LATENTHARNESS: ADAPTIVE LATENT CONTEXT MANAGEMENT FOR MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent communication enables LLM agents in multi-agent systems (MAS) to collaborate by exchanging continuous representations. Methods for organizing and delivering these representations fall into two categories: (1) Training-based methods learn message compression or information allocation but require additional training data and parameter optimization. (2) Training-free methods often deliver complete messages or select representations from predefined positions, limiting adaptation to receiver needs. To address this limitation, we introduce `LatentHarness`, a training-free framework for adaptive latent context management with two components: i) *state selection*, which adaptively selects hidden-state positions across upstream responses by balancing relevance to the question and receiver role against redundancy; and ii) *context construction*, which assembles short spans of consecutive hidden states ending at these positions in upstream response and token order. Experiments on four scientific benchmarks with three LLMs show accuracy gains in most settings. On GPQA Diamond, `LatentHarness` improves Qwen3.5-4B accuracy from 63.77% to 71.74% while reducing textual token volume by 75.57% relative to passing all preceding responses as text. The gains also extend to hierarchical MAS.

1 INTRODUCTION

Multi-agent systems (MAS) can improve the reasoning of large language models (LLMs) by allowing agents to compare proposed solutions and correct errors through mutual feedback (Du et al., 2024; Liu et al., 2025; Gu et al., 2026). Typical MAS organize problem solving through distinct roles and role interactions (Hong et al., 2024; Wan et al., 2025). Role instructions specify what each agent should do, but its ability to carry out that role also depends on the upstream information it receives. Yet exchanging more information does not necessarily improve collaboration: prior work shows that some communication can be removed while maintaining comparable task performance (Zhang et al., 2025; Wang et al., 2025; Zhang et al., 2026). Context management therefore complements role assignment: it determines which upstream information enters each receiver's input and how that information is organized.

In text-based MAS, upstream agents communicate through discrete token sequences, which receivers incorporate into their context (Hong et al., 2024; Wan et al., 2025). However, discrete token communication may lose information during token selection (Pham et al., 2024; Peng et al., 2026). Recent work therefore explores latent communication in MAS, where agents exchange continuous representations to support downstream reasoning. Existing approaches use mixtures of vocabulary embeddings (Pham et al., 2024), key-value (KV) caches (Zou et al., 2026; Shi et al., 2026), or hidden states (Ramesh & Li, 2025; Du et al., 2026; Peng et al., 2026) for inter-agent communication. These advances broaden the forms of communication in MAS, but how to organize the exchanged information for each receiver warrants further investigation.

Existing latent MAS methods can be broadly categorized into two groups: training-based methods and training-free methods. The former learn to compress messages or allocate information, but require additional training data and parameter optimization (Zheng et al., 2025; Du et al., 2026). In contrast, training-free methods avoid optimization by directly delivering continuous representations. However, they typically rely on **fixed communication strategies**, such as transmitting complete latent messages or accumulated working memory (Pham et al., 2024; Zou et al., 2026), or selecting representations

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

from predefined token positions or model layers (Ramesh & Li, 2025; Peng et al., 2026; Shi et al., 2026). Such strategies either introduce redundant information and unnecessary communication overhead, or limit the flexibility of delivered context to different receivers' requirements. This motivates the following question: *Can we dynamically select and organize upstream hidden states according to each receiver's needs without additional training?*

To answer this question, we propose `LatentHarness`, a training-free framework for adaptive latent context management in MAS, comprising two components: **(1) State selection.** To adapt the selected content to each receiver, we score candidate hidden states using the current question and role instructions, combining hidden-space similarity with vocabulary-space token overlap. We iteratively select positions by balancing relevance against redundancy with already selected states. Rather than requiring a fixed number of positions, we adjust the number selected through stopping criteria based on marginal gains and relevance coverage. **(2) Context construction.** To preserve local context, we extract short spans of consecutive hidden states ending at the selected positions. We merge overlapping spans and assemble them, preserving upstream response and token order and adding role labels to identify their sources. Together, these components determine which upstream information each receiver obtains and how it is organized.

We evaluate `LatentHarness` on four scientific benchmarks using three LLMs. Compared with MAS that pass all preceding responses as text, `LatentHarness` improves Qwen3.5-4B accuracy on GPQA Diamond from 63.77% to 71.74% while reducing textual token by 75.57%. On BioProBench, Gemma-4-12B with `LatentHarness` achieves 62.98% accuracy, compared with 50.95% when passing all hidden states from the immediately preceding response and 43.81% when passing only its final states. The gains also extend to hierarchical MAS, where a final agent combines responses from multiple experts that solve the question independently. In this setting, `LatentHarness` improves Qwen3.5-9B accuracy on GPQA Diamond from 74.64% to 76.09% while reducing textual token volume by 74.28% relative to passing all expert responses as text.

Our main contributions are as follows:

- We formulate latent context management in MAS as the problem of dynamically selecting and organizing upstream hidden states according to the current question and receiver role.
- We propose `LatentHarness`, a training-free framework that combines state selection and context construction over multiple upstream responses. It selects hidden-state positions by balancing relevance against redundancy, then constructs consecutive hidden-state spans ending at those positions and assembles them in upstream response and token order under an input budget.
- We evaluate `LatentHarness` on four scientific benchmarks using three LLMs. Results show improved accuracy in most settings and reduced textual token volume.

2 Related Work

Communication in MAS. Text-based MAS coordinate specialized roles through programmable conversations and standardized workflows (Wu et al., 2024; Hong et al., 2024) and refine responses through multi-agent debate (Du et al., 2024; Chen et al., 2024; Liu et al., 2025; Gu et al., 2026). Latent communication instead exchanges continuous representations, including probability-weighted vocabulary embedding mixtures (Pham et al., 2024), KV caches (Zou et al., 2026; Shi et al., 2026; Fu et al., 2026), and hidden states (Ramesh & Li, 2025; Du et al., 2026; Peng et al., 2026).

Context management in MAS. Methods requiring additional training learn to generate concise messages (Chen et al., 2025b) or allocate information across agents and model layers (Zheng et al., 2025; Fu et al., 2026). Interlat trains a reasoning model to generate compact latent messages while keeping the receiver and its communication adapter frozen (Du et al., 2026). Training-free methods deliver complete continuous messages or accumulated working memory (Pham et al., 2024; Zou et al., 2026). Others restrict delivery through non-learned final-token activation fusion or message-suffix extraction (Ramesh & Li, 2025; Peng et al., 2026), or select KV layers through attention-based calibration (Shi et al., 2026). Further work combines attention-based KV eviction with residual compensation (Li et al., 2026). Without additional training, `LatentHarness` selects upstream hidden states using the current question and receiver role, then constructs ordered spans of consecutive states under an input budget.

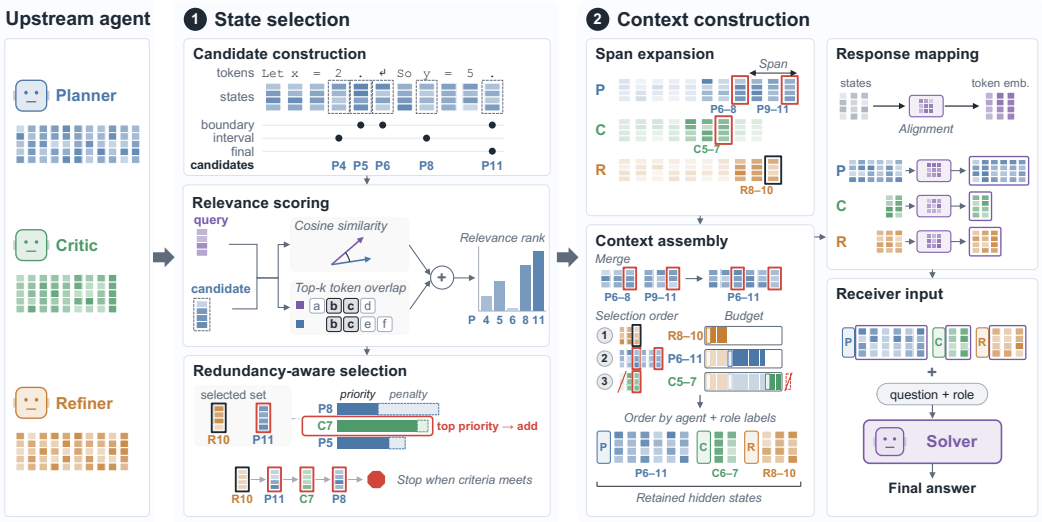

Figure 1: Overview of `LatentHarness` for adaptive latent context management in MAS. (1) State selection identifies relevant hidden-state positions from multiple upstream responses based on the current question and receiver role. (2) Context construction extracts consecutive hidden-state spans ending at these positions and assembles them into an ordered input context.

3 METHOD

3.1 PROBLEM SETUP

We consider a MAS that coordinates agents with distinct roles through a predefined workflow to solve an input task x. A representative configuration comprises four roles: a *planner* that proposes a solution plan, a *critic* that identifies flaws, a *refiner* that incorporates feedback to improve intermediate solutions, and a *solver* that produces the final answer.[1] We refer to the agent currently being invoked as the *receiver*, denoted by r. It generates a response conditioned on the task x, its role instruction p_r, and context from its *upstream agents* $\mathcal{A}_r$. Each upstream agent $i \in \mathcal{A}_r$ has produced a response Y_i of n_i tokens and the corresponding final-layer hidden states $H_i \in \mathbb{R}^{n_i \times d}$, where d denotes the hidden dimension. These responses and hidden states form the upstream information pool $\mathcal{U}_r = \{(Y_i, H_i) : i \in \mathcal{A}_r\}$. The goal of `LatentHarness` is to efficiently select and organize information from $\mathcal{U}_r$ into a receiver-conditioned input context, without additional training.

`LatentHarness` first selects informative token positions from $\mathcal{U}_r$ by favoring states relevant to the task and receiver role while reducing redundancy (§3.2). It then expands the selected positions into short hidden-state spans, assembles the retained spans in upstream response and token order with role labels, and aligns the retained states with the receiver's input embedding space (§3.3). Figure 1 summarizes this process.

3.2 STATE SELECTION

Given the information pool $\mathcal{U}_r$, we aim to identify a compact set of informative hidden-state positions for the receiver. This process consists of three stages: (1) constructing a candidate set from representative positions in upstream responses; (2) scoring candidates according to their relevance to the task and receiver's role; (3) iteratively selecting candidates by balancing relevance and redundancy.

Candidate construction. Inspired by Chen et al. (2025a); Ramesh & Li (2025), we draw candidates from three types of positions in each upstream response: (i) *boundary positions* include sentence-ending punctuation and newlines, (ii) *fixed-interval positions* provide additional candidates between boundaries, and (iii) *final positions* include the last position of each response. After removing

[1] We use a sequential MAS in the main experiments (§4.1) and additionally evaluate `LatentHarness` in a hierarchical MAS (§4.4).

duplicates, these positions form the candidate set $\mathcal{C}_r = \{(i, t)\}$, where i denotes the upstream agent and t denotes the corresponding token position.

Relevance scoring. We encode the receiver input consisting of x and p_r and use the final-layer hidden state at its last position as the receiver query $q_r \in \mathbb{R}^d$, which captures the task and receiver's role. For each candidate $c = (i, t) \in \mathcal{C}_r$, let $h_c \in \mathbb{R}^d$ denotes its corresponding hidden state in H_i. We compare each candidate c with the receiver query q_r in both **hidden space** and **vocabulary space**. In hidden space, cosine similarity measures representation alignment but not output preferences. We therefore complement it with top-ranked token overlap in vocabulary space (Dar et al., 2023). Let $\mathcal{T}_k(h)$ denote the top-k non-special token IDs when h is passed through the model's output head. The overall relevance score is defined as

$$s_c = \alpha \cos(h_c, q_r) + (1 - \alpha) \frac{|\mathcal{T}_k(h_c) \cap \mathcal{T}_k(q_r)|}{|\mathcal{T}_k(h_c) \cup \mathcal{T}_k(q_r)|}, \tag{1}$$

where $\cos(\cdot, \cdot)$ denotes cosine similarity and $\alpha \in [0, 1]$ controls their relative weight. We then obtain a normalized relevance score $\tilde{s}_c \in [0, 1]$ by taking the percentile rank of s_c among candidates from the same upstream response.[2]

Redundancy-aware selection. Relevance alone is insufficient, as highly relevant candidates may contain redundant information. Following maximal marginal relevance (MMR) (Carbonell & Goldstein, 1998), we balance relevance and redundancy through a priority score. Let $\mathcal{S} \subseteq \mathcal{C}_r$ denote the selected set, initially empty except for workflow-specific reserved candidates.[3] For each remaining candidate, we define

$$\pi(c, \mathcal{S}) = \tilde{s}_c - \lambda \max_{c' \in \mathcal{S}} \cos(h_c, h_{c'}), \text{ where } c \in \mathcal{C}_r \setminus S. \tag{2}$$

$\lambda \in [0, 1]$ controls the redundancy penalty, which is zero when $\mathcal{S} = \varnothing$. Each iteration selects the highest-priority candidate $c^\star \in \arg\max_c \pi(c, \mathcal{S})$, and updates the selected set as $\mathcal{S} \leftarrow \mathcal{S} \cup \{c^\star\}$.

The greedy selection iterations terminate under two conditions. **First**, the process stops when the selected set reaches the maximum size $K_{\max}$ or no candidates remain. **Second**, after at least $K_{\min}$ candidates have been selected, we additionally stop when the ① *remaining candidates provide insufficient marginal gains* or the ② *selected candidates have covered sufficient relevance*. Specifically, let c_0 and $\mathcal{S}_0$ denote the initial selected candidate and set. We stop before the next update if

$$\underbrace{\frac{\pi(c, \mathcal{S})}{\pi(c_0, \mathcal{S}_0)} \le \tau}_{\text{① marginal gains}} \quad \text{or} \quad \underbrace{\frac{\sum_{c \in \mathcal{S}} s_c}{\sum_{c \in \mathcal{C}_r} s_c} \ge \rho}_{\text{② sufficient coverage}}, \tag{3}$$

where $\tau, \rho \in [0, 1]$ specify the relative-priority and cumulative-relevance thresholds, respectively. The selected positions in $\mathcal{S}$ serve as endpoints for context construction.

3.3 Context Construction

The selected positions $\mathcal{S}$ only provide endpoints of useful information. To preserve local context around these endpoints, we expand each position into a short span of consecutive hidden states and assemble the spans into a compact latent context under a fixed budget.

Span expansion. For each selected position $c = (i, t) \in \mathcal{S}$, we extract the hidden states at positions

$$\mathcal{I}_c = \{\max(1, t - w + 1), \ldots, t\}, \tag{4}$$

where w is the maximum span length. Within each upstream response, we merge spans and preserve their original order. Let $H_i[1], \ldots, H_i[m]$ denote the m merged hidden-state spans retained from agent i, ordered by their original token positions. We concatenate these spans to form the retained hidden-state sequence for agent i:

$$\mathbf{H}_i = H_i[1] \,\|\, \cdots \,\|\, H_i[m], \tag{5}$$

[2] For responses with multiple candidates, we divide zero-based ascending ranks by the number of candidates minus one; ties receive their average rank. A single candidate receives 1.

[3] In the sequential workflow, the final answering agent starts with the final position of the refinement response. This candidate contributes to redundancy, the selection count, and accumulated relevance.

where $\|$ denotes separators between different spans to form the latent context for the receiver.

Context assembly. We order the merged spans first by upstream response order and then by their original token positions within each response. We then prepend a role label to the sequence of spans from each response. For example, when the receiver is the solver, the resulting latent context can be written as

$$\texttt{Context} = [\texttt{Planner}]\,\mathbf{H}_{\mathsf{P}}\,[\texttt{Critic}]\,\mathbf{H}_{\mathsf{C}}\,[\texttt{Refiner}]\,\mathbf{H}_{\mathsf{R}}, \tag{6}$$

where $\mathbf{H}_{\mathsf{P}}$, $\mathbf{H}_{\mathsf{C}}$, and $\mathbf{H}_{\mathsf{R}}$ denote the concatenated hidden states from the planner, critic, and refiner responses, respectively.

To bridge the mismatch between final-layer states and input embeddings, we follow Peng et al. (2026) to compute a closed-form mapping using paired states and token embeddings from the end of each upstream response. The retained states are then mapped, rescaled, and inserted directly into the receiver's input.

4 Experiments

4.1 Experimental Setup

MAS workflows. We evaluate two four-agent workflows. The main experiments use a sequential workflow: the Planner proposes a solution plan, the Critic checks it, the Refiner revises it, and the Solver produces the final answer. All roles receive the same question; access to preceding responses depends on the communication method. We additionally evaluate a hierarchical workflow (§4.4), where three experts independently solve the same question using identical role instructions. A Summarizer then combines information from all three experts, resolves disagreements, and produces the final answer.

Models and baselines. We evaluate `LatentHarness` with Qwen3.5-4B, Qwen3.5-9B (Qwen Team, 2026), and Gemma-4-12B (Gemma Team, 2026). Within each configuration, all agents use the same frozen model. The sequential experiments use five baselines. *Single-agent* runs only the Solver. *TextMAS-direct* passes the immediate predecessor's response as text, while *TextMAS-accumulated* passes all preceding responses. We also evaluate two baselines based on StateBridge (Peng et al., 2026). *StateBridge-full* delivers the immediate predecessor's complete aligned hidden-state sequence, and *StateBridge-tail64* delivers up to its final 64 states.

Datasets. We evaluate on four scientific benchmarks. *GPQA Diamond* (Rein et al., 2024) contains expert-written questions in biology, chemistry, and physics. For *SuperGPQA* (Du et al., 2025), we use a science and medicine subset. *BioProBench* (Liu et al., 2026) evaluates biological protocol understanding; we use its protocol question answering subset from the BioProBench-PQA release. For *SciCUEval* (Yu et al., 2026), we use the multiple-choice subset for scientific context understanding.

Metrics and implementation. We report final-answer accuracy (Acc.) and textual token volume (Token) per question. Token counts the initial local prompts, transferred text, and complete output of the final answering agent, using each model's tokenizer. We use a context budget of $B = 256$ input positions and a maximum span length of $w = 32$ for `LatentHarness`. The budget includes the delivered states, role labels, and separators. Further experimental details are provided in Appendix A.

4.2 Main Results

Table 1 compares `LatentHarness` with five baselines across three models and four benchmarks, reporting accuracy and Token usage. We draw the following conclusions:

`LatentHarness` improves accuracy while reducing Token usage. `LatentHarness` achieves the highest accuracy in 10 of the 12 model–benchmark settings, including one tie. Compared with TextMAS-accumulated, it improves accuracy in 11 settings and reduces Token usage in all 12 by 10.04–79.01%. On GPQA Diamond, Qwen3.5-4B improves from 63.77% to 71.74% while reducing Token from 15,291 to 3,736 per question. With Qwen3.5-9B on BioProBench, accuracy increases by 5.48 percentage points while Token decreases by 79.01%. The exception is Gemma-4-12B on SciCUEval, where accuracy is 0.17 points lower with a 10.04% Token reduction.

How latent context is organized matters beyond its representation. `LatentHarness` matches or exceeds both StateBridge variants in accuracy across all 12 settings. For Gemma-4-12B on Bio-

Table 1: Accuracy (%) and Token usage (tokens/question) on four scientific benchmarks. Bold marks the highest accuracy across all methods and the lowest Token among MAS methods. Improve reports accuracy changes (percentage points) and Token reductions (%) relative to TextMAS-accumulated.

Method	GPQA Diamond		SuperGPQA		BioProBench		SciCUEval	
	Acc.↑	Token↓	Acc.↑	Token↓	Acc.↑	Token↓	Acc.↑	Token↓
Qwen3.5-4B								
Single-agent	50.72	3,023	35.53	2,517	57.98	748	92.24	5,613
TextMAS-direct	63.77	8,100	40.83	7,504	56.43	2,374	93.00	21,610
TextMAS-accumulated	63.77	15,291	39.83	13,727	56.79	4,333	91.13	23,923
StateBridge-full	58.70	4,126	31.66	3,605	56.90	**1,121**	90.61	20,129
StateBridge-tail64	56.52	**3,711**	36.68	3,409	57.98	1,121	93.00	20,120
`LatentHarness`	**71.74**	3,736	**41.12**	**3,348**	**58.33**	1,147	**93.94**	**20,058**
Improve	↑ 7.97	↓ 75.57%	↑ 1.29	↓ 75.61%	↑ 1.55	↓ 73.54%	↑ 2.82	↓ 16.15%
Qwen3.5-9B								
Single-agent	55.80	2,678	42.84	2,366	61.55	672	92.66	5,521
TextMAS-direct	69.57	7,312	46.56	6,671	55.83	2,661	93.43	21,796
TextMAS-accumulated	67.39	13,891	47.13	11,691	56.79	4,956	93.34	24,003
StateBridge-full	65.94	3,697	45.13	3,348	60.83	1,056	**95.90**	**20,000**
StateBridge-tail64	61.59	3,525	44.27	3,281	61.55	1,091	94.37	20,026
`LatentHarness`	**73.19**	**3,408**	**49.28**	**3,231**	**62.26**	**1,040**	**95.90**	20,021
Improve	↑ 5.80	↓ 75.46%	↑ 2.15	↓ 72.36%	↑ 5.48	↓ 79.01%	↑ 2.56	↓ 16.59%
Gemma-4-12B								
Single-agent	63.77	1,493	43.27	1,602	62.62	534	95.22	5,144
TextMAS-direct	68.12	4,082	48.14	4,182	**63.21**	1,882	95.39	20,263
TextMAS-accumulated	64.49	6,270	47.56	6,222	61.79	2,870	**95.82**	21,396
StateBridge-full	62.32	2,494	41.69	2,823	50.95	1,488	92.66	19,251
StateBridge-tail64	60.87	2,459	34.38	3,066	43.81	2,066	93.17	19,280
`LatentHarness`	**69.57**	**2,304**	**48.42**	**2,552**	62.98	**990**	95.65	**19,249**
Improve	↑ 5.07	↓ 63.26%	↑ 0.86	↓ 58.98%	↑ 1.19	↓ 65.52%	↓ 0.17	↓ 10.04%

ProBench, it reaches 62.98%, compared with 50.95% for StateBridge-full and 43.81% for StateBridge-tail64. Simply delivering complete or trailing hidden-state sequences therefore does not recover the same performance. The comparison supports managing both the available sources and the states delivered to each receiver; we examine these choices separately below.

Long task inputs limit the relative Token savings. Token reductions over TextMAS-accumulated range from 58.98% to 79.01% on GPQA Diamond, SuperGPQA, and BioProBench, but from 10.04% to 16.59% on SciCUEval. For `LatentHarness` on SciCUEval, the four agents' initial local prompts account for approximately 97–98% of Token usage. Because these prompts contain the long scientific context required by each role, reducing upstream textual handoffs affects only a small share of the total measured tokens.

4.3 Ablation Studies

Figure 2 compares the accuracy of `LatentHarness` and Tail on GPQA Diamond with Qwen3.5-4B and Qwen3.5-9B. We find that:

Receiver-conditioned selection provides stronger results than recency alone. Tail selects candidates from the ends of upstream responses without relevance-based ranking or stopping. Both use cumulative upstream sources, the same context budget, context construction, and state alignment. On GPQA Diamond, `LatentHarness` reaches 71.74% with Qwen3.5-4B and 73.19% with Qwen3.5-9B, exceeding Tail by 5.07 and 2.90 percentage points, respectively. The results favor choosing context according to the receiver over relying on response position alone.

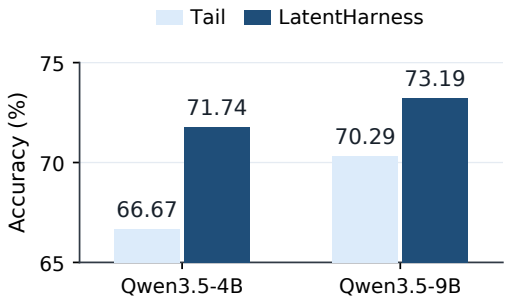

Figure 2: Accuracy (%) on GPQA Diamond for receiver-conditioned selection and Tail, which selects from response endings.

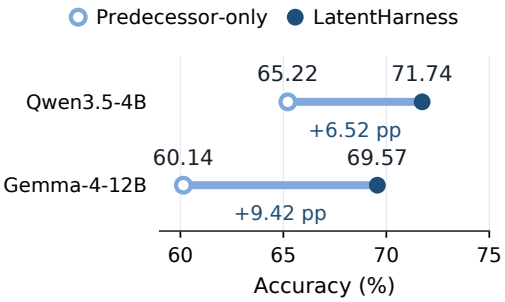

Figure 3: Accuracy (%) on GPQA Diamond when receivers access all preceding responses or only their immediate predecessor.

Table 2: Selection ablations with Qwen3.5-9B. Accuracy (%); drops are in percentage points.

Method	GPQA Diamond	BioProBench
`LatentHarness`	**73.19**	**62.26**
w/o relevance	71.74$_{\downarrow 1.45}$	60.95$_{\downarrow 1.31}$
w/o redundancy	66.67$_{\downarrow 6.52}$	60.00$_{\downarrow 2.26}$

Table 3: Equal-length state replacement. Accuracy (%); drops are in percentage points.

Method	Qwen3.5-9B GPQA Diamond	Gemma-4-12B BioProBench
`LatentHarness`	**73.19**	**63.85**
Replacement	70.29$_{\downarrow 2.90}$	61.92$_{\downarrow 1.92}$

Table 2 reports accuracy for the complete selection rule and its relevance and redundancy ablations on GPQA Diamond and BioProBench with Qwen3.5-9B. We find that:

Combining relevance and redundancy yields the strongest selection results. Removing relevance scoring and normalization retains the redundancy penalty and selects up to 12 candidates before budget packing. Removing the redundancy penalty sets $\lambda = 0$ while retaining relevance scoring, normalization, and adaptive stopping. The complete rule achieves 73.19% on GPQA Diamond and 62.26% on BioProBench. The variant without relevance scores 1.45 and 1.31 points below the complete rule; the variant without redundancy scores 6.52 and 2.26 points lower. Neither ablated rule matches the complete selection strategy on these benchmarks.

Table 3 compares accuracy before and after equal-length state replacement for Qwen3.5-9B on GPQA Diamond and Gemma-4-12B on BioProBench. We find that:

The selected content matters beyond context length. Before the final Solver handoff, we replace approximately one-third of the eligible selected states with unselected, equal-length contiguous spans from the same question and source role. We preserve the Refiner's final 32 states, payload length, and state alignment. Accuracy decreases from 73.19% to 70.29% for Qwen3.5-9B on GPQA Diamond and from 63.85% to 61.92% for Gemma-4-12B on BioProBench. Both conditions use the same evaluation questions: 138 for GPQA Diamond and 780 eligible questions for BioProBench, with the BioProBench reference recomputed on this subset. The lower accuracy under equal-length replacement supports the value of the selected information.

Figure 3 compares accuracy with access to all preceding responses or only the immediate predecessor on GPQA Diamond with Qwen3.5-4B and Gemma-4-12B. We find that:

Earlier sources remain useful beyond the immediate predecessor. We restrict each receiver's available sources to its immediate predecessor while retaining state selection, context construction, and state alignment. Earlier information can still pass indirectly through the predecessor. On GPQA Diamond, cumulative access achieves 71.74% versus 65.22% with Qwen3.5-4B and 69.57% versus 60.14% with Gemma-4-12B. The higher scores support retaining direct access to earlier responses when constructing each receiver's context.

4.4 Hierarchical MAS

Figure 4 compares `LatentHarness`, TextMAS-accumulated, and StateBridge-full in sequential and hierarchical MAS. The horizontal axis shows accuracy (higher is better), and the vertical axis shows Token usage (lower is better). We draw the following conclusions:

The gains extend to hierarchical MAS. With Qwen3.5-9B on GPQA Diamond, `LatentHarness` reaches 76.09% accuracy, compared with 74.64% for TextMAS-accumulated and 72.46% for StateBridge-full. It reduces Token usage by 74.28% relative to TextMAS-accumulated. With Gemma-4-12B on BioProBench, `LatentHarness` achieves 61.19%, compared with 60.83% and 60.95%, respectively, while using 66.89% fewer tokens than TextMAS-accumulated. Thus, the method also supports combining independently generated expert responses with a compact latent context.

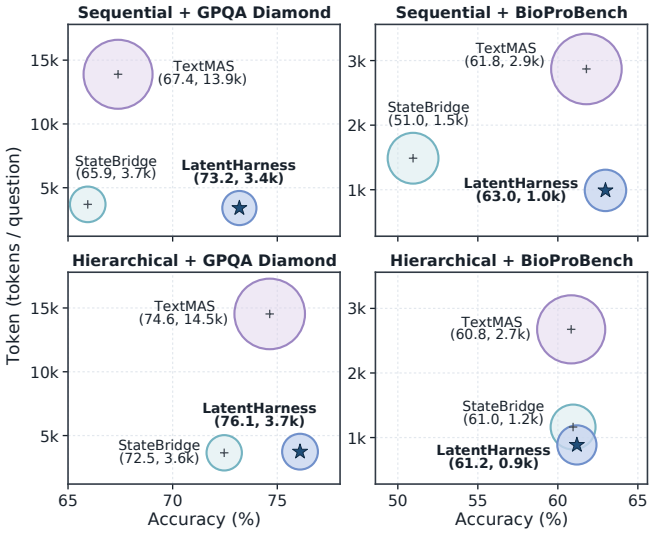

Figure 4: Accuracy and Token usage in sequential and hierarchical MAS: Qwen3.5-9B on GPQA Diamond and Gemma-4-12B on BioProBench. Labels report accuracy (%) and Token (thousands per question). Circle area scales with Token within each column. TextMAS and StateBridge use their accumulated and full variants.

The preferred collaboration structure depends on the setting. For `LatentHarness`, hierarchical and sequential workflows achieve 76.09% and 73.19% on GPQA Diamond, respectively, but 61.19% and 62.98% on BioProBench. Its accuracy advantage over TextMAS-accumulated and StateBridge-full holds in both workflows, whereas changing the workflow does not uniformly improve accuracy.

4.5 Analysis of Selected Context

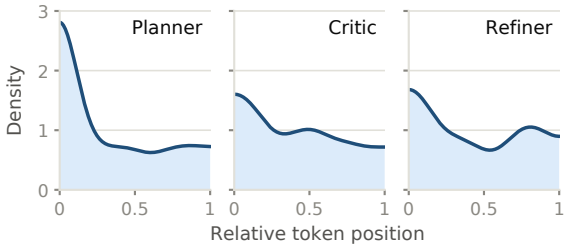

Figure 5: Question-weighted distributions of recovered span endpoints in Qwen3.5-9B Solver context on GPQA Diamond. Reserved Refiner endpoints are excluded.

Table 4: Source-text functions of 243 delivered spans from 30 GPQA Diamond questions.

Source-text function	Spans	Share (%)
Problem conditions	23	9.5
Intermediate reasoning	110	45.3
Exclusion rationale	24	9.9
Corrections	14	5.8
Final conclusions	29	11.9
Generic content	43	17.7

Figure 5 shows the question-weighted distributions of recovered span endpoints in the final Solver context for Qwen3.5-9B on GPQA Diamond. The horizontal axis indicates relative token position within each source response, and the vertical axis shows density. We observe the following:

Selection draws from across upstream responses. The recovered endpoints occur throughout each source's response, with the strongest early concentration in the Planner. Under question-level weighting, the first 10% of response positions account for 34.0% of Planner endpoints, compared with 15.7% for the Critic and 20.0% for the Refiner.

Table 4 summarizes the source-text functions of 243 delivered spans from a fixed sample of 30 GPQA Diamond questions, reporting the count and share of each category. We find that:

Most delivered spans contain question-specific content. We decode the source tokens of these spans from Qwen3.5-9B and use `gpt-6-astra` to assign one primary functional label per span. The annotations identify question-specific information in 200 spans (82.3%). Intermediate reasoning is the largest category, accounting for 110 spans (45.3%).

Question (abridged) An N-sided polygonal aperture has apothem a and is illuminated by monochromatic light of wavelength λ. In the limit $N \to \infty$ and under the small-angle approximation, what is the angular distance between "the first two minima" in the far field?

A. $0.500\,\lambda/a$ B. $0.610\,\lambda/a$ C. $1.220\,\lambda/a$ **D.** $0.506\,\lambda/a$ **(correct)**

Source text	**Original states**	**Replacement states**
Planner 8 states	The shape is an N-sided polygon with apothem a.	$\theta_2 = 1.8309 \times 0.610\,\lambda/a = 1.1168\,\lambda/a$
Critic 32 states	… Let's check the options again. A. 0.500 …	… with a fixed apothem a. In the far field (Fraunhofer diffraction), the intensity pattern for a circular aperture of radius R …
Refiner 28 states	1. Identify the limit case: $N \to \infty$ implies a circular aperture with radius $R = a$.	$\theta_2 = 7.0156\,\lambda/(2\pi a) \approx 1.116\,\lambda/a$
Unchanged	PLANNER … the first dark ring and the second dark ring … REFINER … $2\theta_1$, the diameter of the central spot … REFINER … $\Delta\theta = \theta_2 - \theta_1$. … Answer: D	
Solver output	**Dark-ring separation**	**Central-maximum width**
	… For the first minimum ($m = 1$): … $\theta_1 \approx 0.610\,\lambda/a$. For the second minimum ($m = 2$): … $\theta_2 \approx 1.1166\,\lambda/a$. … The question asks for the angular distance between the first two minima: $\Delta\theta = \theta_2 - \theta_1$. $\Delta\theta \approx (1.1166 - 0.610)\,\lambda/a = 0.5066\,\lambda/a$.	… $\sin\theta_2 = 7.0156\,\lambda/(2\pi a) \approx 1.117\,\lambda/a$. Wait, let's re-evaluate the "first two minima" phrasing. … the "first two minima" usually refers to the first minimum at $+\theta_1$ and the first minimum at $-\theta_1$. The angular distance between them is $\Delta\theta = \theta_1 - (-\theta_1) = 2\theta_1$. …
Answer	**D** ✓ (correct)	**C** ✗ (incorrect)

Figure 6: Jointly replacing 68 states changes the Solver's final computation from dark-ring separation to central-maximum width. Upstream highlights mark replaced positions within source excerpts; gray text provides neighboring source context, and underlining marks retained cues.

Figure 6 presents a GPQA Diamond diffraction example, showing the source excerpts corresponding to the original and replacement states alongside the resulting Solver outputs. This example illustrates the following:

Changing selected states can alter the Solver's interpretation. We jointly replace 68 states across three spans, keeping the upstream responses, alignment, input length, decoding settings, and sampling seed fixed. The original Solver computes the dark-ring separation, $\theta_2 - \theta_1$, and selects the correct option D. After replacement, it still computes both minima but uses the central-maximum width, $2\theta_1$, and selects the incorrect option C. The final interpretation changes even though unchanged spans still specify the dark-ring separation and answer D.

5 Conclusion

We presented `LatentHarness`, a training-free framework for adaptive latent context management in MAS. It selects hidden-state positions from multiple upstream responses by balancing relevance to the current question and receiver role against redundancy, with adaptive stopping determining the number selected. It then constructs consecutive hidden-state spans ending at these positions and assembles them into an ordered context under an input budget. Experiments on four scientific benchmarks using three LLMs show improved accuracy in most settings and reduced textual token volume compared with passing all preceding responses as text. The gains also extend to hierarchical MAS. Controlled ablations further show that the selected content matters beyond context length. Together, these findings support adapting upstream context to each receiver's needs as a complement to latent communication.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish the writing of the manuscript. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The state selection and context construction procedures of `LatentHarness` are described in §3, the experimental setup is summarized in §4.1, and Appendix A provides the implementation details. All experiments use publicly available models and benchmarks. We will publicly release the source code upon acceptance of the paper.

REFERENCES

Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, 1998.

Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo Li, Weiyang Liu, and Chao Huang. SepLLM: Accelerate large language models by compressing one segment into one separator. In *ICML*, 2025a.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs. In *ACL*, 2024.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent system. In *Findings of ACL*, 2025b.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. In *ACL*, 2023.

Xinrun Du, Yifan Yao, Kaijing Ma, et al. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. In *NeurIPS*, 2025.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *ICML*, 2024.

Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space. In *ACL*, 2026.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-Cache: Direct semantic communication between large language models. In *ICLR*, 2026.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Wenyuan Gu, Haowen Wang, Jiale Han, Xiang Li, Zhixuan Wu, Hongru Xiao, and Bo Cheng. Analyze–Compose–Execute: A dynamic dialogue framework for multi-agent debate. In *AAAI*, 2026.

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. MetaGPT: Meta programming for a multi-agent collaborative framework. In *ICLR*, 2024.

Yiping Li, Zhiyu An, and Wan Du. When less latent leads to better relay: Information-preserving compression for latent multi-agent LLM collaboration. *arXiv preprint arXiv:2604.13349*, 2026.

Yexiang Liu, Jie Cao, Zekun Li, Ran He, and Tieniu Tan. Breaking mental set to improve reasoning through diverse multi-agent debate. In *ICLR*, 2025.

Yuyang Liu, Liuzhenghao Lv, Xiancheng Zhang, Jingya Wang, Li Yuan, and Yonghong Tian. BioProBench: A corpus and benchmark for biological protocol reasoning in autonomous science. In *ICML*, 2026.

Yanwen Peng, Delvin Ce Zhang, Xi Wang, and Nikolaos Aletras. StateBridge: Training-free hidden-state alignment for latent communication in LLM multi-agent systems. In *COLM*, 2026.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan A. Plummer, Zhaoran Wang, and Hongxia Yang. Let models speak ciphers: Multiagent debate through embeddings. In *ICLR*, 2024.

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

Vignav Ramesh and Kenneth Li. Communicating activations between language model agents. In *ICML*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *COLM*, 2024.

Xiangyu Shi, Marco Chiesa, Gerald Q. Maguire, Jr., and Dejan Kostić. KVComm: Enabling efficient LLM communication through selective KV sharing. In *ICLR*, 2026.

Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. ReMA: Learning to meta-think for LLMs with multi-agent reinforcement learning. In *NeurIPS*, 2025.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. AgentDropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration. In *ACL*, 2025.

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. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *COLM*, 2024.

Jing Yu, Yuqi Tang, Kehua Feng, Lei Liang, Qiang Zhang, Keyan Ding, and Huajun Chen. SciCUEval: A comprehensive dataset for evaluating scientific context understanding in large language models. *Scientific Data*, 2026.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In *ICLR*, 2025.

Ruijia Zhang, Xinyan Zhao, Ruixiang Wang, Sigen Chen, Guibin Zhang, An Zhang, Kun Wang, and Qingsong Wen. SafeSieve: From heuristics to experience in progressive pruning for LLM-based multi-agent communication. In *AAAI*, 2026.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. In *NeurIPS*, 2025.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. In *ICML*, 2026.

A Experimental Details

A.1 Implementation Details

Algorithm 1 summarizes state selection and context construction for one receiver. The adaptive stopping conditions are the marginal-gain and relevance-coverage criteria in Section 3.2.

Algorithm 1: Receiver-conditioned latent context construction

Input: Task x, receiver role p_r, upstream pool $\mathcal{U}_r$, span length w, and context budget B
Output: Aligned context for receiver r
Construct candidates $\mathcal{C}_r$ from boundary, fixed-interval, and final positions
Encode (x, p_r) to obtain the receiver query q_r
Compute hybrid relevance and normalize it within each upstream response
Initialize ordered selection $\mathcal{L}$ with any workflow-specific reserved candidate
while $|\mathcal{L}| < K_{\max}$ *and unselected candidates remain* **do**
  Compute priorities from relevance and redundancy with selected candidates
  Let $c^\star$ be the highest-priority unselected candidate
  if $|\mathcal{L}| \geq K_{\min}$ *and an adaptive stopping condition is met* **then**
    break
  end
  Append $c^\star$ to $\mathcal{L}$
end
Initialize empty retained intervals for each upstream response
foreach $c \in \mathcal{L}$ *in selection order* **do**
  Expand c into a leftward span of at most w positions
  Retain the longest suffix of this span, if any, that fits within B after merging same-source intervals
end
Align retained states using the mapping for each upstream response
Assemble spans in upstream response and token order with role labels and separators
return *the assembled context*

Selection and context parameters. Candidate positions include sentence-ending punctuation, newlines, fixed-interval positions, and the final position of each response. We use a stride of 8 tokens for GPQA Diamond, 32 for SuperGPQA, and 16 for BioProBench and SciCUEval. The selection parameters are $k = 8$, $\alpha = 0.75$, $\lambda = 0.25$, $K_{\min} = 2$, $K_{\max} = 12$, $\tau = 0.15$, and $\rho = 0.85$. Relevance is normalized by percentile rank within each source response. Context construction uses $w = 32$ and $B = 256$ input positions per handoff. The budget includes retained states, source labels, and separators, but excludes the receiver's local prompt. The selected candidate count can therefore differ from the number of spans that fit within the budget.

Receiver inputs and workflows. Each agent receives its role instructions, the question, and the answer options through the model's chat template. The receiver query is computed before upstream context is inserted. For generation, aligned spans are inserted within the user message after the local task input; the Solver's answer-format instruction follows the spans. Source labels such as `[Planner]` and newline separators use ordinary token embeddings. In the sequential workflow, each receiver can access all completed upstream responses, and the Solver reserves the Refiner's final position before selecting additional candidates. In the hierarchical workflow, the Summarizer receives all three experts' responses without reserving a Refiner-specific position.

State alignment. We fit a separate StateBridge mapping for each source response using its final 64 hidden-state/token-embedding pairs, or all available pairs for shorter responses. The covariance regularization is 10^{-3}. We retain norm calibration and disable vocabulary anchoring for `LatentHarness`. The model parameters remain frozen throughout inference.

A.2 Evaluation Protocol

Evaluation data. We use a fixed, metadata-stratified partition for each benchmark, shared across methods. The evaluation sets contain 138 GPQA Diamond, 698 SuperGPQA, 840 BioProBench, and 1,172 SciCUEval questions.

Communication baselines. Table 5 summarizes the upstream sources and delivered content for the sequential baselines and `LatentHarness`.

Table 5: Upstream communication in the sequential workflow.

Method	Upstream sources	Representation	Delivered content
Single-agent	None	None	No upstream context
TextMAS-direct	Immediate predecessor	Text	Complete response
TextMAS-accumulated	All preceding agents	Text	Complete responses
StateBridge-full	Immediate predecessor	Aligned hidden states	Complete state sequence
StateBridge-tail64	Immediate predecessor	Aligned hidden states	Up to the final 64 states
`LatentHarness`	All preceding agents	Aligned hidden states	Receiver-conditioned spans under budget B

Generation. We use frozen models in BF16 with thinking mode disabled. The sequential workflow allows up to 8,192 generated tokens for each upstream role and 4,608 for the Solver, under a context limit of 65,536 positions. The evaluations use sampling. For sampling, we use a temperature of 0.4, top-$p = 1.0$, and top-$k = 20$.

Answer extraction. The final nonempty line of the Solver output must have the form `ANSWER: <LETTER>`, with a letter in the question's option range. Accuracy is the fraction of questions for which this letter matches the reference answer; malformed answers count as incorrect.

A.3 Token Accounting

We report textual token volume (Token) as the sum of initial local prompt tokens, transferred text tokens, and the final answering agent's complete output tokens. Single-agent includes one initial local prompt, while each multi-agent configuration includes four. Local prompts include the task context, role instructions, and template tokens, excluding upstream messages. Each textual handoff is counted once per recipient, including source labels and separators. The final output includes reasoning, the answer, and any recorded finalizer generation. Upstream internal generation, transferred hidden states, and additional finalizer inputs are excluded. We use each model's native tokenizer and average over questions with available counts, rounding to the nearest integer for display. Counts cover all evaluated questions except the Qwen3.5-4B BioProBench result for `LatentHarness`, which has 838 available counts among 840 questions; its accuracy uses all 840 questions. The Improve rows compare `LatentHarness` with TextMAS-accumulated: accuracy changes are measured in percentage points, and Token reductions are computed as $100(1 - T_{\texttt{LatentHarness}}/T_{\text{baseline}})$ using mean counts over questions available for both methods.

A.4 Selected Context Analysis

Scope and endpoint positions. We analyze the final Solver handoff for Qwen3.5-9B on the 138 GPQA Diamond test questions. For Figure 5, we recover selected endpoints from the right ends of delivered spans and normalize each zero-based position p as $p/(n-1)$, where n is the source response length. Endpoints internal to merged spans cannot be recovered; we also exclude the 138 endpoints belonging to the Refiner's reserved final spans. This leaves 423 Planner, 232 Critic, and 303 Refiner endpoints from 135, 125, and 131 questions, respectively. Within each role, every contributing question receives equal total weight, divided equally among its recovered endpoints. The curves use Gaussian kernel density estimation with boundary reflection. The reported first-decile shares use the unsmoothed positions. These are distributions of recovered endpoints, not selection probabilities over all candidate positions.

Source-text annotations. We analyze all 243 delivered spans from a fixed sample of 30 questions. Unlike the endpoint plot, this content analysis includes the protected Refiner tails. We decode the original token IDs at each delivered interval with the source tokenizer. This recovers the states' source text rather than inverting the hidden vectors. Each span receives one primary functional label, as defined in Table 6. Labels describe the displayed content's function, not its

scientific correctness. The annotations were generated automatically after Solver outcomes were available, without independent human validation.

Table 6: Functional labels for the source text of delivered spans.

Source-text function	Definition
Problem conditions	Given facts, quantities, constraints, or answer options restated from the question.
Intermediate reasoning	Calculations, derivations, explanations, or comparisons that develop a solution.
Exclusion rationale	An identifiable reason for ruling out an option, assumption, or approach.
Corrections	A specific error identified or revised in earlier reasoning or calculations.
Final conclusions	An explicit final answer, numerical result, or option for the question.
Generic content	Procedural language, headings, formatting, or fragments without an identifiable question-specific function.

Qualitative case. For the qualitative analysis, we replace approximately one-third of the selected states outside the protected Refiner tail with equal-length contiguous spans from unselected positions in the same question and source role. Each pair shares the upstream responses, state alignment, payload length, decoding settings, and per-question sampling seed. Figure 6 presents an illustrative GPQA Diamond question selected from the correct-to-incorrect cases. We replace 68 states across three spans while preserving the Refiner's final 32 states. The intervention tests the three spans jointly; individual-span necessity was not tested.

A.5 Role Prompts

Figure 7 gives the role instructions for the sequential workflow. The question and answer options are supplied separately as the task input.

Planner

```
You are the Planner in a scientific multiple-choice reasoning
pipeline. Analyze the question and options, and identify the concepts,
calculations, and discriminating evidence needed to solve it. Do not
select or state an answer letter. Stop exploring once you have a
defensible approach, then give the next agents a compact plan without
repetition.
```

Critic

```
You are the Critic in a scientific multiple-choice reasoning pipeline.
Use any upstream information made available through the communication
channel. Check the proposed approach against the question and
plausible options. Focus on consequential errors, missing evidence,
invalid assumptions, and distractor traps. Do not restate the plan or
state an answer letter. Stop once the key corrections are clear, then
give a compact critique.
```

Refiner

```
You are the Refiner in a scientific multiple-choice reasoning
pipeline. Use the question, options, and any available upstream
information to produce a corrected, self-contained plan for the
Solver. Resolve the strongest critique and retain only useful
evidence. Do not state an answer letter. Stop once the plan is
sufficient, then give a compact refined plan without repetition.
```

Solver

```
You are the Solver in a scientific multiple-choice reasoning pipeline.
Solve the question yourself. Treat any upstream information as
fallible reference: use helpful evidence, ignore irrelevant or
incorrect content, and verify the selected option against the
question. Stop once the choice is justified, then give a concise final
response that follows the task-specific answer format below.
```

Figure 7: Role prompts for the sequential multiple-choice workflow. Each box reproduces the corresponding role instruction.