HEDGE: Gated Hypergraph Memory for Scalable Multi-Agent Debate

Amirhossein Abaskohi[1*], **Mahdi Mostajabdaveh**[2], **Zirui Zhou**[2]

[1]University of British Columbia, [2]Huawei Technologies Canada

arXiv:2610.38902v1 [cs.CL] 28 Oct 2026

Abstract

Multi-agent debate systems require agents to retain and revisit arguments across multiple rounds, yet limited context windows and unstructured memories make relevant information increasingly difficult to recover. Existing flat and pairwise-graph memories also struggle to capture higher-order relations, including multi-agent agreement, conflicting evidence, and evolving consensus. We introduce **HEDGE**, a selective memory framework that represents debate trajectories as semantic hypergraphs. Hyperedges connect related claims across agents, while intra-hyperedge relational graphs preserve fine-grained argumentative structure. A learned *memory selection module* (MSM), trained through supervised learning and reinforcement learning, routes each memory candidate to STORE, BACKUP, or OMIT, retaining useful information while limiting redundancy. Across seven diverse benchmarks, HEDGE consistently outperforms strong single-agent, multi-agent, and memory baselines, improving accuracy by 5.0 points on average over the strongest multi-agent system with GPT-4o-mini. The gains reach 8.2 points on conflict-heavy tasks, while HEDGE reduces memory usage by 48% and the number of debate rounds by 34%.[1]

1 Introduction

Multi-agent debate (MAD) frameworks, in which multiple LLM-based agents propose, critique, and refine arguments over several iterative rounds, have emerged as an effective inference-time strategy for complex reasoning (Du et al., 2024; Liang et al., 2024). By fostering divergent thinking, MAD mitigates the degeneration-of-thought problem, where a single model becomes trapped in a flawed reasoning path (Liang et al., 2024), and produces more reliable consensus than single-agent evaluation (Chan et al., 2024; Chen et al., 2024).

However, as debates grow longer, their expanding histories make earlier evidence, rejected hypotheses, and cross-round dependencies increasingly difficult to recover (Liu et al., 2024a). Important arguments may become buried in the transcript, causing agents to repeat previously rejected claims, overlook unresolved conflicts, or reconsider evidence without its original context. Retaining the full interaction history is equally problematic, as irrelevant and redundant content increases token cost and can degrade reasoning quality (Jiang et al., 2026; Yu et al., 2026). Recent methods address parts of this challenge: MAD-M^2 masks erroneous memories between rounds (Tian et al., 2026), while EquiMem calibrates updates to shared memory (Meng et al., 2026). However, neither jointly organizes debate trajectories into structured argumentative memory, learns the evolving utility of individual memory items, and supports continuous access throughout the debate.

Simply pruning the debate history using heuristic rules is insufficient because an argument's utility changes as the interaction evolves (Yan et al., 2026). Effective debate memory therefore requires a *learned* mechanism that determines what to retain, back up, or discard. It also requires an appropriate structure: flat text or pairwise graphs (Zhang et al., 2025a) cannot naturally represent higher-order relations such as multi-agent agreement, conflicting evidence chains, and evolving consensus. Finally, memory must be integrated into the intra-debate reasoning loop, allowing agents to write, retrieve, and revise information across rounds. Thus, scalable MAD requires three complementary capabilities: structured trajectory representation, learned utility-aware selection, and dynamic intra-debate memory access.

To address these challenges, we propose **HEDGE** (**H**ypergraph-**E**nhanced **D**ebate with **G**ated **E**vidence memory), a framework that maintains a structured, shared memory throughout an on-

*Corresponding author: aabaskoh@cs.ubc.ca.

[1]The code and data will be released upon acceptance.

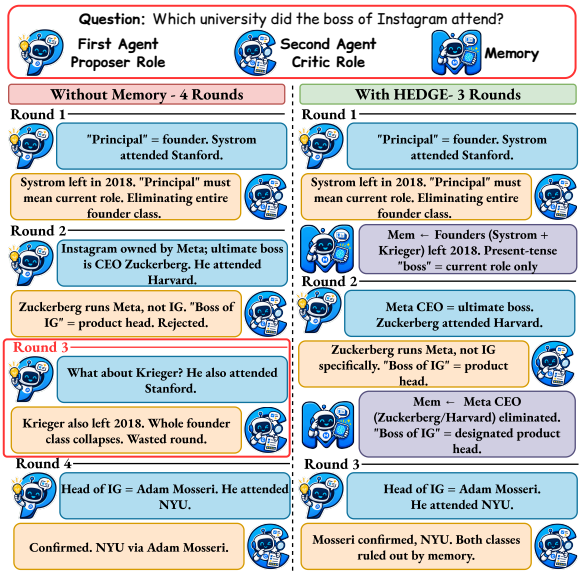


Figure 1: Trajectory-level memory in HEDGE during multi-agent debate. Without memory (***left***), agents revisit a rejected founder hypothesis; with HEDGE (***right***), the stored rationale avoids redundant reasoning and reaches the correct answer in 3 instead of 4 rounds.

going debate. HEDGE represents the debate trajectory as a semantic hypergraph, where hyperedges capture higher-order dependencies among claims and agents beyond pairwise relations. Within each hyperedge, a relational graph preserves fine-grained argumentative structure, including support, refutation, and evidential links. A learned *memory selection module* (MSM), pretrained with supervised learning and refined through reinforcement learning using multi-agent feedback, routes each candidate memory item to STORE, BACKUP, or OMIT. Agents retrieve and update the shared memory across rounds, preserving useful dependencies while reducing redundancy and cost. Figure 1 shows how HEDGE avoids repeated reasoning and improves efficiency.

Empirically, across seven benchmarks spanning mathematical reasoning, knowledge conflicts, and long-context QA, HEDGE consistently outperforms strong baselines. Compared with the best multi-agent system, HEDGE improves accuracy by 5.0 points on average, with gains of up to 8.2 points on conflict-heavy tasks such as ConflictQA (Xie et al., 2024). These gains increase to 11.5 and 16.2 points over the strongest advanced single-agent baseline and the vanilla single-agent using the same backbone, respectively. HEDGE also reduces memory usage by 48% and debate rounds by 34% relative to the strongest multi-agent baseline.

Overall, our contributions are: **(i)** a shared hypergraph-based memory that captures higher-order semantic dependencies and fine-grained argumentative relations across agents; **(ii)** a learned memory selection module, trained with supervised learning and refined through reinforcement learning from debate outcomes, that selectively manages information based on its evolving utility; and **(iii)** an intra-debate memory framework that enables agents to retrieve and update relevant information across rounds.

2 Related Work

Multi-agent debate and debate memory. Multi-agent debate (MAD) improves reasoning by allowing agents to iteratively propose, critique, and refine their answers (Du et al., 2024; Liang et al., 2024; Chan et al., 2024; Chen et al., 2024; Liu et al., 2025). Most MAD frameworks, however, preserve earlier interactions only as an unstructured transcript, making claims, revisions, and cross-round dependencies difficult to track. MeMAD retrieves summaries of completed debates to support new tasks (Ling et al., 2025), while G-Memory models cross-trial collaboration through pairwise graph structures (Zhang et al., 2025a). Recent methods more directly address intra-debate memory: MAD-M^2 masks potentially erroneous information inherited from previous rounds (Tian et al., 2026), and EquiMem calibrates updates to a shared memory based on agent reliability (Meng et al., 2026). Unlike these approaches, HEDGE jointly structures the ongoing debate as a shared semantic hypergraph, preserves higher-order dependencies among multiple agents and claims, and exposes this memory for retrieval and revision throughout the debate.

Memory representation and selection. Existing language-agent memories include flat stores (Zhong et al., 2024; Packer et al., 2024), retrieval-based architectures (Qian et al., 2025b; Wang and Han, 2025), and learned memory-management methods such as Memory-R1, MemAgent, and CoMAM (Yan et al., 2026; Yu et al., 2026; Mao et al., 2026). Flat memories provide limited relational context, while conventional graph memories generally represent only pairwise connections. HyperGraphRAG (Luo et al., 2025) and PRoH (Zai et al., 2026) capture N-ary relations, but focus on retrieving external knowledge rather than maintaining an evolving argumentative state. Moreover,

existing learned memory managers primarily target single-agent long-context reasoning or general memory pipelines. In contrast, HEDGE combines a semantic hypergraph with fine-grained relational graphs and a learned selection module trained from debate outcomes, enabling utility-aware management of higher-order information within an ongoing multi-agent debate.

3 HEDGE

As shown in Figure 2, HEDGE maintains a shared two-level memory for the ongoing debate, with the memory selection module (MSM) controlling how newly generated information is written. At each step t, MSM assigns each candidate node v_t to STORE, BACKUP, or OMIT based on the current debate and memory contents. Stored nodes are organized at the top level into a semantic hypergraph that groups related debate elements, while a directed relational graph within each hyperedge preserves fine-grained support, refutation, evidence, and dependency links. This compact structure evolves throughout the debate and can be retrieved and updated by all agents across rounds.

At debate step t, the memory consists of a hypergraph $\mathcal{H}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$ and directed relational graphs $\{\mathcal{G}_k^{(t)}\}_{k=1}^{|\mathcal{E}^{(t)}|}$. Each node $v \in \mathcal{V}^{(t)}$ represents an atomic debate element, such as a claim, counterclaim, evidence statement, or intermediate conclusion. Each hyperedge $e_k^{(t)} \in \mathcal{E}^{(t)}$ forms a cluster of semantically related nodes. For each hyperedge, we define $\mathcal{G}_k^{(t)} = (\mathcal{N}_k^{(t)}, \mathcal{R}_k^{(t)})$, where $\mathcal{N}_k^{(t)} = \{v \in \mathcal{V}^{(t)} \mid v \text{ belongs to } e_k^{(t)}\}$ and $\mathcal{R}_k^{(t)}$ captures relations such as support and refutation. Nodes may belong to multiple hyperedges, allowing overlapping topics and shared conclusions. Because the memory is updated online (Zhang et al., 2025a; Zhou et al., 2026), its nodes, hyperedges, relations, and corresponding cardinalities refer to a particular step of the evolving debate. Unlike static retrieval memories (Luo et al., 2025; Wang and Han, 2025), it is directly accessed and revised throughout the current interaction.

3.1 Memory Construction and Maintenance

For each node v with text x_v, we compute an ℓ_2-normalized embedding and a sparse semantic signature:

$$\mathbf{h}_v = f_{\text{emb}}(x_v), \qquad \sigma_v = g_{\text{sig}}(x_v), \tag{1}$$

where σ_v contains salient lexical features such as entities, noun phrases, numerical values, and content lemmas. At debate step t, each hyperedge $e_k^{(t)}$ has member set $\mathcal{N}_k^{(t)}$. We maintain the unnormalized sum of its member embeddings and compute the mean embedding as:

$$\mathbf{s}_k^{(t)} = \sum_{u \in \mathcal{N}_k^{(t)}} \mathbf{h}_u, \qquad \bar{\mathbf{h}}_k^{(t)} = \frac{\mathbf{s}_k^{(t)}}{|\mathcal{N}_k^{(t)}|}. \tag{2}$$

The normalized centroid and union signature are then:

$$\mathbf{c}_k^{(t)} = \frac{\mathbf{s}_k^{(t)}}{\|\mathbf{s}_k^{(t)}\|_2}, \qquad \sigma_k^{(t)} = \bigcup_{u \in \mathcal{N}_k^{(t)}} \sigma_u. \tag{3}$$

We cache the unnormalized embedding sums to support incremental centroid updates and hyperedge merging without recomputing embeddings for unchanged nodes.

For an incoming node v, we compare its signature and embedding with each existing hyperedge:

$$J_{vk}^{(t)} = \frac{|\sigma_v \cap \sigma_k^{(t)}|}{|\sigma_v \cup \sigma_k^{(t)}|}, \qquad S_{vk}^{(t)} = \mathbf{h}_v^\top \mathbf{c}_k^{(t)}. \tag{4}$$

Compatible hyperedges are identified using both semantic embeddings and salient entities (e.g., names and numbers):

$$\mathcal{M}_t(v) = \left\{ e_k^{(t)} \in \mathcal{E}^{(t)} \;\middle|\; \begin{array}{l} J_{vk}^{(t)} \geq \tau_{\text{sig}}, \\ S_{vk}^{(t)} \geq \tau_{\text{emb}} \end{array} \right\}. \tag{5}$$

If $\mathcal{M}_t(v) = \varnothing$, we create a new hyperedge; otherwise, v joins every compatible hyperedge. This multi-membership supports overlapping topics and shared conclusions. For each affected hyperedge, the cached embedding sum is updated to reflect its new membership, and the centroid is obtained by normalization. Signatures are likewise updated incrementally.

For each hyperedge, we maintain a directed, typed relational graph $\mathcal{G}_k^{(t)}$ that captures fine-grained connections among its nodes using five relation types: SUPPORTS, REFUTES, EVIDENCE-FOR, DEPENDS-ON, and LEADS-TO, where (v_i, r, v_j) denotes relation r from v_i to v_j. When a node joins a hyperedge, an LLM-based module using `GPT-4o-mini` (OpenAI, 2024) compares it with existing members and predicts one of these relations or no relation using the prompt in Prompt O.1.

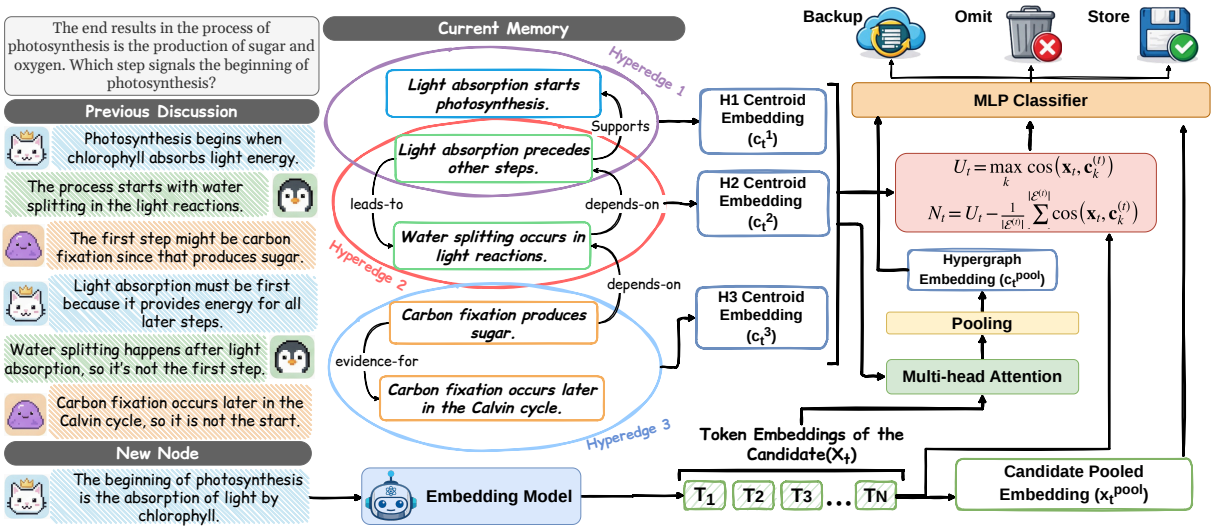


Figure 2: **Overview of HEDGE's memory update pipeline.** The current debate is organized as a semantic hypergraph, while the memory selection module uses candidate features and cross-attention over hyperedge centroids to route new information to STORE, BACKUP, or OMIT.

This local update avoids global recomputation while preserving support, refutation, and dependency chains.

Online Maintenance. We update the memory at the end of each debate round, first reconsidering BACKUP nodes for promotion to STORE. Promoted and newly stored nodes are assigned to compatible hyperedges using Eq. (5), or initialize new hyperedges when no match exists. We then merge redundant hyperedges, prune those with persistently low utility, and reassign their useful nodes when possible. To limit overhead, node embeddings and signatures are cached, unchanged relations are preserved, and only hyperedges whose membership changes are recomputed. Appendix A provides the complete procedure.

3.2 Two-Level Memory Retrieval

Retrieval follows the two-level memory structure. At debate step t, the agent forms a query q with embedding $\mathbf{h}_q$ and signature σ_q. It first selects relevant hyperedges and then retrieves fine-grained argument structures within them.

At the hyperedge level, we select the top-K hyperedges whose signatures are compatible with the query:

$$\mathcal{E}_q^{(t)} = \mathrm{TopK}_{\substack{e_k^{(t)} \in \mathcal{E}^{(t)} \\ J(\sigma_q, \sigma_k^{(t)}) \geq \tau_{\mathrm{sig}}}} \cos\left(\mathbf{h}_q, \mathbf{c}_k^{(t)}\right). \quad (6)$$

Within each selected hyperedge, we retrieve the top-m member nodes by embedding similarity:

$$\mathcal{V}_q^{(k,t)} = \mathrm{TopM}_{v \in \mathcal{N}_k^{(t)}} \cos(\mathbf{h}_q, \mathbf{h}_v). \quad (7)$$

For each $e_k^{(t)} \in \mathcal{E}_q^{(t)}$, we return the induced subgraph $\mathcal{G}_k^{(t)}[\mathcal{V}_q^{(k,t)}]$, containing the retrieved nodes and all relations whose endpoints are both among them. The resulting subgraphs are combined into a bounded memory readout containing relevant claims, evidence, refutations, and dependencies.

3.3 Memory Selection Module

Given a candidate node v_t, extracted as an atomic fact from an agent response using `GPT-4o-mini` (OpenAI, 2024) and Prompt O.2, MSM predicts $a_t \in \{\text{STORE}, \text{BACKUP}, \text{OMIT}\}$ based on the candidate and current memory. Let $\mathbf{X}_t \in \mathbb{R}^{n_t \times d_e}$ denote its token representations and $\mathbf{x}_t = \mathrm{Pool}(\mathbf{X}_t)$ its pooled representation.

For each hyperedge $e_k^{(t)}$, we first compute the candidate-to-hyperedge alignment:

$$u_{t,k} = \cos\left(\mathbf{x}_t, \mathbf{c}_k^{(t)}\right). \quad (8)$$

We then construct the alignment and specificity features:

$$\boldsymbol{\phi}_t^{\mathrm{UN}} = [U_t, N_t], \quad (9)$$

where

$$U_t = \max_k u_{t,k},$$
$$N_t = U_t - \frac{1}{|\mathcal{E}^{(t)}|} \sum_{k=1}^{|\mathcal{E}^{(t)}|} u_{t,k}. \quad (10)$$

Here, $u_{t,k}$ measures the alignment of candidate v_t with hyperedge $e_k^{(t)}$, U_t measures its strongest hyperedge alignment, and N_t measures the specificity of that alignment relative to other hyperedges. The

hyperedge-specific scores $u_{t,k}$ are also used for utility-based pruning, as detailed in Appendix A.

To incorporate memory context, we stack the current hyperedge centroids into $\mathbf{C}_t \in \mathbb{R}^{K_t \times d_e}$, where $K_t = |\mathcal{E}^{(t)}|$. We project the candidate and centroid representations into a shared hidden dimension d_h:

$$\widetilde{\mathbf{X}}_t = \mathbf{X}_t \mathbf{W}_x, \qquad \widetilde{\mathbf{C}}_t = \mathbf{C}_t \mathbf{W}_c. \tag{11}$$

Candidate-to-memory cross-attention is then computed as:

$$\mathbf{Z}_t = \mathrm{MultiHeadAttn}\left(\widetilde{\mathbf{X}}_t, \widetilde{\mathbf{C}}_t, \widetilde{\mathbf{C}}_t\right). \tag{12}$$

The projected candidate and context representations are obtained by pooling:

$$\mathbf{x}_t^{\mathrm{pool}} = \mathrm{Pool}(\widetilde{\mathbf{X}}_t), \qquad \mathbf{c}_t^{\mathrm{pool}} = \mathrm{Pool}(\mathbf{Z}_t). \tag{13}$$

The routing representation combines utility features, candidate semantics, and hypergraph context:

$$\boldsymbol{\phi}_t = \left[\boldsymbol{\phi}_t^{\mathrm{UN}} \,\middle\|\, \mathbf{x}_t^{\mathrm{pool}} \,\middle\|\, \mathbf{c}_t^{\mathrm{pool}}\right]. \tag{14}$$

The action distribution is:

$$\pi_\theta(a_t \mid \boldsymbol{\phi}_t) = \mathrm{Softmax}\left(\mathbf{W}_\pi \,\mathrm{MLP}(\boldsymbol{\phi}_t) + \mathbf{b}_\pi\right). \tag{15}$$

Here, MLP comprises two hidden linear layers, while $\mathbf{W}_\pi$ and $\mathbf{b}_\pi$ parameterize the third, output layer. Further architectural details are provided in Appendix B.

3.4 Training the Memory Selection Module

MSM is a lightweight three-layer MLP with normalization and dropout that predicts $\{\textsc{store}, \textsc{backup}, \textsc{omit}\}$ from the candidate representation and current hypergraph context. Architectural details are provided in Appendix B.

Data and setup. We construct training data from simulated multi-agent debates on the HotpotQA FullWiki training split (Yang et al., 2018). For each sample, we provide the HotpotQA question and context to GPT-4o (OpenAI, 2024), which generates a task-specific roster of 3–6 agents with complementary roles, goals, model sizes, and turn order using Prompt O.3. We then simulate the full debate and log all agent outputs, retrieved evidence, memory references, routing decisions, and final answers. For Stage I, these trajectories are analyzed post hoc to determine whether each candidate memory item is later reused, supports conflict resolution, or contributes to the final answer, yielding 6.5K pseudo-labeled routing examples. For Stage II, we use an additional 1K HotpotQA training questions to simulate online debates and refine the policy through interaction. Appendix C provides full data-generation details.

Stage I: Supervised pretraining. We derive labels post hoc based on whether an item is reused, resolves a conflict, contributes to the final output, or is redundant. Frequently useful and non-redundant items are labeled STORE, potentially useful items BACKUP, and unused items OMIT. MSM is trained with categorical cross-entropy and a class-balancing loss that prevents collapse toward the dominant routing action. This stage provides a general initialization for utility-aware routing based on relevance, specificity, and redundancy.

Stage II: Reinforcement learning. We further optimize MSM with PPO (Schulman et al., 2017) in an online multi-agent debate environment. PPO is suitable because routing decisions have delayed trajectory-level effects: a stored item may become useful several rounds later, while excessive retention gradually increases memory cost. Its value estimates support delayed credit assignment, and clipped updates preserve stability around the supervised initialization. PPO also handles MSM's discrete actions and directly optimizes non-differentiable objectives, including task accuracy, future memory use, and memory size. The reward combines task success, routing utility based on later item usage, and a penalty for memory growth. Training uses the clipped PPO objective with value loss and entropy regularization to support long-horizon adaptation without collapsing to a single action.

Full details on pseudo-label construction, reward design, and optimization are provided in Appendix D.

4 Experiments and Results

4.1 Experimental Settings

Baselines. We compare against vanilla single-agent models (GPT-5.4 (OpenAI, 2026), GPT-4o, GPT-4o-mini (OpenAI, 2024), and Qwen3.5-27B (Qwen, 2026)), self-improvement methods (Self-Refine (Madaan et al., 2023), Self-Consistency (Wang et al., 2023b), and Reflexion (Shinn et al., 2023)), and multi-agent baselines (MAD (Liang et al., 2024) and MoA (Wang et al., 2025b)). Memory-aware baselines include

Method	MATH500	GPQA	FaithEval	ConflictQA	NovelQA	HotpotQA	MMLU-Pro
Single-Agent (Vanilla)							
GPT-5.4	$57.1_{\pm 0.2}$	$56.2_{\pm 0.4}$	$72.7_{\pm 0.3}$	$48.9_{\pm 0.5}$	$62.4_{\pm 0.4}$	$45.3_{\pm 0.3}$	$85.2_{\pm 0.2}$
GPT-4o	$53.2_{\pm 0.3}$	$44.2_{\pm 0.5}$	$66.4_{\pm 0.4}$	$41.5_{\pm 0.6}$	$58.1_{\pm 0.3}$	$38.9_{\pm 0.4}$	$77.6_{\pm 0.3}$
GPT-4o-mini	$49.8_{\pm 0.4}$	$33.8_{\pm 0.6}$	$61.4_{\pm 0.5}$	$29.2_{\pm 0.7}$	$53.3_{\pm 0.4}$	$29.4_{\pm 0.5}$	$66.8_{\pm 0.4}$
Qwen3.5-27B	$51.4_{\pm 0.3}$	$48.9_{\pm 0.4}$	$69.2_{\pm 0.3}$	$36.8_{\pm 0.6}$	$57.6_{\pm 0.5}$	$39.8_{\pm 0.4}$	$83.4_{\pm 0.2}$
Single-Agent (Advanced) (GPT-4o-mini)							
Self-Refine	$53.7_{\pm 0.5}$	$37.6_{\pm 0.6}$	$64.9_{\pm 0.4}$	$33.4_{\pm 0.7}$	$54.1_{\pm 0.5}$	$30.7_{\pm 0.6}$	$71.2_{\pm 0.3}$
Self-Consistency	$54.8_{\pm 0.4}$	$36.5_{\pm 0.7}$	$65.7_{\pm 0.5}$	$34.1_{\pm 0.6}$	$53.8_{\pm 0.6}$	$31.2_{\pm 0.4}$	$72.6_{\pm 0.4}$
Reflexion	$55.6_{\pm 0.5}$	$38.2_{\pm 0.5}$	$66.5_{\pm 0.6}$	$35.3_{\pm 0.8}$	$54.8_{\pm 0.4}$	$31.9_{\pm 0.5}$	$73.8_{\pm 0.3}$
Multi-Agent (GPT-4o-mini)							
MAD	$55.4_{\pm 0.6}$	$38.8_{\pm 0.5}$	$67.4_{\pm 0.6}$	$36.7_{\pm 0.7}$	$55.5_{\pm 0.5}$	$32.4_{\pm 0.4}$	$74.6_{\pm 0.4}$
MAD-M^2	$55.9_{\pm 0.4}$	$39.4_{\pm 0.6}$	$67.8_{\pm 0.5}$	$37.6_{\pm 0.8}$	$56.5_{\pm 0.6}$	$32.8_{\pm 0.5}$	$75.4_{\pm 0.3}$
MoA	$56.2_{\pm 0.5}$	$39.7_{\pm 0.4}$	$68.3_{\pm 0.6}$	$38.1_{\pm 0.7}$	$57.2_{\pm 0.5}$	$33.1_{\pm 0.4}$	$75.8_{\pm 0.3}$
MeMAD	$57.8_{\pm 0.6}$	$41.5_{\pm 0.5}$	$70.2_{\pm 0.4}$	$40.6_{\pm 0.9}$	$58.7_{\pm 0.6}$	$34.9_{\pm 0.5}$	$77.1_{\pm 0.4}$
G-Memory	$58.6_{\pm 0.5}$	$42.8_{\pm 0.6}$	$73.6_{\pm 0.5}$	$46.3_{\pm 0.8}$	$58.9_{\pm 0.7}$	$36.1_{\pm 0.4}$	$78.4_{\pm 0.3}$
G-Memory + EquiMem	$59.4_{\pm 0.4}$	$43.7_{\pm 0.5}$	$74.9_{\pm 0.4}$	$47.8_{\pm 0.7}$	$59.8_{\pm 0.5}$	$37.0_{\pm 0.6}$	$79.0_{\pm 0.4}$
HEDGE	$\mathbf{62.2}_{\pm 0.2}$	$\mathbf{47.7}_{\pm 0.3}$	$\mathbf{79.8}_{\pm 0.3}$	$\mathbf{56.0}_{\pm 0.4}$	$\mathbf{66.0}_{\pm 0.2}$	$\mathbf{42.4}_{\pm 0.2}$	$\mathbf{82.7}_{\pm 0.2}$
Single-Agent (Advanced) (Qwen3.5-27B)							
Self-Refine	$54.2_{\pm 0.4}$	$51.3_{\pm 0.5}$	$72.1_{\pm 0.4}$	$39.6_{\pm 0.6}$	$59.8_{\pm 0.5}$	$42.1_{\pm 0.4}$	$86.0_{\pm 0.2}$
Self-Consistency	$55.1_{\pm 0.3}$	$52.0_{\pm 0.6}$	$72.8_{\pm 0.5}$	$40.3_{\pm 0.7}$	$60.2_{\pm 0.4}$	$42.6_{\pm 0.5}$	$86.9_{\pm 0.3}$
Reflexion	$56.0_{\pm 0.4}$	$53.1_{\pm 0.5}$	$73.6_{\pm 0.4}$	$41.5_{\pm 0.6}$	$61.0_{\pm 0.5}$	$43.4_{\pm 0.4}$	$88.1_{\pm 0.2}$
Multi-Agent (Qwen3.5-27B)							
MAD	$57.3_{\pm 0.4}$	$52.4_{\pm 0.5}$	$73.0_{\pm 0.4}$	$42.1_{\pm 0.6}$	$59.1_{\pm 0.5}$	$42.0_{\pm 0.4}$	$85.8_{\pm 0.3}$
MAD-M^2	$57.7_{\pm 0.5}$	$53.2_{\pm 0.4}$	$73.8_{\pm 0.5}$	$43.1_{\pm 0.7}$	$60.0_{\pm 0.4}$	$42.5_{\pm 0.5}$	$86.3_{\pm 0.2}$
MoA	$58.1_{\pm 0.4}$	$53.6_{\pm 0.5}$	$74.2_{\pm 0.3}$	$43.7_{\pm 0.6}$	$60.4_{\pm 0.5}$	$42.8_{\pm 0.4}$	$86.6_{\pm 0.3}$
MeMAD	$59.3_{\pm 0.5}$	$56.8_{\pm 0.4}$	$76.1_{\pm 0.5}$	$47.0_{\pm 0.7}$	$61.7_{\pm 0.4}$	$44.2_{\pm 0.5}$	$87.4_{\pm 0.2}$
G-Memory	$60.2_{\pm 0.4}$	$57.6_{\pm 0.5}$	$78.3_{\pm 0.4}$	$50.1_{\pm 0.6}$	$62.0_{\pm 0.5}$	$44.6_{\pm 0.4}$	$88.2_{\pm 0.3}$
G-Memory + EquiMem	$61.0_{\pm 0.5}$	$58.7_{\pm 0.4}$	$79.5_{\pm 0.3}$	$51.8_{\pm 0.7}$	$63.1_{\pm 0.4}$	$45.4_{\pm 0.5}$	$88.7_{\pm 0.2}$
HEDGE	$\mathbf{64.3}_{\pm 0.2}$	$\mathbf{62.2}_{\pm 0.3}$	$\mathbf{83.6}_{\pm 0.3}$	$\mathbf{57.9}_{\pm 0.3}$	$\mathbf{67.6}_{\pm 0.4}$	$\mathbf{49.2}_{\pm 0.1}$	$\mathbf{90.3}_{\pm 0.2}$

Table 1: Accuracy (%) across seven benchmarks. Each value reports the mean over **three runs**, with red subscripts indicating the corresponding error. All multi-agent methods use a three-agent topology. HEDGE performs best under both backbone models, with particularly strong gains on FaithEval, ConflictQA, and NovelQA.

MeMAD (Ling et al., 2025), G-Memory (Zhang et al., 2025a), G-Memory + EquiMem (Meng et al., 2026), and the perplexity-based variant of MAD-M^2 (Tian et al., 2026). To test portability, we integrate HEDGE into AutoGen (Wu et al., 2024), G-Designer (Zhang et al., 2025b), DyLAN (Liu et al., 2024b), and MacNet (Qian et al., 2025a), and compare against Voyager (Wang et al., 2023a), Memory Bank (Zhong et al., 2024), Generative Agents (Park et al., 2023), and native memory variants. See Appendix E.1 for baseline details and Appendix F for descriptions of the multi-agent frameworks.

Benchmarks. We evaluate on GPQA (Rein et al., 2024) and MATH500 (Hendrycks et al., 2021) for mathematical and scientific reasoning; FaithEval (Ming et al., 2025) and ConflictQA (Xie et al., 2024) for faithfulness under conflicting evidence; NovelQA (Wang et al., 2025a) and HotpotQA fullwiki (Yang et al., 2018) for long-context and multi-hop reasoning; and MMLU-Pro (Wang et al., 2024) for broad knowledge reasoning. MSM is trained only on the HotpotQA fullwiki training split and evaluated on a disjoint HotpotQA set and six external benchmarks, testing both in-domain robustness and cross-benchmark transfer. We report mean accuracy over three runs. For more information on the benchmarks and baselines refer to Appendix E.2. The prompts used for each benchmark are listed in Appendix O.2.

Experimental setup. We use a temperature of 0.7 and Qwen3-Embedding-4B (Zhang et al., 2025c) for normalized node, query, and hyperedge embeddings. Stage I uses AdamW with learning rate 10^{-4}, batch size 64, weight decay 0.01, and dropout 0.1; Stage II uses learning rate 10^{-5}, $\gamma = 0.99$, GAE parameter 0.95, PPO clipping threshold 0.2, value coefficient 0.5, entropy coefficient 0.01, and four epochs per batch. We set $\tau_{\text{sig}} = 0.30$ and $\tau_{\text{emb}} = 0.70$ for node assignment, $\tau^{\text{sig}}_{\text{merge}} = 0.50$ and $\tau^{\text{emb}}_{\text{merge}} = 0.85$ for merging, and prune hyperedges with utility below 0.10 for three rounds. Backup nodes are retained for at most two rounds. During retrieval, we set $K = 3$, returning the top three hyperedges with up to top eight nodes

Memory	MATH500	GPQA	FaithEval	ConflictQA	NovelQA	HotpotQA	MMLU-Pro
AutoGen 2024							
No-memory	$55.5_{\pm 0.5}$	$38.9_{\pm 0.6}$	$68.0_{\pm 0.4}$	$36.5_{\pm 0.7}$	$56.3_{\pm 0.5}$	$32.7_{\pm 0.4}$	$74.7_{\pm 0.3}$
Voyager	$57.3_{\pm 0.4}$	$40.7_{\pm 0.5}$	$69.8_{\pm 0.6}$	$38.6_{\pm 0.8}$	$58.1_{\pm 0.4}$	$34.2_{\pm 0.5}$	$76.3_{\pm 0.3}$
Memory Bank	$54.9_{\pm 0.6}$	$38.2_{\pm 0.7}$	$67.3_{\pm 0.5}$	$37.2_{\pm 0.6}$	$57.1_{\pm 0.6}$	$33.5_{\pm 0.4}$	$75.3_{\pm 0.4}$
Generative	$57.8_{\pm 0.5}$	$41.4_{\pm 0.6}$	$70.5_{\pm 0.4}$	$39.3_{\pm 0.7}$	$58.9_{\pm 0.5}$	$34.7_{\pm 0.6}$	$77.1_{\pm 0.3}$
MetaGPT-M	$58.4_{\pm 0.4}$	$42.1_{\pm 0.5}$	$71.1_{\pm 0.5}$	$40.2_{\pm 0.6}$	$59.6_{\pm 0.4}$	$35.3_{\pm 0.5}$	$77.9_{\pm 0.3}$
MacNet-M	$56.5_{\pm 0.5}$	$40.3_{\pm 0.6}$	$69.1_{\pm 0.4}$	$38.3_{\pm 0.8}$	$57.9_{\pm 0.6}$	$34.1_{\pm 0.5}$	$76.1_{\pm 0.4}$
ChatDev-M	$54.1_{\pm 0.6}$	$37.5_{\pm 0.7}$	$66.4_{\pm 0.5}$	$35.6_{\pm 0.7}$	$56.0_{\pm 0.5}$	$32.6_{\pm 0.6}$	$74.2_{\pm 0.4}$
MAD-M^2	$58.8_{\pm 0.5}$	$42.7_{\pm 0.4}$	$71.6_{\pm 0.5}$	$41.1_{\pm 0.7}$	$59.8_{\pm 0.4}$	$35.5_{\pm 0.5}$	$78.2_{\pm 0.3}$
MeMAD	$59.2_{\pm 0.4}$	$43.1_{\pm 0.5}$	$72.1_{\pm 0.4}$	$41.9_{\pm 0.6}$	$60.0_{\pm 0.5}$	$35.6_{\pm 0.4}$	$78.6_{\pm 0.3}$
G-Memory	$60.3_{\pm 0.4}$	$44.6_{\pm 0.5}$	$75.9_{\pm 0.4}$	$48.0_{\pm 0.6}$	$60.5_{\pm 0.5}$	$36.1_{\pm 0.4}$	$80.1_{\pm 0.3}$
G-Memory + EquiMem	$60.9_{\pm 0.3}$	$45.4_{\pm 0.4}$	$77.0_{\pm 0.4}$	$49.6_{\pm 0.5}$	$61.4_{\pm 0.4}$	$37.7_{\pm 0.3}$	$80.8_{\pm 0.3}$
HEDGE	$\mathbf{62.9}_{\pm 0.2}$	$\mathbf{48.3}_{\pm 0.3}$	$\mathbf{80.7}_{\pm 0.4}$	$\mathbf{57.8}_{\pm 0.3}$	$\mathbf{67.2}_{\pm 0.3}$	$\mathbf{42.3}_{\pm 0.3}$	$\mathbf{83.5}_{\pm 0.1}$
DyLAN 2024b							
No-memory	$53.2_{\pm 0.6}$	$36.8_{\pm 0.7}$	$65.7_{\pm 0.5}$	$34.6_{\pm 0.8}$	$54.3_{\pm 0.6}$	$31.3_{\pm 0.5}$	$72.6_{\pm 0.4}$
Voyager	$54.9_{\pm 0.5}$	$38.4_{\pm 0.6}$	$67.4_{\pm 0.5}$	$36.5_{\pm 0.7}$	$56.0_{\pm 0.5}$	$32.6_{\pm 0.6}$	$74.1_{\pm 0.3}$
Memory Bank	$52.6_{\pm 0.7}$	$36.2_{\pm 0.6}$	$65.0_{\pm 0.6}$	$35.1_{\pm 0.8}$	$55.1_{\pm 0.5}$	$32.1_{\pm 0.5}$	$73.0_{\pm 0.4}$
Generative	$55.3_{\pm 0.5}$	$39.0_{\pm 0.7}$	$68.0_{\pm 0.4}$	$36.9_{\pm 0.6}$	$56.6_{\pm 0.6}$	$33.0_{\pm 0.5}$	$74.6_{\pm 0.4}$
MetaGPT-M	$55.8_{\pm 0.4}$	$39.6_{\pm 0.6}$	$68.6_{\pm 0.5}$	$37.6_{\pm 0.7}$	$57.2_{\pm 0.5}$	$33.5_{\pm 0.4}$	$75.2_{\pm 0.3}$
MacNet-M	$54.1_{\pm 0.6}$	$37.9_{\pm 0.5}$	$66.7_{\pm 0.6}$	$35.9_{\pm 0.8}$	$55.6_{\pm 0.5}$	$32.5_{\pm 0.6}$	$73.9_{\pm 0.4}$
ChatDev-M	$51.8_{\pm 0.7}$	$35.4_{\pm 0.6}$	$64.1_{\pm 0.5}$	$33.3_{\pm 0.7}$	$53.9_{\pm 0.6}$	$30.8_{\pm 0.5}$	$71.8_{\pm 0.4}$
MAD-M^2	$56.2_{\pm 0.5}$	$40.0_{\pm 0.5}$	$68.9_{\pm 0.4}$	$38.4_{\pm 0.6}$	$57.5_{\pm 0.4}$	$33.8_{\pm 0.5}$	$75.6_{\pm 0.3}$
MeMAD	$56.6_{\pm 0.5}$	$40.3_{\pm 0.6}$	$69.1_{\pm 0.4}$	$39.2_{\pm 0.7}$	$57.8_{\pm 0.5}$	$34.0_{\pm 0.4}$	$76.0_{\pm 0.3}$
G-Memory	$57.5_{\pm 0.4}$	$41.6_{\pm 0.5}$	$72.4_{\pm 0.5}$	$44.8_{\pm 0.6}$	$58.2_{\pm 0.4}$	$34.4_{\pm 0.5}$	$77.3_{\pm 0.3}$
G-Memory + EquiMem	$58.1_{\pm 0.4}$	$42.4_{\pm 0.5}$	$73.5_{\pm 0.4}$	$46.0_{\pm 0.6}$	$59.1_{\pm 0.5}$	$35.2_{\pm 0.4}$	$78.0_{\pm 0.3}$
HEDGE	$\mathbf{60.6}_{\pm 0.2}$	$\mathbf{44.8}_{\pm 0.3}$	$\mathbf{75.1}_{\pm 0.3}$	$\mathbf{50.7}_{\pm 0.3}$	$\mathbf{62.4}_{\pm 0.4}$	$\mathbf{37.4}_{\pm 0.2}$	$\mathbf{80.1}_{\pm 0.2}$

Table 2: Comparison of memory mechanisms under AutoGen and DyLAN. Values are mean accuracy over three runs, with red subscripts indicating standard deviation. HEDGE achieves the best performance across all benchmarks. See Appendix I for the results on MacNet and G-Designer.

per hyperedge ($M = 8$). Memory is updated after each debate round. Further details are provided in Appendix G.

4.2 Performance Across Reasoning Tasks

Performance across backbones. Table 1 reports results in a three-agent setting with two proposers and one critic. HEDGE achieves the best performance among methods using the same backbone on all seven benchmarks. With GPT-4o-mini, it outperforms G-Memory + EquiMem by **5.0 points on average**, with gains ranging from **2.8** on MATH500 to **8.2** on ConflictQA. With Qwen3.5-27B, the average gain is **3.8 points**, ranging from **1.6** on MMLU-Pro to **6.1** on ConflictQA. Improvements are particularly strong on **ConflictQA, NovelQA, and FaithEval**, where agents must retain competing claims, track evidence provenance, and revise conclusions as new information emerges. This pattern suggests that HEDGE is most beneficial when reasoning depends on integrating information distributed across agents and debate rounds. The gains of **5.4** and **3.8** points on HotpotQA further indicate that intra-hyperedge relations help preserve multi-step evidence chains, whereas the smaller gain on MMLU-Pro suggests less benefit when answers rely primarily on parametric knowledge. Notably, HEDGE with Qwen3.5-27B also **surpasses vanilla GPT-5.4 on all seven benchmarks**, showing that structured debate memory can partly compensate for differences in backbone scale. Appendix H provides a qualitative example.

Comparison of memory mechanisms. Table 2 compares memory mechanisms under AutoGen (Wu et al., 2024) and DyLAN (Liu et al., 2024b). HEDGE achieves the best result on every benchmark under both frameworks. MAD-M^2 consistently improves over native and lightweight memory variants, but remains slightly below MeMAD and substantially behind graph-based memories, suggesting that filtering erroneous content alone cannot fully capture cross-agent dependencies. Under AutoGen, HEDGE outperforms the strongest baseline, G-Memory + EquiMem, by **4.3 points on average**, including gains of **2.0** on MATH500, **2.9** on GPQA, **3.7** on FaithEval, and

8.2 on ConflictQA. Under DyLAN, the average gain is **2.7 points**, including improvements of **2.5** on MATH500, **2.4** on GPQA, and **4.7** on ConflictQA. The largest gains again occur on **ConflictQA and NovelQA**, indicating that selective hypergraph memory is particularly effective for preserving and reconciling competing information. The larger gains under AutoGen may reflect its richer interaction pattern, while DyLAN's dynamic orchestration may already reduce some redundant exchanges. Consistent gains across both frameworks show that HEDGE is orchestration-agnostic. Appendix I reports results for G-Designer (Zhang et al., 2025b) and MacNet (Qian et al., 2025a).

4.3 Efficiency Analysis

We evaluate efficiency from two complementary perspectives: **(i)** total token usage and reasoning rounds, and **(ii)** persistent memory footprint. Figure 3 presents representative results, with detailed values for all benchmarks reported in Appendix J.

Token usage and reasoning rounds. The top panel of Figure 3 shows that HEDGE achieves a stronger performance–cost trade-off than structured-memory baselines. Compared with G-Memory + EquiMem, HEDGE reduces token usage by **15–17%** and reasoning rounds by **36–41%** across the displayed settings, while improving accuracy by **2.2–8.2 points**. For example, on ConflictQA with AutoGen, HEDGE reaches 57.8 accuracy using 36.9K tokens and 5.7 rounds, compared with 49.6 accuracy, 43.6K tokens, and 9.2 rounds for G-Memory + EquiMem. MAD-M^2 uses fewer tokens through perplexity-based masking, but its lower accuracy suggests that aggressively removing context cannot replace structured retention of cross-agent dependencies.

Memory footprint. The bottom panel shows that HEDGE avoids the large persistent state required by graph-based memories. Relative to G-Memory + EquiMem, it reduces memory footprint by approximately **47–48%** across the displayed settings while improving accuracy by **1.6–2.7 points**. On MATH500 with AutoGen, for instance, HEDGE improves accuracy from 60.9 to 62.9 while reducing memory from 31.9K to 16.5K tokens. Although lightweight methods retain less memory, they perform worse, indicating that HEDGE provides a more favorable accuracy–memory trade-off. Learned selection and hypergraph organization allow HEDGE to preserve useful argumentative structure while limiting redundant context and accelerating convergence.

4.4 Effect of MSM and Its Training Stages

Table 3 analyzes the contribution of MSM and its two training stages. Storing all candidates produces the largest memory footprint and the weakest accuracy, confirming that indiscriminate retention introduces redundant and distracting context. Stage I supervised training provides substantial gains, improving accuracy by **3.5 points** on ConflictQA and **3.1 points** on HotpotQA while reducing memory by approximately **26%**. Stage II reinforcement-learning refinement yields a further **2.6**- and **1.7**-point improvement, respectively, while additionally reducing rounds, token usage, and memory. Overall, the complete MSM improves accuracy by **6.1 points** on ConflictQA and **4.8 points** on HotpotQA relative to storing all candidates, while reducing memory footprint by approximately **39–41%**. The larger Stage II gain on ConflictQA suggests that feedback from debate outcomes is particularly valuable for retaining useful conflicting evidence and refutations.

Configuration	ConflictQA				HotpotQA			
	Acc.	R	T	M	Acc.	R	T	M
No MSM	51.7	8.1	44.8K	30.6K	37.5	8.0	46.3K	32.1K
Stage I only	55.2	6.5	39.6K	22.7K	40.6	6.6	41.2K	23.8K
Stage I + Stage II	**57.8**	**5.7**	**36.9K**	**18.2K**	**42.3**	**5.8**	**38.1K**	**19.6K**

Table 3: Effect of MSM and its training stages on accuracy, rounds (R), token usage (T), and memory footprint in tokens (M).

Additional analysis. We further evaluate sensitivity to the embedding model, retrieval breadth, and relation extraction model in Appendices K, L, and M, respectively. Appendix N analyzes the complementary effect of sparse signatures and dense embeddings.

5 Conclusion

We introduced HEDGE, a selective intra-debate memory framework that represents evolving debate trajectories as semantic hypergraphs and uses a learned memory selection module to retain useful information while reducing redundancy. Across seven benchmarks, HEDGE outperforms strong single-agent, multi-agent, and memory baselines, improving accuracy by 5.0 points on average over the strongest multi-agent system with GPT-4o-mini and by up to 8.2 points on conflict-heavy tasks. It also reduces memory usage by 48% and debate rounds by 34%, demonstrating that structured,

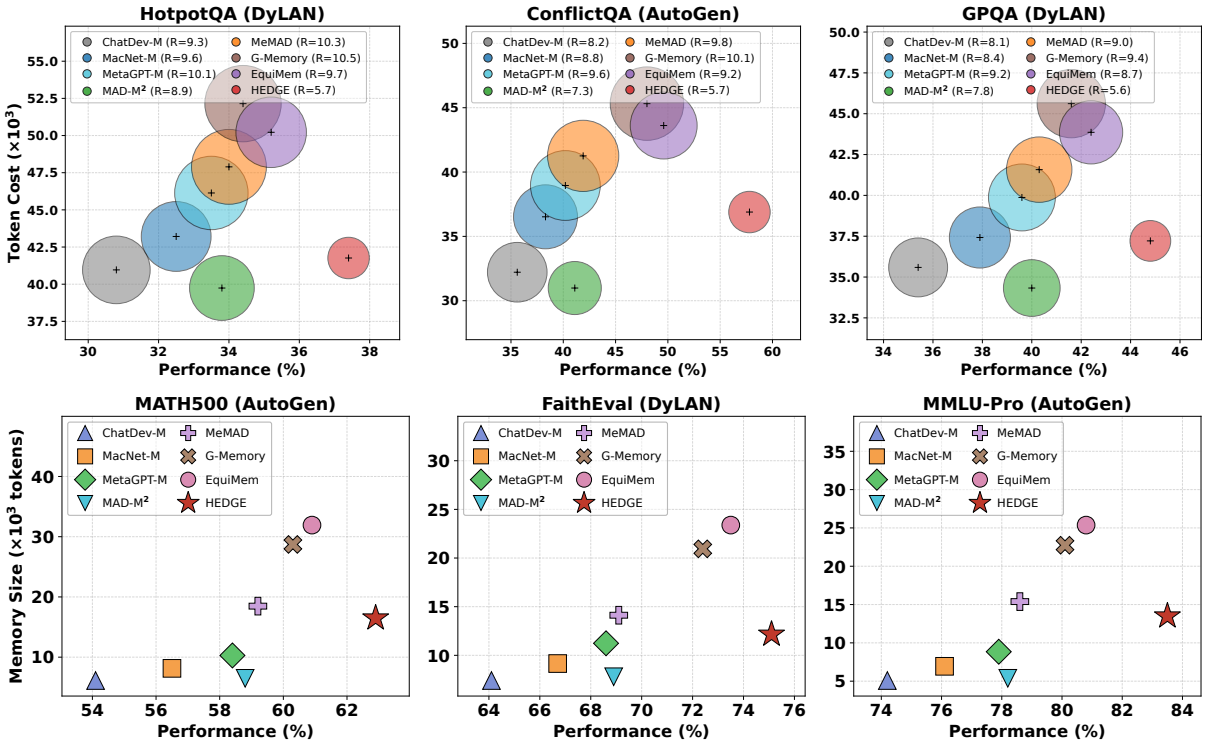


Figure 3: Performance–efficiency trade-offs across representative benchmarks and orchestration frameworks. **Top:** performance versus total token usage, with bubble size indicating the average number of reasoning rounds. **Bottom:** performance versus persistent memory footprint. See Appendix J for full results across benchmarks.

utility-aware memory improves both the effectiveness and efficiency of multi-agent debate.

Limitations

HEDGE has some limitations. First, MSM is trained using pseudo-labels derived from debate trajectories rather than human-annotated memory decisions, so it may discard claims whose utility becomes apparent only in later rounds. Second, memory construction depends on accurately decomposing agent responses into atomic items and representing them through embeddings and predicted relations; errors in either step may propagate through the hypergraph. Third, Stage II relies on reward weights and routing thresholds that balance task accuracy against memory compactness, and different settings may shift this trade-off.

Ethical Considerations

Potential risks. HEDGE inherits the limitations and risks of its underlying language models, including hallucinations, biases, and unsupported reasoning. Because information is shared across agents, an incorrect claim admitted to memory may influence later rounds or create false consensus. Errors in decomposing responses or predicting intra-hyperedge relations may similarly distort the stored argumentative structure. Although MSM is designed to filter low-utility content, it does not guarantee factuality or fairness. Deployments in high-stakes domains should therefore include source verification, human oversight, and auditing of both model outputs and stored memory.

LLM usage and computational cost. We use GPT-4o to generate debate configurations and MSM training trajectories, GPT-4o-mini as a debate backbone and relation predictor, and Qwen3.5-27B as an alternative backbone. These models are used for research evaluation and not for making decisions about individuals. Multi-agent debate remains more computationally expensive than single-agent inference. While HEDGE reduces token usage, memory footprint, and debate rounds relative to strong memory-based systems, its overall computational and environmental costs remain non-trivial. We report detailed efficiency measurements in Appendix J to support assessment of this trade-off.

Use of AI assistance. During manuscript preparation, we used large language models to improve phrasing and presentation and to generate icons used in figures. All technical contributions, exper-

imental designs, analyses, and conclusions were developed and verified by the authors.

Reproducibility statement. We describe the memory construction procedure, MSM architecture, training objectives, hyperparameters, and evaluation protocol in the main paper and appendices. The MSM data-generation pipeline, including debate simulation, memory logging, and pseudo-label construction, is detailed in Appendix C, and all prompts are provided in Appendix O. Upon publication, we will release the implementation, MSM training scripts, generated trajectories, checkpoints, and evaluation code under a permissive research license.

Licensing and artifact release. We use external datasets, models, and baseline implementations according to their original licenses and terms of use. Before release, we will document the source, license, and required attribution for each dependency. Our original code will be released under a permissive license. Generated trajectories and checkpoints will be redistributed only when permitted by the licenses and terms governing their source datasets and models; otherwise, we will release generation scripts, configuration files, and dataset identifiers needed to reproduce them. We will not redistribute third-party model weights, datasets, or baseline artifacts when their licenses prohibit doing so.

Use of existing artifacts and data. All experiments use publicly available benchmarks, models, and baseline implementations according to their original licenses and access conditions. MSM training trajectories are generated from the HotpotQA full-wiki training split (Yang et al., 2018), which is derived from Wikipedia. We do not collect new user data, attempt to identify individuals, or involve human subjects. Naturally occurring names or other entities may remain in the original benchmark content, and our released artifacts are intended solely for research and reproducibility.

References

Anthropic. 2026. Claude sonnet 4.6 system card. Technical report, Anthropic.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better LLM-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*.

Justin Chen, Swarnadeep Saha, and Mohit Bansal. 2024. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7066–7085, Bangkok, Thailand. Association for Computational Linguistics.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Dongming Jiang, Yi Li, Songtao Wei, Jinxin Yang, Ayushi Kishore, Alysa Zhao, Dingyi Kang, Xu Hu, Feng Chen, Qiannan Li, and Bingzhe Li. 2026. Anatomy of agentic memory: Taxonomy and empirical analysis of evaluation and system limitations. *Preprint*, arXiv:2602.19320.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.

Shuai Ling, Lizi Liao, Dongmei Jiang, and Weili Guan. 2025. MeMAD: Structured memory of debates for enhanced multi-agent reasoning. In *Second Conference on Language Modeling*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yexiang Liu, Jie Cao, Zekun Li, Ran He, and Tieniu Tan. 2025. Breaking mental set to improve reasoning through diverse multi-agent debate. In *The Thirteenth International Conference on Learning Representations*.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024b. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*.

Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and Luu Anh Tuan. 2025. Hypergraphrag: Retrieval-augmented generation via hypergraph-structured knowledge representation. *Preprint*, arXiv:2503.21322.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Wenyu Mao, Haoyang Liu, Haosong Tan, Yaorui Shi, Jiancan Wu, An Zhang, and Xiang Wang. 2026. Joint optimization of multi-agent memory system. *Preprint*, arXiv:2603.12631.

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. *Preprint*, arXiv:2605.09278.

Yifei Ming, Senthil Purushwalkam, Shrey Pandit, Zixuan Ke, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. 2025. Faitheval: Can your language model stay faithful to context, even if "the moon is made of marshmallows". In *The Thirteenth International Conference on Learning Representations*.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2026. Introducing GPT-5.4.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. *Preprint*, arXiv:2310.08560.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual ACM Symposium on User Interface Software and Technology*, pages 1–22.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025a. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. 2025b. Memorag: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM Web Conference 2025 (TheWebConf 2025)*, Sydney, Australia. ACM.

Qwen. 2026. Qwen3.5: Accelerating productivity with native multimodal agents.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Hongduan Tian, Xiao Feng, Ziyuan ZHAO, XiangyuZhu, Rolan Yan, and Bo Han. 2026. Multi-agent debate with memory masking. In *The Fourteenth International Conference on Learning Representations*.

Cunxiang Wang, Ruoxi Ning, Boqi Pan, Tonghui Wu, Qipeng Guo, Cheng Deng, Guangsheng Bao, Qian Wang, and Yue Hang. 2025a. NovelQA: Benchmarking question answering on documents exceeding 200k tokens. In *The Thirteenth International Conference on Learning Representations*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Jingjin Wang and Jiawei Han. 2025. PropRAG: Guiding retrieval with beam search over proposition paths. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6212–6227, Suzhou, China. Association for Computational Linguistics.

Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025b. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. 2026. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12805–12825, San Diego, California, United States. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. 2026. Memagent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *The Fourteenth International Conference on Learning Representations*.

Xiangjun Zai, Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, and Wenjie Zhang. 2026. Proh: Dynamic planning and reasoning over knowledge hypergraphs for retrieval-augmented generation. In *Proceedings of the ACM Web Conference 2026*, WWW '26, page 4256–4267, New York, NY, USA. Association for Computing Machinery.

Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng YAN. 2025a. G-memory: Tracing hierarchical memory for multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Heng. 2025b. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Forty-second International Conference on Machine Learning*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025c. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: enhancing large language models with long-term memory. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Chulun Zhou, Chunkang Zhang, Guoxin Yu, Fandong Meng, Jie Zhou, Wai Lam, and Mo Yu. 2026. Hgmem: Hypergraph-based working memory to improve multi-step rag for long-context complex relational modeling. *Preprint*, arXiv:2512.23959.

A Online Memory Maintenance

Memory maintenance is performed once per debate round rather than after every memory operation. We use r to index debate rounds, distinguishing it from the candidate-level step t in Section 3.3. Let $\mathcal{B}^{(r)}$ be the set of nodes assigned to BACKUP during round r. At the end of each round, the memory is updated as follows.

Backup reconsideration and insertion. The MSM reevaluates each node in $\mathcal{B}^{(r)}$ using the updated debate state. Nodes promoted to STORE, together with newly stored nodes, are assigned to all compatible hyperedges according to Eq. (5). A node with no compatible hyperedge initializes a new hyperedge.

Hyperedge merging. We consider two hyperedges $e_k^{(r)}$ and $e_\ell^{(r)}$ redundant when both their signature overlap and centroid similarity exceed the merge thresholds:

$$J_{k\ell}^{(r)} \geq \tau_{\text{merge}}^{\text{sig}} \quad \wedge \quad S_{k\ell}^{(r)} \geq \tau_{\text{merge}}^{\text{emb}}, \tag{16}$$

where $J_{k\ell}^{(r)}$ denotes the Jaccard similarity between their signatures and $S_{k\ell}^{(r)}$ denotes the cosine similarity between their centroids. Their merged membership and signature are:

$$\mathcal{N}_m^{(r)} = \mathcal{N}_k^{(r)} \cup \mathcal{N}_\ell^{(r)}, \qquad \sigma_m^{(r)} = \sigma_k^{(r)} \cup \sigma_\ell^{(r)}. \tag{17}$$

Consistent with Section 3.1, we maintain the cached unnormalized embedding sum $\mathbf{s}_k^{(r)}$ for each hyperedge. Rather than combining normalized centroids directly, we merge these sums:

$$\begin{aligned} \mathbf{s}_m^{(r)} &= \sum_{v \in \mathcal{N}_m^{(r)}} \mathbf{h}_v \\ &= \mathbf{s}_k^{(r)} + \mathbf{s}_\ell^{(r)} - \sum_{v \in \mathcal{N}_k^{(r)} \cap \mathcal{N}_\ell^{(r)}} \mathbf{h}_v. \end{aligned} \tag{18}$$

The overlap correction prevents shared nodes from being counted twice. The merged centroid is then obtained by normalization:

$$\mathbf{c}_m^{(r)} = \frac{\mathbf{s}_m^{(r)}}{\|\mathbf{s}_m^{(r)}\|_2}. \tag{19}$$

Utility-based pruning. Let $\mathcal{V}_{\text{cand}}^{(r)}$ denote the set of candidate nodes extracted during debate round r. For each hyperedge, we compute its round-level utility as the maximum candidate-to-hyperedge similarity:

$$U_k^{(r)} = \max_{v_i \in \mathcal{V}_{\text{cand}}^{(r)}} \cos\left(\mathbf{x}_i, \mathbf{c}_k^{(r)}\right), \tag{20}$$

where $\mathbf{x}_i$ is the unprojected pooled representation of candidate v_i. This uses the same hyperedge-specific alignment underlying Eq. (10). However, while the candidate-level feature U_t takes the maximum across hyperedges for a given candidate, $U_k^{(r)}$ takes the maximum across candidates within a debate round for a given hyperedge.

For each hyperedge, we maintain a counter $q_k^{(r)}$ recording consecutive rounds of low utility:

$$q_k^{(r)} = \begin{cases} q_k^{(r-1)} + 1, & U_k^{(r)} < \tau_{\text{util}}, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

A hyperedge is removed when $q_k^{(r)} \geq P$, where P denotes the pruning patience. Each of its nodes is then matched against the surviving hyperedges using Eq. (5). Compatible nodes are reassigned to the highest-scoring surviving hyperedge, while unmatched nodes are discarded.

Incremental relational update. When hyperedges are merged or nodes are reassigned, previously inferred relations between unchanged node pairs are retained. The relation module is invoked only for pairs involving newly added or moved nodes. Centroids and signatures are likewise updated only for affected hyperedges. Thus, existing node embeddings, signatures, memberships, and relation labels are reused across rounds, and maintenance depends primarily on the round-level changes rather than the complete memory size.

B MSM Classifier Architecture

The MSM routing policy is implemented as a three-layer multi-layer perceptron (MLP). Its input combines candidate content, the current hypergraph context, and two scalar utility features.

Input representation. At candidate step t, the classifier input contains three components:

- **Utility features** $\boldsymbol{\phi}_t^{\text{UN}} = [U_t, N_t] \in \mathbb{R}^2$, where U_t measures alignment with the current memory and N_t measures candidate specificity, as defined in Eq. (10).
- **Candidate representation** $\mathbf{x}_t^{\text{pool}} \in \mathbb{R}^{d_h}$, obtained by pooling the projected token representations of the candidate.

- **Hypergraph context** $\mathbf{c}_t^{\text{pool}} \in \mathbb{R}^{d_h}$, obtained by attending from the candidate tokens to the current hyperedge centroids.

Let $\mathbf{X}_t \in \mathbb{R}^{n_t \times d_e}$ denote the candidate token representations, and let $\mathbf{C}_t \in \mathbb{R}^{K_t \times d_e}$ stack the centroids of the K_t hyperedges available at step t. Here, d_e is the embedding dimension; for Qwen3-Embedding-4B (Zhang et al., 2025c), $d_e = 2560$.

The utility features are computed using the unprojected pooled representation $\mathbf{x}_t = \text{Pool}(\mathbf{X}_t)$ and the original hyperedge centroids. For cross-attention, candidate and centroid representations are projected into a shared dimension $d_h < d_e$:

$$\widetilde{\mathbf{X}}_t = \mathbf{X}_t \mathbf{W}_x, \qquad \widetilde{\mathbf{C}}_t = \mathbf{C}_t \mathbf{W}_c. \tag{22}$$

We compute candidate-to-memory cross-attention using

$$\mathbf{Z}_t = \text{MultiHeadAttn}\left(\widetilde{\mathbf{X}}_t, \widetilde{\mathbf{C}}_t, \widetilde{\mathbf{C}}_t\right), \tag{23}$$

where candidate tokens form the queries and hyperedge centroids form the keys and values.

The projected candidate and memory representations are then pooled:

$$\mathbf{x}_t^{\text{pool}} = \text{Pool}(\widetilde{\mathbf{X}}_t), \qquad \mathbf{c}_t^{\text{pool}} = \text{Pool}(\mathbf{Z}_t). \tag{24}$$

This cross-attention allows MSM to emphasize hyperedges relevant to the incoming candidate. Because the hypergraph evolves during debate, $\mathbf{C}_t$ and K_t reflect the memory state at step t.

The three components are concatenated to form

$$\boldsymbol{\phi}_t = \left[\boldsymbol{\phi}_t^{\text{UN}} \,\middle\|\, \mathbf{x}_t^{\text{pool}} \,\middle\|\, \mathbf{c}_t^{\text{pool}}\right], \qquad d_{\text{in}} = 2d_h + 2. \tag{25}$$

Network structure. The classifier contains three linear layers:

- $\text{Linear}(d_{\text{in}}, d_h)$, followed by LayerNorm, GELU, and Dropout;
- $\text{Linear}(d_h, d_h/2)$, followed by LayerNorm, GELU, and Dropout;
- $\text{Linear}(d_h/2, 3)$.

The first two layers constitute $\text{MLP}(\boldsymbol{\phi}_t)$. The final layer, parameterized by $\mathbf{W}_\pi$ and $\mathbf{b}_\pi$, produces logits over $\{\text{STORE}, \text{BACKUP}, \text{OMIT}\}$. The resulting policy distribution is

$$\pi_\theta(a_t \mid \boldsymbol{\phi}_t) = \text{Softmax}\left(\mathbf{W}_\pi \,\text{MLP}(\boldsymbol{\phi}_t) + \mathbf{b}_\pi\right). \tag{26}$$

By combining candidate semantics, context-aware hypergraph information, and utility features, MSM makes routing decisions conditioned on both incoming content and the evolving debate memory.

C Data Generation via Debate Simulation for Training MSM

We provide detailed information on the data generation pipeline used to construct training trajectories for the memory selection module (MSM).

C.1 Dataset

We generate training data using the **HotpotQA** dataset (Yang et al., 2018), specifically the *train split of the fullwiki variant*. Each instance consists of a multi-hop question paired with a set of supporting Wikipedia passages. During simulation, agents are provided only with the question and its associated context passages; gold answers and supporting facts are withheld. For reproducibility, we store the full original HotpotQA record (including answers and annotations) alongside each generated trajectory, although these fields are never exposed to the agents during simulation.

C.2 Retrieval Setup

To support grounded reasoning, we construct a lightweight retrieval module over the provided context. Each context passage is segmented into sentence-level chunks, and each sentence is assigned a unique identifier. Agents may issue retrieval queries during the debate, which are handled using a sparse similarity method (BM25). Retrieved sentences and their identifiers are returned to the agent and logged. This design ensures that all external information access is explicit, traceable, and grounded in the dataset-provided evidence.

C.3 Agent Configuration

For each sample, we instantiate a multi-agent debate consisting of 3 to 6 agents. The debate configuration is generated once per sample using a high-capacity model (GPT-4o) (OpenAI, 2024), which specifies: **(i)** the number of agents, **(ii)** each agent's role and objective, **(iii)** a fixed turn order, and **(iv)** the model size assigned to each agent (Qwen3-4B or Qwen3-8B (Yang et al., 2025)).

This procedure introduces diversity in agent behavior and reasoning strategies while maintaining a consistent and reproducible interaction protocol.

C.4 Debate Protocol

Each debate proceeds for at most 15 rounds, with agents acting sequentially according to a predefined turn order. At each turn, an agent may PROPOSE, SUPPORT, REFUTE, QUESTION, REVISE, SUMMARIZE, or RETRIEVE.

When an agent issues a REFUTE action or expresses strong disagreement, the simulator initiates a conflict-resolution sub-dialogue. This consumes one round from the global budget and continues until the agents reach a consistent interpretation or produce an explicit resolution. The outcome is summarized and written to memory before the main debate resumes.

C.5 Memory Logging and Traceability

Throughout the simulation, we maintain a global, append-only memory that records all agent outputs. Each memory item is assigned a unique identifier and stored together with metadata, including the round index, agent role, action type, and references to previously used memory items or retrieved evidence.

To enable fine-grained supervision, agents are required to explicitly reference the memory items or retrieved sentences they rely on when generating new outputs. This design allows us to track how information propagates through the debate and to identify which memory items contribute to future reasoning steps, conflict resolution, and final outputs.

While dependency annotations are fully logged, only the textual content, identifiers, and action types are exposed to agents during simulation. This ensures that training signals are derived from observable interactions rather than injected supervision.

C.6 Trajectory-Level Supervision and Pseudo-Labeling

Training examples are constructed from full multi-agent debate trajectories. For each candidate memory item x_t, we compute trajectory-level signals *post hoc*, capturing its role in future reasoning. These signals include: **(i)** whether the item is retrieved or explicitly referenced in subsequent steps, **(ii)** whether it participates in conflict resolution, and **(iii)** whether it contributes to the final answer or summary.

Based on these signals, we derive pseudo-labels $a_t^* \in \{\text{STORE}, \text{BACKUP}, \text{OMIT}\}$ in conjunction with redundancy and novelty statistics. Items that are frequently reused and provide non-redundant information are labeled STORE, partially useful or redundant items are labeled BACKUP, and items that are never reused are labeled OMIT. These pseudo-labels provide supervised targets for training MSM in Stage I.

D MSM Training Procedure

Dataset. Both training stages use simulated multi-agent debates from the HotpotQA FullWiki training split (Yang et al., 2018). For each sample, we instantiate three to six agents that collaboratively reason over the question and its associated passages. Agent responses are decomposed into atomic candidate items, and agents explicitly reference previously retrieved evidence or memory items whenever applicable. This makes memory usage traceable across the trajectory and allows us to determine whether each candidate is later reused, involved in conflict resolution, or incorporated into the final answer. These signals are used to construct pseudo-labels for STORE, BACKUP, and OMIT. Further details are provided in Appendix C. After training, the same MSM policy is applied to all evaluation benchmarks without target-specific adaptation; HotpotQA evaluation uses examples disjoint from its training split.

Training. We train MSM in two stages. Stage I uses supervised learning to initialize the routing policy from trajectory-derived pseudo-labels, while Stage II refines this policy through reinforcement learning on additional simulated HotpotQA debates.

Stage I: Supervised pretraining. For each candidate node v_t, MSM predicts an action $a_t \in \{\text{STORE}, \text{BACKUP}, \text{OMIT}\}$ from its representation $\boldsymbol{\phi}_t$, which encodes the candidate and current memory state. Pseudo-labels a_t^* are derived *post hoc* from the complete trajectory. Candidates that are repeatedly reused, support conflict resolution, or contribute to the final answer are labeled STORE; candidates with potential but limited future use are labeled BACKUP; and redundant candidates that are neither reused nor contribute downstream are labeled OMIT. Redundancy and specificity signals are also considered when resolving ambiguous cases.

The routing policy is

$$\pi_\theta(a_t \mid \boldsymbol{\phi}_t) = \text{Softmax}\left(\mathbf{W}_\pi \,\text{MLP}(\boldsymbol{\phi}_t) + \mathbf{b}_\pi\right), \tag{27}$$

where MLP consists of two hidden linear layers, while $\mathbf{W}_\pi$ and $\mathbf{b}_\pi$ parameterize the third, output layer, as detailed in Appendix B. We optimize

$$\mathcal{L}_{\mathrm{I}} = -\sum_t \log \pi_\theta(a_t^* \mid \boldsymbol{\phi}_t) + \lambda_b \mathcal{L}_{\mathrm{bal}}, \quad (28)$$

with the class-balancing regularizer

$$\mathcal{L}_{\mathrm{bal}} = \sum_a \left(\mathbb{E}_t[\pi_\theta(a \mid \boldsymbol{\phi}_t)] - \rho_a\right)^2, \quad (29)$$

where a ranges over the three routing actions and ρ_a is its empirical frequency in the training data. This term prevents degenerate policies caused by label imbalance, such as consistently predicting OMIT.

Stage II: Reinforcement learning with multi-agent feedback. Starting from the Stage I checkpoint, we further optimize MSM using PPO on simulated debates from the HotpotQA FullWiki training split. No trajectories or labels from the downstream evaluation benchmarks are used for target-specific refinement. Each atomic candidate forms a sequential decision point, and the resulting routing decisions determine how the shared memory evolves throughout the debate.

At the end of each trajectory τ, we compute

$$R_\tau = \lambda_{\mathrm{task}} R_{\mathrm{task}} + \lambda_{\mathrm{route}} R_{\mathrm{route}} + \lambda_{\mathrm{eff}} R_{\mathrm{eff}}, \quad (30)$$

where R_{task} measures final-answer correctness. The routing reward is

$$R_{\mathrm{route}} = w_s p_{\mathrm{store}} + w_b p_{\mathrm{backup}} - w_o p_{\mathrm{miss}}, \quad (31)$$

where p_{store} and p_{backup} are the proportions of stored and backed-up items subsequently used, and p_{miss} is the proportion of omitted items later identified as useful. Ratios with no corresponding routed items are defined as zero.

The efficiency reward penalizes unnecessary memory growth:

$$R_{\mathrm{eff}} = -\left(\alpha_c \widetilde{n}_{\mathrm{store}} + \alpha_a \widetilde{m}_{\mathrm{avg}} + \alpha_f \widetilde{m}_{\mathrm{final}}\right), \quad (32)$$

where $\widetilde{n}_{\mathrm{store}}$, $\widetilde{m}_{\mathrm{avg}}$, and $\widetilde{m}_{\mathrm{final}}$ denote normalized counts of stored candidates, average memory size, and final memory size, respectively. The trajectory reward is propagated to earlier routing decisions through discounted returns and generalized advantage estimation.

The PPO loss is

$$\mathcal{L}_{\mathrm{II}} = -\mathbb{E}_t[\min(\varrho_t A_t, \mathrm{clip}(\varrho_t, 1-\epsilon, 1+\epsilon) A_t)] + c_v \mathcal{L}_{\mathrm{value}} - c_e \mathcal{H}(\pi_\theta), \quad (33)$$

where

$$\varrho_t = \frac{\pi_\theta(a_t \mid \boldsymbol{\phi}_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid \boldsymbol{\phi}_t)} \quad (34)$$

is the policy probability ratio and A_t is the generalized advantage estimate. Clipping limits abrupt policy changes between updates.

A critic head estimates the expected future return:

$$\mathcal{L}_{\mathrm{value}} = \mathbb{E}_t\left[\left(V_\psi(\boldsymbol{\phi}_t) - \widehat{G}_t\right)^2\right], \quad (35)$$

where $\widehat{G}_t$ is the return target obtained from the trajectory rewards. The entropy term is

$$\mathcal{H}(\pi_\theta) = -\mathbb{E}_t\left[\sum_a \pi_\theta(a \mid \boldsymbol{\phi}_t) \log \pi_\theta(a \mid \boldsymbol{\phi}_t)\right]. \quad (36)$$

This term encourages exploration and prevents premature collapse to a single routing action, particularly early in Stage II.

E Baselines and Benchmarks

E.1 Baselines

Single-agent baselines. We evaluate GPT-5.4 (OpenAI, 2026), GPT-4o, GPT-4o-mini (OpenAI, 2024), and Qwen3.5-27B (Qwen, 2026) in a standard single-pass setting. We also include Self-Refine (Madaan et al., 2023), Self-Consistency (Wang et al., 2023b), and Reflexion (Shinn et al., 2023), which improve reasoning through iterative refinement, multiple reasoning paths, or verbal feedback but do not maintain structured shared memory.

Multi-agent and memory-aware baselines. We compare against MAD (Liang et al., 2024) and MoA (Wang et al., 2025b), which use debate and layered aggregation, respectively. The closest memory-aware baselines are MeMAD (Ling et al., 2025), which reuses experiences from completed debates; G-Memory (Zhang et al., 2025a), which represents collaboration through pairwise graphs; MAD-M^2 (Tian et al., 2026), for which we use the perplexity-based masking variant; and EquiMem (Meng et al., 2026), which calibrates shared-memory updates and is evaluated together with G-Memory. These methods cover cross-episode reuse, graph-based memory, erroneous-memory masking, and shared-memory calibration.

Memory mechanisms and orchestration frameworks. To evaluate HEDGE as a modular memory component, we integrate it into AutoGen (Wu et al., 2024), G-Designer (Zhang et al., 2025b), DyLAN (Liu et al., 2024b), and MacNet (Qian et al., 2025a), which differ in agent roles, communication structures, and orchestration policies. Within these frameworks, we compare against Voyager (Wang et al., 2023a), Memory Bank (Zhong et al., 2024), Generative Agents (Park et al., 2023), MAD-M^2 (Tian et al., 2026), MeMAD, G-Memory, and G-Memory + EquiMem (Meng et al., 2026), as well as the native memory mechanisms of MetaGPT (Hong et al., 2024), MacNet (Qian et al., 2025a), and ChatDev (Qian et al., 2024). This setup evaluates whether the benefits of HEDGE generalize across different interaction and control strategies.

E.2 Benchmarks

We evaluate seven benchmarks covering mathematical reasoning, scientific knowledge, contextual faithfulness, conflicting evidence, long-context understanding, multi-hop retrieval, and broad-domain reasoning. Table 4 summarizes their scale and evaluation setting; in aggregate the suite spans 33,234 evaluation instances, ranging from single-sentence prompts to inputs exceeding 200K tokens.

MATH500. MATH500 (Hendrycks et al., 2021) contains 500 competition-level mathematics problems requiring multi-step symbolic reasoning and precise intermediate computation. The subset spans seven subjects (Algebra 124, Intermediate Algebra 97, Prealgebra 82, Number Theory 62, Precalculus 56, Geometry 41, Counting and Probability 38) and five difficulty levels (Level 1–5: 43, 90, 105, 128, 134), so accuracy is not dominated by any single topic or difficulty band.

GPQA. GPQA (Rein et al., 2024) contains 448 graduate-level science questions in biology, physics, and chemistry, each with four options, requiring deep domain knowledge and conceptual reasoning. Questions are validated to remain difficult for non-experts even with unrestricted web access, making the benchmark a test of knowledge rather than of retrievability.

FaithEval. FaithEval (Ming et al., 2025) evaluates contextual faithfulness under unanswerable, inconsistent, and counterfactual evidence, comprising 4.9K problems in total. We use its counterfactual subset (1,000 instances), in which each question is paired with a single passage asserting a fact that contradicts world knowledge, to test whether agents follow the provided context despite conflicts with parametric knowledge.

ConflictQA. ConflictQA (Xie et al., 2024) measures the ability to identify and resolve conflicting evidence. We use the PopQA setting, derived from the entity-centric questions of PopQA, in which each instance pairs an elicited memory passage with a generated counter-memory supporting an incompatible answer. We evaluate on the `ConflictQA-popQA-gpt4` configuration (9,544 instances). We treat this as an evidence-conflict task rather than a parametric-versus-contextual one, since the memories were elicited from GPT-4 rather than from the backbones we evaluate; fixing a single configuration also keeps the evaluation set identical across systems.

NovelQA. NovelQA (Wang et al., 2025a) evaluates question answering over full-length novels, with 2,305 manually annotated questions over 89 English novels and an average input length exceeding 200K tokens. Questions are annotated by complexity (detail, single-hop, multi-hop) and by aspect (character, plot, relation, setting, span, times, meaning), requiring retrieval and reasoning across long narrative contexts.

HotpotQA. HotpotQA (Yang et al., 2018) requires multi-hop reasoning over multiple Wikipedia documents. We use the FullWiki development set (7,405 questions), where relevant evidence must be retrieved from a corpus of roughly 5.2M Wikipedia introductory paragraphs rather than from a small distractor pool.

MMLU-Pro. MMLU-Pro (Wang et al., 2024) extends MMLU with 12,032 test questions across 14 domains, expanding the option set from four to ten and filtering trivial items, which lowers the random-guess floor from 25% to 10%. Together, MATH500 and GPQA test structured reasoning; FaithEval and ConflictQA stress faithfulness and contradiction handling; NovelQA and HotpotQA evaluate long-context and multi-hop evidence integration; and MMLU-Pro measures broad reasoning robustness.

Benchmark	Capability probed	Split / subset	#Eval	Context scale	Answer format
MATH500	Symbolic reasoning	test	500	short problem statement	free-form
GPQA	Expert scientific knowledge	main	448	short question	4-way multiple choice
FaithEval	Contextual faithfulness	counterfactual	1,000	single passage	short-form extractive
ConflictQA	Evidence conflict resolution	popQA (gpt4)	9,544	two conflicting passages	short-form
NovelQA	Long-context comprehension	full	2,305	>200K tokens per novel	free-form (generative)
HotpotQA	Multi-hop retrieval	dev (fullwiki)	7,405	5.2M-passage corpus	short-form extractive
MMLU-Pro	Broad-domain reasoning	test	12,032	short question	10-way multiple choice
Total			**33,234**		

Table 4: Statistics of the seven evaluation benchmarks. Context scale refers to the evidence available at inference time rather than to the question length.

F Multi-Agent Frameworks

We briefly introduce the multi-agent frameworks used to evaluate **HEDGE** as a modular memory component. These frameworks differ in how agents are organized, communicate, and coordinate, providing diverse settings for assessing memory effectiveness.

AutoGen. AutoGen (Wu et al., 2024) is a flexible framework for building LLM-based multi-agent systems through structured conversations. Agents interact via message passing, enabling modular composition of roles such as planners, executors, and critics. Its general-purpose design makes it a strong baseline for evaluating plug-in components like memory modules.

DyLAN. DyLAN (Liu et al., 2024b) introduces a **dynamic agent network** where agents are selected and connected adaptively based on the task. Instead of using a fixed communication pattern, DyLAN learns to route information between agents, enabling more efficient and task-specific collaboration.

G-Designer. G-Designer (Zhang et al., 2025b) formulates multi-agent collaboration as a graph design problem. It leverages graph neural networks to **learn communication topologies** among agents, optimizing how information flows between them. This allows for structured and adaptive coordination beyond predefined interaction patterns.

MacNet. MacNet (Qian et al., 2025a) focuses on **scaling multi-agent collaboration** by organizing agents into structured communication and coordination protocols. It emphasizes efficiency and scalability, enabling large numbers of agents to collaborate effectively while maintaining coherent reasoning.

Discussion. These frameworks span a spectrum from flexible conversational systems (AutoGen) to dynamically structured (DyLAN), graph-optimized (G-Designer), and scalable coordinated architectures (MacNet). This diversity ensures that our evaluation of **HEDGE** is not tied to a specific interaction paradigm, but instead reflects its general applicability across different multi-agent settings.

G Experimental and Training Settings

We provide the full experimental configuration for HEDGE, including supervised pretraining (Stage I), reinforcement-learning refinement (Stage II), memory construction, and retrieval.

Models and compute. We use a temperature of 0.7 and Qwen3-Embedding-4B (Zhang et al., 2025c) to obtain normalized node, query, and hyperedge embeddings. Stage I takes approximately three hours, while Stage II requires approximately five hours.

Training data. Both stages use simulated debates generated from the HotpotQA FullWiki training split (Yang et al., 2018). Stage II continues training on HotpotQA rather than adapting MSM to any target evaluation benchmark.

Stage I configuration. We optimize the supervised routing objective using AdamW with learning rate 10^{-4}, batch size 64, weight decay 0.01, and dropout 0.1.

Stage II configuration. We refine the Stage I checkpoint using PPO over approximately 10K debate episodes. We use a learning rate of 10^{-5}, discount factor $\gamma = 0.99$, GAE parameter 0.95, clipping threshold $\epsilon = 0.2$, value-loss coefficient 0.5, entropy coefficient 0.01, batch size 64, and four optimization epochs per batch.

Reward configuration. The trajectory reward is

$$R = \lambda_{\text{task}} R_{\text{task}} + \lambda_{\text{route}} R_{\text{route}} + \lambda_{\text{eff}} R_{\text{eff}}, \quad (37)$$

with $\lambda_{\text{task}} = 1.0$, $\lambda_{\text{route}} = 0.5$, and $\lambda_{\text{eff}} = 0.3$. For the efficiency term, we set $\alpha_c = 0.1$, $\alpha_a = 0.05$, and $\alpha_f = 0.01$ to penalize the number of stored candidates, average memory size, and final memory size, respectively.

Memory construction and retrieval. We set $\tau_{\text{sig}} = 0.30$ and $\tau_{\text{emb}} = 0.70$ for node assignment, and $\tau^{\text{sig}}_{\text{merge}} = 0.50$ and $\tau^{\text{emb}}_{\text{merge}} = 0.85$ for hyperedge merging. Hyperedges with utility below 0.10 for three consecutive rounds are pruned, and backup nodes are retained for at most two rounds. During retrieval, we use $K = 3$ hyperedges and retrieve up to $M = 8$ nodes from each hyperedge. Memory is updated after every debate round.

H Qualitative Analysis

We examine the hypergraph memory constructed for a HotpotQA question: *"What government position was held by the woman who portrayed Corliss Archer in the film Kiss and Tell?"* Answering this question requires identifying the actress from the film and connecting her identity to a government position. Table 5 presents the corresponding memory structure, including retained claims, hyperedge memberships, and directed argumentative relations.

The memory organizes information into three hyperedges. H1 captures the film and cast information, where n_1 provides evidence for n_2, establishing that Shirley Temple portrayed Corliss Archer. H2 organizes information about her government career. Here, n_3 provides evidence for n_5 by establishing that Shirley Temple served as Chief of Protocol, while n_4 supports n_5 by establishing that Chief of Protocol is a government position. These relations preserve the evidential basis for the relevant intermediate claims.

H3 connects the two reasoning contexts through *overlapping node membership*. Specifically, n_2 belongs to both H1 and H3, while n_5 belongs to both H2 and H3. These are the same memory nodes participating in multiple hyperedges, rather than duplicated claims. Within H3, the final conclusion n_6 depends on both shared nodes, connecting the actress identified through the film to her government position. Importantly, these dependencies are represented as individual directed DEPENDS-ON relations. The higher-order structure arises from the shared hyperedge membership of multiple related claims, rather than from separate cross-hyperedge or n-ary argumentative relations.

The retained claims also demonstrate how the memory organizes information for subsequent retrieval. Nodes n_2 and n_5 preserve the two intermediate facts required to answer the question, while n_6 captures their combined conclusion. Retrieving H3 therefore provides access to both supporting claims and their argumentative dependencies without requiring the complete debate history. This organization supports the correct answer, *Chief of Protocol*, while avoiding the need to reconstruct the connection between the film and the government position from unstructured dialogue.

I Generalization of HEDGE on Different Multi-agent Frameworks

Table 6 extends the comparison to G-Designer (Zhang et al., 2025b) and MacNet (Qian et al., 2025a). HEDGE achieves the best result on all seven benchmarks under both frameworks, outperforming G-Memory + EquiMem by **3.5 points on average** in each case. Under G-Designer, gains range from **1.7** points on MATH500 to **5.5** on ConflictQA; under MacNet, they range from **2.0** to **5.4** points on the same tasks. The largest improvements occur on ConflictQA, NovelQA, and HotpotQA, where agents must preserve competing claims and multi-step evidence across rounds. Improvements on MATH500, GPQA, and MMLU-Pro further show that the benefits are not limited to conflict-heavy tasks. The similar gains and generally lower run-to-run variation across both frameworks reinforce that HEDGE is robust to different multi-agent orchestration designs.

J Detailed Results on Efficiency

Tables 7, 8, and 9 report the average reasoning rounds, total token usage, and memory footprint across all benchmarks.

Token accounting. The total token usage (T) in Tables 7, 8, and 9 includes debate interactions and auxiliary LLM calls for memory construction. Specifically, HEDGE's counts include GPT-4o-mini calls for atomic-fact and relation extraction; the G-Memory+EquiMem counts likewise include its atomic extraction calls. Auxiliary tokens are a subset of the reported totals, not an additional cost to be added. Embedding computation, memory maintenance, and MSM inference consume

Hyperedge	Nodes (Claims)	Connections
H1: Film and Cast	n_1: Kiss and Tell stars Shirley Temple as Corliss Archer n_2: Shirley Temple portrayed Corliss Archer in Kiss and Tell	$n_1 \xrightarrow{\text{EVIDENCE-FOR}} n_2$
H2: Government Position	n_3: Shirley Temple served as U.S. Chief of Protocol n_4: Chief of Protocol is a U.S. government position n_5: Shirley Temple held the government position of Chief of Protocol	$n_3 \xrightarrow{\text{EVIDENCE-FOR}} n_5$ $n_4 \xrightarrow{\text{SUPPORTS}} n_5$
H3: Actress–Government Role Linking	n_2: Shirley Temple portrayed Corliss Archer in Kiss and Tell n_5: Shirley Temple held the government position of Chief of Protocol n_6: The actress portraying Corliss Archer held the position of Chief of Protocol	$n_6 \xrightarrow{\text{DEPENDS-ON}} n_2$ $n_6 \xrightarrow{\text{DEPENDS-ON}} n_5$

Table 5: Hypergraph memory constructed by HEDGE. Nodes represent atomic claims, while typed directed relations encode evidential support and argumentative dependencies within each hyperedge. Nodes n_2 and n_5 belong to multiple hyperedges, connecting entity-specific reasoning with the final conclusion through overlapping membership.

no additional LLM tokens, but their overhead contributes to wall-clock latency. Table 10 reports these auxiliary costs and end-to-end latency for the two evaluated tasks with available timing measurements.

Auxiliary costs and wall-clock latency. On ConflictQA, HEDGE uses 8,746 auxiliary tokens, compared with 4,312 for G-Memory+EquiMem, yet reduces total tokens by 15.4%, memory footprint by 49.2%, and latency from 49.3 to 41.2 seconds (16.4%). On HotpotQA, it uses 9,381 auxiliary tokens, compared with 4,587, while reducing total tokens by 15.6%, memory by 49.3%, and latency from 51.1 to 43.5 seconds (14.9%). The additional relation-extraction cost is therefore outweighed by fewer debate rounds and more targeted memory retrieval in these settings.

Scalability and latency. Table 11 reports costs when varying the number of debating agents and the debate length. Increasing the agent count from three to six maintains similar accuracy (57.8% to 58.6%) and reduces the average number of debate rounds from 5.7 to 4.8, but increases total tokens from 36,891 to 57,962, memory tokens from 18,214 to 29,741, and wall-clock latency from 41.2 to 64.1 seconds. With three agents and the debate length increased from three to twelve rounds, accuracy rises from 56.1% to 58.3% and largely saturates by nine rounds; total tokens rise from 25,284 to 68,967, memory tokens from 13,472 to 29,314, and latency from 29.4 to 75.2 seconds. These results show that the accuracy gains stabilize while additional agents and longer debates impose increasing computational cost.

Non-memory systems. Computational cost increases with iterative and multi-agent reasoning. Vanilla GPT-4o-mini uses one round and averages 7.3K tokens, whereas advanced single-agent methods average 2.4–4.3 rounds and 13.5K–21.2K tokens. MAD and MoA further increase these averages to 5.9 and 6.9 rounds and 27.0K and 30.7K tokens, respectively. The highest non-memory cost occurs on HotpotQA, where MoA uses 8 rounds and 36,208 tokens. Thus, longer reasoning alone substantially expands the interaction context even without persistent memory.

Memory-based systems. Memory introduces an additional trade-off between retaining useful state and controlling context growth. Under AutoGen, MAD-M^2 has the smallest average token usage and memory footprint among memory baselines, at 27.7K and 6.4K tokens, respectively, while G-Memory averages 40.5K total tokens and 27.9K memory tokens. Adding EquiMem reduces G-Memory's average rounds from 8.3 to 7.5 and token usage from 40.5K to 39.0K, but increases its memory footprint to 31.1K. The same pattern appears under DyLAN: G-Memory + EquiMem lowers average rounds and total tokens from 9.3 and 45.1K to 8.6 and 43.4K, while increasing memory from 31.8K to 35.5K. Costs are generally higher under DyLAN than AutoGen, showing that memory overhead also depends on the underlying orchestration framework.

HEDGE versus structured memory baselines. HEDGE achieves the fewest rounds among all memory-based methods on every benchmark. Under AutoGen, it averages 4.9 rounds, 32.8K total tokens, and 16.1K memory tokens. Relative to G-Memory + EquiMem, this corresponds to reductions of **34.5%** in rounds, **15.9%** in token usage, and **48.1%** in memory footprint. Under DyLAN, HEDGE averages 5.1 rounds, 36.2K total tokens, and 18.5K memory tokens, reducing

Memory	MATH500	GPQA	FaithEval	ConflictQA	NovelQA	HotpotQA	MMLU-Pro
G-Designer 2025b							
No-memory	$55.3_{\pm 0.5}$	$38.6_{\pm 0.6}$	$67.5_{\pm 0.4}$	$36.1_{\pm 0.7}$	$55.8_{\pm 0.5}$	$32.6_{\pm 0.4}$	$74.2_{\pm 0.3}$
Voyager	$57.0_{\pm 0.4}$	$40.4_{\pm 0.5}$	$69.3_{\pm 0.6}$	$38.2_{\pm 0.7}$	$57.6_{\pm 0.4}$	$33.9_{\pm 0.5}$	$75.9_{\pm 0.3}$
Memory Bank	$54.8_{\pm 0.6}$	$37.9_{\pm 0.7}$	$66.9_{\pm 0.5}$	$36.8_{\pm 0.6}$	$56.5_{\pm 0.6}$	$33.1_{\pm 0.4}$	$74.6_{\pm 0.4}$
Generative	$57.3_{\pm 0.5}$	$41.1_{\pm 0.6}$	$70.1_{\pm 0.4}$	$38.9_{\pm 0.8}$	$58.2_{\pm 0.5}$	$34.3_{\pm 0.6}$	$76.4_{\pm 0.3}$
MetaGPT-M	$57.7_{\pm 0.4}$	$41.8_{\pm 0.5}$	$70.7_{\pm 0.5}$	$39.6_{\pm 0.6}$	$58.8_{\pm 0.4}$	$34.8_{\pm 0.5}$	$77.0_{\pm 0.3}$
MacNet-M	$56.0_{\pm 0.5}$	$40.0_{\pm 0.6}$	$68.8_{\pm 0.4}$	$37.9_{\pm 0.7}$	$57.2_{\pm 0.6}$	$33.8_{\pm 0.5}$	$75.6_{\pm 0.4}$
ChatDev-M	$54.0_{\pm 0.6}$	$37.3_{\pm 0.7}$	$66.2_{\pm 0.5}$	$35.2_{\pm 0.8}$	$55.6_{\pm 0.5}$	$32.3_{\pm 0.6}$	$73.6_{\pm 0.4}$
MAD-M^2	$58.2_{\pm 0.5}$	$42.2_{\pm 0.4}$	$70.9_{\pm 0.5}$	$40.8_{\pm 0.7}$	$59.2_{\pm 0.4}$	$35.3_{\pm 0.5}$	$77.6_{\pm 0.3}$
MeMAD	$58.6_{\pm 0.4}$	$42.6_{\pm 0.5}$	$71.1_{\pm 0.4}$	$41.5_{\pm 0.6}$	$59.5_{\pm 0.5}$	$35.6_{\pm 0.4}$	$78.0_{\pm 0.3}$
G-Memory	$59.5_{\pm 0.4}$	$44.0_{\pm 0.5}$	$74.8_{\pm 0.4}$	$47.5_{\pm 0.6}$	$59.9_{\pm 0.5}$	$36.0_{\pm 0.4}$	$79.3_{\pm 0.3}$
G-Memory+EquiMem	$60.2_{\pm 0.3}$	$45.0_{\pm 0.4}$	$76.1_{\pm 0.4}$	$49.0_{\pm 0.5}$	$60.8_{\pm 0.4}$	$37.1_{\pm 0.3}$	$80.0_{\pm 0.3}$
HEDGE	$\mathbf{61.9}_{\pm 0.2}$	$\mathbf{48.1}_{\pm 0.3}$	$\mathbf{79.3}_{\pm 0.2}$	$\mathbf{54.5}_{\pm 0.3}$	$\mathbf{65.1}_{\pm 0.2}$	$\mathbf{41.0}_{\pm 0.3}$	$\mathbf{83.1}_{\pm 0.2}$
MacNet 2025a							
No-memory	$53.0_{\pm 0.6}$	$37.0_{\pm 0.7}$	$67.1_{\pm 0.5}$	$35.3_{\pm 0.8}$	$55.6_{\pm 0.6}$	$32.4_{\pm 0.5}$	$72.7_{\pm 0.4}$
Voyager	$54.8_{\pm 0.5}$	$38.8_{\pm 0.6}$	$69.0_{\pm 0.5}$	$37.4_{\pm 0.7}$	$57.3_{\pm 0.5}$	$33.8_{\pm 0.6}$	$74.3_{\pm 0.3}$
Memory Bank	$52.4_{\pm 0.7}$	$36.3_{\pm 0.6}$	$66.4_{\pm 0.6}$	$35.9_{\pm 0.8}$	$56.3_{\pm 0.5}$	$33.2_{\pm 0.5}$	$73.3_{\pm 0.4}$
Generative	$55.1_{\pm 0.5}$	$39.6_{\pm 0.7}$	$69.8_{\pm 0.4}$	$38.1_{\pm 0.6}$	$58.0_{\pm 0.6}$	$34.2_{\pm 0.5}$	$74.8_{\pm 0.4}$
MetaGPT-M	$55.6_{\pm 0.4}$	$40.2_{\pm 0.6}$	$70.4_{\pm 0.5}$	$38.8_{\pm 0.7}$	$58.6_{\pm 0.5}$	$34.7_{\pm 0.4}$	$75.5_{\pm 0.3}$
MacNet-M	$54.0_{\pm 0.6}$	$38.6_{\pm 0.5}$	$68.6_{\pm 0.6}$	$37.2_{\pm 0.8}$	$57.0_{\pm 0.5}$	$33.7_{\pm 0.6}$	$74.1_{\pm 0.4}$
ChatDev-M	$51.9_{\pm 0.7}$	$35.9_{\pm 0.6}$	$66.0_{\pm 0.5}$	$34.5_{\pm 0.7}$	$55.4_{\pm 0.6}$	$32.2_{\pm 0.5}$	$72.2_{\pm 0.4}$
MAD-M^2	$56.0_{\pm 0.4}$	$40.5_{\pm 0.5}$	$70.5_{\pm 0.5}$	$39.8_{\pm 0.6}$	$58.8_{\pm 0.4}$	$35.0_{\pm 0.5}$	$76.2_{\pm 0.3}$
MeMAD	$56.4_{\pm 0.5}$	$41.0_{\pm 0.6}$	$70.8_{\pm 0.4}$	$40.5_{\pm 0.7}$	$59.2_{\pm 0.5}$	$35.4_{\pm 0.4}$	$76.5_{\pm 0.3}$
G-Memory	$57.3_{\pm 0.4}$	$42.5_{\pm 0.5}$	$74.1_{\pm 0.5}$	$46.4_{\pm 0.6}$	$59.6_{\pm 0.4}$	$35.9_{\pm 0.5}$	$77.8_{\pm 0.3}$
G-Memory+EquiMem	$58.0_{\pm 0.4}$	$43.3_{\pm 0.4}$	$75.4_{\pm 0.4}$	$47.8_{\pm 0.5}$	$60.4_{\pm 0.4}$	$36.8_{\pm 0.4}$	$78.5_{\pm 0.3}$
HEDGE	$\mathbf{60.0}_{\pm 0.2}$	$\mathbf{46.0}_{\pm 0.3}$	$\mathbf{79.1}_{\pm 0.2}$	$\mathbf{53.2}_{\pm 0.3}$	$\mathbf{64.0}_{\pm 0.3}$	$\mathbf{41.1}_{\pm 0.2}$	$\mathbf{81.4}_{\pm 0.2}$

Table 6: Additional results under the G-Designer and MacNet multi-agent frameworks. Values report the mean over three runs, with red subscripts indicating the standard deviation. EquiMem consistently improves G-Memory, while HEDGE achieves the strongest performance across all benchmarks with lower run-to-run variability.

the same metrics by **39.9%**, **16.6%**, and **47.8%**, respectively. For example, on HotpotQA under DyLAN, HEDGE uses 5.7 rounds, 41,762 tokens, and 22,631 memory tokens, compared with 9.7 rounds, 50,219 tokens, and 43,816 memory tokens for G-Memory + EquiMem.

Trade-off with lightweight masking. MAD-M^2 remains one of the most storage-efficient methods because its perplexity-based variant masks prior content rather than maintaining a richer structured state. Accordingly, HEDGE uses 18.3% more total tokens than MAD-M^2 under AutoGen and 6.4% more under DyLAN. However, it requires **17.4%** and **32.7% fewer rounds**, respectively, while providing substantially stronger task performance. Overall, the results show that HEDGE does not merely minimize storage; it selectively maintains sufficient structured context to improve reasoning while preventing the uncontrolled growth observed in stronger graph-memory baselines.

K Sensitivity to the Embedding Model

Table 12 evaluates HEDGE with six embedding models while keeping all other components fixed. Performance generally improves with embedding quality, but the overall trends remain stable across models. Compared with text-embedding-ada-002, the default Qwen3-Embedding-4B improves accuracy by **1.9–3.4 points**, with the largest gains on FaithEval and ConflictQA, where accurate semantic grouping is particularly important for separating competing claims and evidence.

Scaling from Qwen3-Embedding-4B to Qwen3-Embedding-8B yields only **0.2–0.3 points** of additional accuracy across benchmarks. We therefore use the 4B model as the default because it provides a better performance–computation trade-off while remaining close to the strongest embedding model. These results indicate that HEDGE benefits from stronger semantic representations but is not dependent on a particular embedding backbone.

Method	MATH500		GPQA		FaithEval		ConflictQA		NovelQA		HotpotQA		MMLU-Pro	
	R	T	R	T	R	T	R	T	R	T	R	T	R	T
Single-Agent (Vanilla)														
GPT-4o-mini	1	8566	1	7312	1	5489	1	8235	1	5791	1	9147	1	6623
Single-Agent (Advanced) (GPT-4o-mini)														
Self-Refine	2	14273	2	13158	2	9276	3	15342	3	14895	3	16211	2	11367
Self-Consistency	4	22491	4	20358	3	14872	5	24137	5	23705	5	25164	4	17628
Reflexion	3	18342	3	17486	3	12657	4	19372	4	19108	4	20476	3	14392
Multi-Agent (GPT-4o-mini)														
MAD	5	28417	6	26783	4	18954	7	30126	7	29688	7	32451	5	22379
MoA	6	32714	7	30892	5	21865	8	34127	8	33654	8	36208	6	25491

Table 7: Average number of reasoning rounds (R) and total token usage (T) for **non-memory** systems built on **GPT-4o-mini**. Methods are grouped into single-agent, advanced single-agent, and multi-agent settings.

L Sensitivity to Retrieval Breadth

We study the effect of the number of retrieved hyperedges K and the maximum number of nodes retrieved from each hyperedge M. Table 13 varies M while fixing $K = 3$. Performance improves consistently from $M = 2$ to $M = 8$, with particularly large gains on ConflictQA (**+10.2 points**) and NovelQA (**+8.7 points**). Retrieving too few nodes therefore omits relevant evidence and argumentative links. Performance declines slightly at $M = 12$ and more noticeably at $M = 16$, suggesting that excessive within-hyperedge retrieval introduces redundant or weakly relevant content. We therefore use $M = 8$.

Table 14 varies K while fixing $M = 8$. Increasing K from 1 to 3 improves all four benchmarks, indicating that reasoning often requires evidence distributed across multiple semantic groups. However, retrieving four or five hyperedges reduces performance, likely because less relevant groups introduce distracting context. The effect is strongest on ConflictQA and NovelQA, where preserving multiple related perspectives is useful but excessive retrieval can obscure the relevant conflicts. Overall, $K = 3$ and $M = 8$ provide the best balance between information coverage and retrieval noise, and are used as the default configuration.

M Sensitivity to the Relation Extraction Model

Table 15 evaluates the sensitivity of HEDGE to the LLM used for predicting intra-hyperedge relations while keeping all other components fixed. Performance varies by at most **0.6 points** across the evaluated models, indicating that HEDGE is not strongly dependent on a particular relation extractor. Larger closed-source models provide small gains, particularly on ConflictQA, where distinguishing support and refutation relations is more important. However, GPT-4o-mini remains within **0.1–0.4 points** of the strongest model across all benchmarks, providing a substantially more economical default with minimal performance loss.

N Effect of Signature-Based Hyperedge Matching

We evaluate whether sparse semantic signatures provide useful information beyond dense embedding similarity. The default configuration uses both signals: Jaccard similarity over signatures first identifies lexically compatible hyperedges, after which dense similarity measures semantic compatibility. The ablated configuration removes the signature gate and assigns nodes using embeddings alone.

Table 16 shows that combining sparse signatures with dense embeddings consistently improves performance. Removing the Jaccard signature gate reduces accuracy across all benchmarks, with the largest drops on MMLU-Pro, NovelQA, and FaithEval. Sparse signatures retain exact lexical cues such as entities, numerical values, and salient content terms that dense representations may blur, making them particularly useful for separating closely related or conflicting claims.

Embedding-only matching also produces fewer hyperedges. This suggests that dense similarity tends to merge semantically related but argumentatively distinct items into broad groups. The signature gate prevents such over-merging, produc-

Method	MATH500			GPQA			FaithEval		
	R	T	M	R	T	M	R	T	M
ChatDev-M	5.2	30185	6124	6.1	28462	5783	4.3	20137	4215
MacNet-M	6.3	33982	8142	6.8	32174	7685	5.1	22648	5562
MetaGPT-M	6.1	36125	10284	7.3	34418	9681	5.4	24102	7025
MAD-M^2	5.4	29164	6487	5.8	27546	6038	4.2	19482	4476
MeMAD	7.2	38261	18472	7.8	36847	17365	6.1	25973	12498
G-Memory	7.4	42673	28714	8.2	40921	26903	6.3	28954	18456
G-Memory+EquiMem	6.7	40984	31926	7.5	39368	29875	5.7	27842	20583
HEDGE	**4.8**	34712	16483	**4.2**	32985	15372	**3.9**	22468	10895

Method	ConflictQA			NovelQA			HotpotQA			MMLU-Pro		
	R	T	M	R	T	M	R	T	M	R	T	M
ChatDev-M	8.2	32214	6982	7.1	31785	7126	7.3	33942	7451	5.2	23618	5087
MacNet-M	8.8	36521	9384	8.2	35894	9621	8.1	37712	10173	6.2	27365	6941
MetaGPT-M	9.6	38954	11873	8.4	38211	12145	8.5	40137	12988	6.4	29108	8842
MAD-M^2	7.3	30985	7324	6.8	30416	7463	7.1	33278	7835	5.3	22964	5376
MeMAD	9.8	41256	20311	9.2	40538	20954	9.1	42814	22417	7.1	30965	15386
G-Memory	10.1	45318	32175	9.4	44782	31492	9.3	46811	34788	7.3	33867	22741
G-Memory+EquiMem	9.2	43615	35864	8.6	43206	35091	8.5	45187	38742	6.6	32641	25368
HEDGE	**5.7**	36891	18214	**5.6**	36277	18763	**5.8**	38145	19628	**4.6**	27854	13491

Table 8: Average reasoning rounds (R), average total token usage (T), and average memory footprint (M) for memory-based multi-agent systems implemented with **AutoGen** using **GPT-4o-mini**.

ing a more fine-grained memory in which distinct claims and evidence chains remain separable. Thus, sparse and dense matching provide complementary signals: embeddings capture semantic relatedness, while signatures preserve precise lexical distinctions needed for reliable hypergraph construction.

O Prompts

We provide all prompts used by HEDGE and for benchmark evaluation to support reproducibility. The method-specific prompts define the construction of the debate roster and structured memory, while the benchmark prompts follow each dataset's task format and answer requirements.

O.1 Prompts Used by HEDGE

HEDGE uses three prompts during debate and memory construction. The roster-generation prompt designs a team of three to six agents with complementary roles, model sizes, and permitted debate actions (Prompt O.3). Agent responses are then decomposed into self-contained atomic claims using Prompt O.2. For each pair of claims within a hyperedge, Prompt O.1 predicts a directed argumentative relation—SUPPORTS, REFUTES, EVIDENCE-FOR, DEPENDS-ON, or LEADS-TO—or determines that no relation exists. Together, these prompts define the agent configuration, memory nodes, and intra-hyperedge relational structure used by HEDGE.

O.2 Benchmark Evaluation Prompts

The evaluation prompts are adapted to the structure and output format of each benchmark. MATH500 and GPQA request concise but precise reasoning, with final answers returned in a standardized format (Prompts O.4 and O.5). MMLU-Pro similarly constrains the final output to one of its predefined answer choices (Prompt O.7). ConflictQA uses a strict entity-extraction format that permits only the target entity in the response (Prompt O.8).

For context-intensive tasks, HotpotQA and NovelQA instruct agents to integrate evidence across the provided context while avoiding unnecessary explanation (Prompts O.10 and O.9). For FaithEval, agents receive benchmark-provided counterfactual contexts and are instructed to answer questions using only the supplied information, even when it contradicts their parametric knowledge (Prompt O.6). No gold answers are exposed during evaluation. These task-specific prompts preserve each benchmark's evaluation requirements while maintaining

Method	MATH500			GPQA			FaithEval		
	R	T	M	R	T	M	R	T	M
ChatDev-M	7.2	37210	11245	8.1	35590	10384	6.2	24830	7412
MacNet-M	7.5	39185	13672	8.4	37421	12755	6.5	26194	9156
MetaGPT-M	8.3	41562	16841	9.2	39873	15892	7.1	27945	11237
MAD-M^2	6.9	35984	11736	7.8	34327	10942	5.9	23916	7798
MeMAD	8.1	43218	21495	9.0	41576	20184	7.3	29163	14122
G-Memory	8.5	47294	32871	9.4	45618	30955	7.6	32107	20944
G-Memory+EquiMem	7.8	45481	36692	8.7	43862	34531	6.9	30916	23384
HEDGE	**4.7**	38975	18642	**5.6**	37214	17530	**4.4**	25196	12158

Method	ConflictQA			NovelQA			HotpotQA			MMLU-Pro		
	R	T	M	R	T	M	R	T	M	R	T	M
ChatDev-M	9.2	39584	12845	8.9	38872	13211	9.3	40963	13894	7.2	29541	9123
MacNet-M	9.5	41827	15873	9.2	41055	16324	9.6	43211	17095	7.5	31284	11842
MetaGPT-M	10.2	44591	19218	9.8	43802	19871	10.1	46137	20744	8.1	33315	14392
MAD-M^2	8.7	38126	13472	8.5	37365	13796	8.9	39742	14538	6.8	28617	9564
MeMAD	10.0	46275	23891	9.7	45482	24563	10.3	47894	25817	8.3	34876	17845
G-Memory	10.4	50538	36472	10.1	49821	35890	10.5	52147	39284	8.6	37952	26108
G-Memory+EquiMem	9.6	48642	40685	9.3	48036	40072	9.7	50219	43816	7.9	36528	29174
HEDGE	**5.8**	40218	21184	**5.0**	39475	21763	**5.7**	41762	22631	**4.8**	30485	15627

Table 9: Average reasoning rounds (R), average total token usage (T), and average memory footprint (M) for memory-based multi-agent systems implemented with **DyLAN** using **GPT-4o-mini**.

Benchmark	Method	R	Atomic calls	Relation calls	Aux. tokens	Total tokens	Memory tokens	Latency (s)
ConflictQA	G-Memory+EquiMem	9.2	27.6	0	4,312	43,615	35,864	49.3
ConflictQA	**HEDGE**	**5.7**	17.1	26.8	8,746	**36,891**	**18,214**	**41.2**
HotpotQA	G-Memory+EquiMem	8.5	25.5	0	4,587	45,187	38,742	51.1
HotpotQA	**HEDGE**	**5.8**	17.4	29.6	9,381	**38,145**	**19,628**	**43.5**

Table 10: End-to-end efficiency comparison under AutoGen with GPT-4o-mini on ConflictQA and HotpotQA. Values are per-question averages. R denotes debate rounds; atomic and relation calls count auxiliary LLM invocations; auxiliary tokens include atomic-fact and relation-extraction usage. Total tokens include auxiliary LLM calls, while memory tokens measure the persistent memory footprint. Latency is end-to-end wall-clock time.

consistent, concise interaction across methods.

Scaling factor	Setting	Accuracy (%)	Total tokens	Memory tokens	Relation calls	Latency (s)
Agent count	3 agents	57.8	36,891	18,214	26.8	41.2
Agent count	4 agents	58.5	43,276	21,637	37.1	47.9
Agent count	5 agents	58.9	50,418	25,483	49.6	55.8
Agent count	6 agents	58.6	57,962	29,741	63.4	64.1
Debate length	3 rounds	56.1	25,284	13,472	16.3	29.4
Debate length	6 rounds	57.9	37,426	18,608	28.7	42.3
Debate length	9 rounds	58.4	53,218	24,576	43.5	58.7
Debate length	12 rounds	58.3	68,967	29,314	57.8	75.2

Table 11: Scaling analysis of HEDGE on ConflictQA under AutoGen with GPT-4o-mini. The agent-count sweep uses the default debate-stopping procedure; the debate-length sweep keeps three agents fixed. All token, memory, call, and latency measurements are averages per question.

Embedding Model	MATH500	GPQA	FaithEval	ConflictQA	NovelQA	HotpotQA	MMLU-Pro
text-embedding-ada-002	60.3	45.2	76.5	52.6	63.7	39.9	80.2
text-embedding-3-small	61.0	46.1	77.6	54.1	64.6	40.8	81.3
text-embedding-3-large	61.6	46.9	78.7	55.4	65.4	41.7	82.1
Qwen3-Embedding-0.6B	60.8	45.8	78.2	54.6	65.0	41.2	81.5
Qwen3-Embedding-4B	62.2	47.7	79.8	56.0	66.0	42.4	82.7
Qwen3-Embedding-8B	**62.4**	**47.9**	**80.1**	**56.3**	**66.2**	**42.6**	**83.0**

Table 12: Sensitivity of HEDGE to the embedding model. Performance remains stable across models, with only modest gains from increasing embedding-model scale. Qwen3-Embedding-4B is the default.

M	MATH500	GPQA	ConflictQA	NovelQA
2	57.8	41.9	47.6	58.5
4	59.2	43.9	50.6	61.9
8	**62.9**	**48.3**	**57.8**	**67.2**
12	62.6	47.9	57.1	66.7
16	61.9	47.1	55.8	65.6

Table 13: Effect of the number of retrieved nodes per group M on HEDGE with GPT-4o-mini, keeping the number of retrieved groups fixed at $K = 3$.

K	MATH500	GPQA	ConflictQA	NovelQA
1	60.8	45.9	53.1	63.8
2	62.1	47.2	56.0	65.9
3	**62.9**	**48.3**	**57.8**	**67.2**
4	62.5	47.8	57.0	66.5
5	61.7	46.9	55.4	65.1

Table 14: Effect of the number of retrieved groups k on HEDGE with GPT-4o-mini, keeping the number of retrieved nodes per group fixed at $M = 8$.

Relation Extraction Model	MATH500	GPQA	FaithEval	ConflictQA	NovelQA	HotpotQA	MMLU-Pro
Qwen3-32B (Yang et al., 2025)	62.1	47.6	79.7	55.8	65.9	42.3	82.6
GPT-4o-mini	62.2	47.7	79.8	56.0	66.0	42.4	82.7
GPT-4o	62.3	47.8	79.9	56.2	66.1	42.5	82.8
GPT-5.4	62.4	47.9	80.0	56.3	66.2	42.6	82.9
Claude-Sonnet-4.6 (Anthropic, 2026)	**62.4**	**47.9**	**80.1**	**56.4**	**66.2**	**42.7**	**83.0**

Table 15: Sensitivity of HEDGE to the relation extraction model. Performance remains stable across models, with only marginal gains from larger relation extractors. GPT-4o-mini is used by default.

Benchmark	Accuracy		Avg. Number of Hyperedges	
	Signature + Embedding	Embedding Only	Signature + Embedding	Embedding Only
MATH500	**62.2**	60.8	5.2	3.8
GPQA	**47.7**	46.0	5.7	4.1
FaithEval	**79.8**	77.3	6.5	4.6
ConflictQA	**56.0**	54.2	7.3	5.1
NovelQA	**66.0**	63.2	6.9	4.8
HotpotQA	**42.4**	40.5	5.8	3.9
MMLU-Pro	**82.7**	78.9	4.6	3.4

Table 16: Effect of combining sparse signatures with dense embeddings for hyperedge matching. The embedding-only variant removes the Jaccard signature gate.

Prompt O.1: Relation Extraction

Given two claims from an ongoing multi-agent debate, identify the directed relation from node A to node B.
Choose exactly one relation:

- SUPPORTS: A strengthens B.
- REFUTES: A contradicts or weakens B.
- EVIDENCE-FOR: A directly provides evidence for B.
- DEPENDS-ON: A logically requires B.
- LEADS-TO: A causes or implies B.
- NO-RELATION: no clear relation exists.

Use only information explicitly stated in the claims. When the relation is ambiguous or weak, return NO-RELATION. The direction is always from A to B.

Input

```
{
  "node_A": {
    "id": "<node_id>",
    "text": "<claim text>"
  },
  "node_B": {
    "id": "<node_id>",
    "text": "<claim text>"
  }
}
```

Output

```
{
  "node_A_id": "<node_id>",
  "node_B_id": "<node_id>",
  "relation": "<relation type>",
  "confidence": "<high | medium | low>"
}
```

Return only the JSON object.

<input_nodes>
{{input_nodes}}
</input_nodes>

Prompt O.2: Atomic Fact Extraction

```
You are given a passage from an ongoing multi-agent debate. Extract all atomic facts as the smallest
independently verifiable claims.
Each fact must be self-contained, express one claim, remain faithful to the passage, and be concise.
Remove greetings, filler, opinions, uncertainty markers, and agent meta-commentary. Do not use
unresolved pronouns or combine multiple claims.
Return only JSON in the following format:
{
  "atomic_facts": [
    {
      "id": "f1",
      "text": "Atomic fact."
    }
  ]
}
Assign unique IDs as f1, f2, and so on.
<input_passage>
{{passage}}
</input_passage>
```

Prompt O.3: Roster Generation

```
You are a multi-agent debate orchestrator. Design an optimal team of AI agents to collaboratively
answer a complex multi-hop question through debate.
Output only valid JSON with the following structure:
{
  "agents": [
    {
      "name": "<unique agent name>",
      "role": "<brief role description>",
      "goals": ["<goal 1>", "<goal 2>"],
      "preferred_actions": ["<action 1>", "<action 2>"],
      "avoid": ["<behavior to avoid>"],
      "model_size": "<4b or 8b>"
    }
  ],
  "turn_order": ["<agent1>", "<agent2>", "..."],
  "notes": "<system-level guidance>"
}
```

Constraints

- Create 3–6 agents, each with a unique name.
- Include every agent exactly once in turn_order.
- Use 8b for complex reasoning roles and 4b for supporting roles.
- Allowed actions are Propose, Support, Refute, Question, Revise, Summarize, and Retrieve.

Design principles

- Include at least one proposer, one critic, and one synthesizer.
- Balance evidence gathering, inference, and verification.
- Select model sizes according to the complexity of each role.

Return only the JSON object without explanation.

Question

{{question}}

Context preview

{{context_preview}}

Design the optimal team for answering this HotpotQA question through multi-agent debate.

Prompt O.4: MATH500 Prompt

```
Can you solve the following math problem as accurately as possible?
<question>{{question}}</question>
Provide a concise solution using only essential reasoning steps. Avoid unnecessary explanations.
Return the final answer in double parentheses at the end of your response:
((final answer))
```

Prompt O.5: GPQA Prompt

```
Please analyze the following multiple-choice question and select the correct answer.
<question>{{question}}</question>
Use concise reasoning with only essential steps.
Return your final answer in double parentheses:
((answer))
where answer is one of A, B, C, or D.
```

Prompt O.6: FaithEval Prompt

```
You are an expert in retrieval-based question answering. Please respond with the exact answer, using
only the information provided in the context.
Context:
{{context}}
Question:
{{question}}
```

Prompt O.7: MMLU-Pro Prompt

```
Please analyze the following {{topic}} question and select the correct answer.
<question>{{question}}</question>
Use concise reasoning with only essential steps.
Return your final answer in double parentheses:
((answer))
where answer is one of A, B, C, D, E, F, G, H, I, or J.
```

Prompt O.8: ConflictQA Prompt

```
You are a precise knowledge engine.
Your task is to answer the question using only the provided context.
Rules:
1. Output ONLY the entity name.
2. Do NOT use full sentences.
3. Do NOT provide explanations.
4. Be concise and exact.
Context:{{context}}
Question: {{question}}
```

Prompt O.9: NovelQA Prompt

```
Answer the following question based on the provided context.
<context>
{{context}}
</context>
<question>
{{question}}
</question>
Provide a concise answer using only the necessary reasoning steps. Avoid unnecessary details.
Return the final answer clearly at the end of your response.
```

Prompt O.10: HotpotQA Prompt

```
Answer the following multi-hop question using the provided context.
<context>
{{context}}
</context>
<question>
{{question}}
</question>
Use concise reasoning to combine evidence from multiple parts of the context.
Return the final answer at the end of your response.
```