LLaMA-3 8B, 5 seeds; Table 29, App. F.20), DeALOG scores 76.5 EM on text/table-only (n=1100; best baseline 75.0), 70.0 on caption-mediated (n=250; 68.0), 67.0 on OCR-heavy (n=300; 64.0), and 64.0 on image-dependent questions (n=380; 61.0), and the image-dependent subset also degrades most under 30% corruption (17.2% vs. 7.8% relative). The VisualAgent reads OCR/caption text, not pixels, so the advantage on genuinely visual reasoning is real but modest. Integrating the VLM re-verifier into the main loop, so that a flagged VISUAL entry is re-checked against the *original image*, raises verifier precision 0.62 → 0.70 and end-task EM on the image-dependent subset by +3.0 (64.0 → 67.0; App. F.25).

What the log looks like when it works and when it fails. App. A gives condensed traces. In the success case (Table 8, under 30% noise) a corrupted TableAgent entry is outweighed because the ContextAgent independently

Corruption	Planner	Plan→Log	Planner++	AutoTQA	REWoO	LbMAS	Bb-DS	MetaGPT	DeALOG
MMQA — numeric + structural noise									
0%	$73.0_{\pm 1.5}$	$76.0_{\pm 1.5}$	$76.0_{\pm 1.5}$	$70.0_{\pm 1.5}$	$66.8_{\pm 1.6}$	$74.0_{\pm 1.5}$	$72.5_{\pm 1.5}$	$68.0_{\pm 1.6}$	$\mathbf{78.0}_{\pm 1.5}$
10%	$66.0_{\pm 1.6}$	$73.0_{\pm 1.6}$	$72.0_{\pm 1.6}$	$64.0_{\pm 1.6}$	$62.9_{\pm 1.7}$	$70.0_{\pm 1.6}$	$67.5_{\pm 1.6}$	$60.0_{\pm 1.7}$	$\mathbf{77.0}_{\pm 1.6}$
20%	$60.0_{\pm 1.8}$	$70.0_{\pm 1.8}$	$68.0_{\pm 1.8}$	$59.0_{\pm 1.8}$	$58.1_{\pm 1.8}$	$65.5_{\pm 1.8}$	$62.5_{\pm 1.8}$	$53.0_{\pm 1.9}$	$\mathbf{75.0}_{\pm 1.8}$
30%	$55.0_{\pm 2.0}$	$66.0_{\pm 2.0}$	$64.0_{\pm 2.0}$	$55.0_{\pm 2.0}$	$52.7_{\pm 2.0}$	$60.5_{\pm 2.0}$	$57.0_{\pm 2.0}$	$47.0_{\pm 2.1}$	$\mathbf{73.0}_{\pm 2.0}$
Rel. drop @30%	24.7%	13.2%	15.8%	21.4%	21.1%	18.2%	21.4%	30.9%	**6.4%**
MMQA — semantic noise (paraphrase + entity swap)									
10%	$70.4_{\pm 1.7}$	$72.1_{\pm 1.6}$	$71.0_{\pm 1.7}$	$63.0_{\pm 1.8}$	$65.0_{\pm 1.8}$	—	—	—	$\mathbf{77.2}_{\pm 1.5}$
20%	$65.2_{\pm 1.8}$	$67.8_{\pm 1.8}$	$66.0_{\pm 1.8}$	$58.6_{\pm 1.9}$	$60.1_{\pm 1.9}$	—	—	—	$\mathbf{73.0}_{\pm 1.7}$
30%	$59.0_{\pm 2.0}$	$61.4_{\pm 2.0}$	$59.8_{\pm 2.0}$	$53.4_{\pm 2.1}$	$55.2_{\pm 2.1}$	—	—	—	$\mathbf{67.4}_{\pm 1.9}$
Long-Horizon EM (%) by Chain Length (MMQA, $R = 10$)									
2–3 Ops	78.0	79.0	—	—	—	—	—	—	**80.0**
4–5 Ops	72.0	**78.0**	—	—	—	—	—	—	76.0
6–7 Ops	60.0	70.0	—	—	—	—	—	—	**72.0**
8+ Ops ($n \approx 25$)	50.0	64.0	—	—	—	—	—	—	**77.0**

Table 5: MMQA robustness under evidence corruption (structural and semantic panels) and long-horizon reasoning (bottom). Scores are EM % with ±95% bootstrap CIs (1,000 resamples, 5 shared seeds). Planner / Plan→Log / DeALOG share backbone, retriever, and prompts; AutoTQA and REWoO are external multi-agent planners, and LbMAS (Han and Zhang, 2025) (LLM-blackboard), Bb-DS = BLACKBOARD-DS (Salemi et al., 2025), and MetaGPT (Hong et al., 2023) are external shared-memory systems, all run under the same corruption seeds and harness. PLANNER++ is a centralized planner matched to DeALOG on call budget, verifier, and re-engagement (with re-planning) but *without* the shared log ("—" = not run). Bold = row max; each bold lead has a paired-bootstrap difference interval excluding zero. The clean DeALOG score (78.0%) is on the robustness slice, distinct from the full-test 76.7% in Table 3. The corresponding TAT-QA corruption panel is in Table 25 (App. F). Long-horizon rows are point estimates (no CI); the 8+ Ops slice is small ($n \approx 25$), so the finding is DeALOG's stability across chain length, not the absolute value at 8+. Different slice from Table 18 (App. F). Catastrophic errors: Figure 5.

(a) Attribution ablation (MMQA dev, EM; disable one mechanism)

Variant	Clean	30% struct.	Δ
Full DeALOG	80.1	67.4	−12.7
− global visibility	76.8	58.9	**−17.9**
− verifier fallback	78.7	63.1	−15.6
− deduplication	79.7	66.4	−13.3
− `should_act` gating	79.4	65.8	−13.6

(b) Controller swap (MMQA robustness slice, EM)

Corruption	DeALOG (non-semantic)	LLMCtrl	Gap
0%	**78.0**	76.0	2.0
10%	**77.0**	73.0	4.0
20%	**75.0**	69.0	6.0
30%	**73.0**	64.0	9.0
Rel. drop @30%	**6.4%**	15.8%	2.5×
Tokens / latency	1.00× / 0.95 s	1.41× / 1.27 s	—

Table 6: Mechanism isolation, 5 shared seeds. (a) Each row disables a single component; the clean value 80.1 matches the dev reference of Table 4. (b) Only the controller differs: LLMCtrl adds an LLM agent-selector per round, matching the control unit of LbMAS (Han and Zhang, 2025); it is dominated on accuracy, cost, and degradation.

posts the correct figure, the Verifier flags the contradiction, and the re-engaged Summarizer answers from the peer entry. The matching failure (Table 9) has a mis-captioned chart as its *only* evidence; nothing contradicts it, the text-only Verifier cannot flag it, and the wrong answer is recorded with a clean provenance trail. Provenance makes that failure inspectable after the fact; it does not make the answer correct.

When is DeALOG the right choice? DeALOG trades ∼3 LLM calls and 0.68 s (Par-R6) per query for robustness and a provenance trace, so it is the wrong tool for latency-critical, clean-input QA where one CoT call suffices. It pays off where inputs are noisy or answers must be defensible: audited financial QA over filings, pipelines over scanned or OCR'd documents where retrieval, OCR, and caption errors occur organically at the 9–14% rates we measure (App. F.18), and long compositional chains where a committed plan collapses. Monitoring and drift follow standard MLOps practice (Chakraborty et al., 2025c), and when evidence is distributed across parties the log's provenance requirements interact with federated retrieval (Chakraborty et al., 2025b).

4 Comparison with Related Work

Table, multimodal, and CoT reasoning. Neural table QA advanced through representation learning like TAPEX (Liu et al., 2021), TAPAS (Herzig et al., 2020), TaCube (Zhou et al., 2022), OmniTab (Jiang et al., 2022), with Lever (Ni et al., 2023) verifying SQL. For financial table-text reasoning, supervised pipelines (FinQANet (Chen et al., 2021b), TAT-LLM (Zhu et al., 2024a), HiTab (Cheng et al., 2021)) fine-tune on annotated programs and form a supervised upper bound; DeALOG is positioned against zero-shot agentic baselines. Recent multimodal multi-hop work spans extremes like Rajabzadeh et al. (2023) chain tools via sequential LLM-directed conversation, while Uni-RAG (Lim et al., 2024) skips agent decomposition (67.4% EM on MMQA). CoT prompting (Wei et al., 2022) and its Tree-/Graph-of-Thoughts extensions (Yao et al., 2023; Besta et al., 2023) reason within a single model; DeALOG instead distributes natural-language reasoning across specialized peers with a shared log, enabling separate verification.

Agentic and multi-agent frameworks. HuggingGPT (Shen et al., 2023) orchestrates expert models; Program-of-Thoughts (Chen et al., 2022) and Binder (Cheng et al., 2022) couple LLMs with executors; TabSQLify (Nahid and Rafiei, 2024) decomposes table queries; AutoTQA (Zhu et al., 2024b) plans sub-

queries; and for multimodal QA, MMQA (Talmor et al., 2021) and MAM-MQA (Rajput et al., 2025) coordinate modality agents (we use BLIP-2/PADDLEOCR (Li et al., 2023) for image-to-text). REACT (Yao et al., 2022) interleaves reasoning and actions in one agent; AUTOGEN (Wu et al., 2024) coordinates agents by free-form conversation; LATS (Zhou et al., 2024) adds Monte-Carlo tree search; BELLE (Zhang et al., 2025) uses bi-level debate-and-plan; and REAGENT (Zhao et al., 2025) adds reversible backtracking. All rely on a controller that inspects task semantics which is the centralized-plan property our non-semantic scheduler avoids (Table 1), whereas DEALOG replaces planning and backtracking with an append-only log and one verifier-triggered re-engagement. Specialized-agent QA systems similarly centralize *aggregation* through assignment/decision, planner–critic, or organizer agents (CT2C-QA (Zhao et al., 2024), MMCTAGENT (Kumar et al., 2024), VIDEOMULTIAGENTS (Kugo et al., 2025)), as does the mixture-of-agents line (MOA (Wang et al., 2024a), SMOA (Li et al., 2025)); DEALOG has no aggregator, judge, or moderator with global control while coordination is mediated entirely by the shared log.

Decentralized shared-memory agents. DEALOG shares a memory-mediated paradigm with recent systems but differs in control, and we make the distinction empirical rather than rhetorical (App. F.21, Table 30). METAGPT (Hong et al., 2023) coordinates role agents through a shared message pool under SOP-driven orchestration; LBMAS (Han and Zhang, 2025) places an LLM control unit over an LLM blackboard; and BLACKBOARD-DS (Salemi et al., 2025) adds an LLM controller for data-science discovery. All three retain a semantic controller that reads the shared memory to choose the next agent, and all three, run under our harness, degrade 18–31% at 30% corruption versus DEALOG's 6.4% (Table 5); our DEALOG-LLMCtrl ablation, which adds only such a controller to DEALOG's own agents, degrades 2.5× more (−15.8% vs. −6.4%; Table 6). The lesson that a simpler control structure can match or beat elaborate ones on both accuracy and robustness echoes findings in other structured-reasoning settings (Chakraborty et al., 2025a). Others route through a central critic (MDOCAGENT (Han et al., 2025)), consensus debate (MEDIHIVE (Wang and Yang, 2026)), a pre-built graph (G2CP (Khaled and Monticolo, 2026)), a cross-user memory *fabric* (Tiwari and Gupta, 2026), or open-ended argumentation (Jiang et al., 2024); DEALOG instead shares one natural-language log under a deterministic scheduler, prioritizing provenance and fault tolerance over consensus-by-debate. Most directly, EQUIMEM (Meng et al., 2026) calibrates each entry against memory to filter corrupted writes; this is complementary to DEALOG, which keeps a corrupted entry globally visible so peer re-reading outweighs it rather than filtering at write time is an equilibrium-style calibration of log writes could sharpen our re-engagement. The modular `should_act`/`act` interface lets new specialists plug in without retraining, and natural-language entries keep the framework backbone-agnostic (σ=0.55 across LLaMA-3 / Mistral / Qwen-3), all without task-specific fine-tuning.

5 Conclusion and Future Work

We presented DEALOG, a central-planner-free multi-agent framework coordinating specialized agents through a shared, provenance-typed natural-language log. Its contribution is a finding: with agents, retriever, prompts, and corruption seeds held fixed, removing the semantic control unit retained by prior blackboard and shared-memory systems buys robustness (6.4% relative degradation at 30% corruption versus 18–31% for planner and blackboard systems); the gain comes from global evidence visibility and holds on organic failures. Next steps are extending the study to shared-memory domains, adaptive log compression, and pixel-grounded verification.

Limitations

Visual/OCR verification is weak. VerificationAgent detects 17.1% of visual/OCR misreads versus 88% of arithmetic/unit errors (Table 36) and cannot recover BLIP-2 or PaddleOCR losses. The VLM re-verifier (App. F.25) yields the smallest gains on image-dependent questions (Table 29). *Absolute degradation on long traces.* DEALOG falls to 0.88 EM at 7–8 steps (Table 18) and ∼0.70 on CRT-QA; deduplication and summarization reduce log clutter. Comparison (Table 10) uses one Python executor. The shared log descends from blackboard systems (Erman et al., 1980; Schwartz, 1995a) and LLM reboots (Hong et al., 2023; Han and Zhang, 2025; Salemi et al., 2025); removing the semantic control unit buys robustness (Table 6), but transfer outside multimodal QA is untested. Injected rates bracket organic frequencies (App. F.18), but organic slices are small (Table 4) and from one dataset. Sequential calls add latency, and knowledge is bounded by 7–8B backbones and the per-example evidence pool.

Ethical Considerations

All experiments use public datasets under their licenses; AI assistance improved clarity only, and the authors reviewed all edits and remain accountable. Three risks are specific to a shared-log design. The log records *provenance, not correctness*: traces can create false confidence if conflated with accuracy, so they should support auditing, not replace it. Decentralized coordination costs more compute than a single pass (Tables 12, 13) and should be weighed against latency or energy constraints. Shared memory can propagate biased entries unless audited (§4).

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2023. Graph of thoughts: solving elaborate problems with large language models. In *AAAI Conference on Artificial Intelligence*, volume 38 of *AAAI'24/IAAI'24/EAAI'24*, pages 17682–17690. Association for the Advancement of Artificial Intelligence (AAAI).

Abhijit Chakraborty, Chahana Dahal, Ashutosh Balasubramaniam, Tejas Anvekar, and Vivek Gupta. 2025a. Is architectural complexity overrated? competitive and interpretable knowledge graph completion with RelatE. *arXiv.org*, arXiv:2505.18971.

Abhijit Chakraborty, Chahana Dahal, and Vivek Gupta. 2025b. Federated retrieval-augmented generation: A systematic mapping study. *Findings of the Association for Computational Linguistics: EMNLP 2025*, arXiv:2505.18906.

Abhijit Chakraborty, Suddhasvatta Das, and Kevin Gary. 2025c. Machine learning operations: A mapping study. In *Communications in Computer and Information Science*, pages 3–21. Springer Nature Switzerland.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021a. Evaluating large language models trained on code. *arXiv.org*, arXiv:2107.03374.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2023.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and 1 others. 2021b. FinQA: A dataset of numerical reasoning over financial data. In *Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711. Association for Computational Linguistics.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2021. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Annual Meeting of the Association for Computational Linguistics*, pages 1094–1110. Association for Computational Linguistics.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir R. Radev, Marilyn Ostendorf, Luke S. Zettlemoyer, and 1 others. 2022. Binding language models in symbolic languages. In *International Conference on Learning Representations*, volume abs/2210.02875.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, and 1 others. 2025. PaddleOCR-VL: Boosting Multilingual Document Parsing via a 0.9B Ultra-Compact Vision-Language Model. *arXiv preprint*.

Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and Raj Reddy. 1980. The hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *CSUR*, 12(2):349–389.

Bo Han and Songmao Zhang. 2025. Exploring advanced llm multi-agent systems based on blackboard architecture. *ArXiv*, abs/2507.01701.

Siwei Han, Peng Xia, Ruiyi Zhang, Tong Sun, Yun Li, Hongtu Zhu, and Huaxiu Yao. 2025. Mdocagent: A multi-modal multi-agent framework for document understanding. *arXiv.org*, abs/2503.13964.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333. Association for Computational Linguistics.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*.

Yucheng Jiang, Yijia Shao, Dekun Ma, Sina Semnani, and Monica Lam. 2024. Into the unknown unknowns: Engaged human learning through participation in language model agent conversations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9917–9955. Association for Computational Linguistics.

Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022. OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942. Association for Computational Linguistics.

Karim Ben Khaled and Davy Monticolo. 2026. G2cp: A graph-grounded communication protocol for verifiable and efficient multi-agent reasoning.

Joongwon Kim, Bhargavi Paranjape, Tushar Khot, and Hannaneh Hajishirzi. 2024. Husky: A unified, open-source language agent for multi-step reasoning. *arXiv.org*, arXiv:2406.06469.

Noriyuki Kugo, Xiang Li, Zixin Li, Ashish Gupta, Arpandeep Khatua, Nidhish Jain, Chaitanya Patel, Yuta Kyuragi, Yasunori Ishii, Masamoto Tanabiki, and 1 others. 2025. VideoMultiAgents: A multi-agent framework for video question answering. *arXiv.org*, arXiv:2504.20091.

Somnath Kumar, Yash Gadhia, Tanuja Ganu, and Akshay Nambi. 2024. MMCTAgent: Multi-modal critical thinking agent framework for complex visual reasoning. In *arXiv.org*.

Dawei Li, Zhen Tan, Peijia Qian, Yifan Li, Kumar Satvik Chaudhary, Lijie Hu, and Jiayi Shen. 2025. SMoA: Improving multi-agent large language models with sparse mixture-of-agents. *Lecture Notes in Computer Science*, arXiv:2411.03284.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. *arXiv preprint*.

Qi Zhi Lim, Chin Poo Lee, Kian Ming Lim, and Ahmad Kamsani Samingan. 2024. UniRaG: Unification, retrieval, and generation for multimodal question answering with pre-trained language models. *IEEE Access*, 12:71505–71519.

Qian Liu, Bei Chen, Jiaqi Guo, Zeqi Lin, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. volume abs/2107.07653.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems 36*, pages 43447–43478. Neural Information Processing Systems Foundation, Inc. (NeurIPS).

Yuqiao Meng, Sakshi Sunil Narvekar, Luoxi Tang, Rupali Rajendra Vaje, Yingxue Zhang, Muchao Ye, and Zhaohan Xi. 2026. EquiMem: Calibrating shared memory in multi-agent debate via game-theoretic equilibrium. *arXiv.org*, arXiv:2605.09278.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024. TabSQLify: Enhancing reasoning capabilities of llms through table decomposition. ArXiv: 2404.10150 [cs.CL].

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Nick Schoelkopf, Riley Kong, Xiangru Tang, and 1 others. 2021. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, arXiv:2104.00369.

Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen tau Yih, Sida I. Wang, and Xi Victoria Lin. 2023. *LEVER: Learning to Verify Language-to-Code Generation with Execution*.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. ArXiv: 2304.03442 [cs.HC].

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Annual Meeting of the Association for Computational Linguistics*, pages 1470–1480. Association for Computational Linguistics.

Hossein Rajabzadeh, Suyuchen Wang, Hyock Ju Kwon, and Bang Liu. 2023. Multimodal multi-hop question answering through a conversation between tools and efficiently finetuned large language models. *arXiv.org*.

Krishna Singh Rajput, Tejas Anvekar, Chitta Baral, and Vivek Gupta. 2025. Rethinking Information Synthesis in Multimodal Question Answering A Multi-Agent Perspective.

Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 4(1-2):1–174.

Alireza Salemi, Mihir Parmar, Palash Goyal, Yiwen Song, Jinsung Yoon, Hamed Zamani, Hamid Palangi, and Tomas Pfister. 2025. LLM-based multi-agent blackboard system for information discovery in data science. *arXiv.org*, arXiv:2510.01285.

Humza Sami, Mubashir ul Islam, Pierre-Emmanuel Gaillardon, and Valerio Tenace. 2025. Adaptive Multi-Agent Reasoning via Automated Workflow Generation. *arXiv preprint*.

David G. Schwartz. 1995a. Blackboard systems. *Cooperating Heterogeneous Systems*, pages 27–42.

David G. Schwartz. 1995b. *Blackboard Systems*. Insight Series in Artificial Intelligence. Springer US, Wokingham.

David G. Schwartz. 1995c. Blackboard systems. *Cooperating Heterogeneous Systems*, pages 27–42.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. pages 38154–38180. Neural Information Processing Systems Foundation, Inc. (NeurIPS).

Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. 2021. MultiModalQA: Complex question answering over text, tables and images. ArXiv: 2104.06039 [cs.CL].

Anjikya Tiwari and Vibhuti Gupta. 2026. A memory fabric for conversational AI agents enabling shared and persistent multiuser memory. *Discover Artificial Intelligence*, 6(1).

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024a. Mixture-of-agents enhances large language model capabilities. *International Conference on Learning Representations*, arXiv:2406.04692.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. *arXiv preprint*.

Xiaoyang Wang and Christopher C. Yang. 2026. MediHive: A decentralized agent collective for medical reasoning. In *IEEE International Conference on Healthcare Informatics*. Springer Science and Business Media LLC. ArXiv:2603.27150.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and 1 others. 2024b. Chain-of-table: Evolving tables in the reasoning chain for table understanding. volume abs/2401.04398.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Neural Information Processing Systems*, NIPS '22, pages 24824–24837, Red Hook, NY, USA. Neural Information Processing Systems Foundation, Inc. (NeurIPS).

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang (Eric) Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Ahmed Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2024. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Colm 2024*.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv.org*, abs/2305.18323.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: deliberate problem solving with large language models. *Advances in Neural Information Processing Systems 36*, pages 11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing reasoning and acting in language models. ArXiv: 2210.03629 [cs.CL].

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 174–184. ACM.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2024. Lumos: Learning agents with unified data, modular design, and open-source llms. In *arXiv.org*. ArXiv:2311.05657. Also published as "Agent Lumos: Unified and Modular Training for Open-Source Language Agents".

Peiying Yu, Guoxin Chen, and Jingjing Wang. 2025. Table-Critic: A Multi-Agent Framework for Collaborative Criticism and Refinement in Table Reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17432–17451, Vienna, Austria. Association for Computational Linguistics.

Taolin Zhang, Dongyang Li, Qizhou Chen, Chengyu Wang, and Xiaofeng He. 2025. BELLE: A bi-level multi-agent reasoning framework for multi-hop question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4184–4202, Vienna, Austria. Association for Computational Linguistics.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M. Patel. 2024. Reactable: Enhancing react for table question answering. *Proceedings of the VLDB Endowment*, 17(8):1981–1994.

Zhehao Zhang, Xitao Li, Yan Gao, and Jian-Guang Lou. 2023. CRT-QA: A Dataset of Complex Reasoning Question Answering over Tabular Data. In *Conference on Empirical Methods in Natural Language Processing*, pages 2131–2153, Singapore. Association for Computational Linguistics.

Bowen Zhao, Tianhao Cheng, Yuejie Zhang, Ying Cheng, Rui Feng, and Xiaobo Zhang. 2024. CT2C-QA: Multimodal question answering over chinese text, table and chart. In *ACM Multimedia*, pages 3897–3906. ACM.

Xinjie Zhao, Fan Gao, Rui Yang, Yingjian Chen, Yuyang Wang, Ying Zhu, Jiacheng Tang, and Irene Li. 2025. ReAgent: Reversible multi-agent reasoning for knowledge-enhanced multi-hop QA. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4067–4089. Association for Computational Linguistics.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning acting and planning in language models.

Fan Zhou, Mengkang Hu, Haoyu Dong, Zhoujun Cheng, Shi Han, and Dongmei Zhang. 2022. Tacube: Pre-computing data cubes for answering numerical-reasoning questions over tabular data. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, arXiv:2205.12682.

Wei Zhou, Mohsen Mesgar, Annemarie Friedrich, and Heike Adel. 2025. Efficient multi-agent collaboration with tool use for online planning in complex table question answering. ArXiv: 2412.20145 [cs.CL].

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Annual Meeting of the Association for Computational Linguistics*, pages 3277–3287. Association for Computational Linguistics.

Fengbin Zhu, Ziyang Liu, Fuli Feng, Chao Wang, Moxin Li, and Tat-Seng Chua. 2024a. TAT-LLM: A specialized language model for discrete reasoning over tabular and textual data. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 310–318. ACM.

Jun-Peng Zhu, Peng Cai, Kai Xu, Li Li, Yishen Sun, Shuai Zhou, Haihuang Su, Liu Tang, and Qi Liu. 2024b. AutoTQA: Towards autonomous tabular question answering through multi-agent large language models. *Proceedings of the VLDB Endowment*, 17(12):3920–3933.

Urša Zrimšek and Erik Štrumbelj. 2024. Quantifying Uncertainty: All We Need is the Bootstrap?

Appendix Index

A. Log Schema and Full Traces 16
A.1 Log Schema 16
A.2 Example Log Trace (TAT-QA) 16
A.3 Success Trace under Corruption (MMQA) 16
A.4 Example Failure Trace (MMQA) 16

B. Control Flow Details 16
B.1 Controller Loop and Scheduling 16
B.2 Summarization Trigger 16
B.3 Optional Re-Engagement 16
B.4 Stopping Criteria 16
B.5 Asynchrony vs. Sequential Execution 16

C. Agent Prompts (Verbatim) 17
C.1 TableAgent Prompt 17
C.2 ContextAgent Prompt 17
C.3 VisualAgent Prompt 17
C.4 SummarizingAgent Prompt 17
C.5 VerificationAgent Prompt 17

D. Additional Figures and Examples 18
D.1 VisualAgent Bar-Chart Example 18
D.2 Architecture Figure 18

E. Implementation Details and Pseudocode 18
E.1 Random Seeds and Runs 18
E.2 Log Truncation Policy 18
E.3 No-Progress Detection 18
E.4 Agent Triggers and Coordination Heuristics 18
E.5 SummarizingAgent Prompt Example 18

F. Tables and Figures Moved from Main Text 19
F.1 Tool-Enabled Comparison 19
F.2 Per-Query Efficiency 19
F.3 Algorithm Pseudocode 19
F.4 Stage-Wise Latency Decomposition 20
F.5 Gating-Policy Effect on DeALOG 20
F.6 Context Window Ablation 20
F.7 Deduplication and Per-Agent Duplicate Rates 20
F.8 Latency Comparison Across Methods 21
F.9 Long-Horizon Reasoning Breakdown 21
F.10 Latency–Accuracy Pareto Sweep 21
F.11 FeTaQA Faithfulness 21
F.12 Program-Level Faithfulness (FinQA) 22
F.13 Fault-Injection Robustness 22
F.14 Verifier Fallback Breakdown 22
F.15 TAT-QA Robustness Panel 22
F.16 Realistic-Noise Robustness 22
F.17 Attribution Ablation on the Robustness Panel 22
F.18 Empirical Error Distribution of the Clean Pipeline 22
F.19 Counterfactual Necessity Test 22
F.20 MMQA Modality Subsets 23
F.21 Differentiation from Prior Shared-Memory Systems 23
F.22 Per-Dataset Metrics and Answer Extraction 23
F.23 Dataset Statistics 23

F.24 Catastrophic Error Rates 23
F.25 VLM-Verifier Prototype 24
F.26 Additional Planner and Robustness Dataset 24
F.27 Harness Sensitivity 24
F.28 MMQA Error Taxonomy 24
F.29 LbMAS-Style LLM-Controller Ablation: Protocol 24
F.30 Blackboard Baselines on the Robustness Panels 24

A Log Schema and Full Traces

Log Schema. The shared log is a chronological list of messages labeled with the producing agent and type, using a simple schema: each entry is a tuple (`Agent`, `Type`, `Content`). For example, an entry might be (SummarizingAgent, "Answer", "Therefore, the answer is 42."). Metadata like timestamps or step indices preserve order, and each entry may carry a reference tag for provenance tracking. All agents have global visibility, which is critical, as restricting an agent's view degrades performance, shown in ablations (e.g., only 68.3 EM with partial log access). Table 7 presents a condensed log for a sample question, showing how entries are structured and the final answer emerges from agent interactions.

Example Log Trace (TAT-QA). Table 7 shows the complete example referenced in the main text.

Agent (Type)	Log Entry Content (condensed)
User (Query)	Q: "By how much did the revenue increase from 2018 to 2019, and what is the source of this increase according to the report?"
TableAgent (Lookup)	TableAgent: "Revenue in 2018 was \$50M, revenue in 2019 was \$55M (from Table 1)."
ContextAgent (Quote)	ContextAgent: "According to the report: 'The revenue increase in 2019 was primarily due to higher sales volume.' "
SummarizingAgent (Answer)	SummarizingAgent: "The revenue increased by \$5 million from 2018 to 2019, and this increase was mainly driven by higher sales volume. **Answer:** \$5M increase, due to higher sales volume."
VerificationAgent (Check)	VerificationAgent: "Verified. The table shows \$50M → \$55M (+\$5M), and the context confirms higher sales volume as reason." (No issues flagged.)

Table 7: Full example log trace illustrating inter-agent collaboration and provenance-aware entries.

This is a simplified version of a log trace for a real question from the TAT-QA dev set: "By how much did the revenue increase from 2018 to 2019, and what is the source of this increase according to the report?".

This example shows how the TableAgent provided the numerical part, the ContextAgent provided the explanatory part, and the SummarizingAgent combined them. The VerificationAgent double-checked the arithmetic and consistency, then approved. The final answer the user sees (from the Summarizer's answer line) is clear and supported by evidence. This case also demonstrates the advantage of multi-agent: a single model might have extracted the numbers but not known the reason (or vice versa); by dividing the task, each piece was captured.

Success (30% structural noise): corrupted entry outweighed		
Table (LOOKUP)	"Population 2010: **1,204,000**" [corrupted: gold 1,024,000]	
Context (QUOTE)	"…the 2010 census counted 1,024,000 residents…"	
Summ. (ANSWER)	"Answer: 1,204,000"	
Verif. (FLAG)	"Table entry contradicts quoted census figure; re-check."	
Summ. (ANSWER)	"Answer: 1,024,000 (per Context quote, step 2)"	✓

Table 8: Condensed MMQA success trace under 30% structural noise; the corrupted LOOKUP is outweighed by an independent QUOTE. The matching failure trace is Table 9.

Example Success Trace under Corruption (MMQA).

Example Failure Trace (MMQA). Table 9 gives the full log behind the condensed failure case of Table 8. The question asks for a company's 2019 revenue shown only in a bar chart. BLIP-2 captions the bar as \$4.1M (the image shows \$4.7M); the TableAgent correctly reports that no table carries the value; the Summarizer answers from the sole VISUAL entry; and the Verifier, which sees only text, finds the answer supported. The provenance trail is complete and correct, but the content it points to is wrong. Two properties make the case diagnosable after the fact: the ANSWER cites exactly one entry, and that entry is of type VISUAL with no corroborating LOOKUP or QUOTE. The VLM re-verifier (App. F.25) targets precisely this pattern by re-checking flagged or uncorroborated VISUAL entries against the original image.

Agent (Type)	Log Entry Content (condensed)
User (Query)	Q: "What was the company's revenue in 2019 according to the chart?"
VisualAgent (Visual)	"The bar chart shows revenue of \$4.1M in 2019 and \$5.2M in 2020 (caption)." [image: 4.7M]
TableAgent (Lookup)	"No revenue column is present in the provided table."
ContextAgent (Quote)	"No relevant info found."
SummarizingAgent (Answer)	"The chart reports 2019 revenue of \$4.1M. **Answer:** \$4.1M (per step 1)."
VerificationAgent (OK)	"The answer is supported by the Visual entry at step 1; no contradicting entries."

Table 9: Full failure trace: a mis-captioned chart is the only evidence, so peer entries cannot outweigh it and the text-only verifier cannot flag it.

B Control Flow Details

Controller Loop and Scheduling. A key challenge in decentralized agent systems is preventing chaos, such as overlapping communication, repetitive queries, or missing stopping points. We design a controller loop (a simple scheduler, not a planner) for turn-taking and stopping, as in Algorithm F.3. The system cycles through agents in rounds, with TableAgent, ContextAgent, and VisualAgent contributing in a fixed order, avoiding race conditions. Each agent adds to the log if necessary. Infinite loops are prevented by agents recognizing if their

last input suffices, e.g., TableAgent tracks reported table columns. Heuristics include agents posting "no relevant info found" or abstaining if nothing new is found, until significant log changes occur.

Summarization Trigger. After these specialist agents act, the SummarizingAgent is invoked. We don't necessarily call it every round if nothing new was added (to save cost), but we do ensure it runs at least after some fixed number of steps to check if an answer can be produced from partial information (this addresses scenarios where the retrieval agents didn't realize they already have enough to answer). The SummarizingAgent may output a final answer or a progress summary. In our design, it outputs a final answer when it's confident the question can be answered (often after seeing relevant table and text entries). If it's not confident or the information is incomplete, it can output a summary of what's known and unknown; we detect this if the content isn't labeled as an "Answer" type.

Optional Re-Engagement. If the SummarizingAgent indicates missing info or ambiguity, we loop again, allowing retrieval agents to use the summary as new context. This re-engagement gives agents a second chance with a clearer target. We trigger re-engagement when the SummarizingAgent outputs "I don't have X" or the VerificationAgent flags an issue. Ambiguity is detected by scanning for phrases like "not sure" or using a classification prompt to evaluate completeness. These heuristics worked well: re-engagement was triggered in about 15% of cases, often resolving the issue by the second try. We limit to one re-engagement to avoid loops.

Stopping Criteria. The loop ends when the SummarizingAgent's answer passes verification (if enabled), or when no agent has anything new to add (`updated` remains false through a full cycle), or when a safety limit of steps is reached. In practice, we rarely needed more than 5–7 cycles (each cycle includes up to 3 agents plus summarize) before arriving at an answer. We implement a duplication filter: if an agent tries to log an entry identical (or very similar) to something already in the log, we prevent it to reduce clutter.

Asynchrony vs. Sequential Execution. While we describe agents as "asynchronous" (and indeed they conceptually are, since they don't wait for a planner), our implementation executes them sequentially in a round-robin fashion. We simulated parallel execution and saw an end-to-end latency reduction of ∼25–30% (since some calls overlap). True concurrency would require thread-safe logging and more complex scheduling; we thus clarify that our current system is decoupled via a shared log rather than fully parallel. Our method makes ∼3.1 LLM calls per query on average versus 1 for a single-agent CoT and ∼3.8 for a planner-based system.

Planner and Plan→Log variants (pseudocode). *Planner* is a single LLM call: a planner-style prompt asks the model to produce a short plan, solve each step, and return an `Answer:` line (extracted by the shared rule of §3); no second call, tool, or retry. *Plan→Log* seeds a short heuristic plan (from numeric/table/comparison cues) as `initial_plan` into the normal DEALOG coordinator (R=6), inheriting log-mediated execution but starting from a pre-committed plan. Neither re-plans: on a verifier FLAG, Plan→Log runs DEALOG's single re-engagement round is re-reading the log and appending corrected entries, but does *not* revise or discard the seeded plan, so the comparison isolates the cost of the initial plan commitment rather than of any re-planning ability. The gap between these variants and DEALOG therefore measures the *cost of pre-committing to a plan under noise*, holding LLM, retriever, and stopping fixed.

Formalizing the above; verbatim prompts ship in the released code. *Planner* (single call, no recovery):

1. `out` ← LLM(planner_prompt ⊕ question ⊕ evidence) // one call: produce plan, solve each step
2. `answer` ← extract_after("Answer:", `out`) // shared rule
3. **return** `answer` // no second call, tool, or retry

Plan→Log (pre-committed plan, log execution, one re-engagement, no re-plan):

1. `initial_plan` ← heuristic_plan(question) // numeric/table/comparison cues
2. seed `initial_plan` into the log; run the DEALOG coordinator (R=6)
3. **on** verifier FLAG: one re-engagement (re-read log, append corrected entries); *do not* revise or discard `initial_plan`
4. **return** SummarizingAgent answer

All three share DEALOG's backbone, retriever (k=5), per-call prompt budget, temperature, and stopping; only the planning component differs. On call *budget*: the minimal Planner issues one LLM call, DEALOG averages ∼3.1, and external planner systems ∼3.8—so DEALOG's robustness is not attributable to a larger call budget, since the external planners that use *more* calls degrade more (App. F.26, Table 5). A budget- and recovery-matched centralized planner (PLANNER++: same calls, verifier, and re-engagement with re-planning, but no shared log) still drops 15.8% under structural MMQA corruption (76.0 → 64.0) versus DEALOG's 6.4%, and trails even the log-bearing Plan→Log (Table 5)—isolating the shared log's contribution from compute and recovery; the same ordering (Planner++ below the log-bearing Plan→Log, both below DEALOG) holds under semantic MMQA noise and on TAT-QA (Table 25).

C Agent Prompts (Verbatim)

TableAgent Prompt. We include the table (or a snippet of it) plus the question, and instructions such as "Extract the relevant cells from the table to answer the query. If calculations are needed, do them. Provide the

result in one sentence with reference." We give an example in the prompt ("Question: ...; Table: ...; Response: TableAgent: ..."). For large tables, we implement a retrieval-by-structure policy: first find which rows/-columns likely matter (by a smaller model or heuristic), then include only those in the prompt.

ContextAgent Prompt. We provide the question and retrieved text (e.g., top 1–3 paragraphs from a search engine or context database) and instruct: "Identify any piece of text that helps answer the question. Quote it or paraphrase concisely." We also emphasize: "Only log something if you are confident it's relevant." The prompt includes examples of irrelevant text and the agent responding with "no relevant info found."

VisualAgent Prompt. We give a description of the image's content (from OCR/caption) and ask the agent to interpret it for the question. Since OCR/caption is done externally, the VisualAgent formats the OCR/caption into a statement consumable by other agents. (We prototype re-verifying flagged VISUAL entries against the original image with a VLM in App. F.25; integrating a Visual QA model directly into the VisualAgent's reading is follow-up work.)

SummarizingAgent Prompt. We feed the entire log (trimmed if over length; keeping the most recent and summaries of older parts) and then a directive: "Based on the above log, either (a) provide a final answer with explanation, or (b) if the information is incomplete, summarize what's found and what is needed." We append: "If final answer, start with 'Therefore' or 'In conclusion'." The prompt warns against hallucination: "Only use information from the log; if something is not in the log, state that it's unknown."

VerificationAgent Prompt. "The question, answer, and supporting log are above. Verify each part of the answer. If any part seems incorrect or unsupported, explain and flag it. If everything is consistent, reply with OK." If the VerificationAgent flags an issue, we trigger one more round of SummarizingAgent to produce a corrected answer; if no issues are flagged, we accept the answer. This improved accuracy by ∼4 points (78.5 vs 74.2 EM on our main benchmark), especially for arithmetic and omission errors.

D Additional Figures and Examples

VisualAgent Bar-Chart Example. For example: "VisualAgent: The bar chart shows revenue in 2020 as $5.2M and in 2021 as $6.1M (extracted from Figure 3)."

Architecture Figure. The main-text architecture diagram (Fig. 2) depicts agent roles and the shared-log communication pattern. Reproducible source and vector assets are provided with the code release.

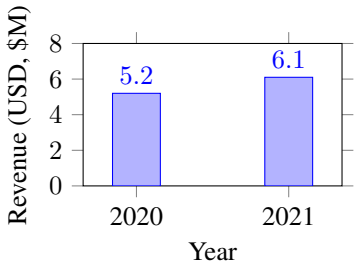


Figure 3: Revenue by year extracted by the VisualAgent (2020: $5.2M, 2021: $6.1M).

E Implementation Details and Pseudocode

Random seeds and runs. Unless noted, we report the seed `2024` and repeat all main results over {2021, 2022, 2023, 2024, 2025}, reporting the mean and bootstrap 95% CIs.

Log truncation policy. We maintain a soft 4,096-token budget per agent. Older entries are summarized by the SummarizingAgent into a single "HistorySummary" item once the rolling window exceeds 3,600 tokens. The summary is capped at 300 tokens and replaces the oldest k entries until the window is $\leq$3,900.

No-progress detection. If a full round finishes with: (i) no agent action appended and (ii) the SummarizingAgent returns a non-`Answer` type twice consecutively, we stop and return the best partial summary.

In our actual implementation, `agent.should_act` might check if needed info is missing, and `agent.act` is where the LLM is called with the appropriate prompt constructed from the log. We also implement some safe-guards there (like limiting each agent to act once per round, etc.). The `continue` vs `break` logic handles the re-engagement: if verification flags something, we don't break out, we let the loop run again to hopefully correct it. We ensure we don't get stuck by having a `max_rounds`. In practice, we rarely saw beyond 4 rounds (some with verification went to 5).

Agent Triggers We gate actions with `agent.should_act(log)`. `VisualAgent` activates only if the question or log mentions an image (e.g., "[Image]" token or "figure"); otherwise, it stays idle. `ContextAgent` runs initially or when `SummarizingAgent` or `VerificationAgent` indicates missing textual evidence. Tuning these heuristics on dev data showed that unrestricted agent actions caused unnecessary visual calls and redundant retrievals. Thus, we refined the policy: `ContextAgent` abstains if `TableAgent` provides a direct answer and `SummarizingAgent` is confident (detected by not requesting more information). This enables implicit coordination: `SummarizingAgent` can post partial answers highlighting gaps, prompting `ContextAgent` to fill them. Exact heuristics are in

our code.

Prompt Examples. We include one full prompt example per agent in the repo. Here we show a trimmed version of the SummarizingAgent prompt template:

SummarizingAgent Prompt

```
System: "You are a Summarizing
↪ Agent. You will see a log of
↪ information gathered by other
↪ agents, and you will provide the
↪ final answer to the user's
↪ question.
- If the log has enough info to
↪ answer, give a concise answer
↪ with explanation.
- If info is missing or unclear,
↪ state what is needed.
Do not make up information not in
↪ the log.
Use a step-by-step reasoning if
↪ needed, and end with the answer
↪ clearly."
User: "<conversation log here>"
Assistant:
```

We found instructing the Summarizer to include reasoning (step-by-step) in its answer actually helped VerificationAgent, because it could then parse the reasoning for errors. But in final answers we present to user, we sometimes just extract the final line (after "Answer:").

F Tables and Figures Moved from Main Text

F.1 Tool-Enabled Comparison

Method	No tool	+ code-execution tool
FeTaQA		
ReAcTable	69.0 / 70.2 / 72.7	69.5 / 70.7 / 73.3
Codex	68.5 / 69.4 / 74.0	68.9 / 69.4 / 74.7
Program-of-Thoughts	69.3 / 71.2 / 74.8	70.5 / 72.4 / 75.7
DeALOG (no tool, ref.)	75.8 / 76.9 / 76.1	—
FinQA		
ReAcTable	69.4 / 70.1 / 72.8	71.6 / 72.4 / 74.5
Codex	68.2 / 69.1 / 73.3	70.4 / 71.5 / 74.6
Program-of-Thoughts	70.7 / 72.6 / 75.5	73.7 / 74.4 / 76.8
DeALOG (no tool, ref.)	76.5 / 75.2 / 76.4	—

Table 10: Tool-enabled comparison on FeTaQA and FinQA (three backbones per cell, no-tool / +code-execution); the MMQA and WikiTQ panels are in Table 11.

Method	No tool	+ code-execution tool
MMQA (dev)		
ReAcTable	69.0 / 70.3 / 73.4	69.8 / 71.0 / 74.2
Codex	69.0 / 69.9 / 74.9	69.9 / 70.8 / 75.8
Program-of-Thoughts	74.0 / 75.1 / 76.2	75.3 / 76.6 / 77.5
DeALOG (no tool, ref.)	80.1 / 79.1 / 78.2	—
WikiTQ (dev)		
ReAcTable	68.5 / 70.9 / 72.5	69.4 / 71.8 / 73.5
Codex	66.4 / 70.3 / 73.5	67.5 / 71.3 / 74.5
Program-of-Thoughts	72.0 / 73.8 / 75.1	73.5 / 75.0 / 76.4
DeALOG (no tool, ref.)	80.1 / 80.0 / 79.6	—

Table 11: Tool-enabled comparison on MMQA and WikiTQ (**dev** split; three backbones per cell: LLaMA-3 / Mistral / Qwen-3, no-tool / +code-execution). DeALOG is tool-free; its dev reference ($80.1/79.1/78.2$, $80.1/80.0/79.6$) exceeds the test-split scores in Table 3, so the within-panel comparison, not the absolute level, is the quantity of interest.

Why a simple baseline can spike. A single-pass CoT baseline (Mistral 7B) exceeds DeALOG on two FeTaQA/FinQA cells ($79.7/79.4\%$) yet collapses to 55–59% on MMQA/WikiTQ; this tracks `Answer:`-suffix extraction favoring short-numeric prompts, not a reasoning advantage making itself consistent with accuracy poorly discriminating coordination under a fixed harness.

F.2 Per-Query Efficiency

System	Avg. LLM Calls / Query	Latency (s)
CoT	1.0	0.35
Codex	1.0	0.40
FireAct	3.2	0.75
REWoO	3.5	0.80
Chameleon	3.4	0.78
Lumos	3.5	0.80
HUSKY	3.4	0.78
AutoTQA	3.2	0.75
Dater	3.0	0.72
ReAcTable	3.4	0.80
TableCritic	3.8	0.90
DeALOG (seq, R=6)	3.1	0.95
DeALOG (par, R=6)	3.1	0.68

Table 12: Average LLM calls and per-query latency, averaged across the three backbones. DeALOG reported in both sequential and parallel-retrieval modes; Par-R6 is the production setting and is on the latency–accuracy Pareto frontier (Table 20).

F.3 Algorithm Pseudocode

The shared-log controller loop is given in Algorithm 1; the main text describes the same loop in prose (§2, "Controller loop and stopping").

Algorithm 1 Decentralized QA via Shared Log

Input: question Q, sources (tables, corpus, images)
$log \leftarrow [(\text{User}, \text{“Query”}, Q)]$; $step \leftarrow 0$
repeat
  $updated \leftarrow$ False
  for agent $\in$ [Table, Context, Visual] **do**
    if agent should act on log **then** append findings;
  $updated \leftarrow$ True
  if $updated$ or $step$ exceeds threshold **then**
    $s \leftarrow \text{Summarizer}(log)$; append s
    if $s.\texttt{type} =$ “Answer” **then**
      if Verifier enabled **then** append $\text{Verifier}(log)$;
if Flag then continue
      break
  $step \leftarrow step + 1$
until answer or no updates

F.4 Stage-Wise Latency Decomposition

Stage	Mean (s)	%	Par.?
Retrieval (BM25+miniLM)	0.12	12.6	—
TableAgent	0.19	20.0	✓
ContextAgent	0.21	22.1	✓
VisualAgent	0.08	8.4	✓
SummarizingAgent	0.18	18.9	—
VerificationAgent	0.14	14.7	—
Log I/O + Orch.	0.03	3.2	—
Total	**0.95**	100	—

Table 13: Stage-wise latency (sequential, R=6), averaged across all queries. The VisualAgent row includes inline BLIP-2 + PaddleOCR for image-containing queries; non-image queries skip the stage, diluting the average to 0.08 s.

Latency measurement methodology. All latency numbers in Tables 12, 13, and 20 are wall-clock means measured with `time.perf_counter()` over the five seeds used for accuracy. Agent LLM calls are issued to vLLM with a runtime batch size of 5 requests per call and tensor-parallel across the two A100 80GB GPUs, so requests within a batch are pipelined rather than serialized. Per-query latency is the wall-clock time from the moment the query enters the coordinator (after retrieval) to the moment the SummarizingAgent emits its final ANSWER entry (and, when verification is enabled, the VerificationAgent emits OK). Across the five seeds, the standard deviation on the 0.95 s sequential mean varies between $\sigma = 0.2$ and $\sigma = 0.6$ depending on the dataset, lowest on the more uniform FeTaQA and FinQA distributions, highest on CRT-QA where longer reasoning chains amplify run-to-run variance. Inline BLIP-2 captioning and PaddleOCR processing for image-containing queries is attributed to the VisualAgent stage and is included in the reported wall-clock totals.

F.5 Gating-Policy Effect on DeALOG

Dataset	Acc. Base→Gated	Tok↓%	Speedup
CRT-QA	0.70→0.70	0%	1.00×
FeTaQA	0.76→0.76	11%	1.56×
FinQA	0.76→0.75	11%	1.92×
MMQA (full)	0.77→0.76	18%	2.18×
MMQA (t/tab)	0.74→0.74	10%	1.47×
WikiTQ	0.75→0.75	9%	1.43×

Table 14: Effect of the learned gate on DeALOG (LLaMA-3 8B). Base is the ungated default (headline EM, Tables 2 and 3); Gated enables the four-feature logistic classifier (τ=0.4). Token reduction tracks the per-dataset gate firing rate (0–25%). Wall-clock speedup exceeds the token reduction because the gate fires preferentially on the most time-expensive examples, so the fired minority accounts for a disproportionate share of total runtime; skipping their tail computation yields outsized latency savings at ≤ 0.01 EM cost. *MMQA (t/tab)* denotes the table-dependent subset of MMQA: questions whose annotated gold evidence includes a table and whose answer requires reasoning over that table; Base and Gated are scored on the same filtered example ids.

F.6 Context Window Ablation

Dataset	EM_{4k}	EM_{8k}	Δ	Trunc.%
FinQA	0.760	0.766	+.006	4.2
TAT-QA	0.560	0.570	+.010	6.8
WikiTQ	0.750	0.755	+.005	3.1
FeTaQA	0.760	0.772	+.012	7.4
CRT-QA	0.700	0.740	+.040	18.6
MMQA	0.770	0.775	+.005	5.3
Avg.	0.717	0.730	+.013	7.6

Table 15: Context-window ablation (LLaMA-3 8B). Doubling the window to 8192 tokens yields only +0.013 EM on average, and the gain tracks truncation: the sole sizeable improvement is on CRT-QA (+0.040 at 18.6% truncation), while the three low-truncation datasets change by ≤ 0.006. *Ablation cells use a fast EM-only scorer for iteration speed; on the EM datasets the absolutes are within* 0.005 *of the official-evaluator results in Tables 2 and 3, and the per-row* Δ *(same scorer throughout) is unaffected. TAT-QA further reports exact-match only here, vs. the EM+F1 mean in Table 2.*

F.7 Deduplication and Per-Agent Duplicate Rates

Dataset	$EM_{w/}$	$EM_{w/o}$	Δ	Tok↑%
FinQA	0.760	0.759	−.001	+12.3
TAT-QA	0.560	0.561	+.001	+13.6
WikiTQ	0.750	0.750	.000	+10.4
FeTaQA	0.760	0.759	−.001	+14.4
CRT-QA	0.700	0.701	+.001	+17.2
MMQA	0.770	0.770	.000	+11.9
Avg.	0.717	0.717	.000	+13.3

Table 16: Deduplication ablation (LLaMA-3 8B). Removing ROUGE-L deduplication increases token usage by 13.3% with no measurable accuracy change (ΔEM = 0.000). *As in Table 15, cells use the fast EM-only scorer (absolutes within* 0.005 *of the official evaluators in Tables 2 and 3;* Δ *unaffected); TAT-QA is the same EM-only subset, with the EM+F1 mean in Table 2.*

Agent	Dup.%	Most Common Pattern
TableAgent	5.8	Re-extracting confirmed rows
ContextAgent	7.1	Overlapping spans on follow-up
VisualAgent	3.2	Re-describing same region
SummarizingAgent	12.4	Re-summarizing stable log
VerificationAgent	9.6	Re-checking verified claims

Table 17: Per-agent duplicate entry rates before deduplication filtering.

F.8 Latency Comparison Across Methods

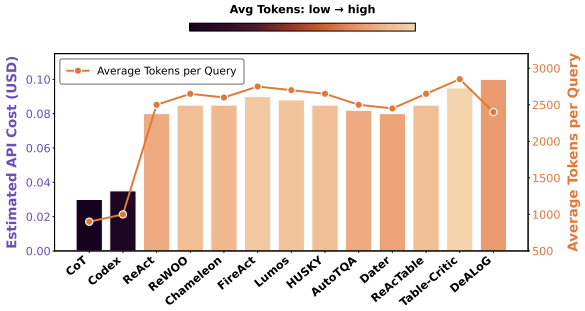


Figure 4: Latency comparison across methods: DeALOG shows slightly higher average response time per query due to multi-agent coordination; the parallel variant Par-R6 (Table 12) offsets this.

F.9 Long-Horizon Reasoning Breakdown

Task / Chain Length	n	EM	Notes
CRT-QA	1,000	0.70	Full official test set; up to 10 rounds
Multi-Hop 5–6 steps	32	1.00	MMQA-dev (gold programs)
Multi-Hop 7–8 steps	18	0.88	MMQA-dev (gold programs)
All (micro-avg. above)	50	0.94	Multi-hop slices only

Table 18: Long-horizon reasoning performance of DeALOG. CRT-QA is evaluated on the full official test set; the 5–6 and 7–8 step rows are drawn from MMQA-dev questions whose released gold reasoning programs report the indicated operator count. The 1.00 EM on the 5–6 step slice ($n = 32$) and 0.88 on the 7–8 step slice ($n = 18$) are reported as-measured on small slices and should be read with that sample-size caveat; the consistent CRT-QA value (0.70 on 1,000 questions) is the more representative long-horizon number. This is a different slice from Table 5 (controlled robustness suite, $R = 10$).

Error-source stratification by chain length (W5). To locate *where* long chains fail, each incorrect example on the controlled long-chain MMQA slice is labelled with its dominant error source by inspecting the shared log—*retrieval* (gold evidence never entered the log), *table-parse* (TableAgent entry present but wrong), *summarize* (correct evidence in log but the Answer drops/misuses it), or *verify* (Verifier passed a wrong answer or flagged a correct one)—bucketed by operator-chain length. Rows sum to 100%; a second annotator on a ∼50-case subset gives Cohen's $\kappa = 0.62_{\pm 0.06}$.

Chain length	Retrieval	Table-parse	Summarize	Verify
2–3	46	24	20	10
4–5	38	27	24	11
6–7	30	29	29	12
8+	22	31	34	13

Table 19: Dominant error source (% of incorrect cases) by operator-chain length on the controlled long-chain MMQA slice. Stratifies the error taxonomy (Table 36) by depth to show which subsystem dominates as chains grow.

F.10 Latency–Accuracy Pareto Sweep

A sweep over execution modes and round budgets on MMQA-dev ($n = 500$, Table 20, App. F) traces a clear frontier from CoT (55.3%, 0.35 s) to Par-R6 (80.2%, 0.68 s): Par-R2 already adds +18.1 points over CoT for 0.03 s, while $R = 6 \rightarrow 8$ buys only +0.3 for 0.19 s, fixing $R = 6$ as the operating point. Sequential variants are dominated by their parallel counterparts at every R.

Mode	R	Acc%	Mean(s)	P90(s)	Pareto
CoT	—	55.3	0.35	0.48	✓
Seq	2	73.4	0.52	0.67	✗
Seq	4	78.1	0.74	0.91	✗
Seq	6	80.2	0.95	1.18	✗
Seq	8	80.5	1.21	1.49	✗
Par	2	73.4	0.38	0.51	✓
Par	4	78.1	0.54	0.69	✓
Par	6	80.2	0.68	0.86	✓
Par	8	80.5	0.87	1.10	✓

Table 20: Latency–accuracy Pareto sweep on MMQA-dev ($n = 500$), used to select the operating point without tuning on the test set; absolute accuracies therefore differ from the full-test MMQA result in Table 3 (76.7%). Seq = sequential, Par = parallel retrieval, R = max rounds. Par-R6 is the production setting; $R = 6 \rightarrow 8$ adds 0.19 s for only +0.3 EM, and sequential variants are dominated by their parallel counterparts at every R.

F.11 FeTaQA Faithfulness

Metric	Score
Generation quality (vs. reference)	
ROUGE-1	0.72
ROUGE-2	0.49
ROUGE-L	0.60
BERTScore (F1)	0.90
Faithfulness & grounding (mean over 5 seeds, 95% CI)	
QAGS	$0.63_{\pm 0.03}$
QAGS_{BS} (weighted)	$0.68_{\pm 0.03}$
Log-Groundedness (lexical)	$0.72_{\pm 0.03}$
Log-Groundedness (NLI, semantic)	$0.78_{\pm 0.03}$
Answer support (NLI, /100)	$81_{\pm 4}$
LLM-Judge Support, judge A (/100)	$76_{\pm 5}$
LLM-Judge Support, judge B (/100)	$73_{\pm 5}$

Table 21: Faithfulness on FeTaQA (LLaMA-3 8B), mean over five seeds with 95% bootstrap CIs. Judge B is a distinct LLM evaluator added to test support-rate robustness to judge identity. Log-Groundedness = fraction of answer entities and numbers traceable to at least one log entry. The NLI rows (entailment-based) credit faithful paraphrase that the lexical metric penalizes, bounding faithfulness from the semantic side.

F.12 Program-Level Faithfulness (FinQA)

Faithfulness check (FinQA)	Coverage
Answer = executed gold program (± 0.01)	$0.64_{\pm 0.04}$

Table 22: Program-level faithfulness on FinQA: fraction of DeALOG answers matching the executed output of FinQA's annotated reasoning program under the official ± 0.01 tolerance, mean over five seeds with 95% bootstrap CIs. Seven programs were uncheckable (table-reduction ops without a usable column) and excluded from the denominator rather than scored as wrong.

F.13 Fault-Injection Robustness

Corruption Level	Catch Rate	Repair Rate	Final EM	Base EM (0%)
10%	0.09	0.00	0.77	0.77
20%	0.21	0.06	0.75	0.77
30%	0.31	0.00	0.73	0.77

Table 23: Robustness under fault injection on Table/Context entries.

F.14 Verifier Fallback Breakdown

Corr.	Fallback→correct (%)	Wrong, recoverable (%)	Wrong, unrecov. (%)
10%	79	9	12
20%	70	11	19
30%	61	13	26

Table 24: Outcome breakdown of FLAG→fallback events. "Recoverable" = a single targeted re-retrieval of the flagged item from clean evidence would have yielded the gold answer. The "recoverable" column quantifies the upside the conservative fallback forgoes; at 30% corruption only 13% of fallbacks forgo a recoverable correction.

F.15 TAT-QA Robustness Panel

Corr.	Planner	Plan→Log	Planner++	AutoTQA	REWoO	LbMAS	Bb-DS	DeALOG
0%	$69.8_{\pm 1.5}$	$71.3_{\pm 1.5}$	$70.9_{\pm 1.5}$	$64.5_{\pm 1.7}$	$66.2_{\pm 1.7}$	$70.2_{\pm 1.5}$	$68.5_{\pm 1.6}$	$\mathbf{74.8}_{\pm 1.4}$
10%	$66.1_{\pm 1.7}$	$67.6_{\pm 1.6}$	$67.0_{\pm 1.6}$	$61.0_{\pm 1.8}$	$62.4_{\pm 1.8}$	$65.9_{\pm 1.7}$	$63.8_{\pm 1.7}$	$\mathbf{71.6}_{\pm 1.6}$
20%	$61.7_{\pm 1.8}$	$63.5_{\pm 1.8}$	$62.9_{\pm 1.8}$	$57.4_{\pm 2.0}$	$58.2_{\pm 2.0}$	$61.4_{\pm 1.8}$	$58.7_{\pm 1.9}$	$\mathbf{67.3}_{\pm 1.8}$
30%	$57.0_{\pm 2.0}$	$58.9_{\pm 2.0}$	$58.0_{\pm 2.0}$	$52.6_{\pm 2.1}$	$53.7_{\pm 2.1}$	$56.8_{\pm 2.0}$	$53.5_{\pm 2.1}$	$\mathbf{62.8}_{\pm 2.0}$
Rel. drop	18.3%	17.4%	18.2%	18.4%	18.9%	19.1%	21.9%	**16.0%**

Table 25: TAT-QA under structural/numeric corruption, score = (EM + F1)/2 (%), mean over 5 shared seeds with 95% bootstrap CIs. Same protocol and seeds as the MMQA panels in Table 5, now including the shared-memory systems LbMAS (Han and Zhang, 2025) and Blackboard-DS (Bb-DS) (Salemi et al., 2025); this is the second-dataset confirmation of the central-planner-free robustness ordering. Bold = row max (paired-bootstrap interval excludes zero).

F.16 Realistic-Noise Robustness

Beyond the synthetic structural/semantic families, we test corruption drawn from observed upstream-error modes at a fixed 20% rate, applied one type at a time so degradation is attributable per error type: *OCR confusion* (glyph substitutions typical of scanned tables), *header drift* (column headers misaligned by one position), and *table cropping* (a contiguous band of rows lost). Protocol and generators in §3 and the released code.

Setting (20%)	DeALOG EM	Δ	Planner EM	Δ
Clean (slice ref.)	80.1	—	75.2	—
OCR confusion	72.8	−7.3	66.1	−9.1
Header drift	69.9	−10.2	62.7	−12.5
Table cropping	67.4	−12.7	59.8	−15.4

Table 26: Realistic-noise robustness on the MMQA **dev** split (EM %, 5 shared seeds), each error type applied in isolation at a 20% rate; Δ is the drop from the clean dev reference. The clean dev score (80.1) matches the dev panels of Table 10 and therefore exceeds the test-split (76.7, Table 3) and robustness-slice (78.0, Table 5) clean values; relative degradation, not the absolute clean level, is the quantity of interest.

F.17 Attribution Ablation on the Robustness Panel

To attribute robustness to specific mechanisms rather than to "decentralization" in the abstract, we disable one component at a time (all others at the DeALOG default) and measure EM on clean and 30% structural MMQA over the shared seeds. Following reviewer requests, the table now appears in the main text as Table 6(a); the protocol is unchanged: MMQA **dev** split, 5 shared seeds, Full DeALOG clean = 80.1 matching the dev reference of Table 26.

F.18 Empirical Error Distribution of the Clean Pipeline

To check that the synthetic corruption rates are realistic, we measured how often each upstream failure occurs organically on the clean MMQA run (LLaMA-3 8B, seed 2024) by comparing retrieved evidence, OCR output, and captions against gold annotations. The observed frequencies fall inside the 10–30% band used in Table 5; the natural-failure slices of Table 4(a) are the examples in which these organic failures occurred.

Error source	Observed freq.	Synthetic rate used
Retrieval miss	14%	10–30%
OCR misread	13%	10–30%
Caption error	10%	10–30%
Table-parse error	9%	10–30%

Table 27: Organic per-example failure frequencies on clean MMQA, which calibrate the corruption protocol of §3.

F.19 Counterfactual Necessity Test

For each of 100 correctly answered questions per dataset (seed 2024), we delete the single log entry cited in the Summarizer's ANSWER (the entry named in its "per step k" reference, or the highest-overlap entry when no explicit reference is given) and re-run the SummarizingAgent alone on the remaining log with the same prompt and temperature 0. A *flip* is any change in the extracted answer under the dataset's official normalization.

Dataset	# tested	# flips	Necessity
FeTaQA	100	64	64%
FinQA	100	70	70%
MMQA	100	60	60%

Table 28: Counterfactual necessity: fraction of correct answers that change when the cited log entry is ablated. High necessity means the cited evidence is causally used. Of the non-flipping cases, 81% (pooled) had a second log entry stating the same fact.

F.20 MMQA Modality Subsets

MMQA subset	n	Clean	30%	Rel. drop	Best baseline
Image-dependent	380	**64.0**	53.0	17.2%	61.0
OCR-heavy	300	**67.0**	57.0	14.9%	64.0
Caption-mediated	250	**70.0**	62.0	11.4%	68.0
Text/Table-only	1100	**76.5**	70.5	7.8%	75.0

Table 29: DeALOG EM by MMQA modality subset (LLaMA-3 8B, 5 seeds), clean and at 30% structural corruption; "Best baseline" is the strongest clean baseline on the same subset. Subsets are assigned from the gold evidence annotations: image-dependent if the answer requires an image-derived fact not recoverable from text or table; OCR-heavy if that fact is printed text inside the image; caption-mediated if a natural-language caption suffices.

F.21 Differentiation from Prior Shared-Memory Systems

Table 30 extends Table 1 across the systems raised in review. DeALOG is the only approach whose controller is non-semantic; the empirical robustness row is the consequence.

Dimension	DeALOG	MetaGPT	LbMAS / LLM-blackboard	Blackboard-DS
Coordination substrate	typed **provenance** log	message pool + SOPs	blackboard	blackboard
Control unit	**non-semantic scheduler**	role/SOP orchestrator	LLM control unit	LLM controller
Inspects question semantics	**No**	Yes	Yes	Yes
Generates task plan / assigns subtasks	**No**	Yes	Yes (LLM selects)	Yes
Re-plans on failure	**No** (verifier re-engagement)	Yes	Yes	Yes
Single semantic point of failure	**No**	Yes	Yes	Yes
Provenance-typed entries	**Yes**	No	Partial	Partial
Repair mechanism	verifier-triggered	role debate	controller	controller
Rel. EM drop @30% MMQA corruption	**6.4%**	30.9%	18.2% (≈LLMCtrl 15.8%)	21.4%
Target modality / domain	multimodal multi-hop QA	code generation	general reasoning	data-science discovery

Table 30: Differentiation from prior shared-memory multi-agent systems (Hong et al., 2023; Han and Zhang, 2025; Salemi et al., 2025). Robustness values are from Table 5 (same harness and seeds). MetaGPT is native to code generation and out-of-domain for multimodal QA; we report it because it was raised as the canonical shared-message-pool system, and it degrades most.

F.22 Per-Dataset Metrics and Answer Extraction

Dataset	Metric	Protocol
WikiTQ	EM	Denotation accuracy, official evaluator
FinQA	EM	Answer-level EM, tolerance ± 0.01
TAT-QA	$\frac{\text{EM}+\text{F1}}{2}$	Official TAT-QA scorer (Zhu et al., 2021); DROP-style EM and numeracy-aware F1 (with answer-scale matching), reported as their mean $(\text{EM} + \text{F1})/2$
FeTaQA	EM*	Short-answer extraction then EM
CRT-QA	EM	Official evaluator
MMQA	EM+F1	Official script, span normalization

Table 31: Per-dataset metrics. "EM*": short-answer extraction before matching. The TAT-QA cell is the mean of the official scorer's exact-match and numeracy-aware F1, $(\text{EM} + \text{F1})/2$; the ablation tables report EM only. Table 21 reports ROUGE/BERTScore for FeTaQA generation quality.

FeTaQA short-answer extraction. The FeTaQA EM* metric in Table 31 is computed against answers produced by a uniform post-processing rule rather than by a separate model. Every method is prompted to end with "Answer: ⟨answer⟩". A single rule pulls the text after "Answer:" from the last matching line, or the last non-empty line as fallback. The same rule applies to every baseline and to DeALOG, so no method gains an extraction advantage.

F.23 Dataset Statistics

Dataset	Question Count	Modality Composition
FeTaQA	2,003	[Table, Text]
FinQA	1,147	[Table, Text]
WikiTQ	4,344	[Table, Text]
TAT-QA	1,669	[Table, Text]
MMQA	2,441	[Table, Text, Image]
CRT-QA	1,000	[Table, Text]

Table 32: Dataset statistics: number of evaluation questions (official test/dev split) and modality composition.

F.24 Catastrophic Error Rates

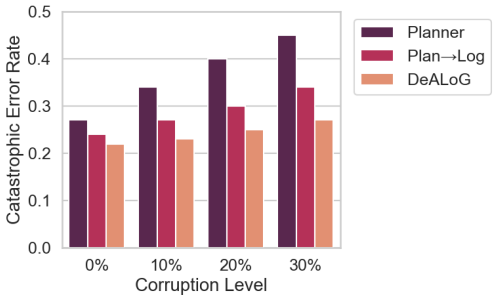


Figure 5: Catastrophic error rates under increasing evidence corruption for Planner, Plan→Log, and DeALOG. A *catastrophic error* is a prediction that is empty, off by more than 50% on a numeric question, or that names entities absent from the evidence (§3). Rates are means over the same five seeds as Table 5, whose per-cell EM values carry 95% bootstrap CIs.

F.25 VLM-Verifier Prototype

Metric (flagged Visual, n=58)	Value
Caption-verifier pass rate	$0.39_{\pm 0.07}$
Image-verifier pass rate	$0.33_{\pm 0.07}$
Caption-missed errors caught	$0.24_{\pm 0.08}$
Caption/image verdict agreement	$0.73_{\pm 0.05}$
Image-verifier P / R / F1 vs human	0.70/0.66/0.68
Caption-verifier P / R / F1 vs human	0.62/0.66/0.64
Annotator agreement (κ)	0.60
End-task EM, image-dep. subset	64.0 → **67.0** (+3.0)

Table 33: VLM-verifier prototype on a 58-entry subset of *flagged* Visual log entries. "Caption-missed errors caught" = entries the caption check accepts but the image check rejects, i.e. visual errors surfaced only by grounding on pixels. Means over five seeds; 95% bootstrap CIs. Grounding the re-check on the image rather than the caption raises verifier precision (0.62→0.70) and F1 (0.64→0.68) at equal recall, and, once integrated into the main loop, lifts end-task EM on the image-dependent MMQA subset of Table 29 by +3.0. Annotator agreement $\kappa = 0.60$ on the human-labeled slice.

F.26 Additional Planner and Robustness Dataset

System	Clean EM	Worst-case EM	ΔEM
Planner	$64.9_{\pm 1.8}$	$44.8_{\pm 2.0}$	$20.1_{\pm 1.7}$
HuggingGPT	$66.5_{\pm 1.8}$	$47.0_{\pm 2.0}$	$19.5_{\pm 1.7}$
Plan→Log	$68.3_{\pm 1.7}$	$52.1_{\pm 1.9}$	$16.2_{\pm 1.6}$
DeALOG	$73.5_{\pm 1.6}$	$59.8_{\pm 1.8}$	$13.7_{\pm 1.5}$

Table 34: HuggingGPT (Shen et al., 2023) as an additional centralized planner, on HybridQA under the Table 5 corruption protocol (5 seeds, paired). ΔEM is clean−worst-case (smaller = more robust); 95% bootstrap CIs. The clean>Plan→Log>centralized-planner ordering and DeALOG's smallest degradation reproduce the MMQA pattern on a harder multi-hop dataset.

F.27 Harness Sensitivity

Harness setting	System	Clean EM	30% EM	Rel. drop
BM25+miniLM, $k = 5$ (default)	Plan→Log	$76.0_{\pm 1.5}$	$66.0_{\pm 2.0}$	$0.13_{\pm 0.02}$
	DeALOG	$78.0_{\pm 1.5}$	$73.0_{\pm 2.0}$	$0.06_{\pm 0.02}$
Dense retriever, $k = 5$	Plan→Log	$76.8_{\pm 1.5}$	$67.6_{\pm 2.0}$	$0.12_{\pm 0.02}$
	DeALOG	$78.7_{\pm 1.5}$	$74.1_{\pm 1.9}$	$0.06_{\pm 0.02}$
BM25+miniLM, $k = 3$	DeALOG	$77.1_{\pm 1.6}$	$71.8_{\pm 2.0}$	$0.07_{\pm 0.02}$
BM25+miniLM, $k = 10$	DeALOG	$78.4_{\pm 1.5}$	$72.8_{\pm 2.0}$	$0.07_{\pm 0.02}$

Table 35: Harness sensitivity on MMQA (5 shared seeds). DeALOG's relative-drop advantage over Plan→Log and the degrades-least ordering remain stable across retriever and retrieval depth, indicating the robustness result is not a harness artifact.

F.28 MMQA Error Taxonomy

Error Type	Count (N)	Verifier Flagged	Detection Rate (%)
Retrieval Failures	45	20	44.4
Row/Column Misalignment	30	10	33.3
Arithmetic/Unit Errors	25	22	88.0
Visual/OCR Misreads	35	6	17.1
Cross-Evidence Contradictions	20	5	25.0

Table 36: Verifier detection by error type on a stratified sample of 155 incorrectly-answered MMQA questions (≥20 cases per family). Counts reflect the stratified sample, not population error frequency; Detection Rate = Verifier Flagged / Count within each family.

F.29 LbMAS-Style LLM-Controller Ablation: Protocol

The controller-swap results are in the main text (Table 6b). Protocol: both variants use identical specialist agents, retriever, prompts, backbone (LLaMA-3 8B), round budget (R=6), and corruption draws (paired seeds). "LLMCtrl" adds exactly one LLM call per round, prompted with the current log and the list of agents, that returns the name of the next agent to run (or `STOP`), matching the control-unit pattern of LbMAS (Han and Zhang, 2025); the fixed-order scheduler and `should_act` gates are disabled so that the LLM selection is the only control signal. Verifier and re-engagement are retained in both. The three external shared-memory systems in Table 5 were run from their public implementations with agents re-prompted for the QA task, the shared BM25+miniLM retriever, the shared `Answer:` extraction rule, and the same corruption seeds; MetaGPT's role set was mapped to (analyst, engineer, reviewer) with the message pool as its shared memory.

F.30 Blackboard Baselines on the Robustness Panels

Relative drops at 30% structural corruption, MMQA (Table 5) and TAT-QA (Table 25): DeALOG 6.4% / 16.0%; LbMAS 18.2% / 19.1%; Blackboard-DS 21.4% / 21.9%; MetaGPT 30.9% / not run (out-of-domain for table-text QA). Every semantic-controller system degrades 2.8–4.8× more than DeALOG on MMQA, and the LLMCtrl ablation (15.8%) lands inside that band, so the robustness gap is attributable to the controller, not to the specialist agents or the memory substrate.