EVIDENCE AND POSITION-AWARE CHAIN-OF-AGENTS FOR LONG-CONTEXT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models frequently encounter challenges when processing long-context tasks, often neglecting significant information that may be dispersed throughout extensive inputs. Traditional multi-agent frameworks, such as Chain-of-Agents, attempt to address this issue by partitioning long inputs among multiple agents. However, these approaches fail to account for the relative importance and positional vulnerability of the evidence when delegating tasks to agents. To overcome these limitations, we propose a framework that strategically assigns long-context evidence to appropriate agents based on its significance, susceptibility to positional displacement, and agent capabilities. Furthermore, a worker summary index provides non-sequential access to earlier evidence, while a scoped, non-regressive escalation mechanism corrects intermediate errors without degrading prior results. The effectiveness of the framework is assessed through rigorous evaluations on long-context question-answering and summarization tasks, utilizing established benchmarks including Qasper, QuALITY, HotpotQA, NarrativeQA, and GovReport. Performance analysis compares the proposed framework with baseline methods, such as Vanilla, CoA, Tree-of-Agents, G-Designer, and AgentRouter. Additionally, ablation studies are conducted to assess the impact of evidence importance and positional awareness on overall performance. Our findings indicate that the proposed framework provides a cohesive solution for evidence processing in long-context reasoning, significantly enhancing the retention of critical information and fostering improved multi-agent reasoning capabilities.

1 INTRODUCTION

Large language models are increasingly applied to documents that exceed their context windows: multi-hop question answering across articles, comprehension of book-length narratives, and summarization of long reports. Two common remedies each have a blind spot. Context-window extension enlarges the input a model can consume but does not ensure the model attends to the relevant part of it; retrieval reduces the input to a few passages but risks discarding evidence the retriever scores poorly (Chang et al., 2025). Underlying both is a positional effect: accuracy depends on *where* relevant content sits, degrading when it falls in the middle of a long input (Liu et al., 2024a; Wang et al., 2025b).

Multi-agent decomposition offers a third route. Chain-of-Agents (CoA) (Zhang et al., 2024) splits a long input into chunks, assigns each to a worker that reads its chunk and a summary passed from the previous worker, and lets a manager synthesize the final worker's output. This enlarges the effective receptive field without enlarging any single window, and is training-free. However, CoA treats every chunk as interchangeable and processes chunks in document order: it models neither which segments carry the evidence that answers the query nor how vulnerable a segment is to positional neglect.

This paper asks whether making the assignment *evidence and position-aware* improves long-context Chain-of-Agents reasoning. We propose EP-CoA (Evidence and Position-Aware Chain-of-Agents), which scores each segment for importance and positional vulnerability, routes segments to positionally advantaged slots under a capacity budget, provides non-sequential access to earlier evidence through a worker summary index, and adds scoped escalation gated by a non-regression rule. We implement the full system and evaluate it across three backbones and five benchmarks, with a cumu-

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.

lative ablation that isolates each mechanism. EP-CoA performs competitively throughout, achieving the best result on a plurality of settings, and its non-regression escalation rule emerges as a reliable, low-cost contributor that makes intermediate self-correction dependable. To ensure these comparisons are trustworthy, we characterize run-to-run variation under batched, prefix-cached serving (Kwon et al., 2023) and adopt a per-task reliability threshold, reporting every claim against it. We also analyze the conditions under which evidence- and position-aware assignment is and is not beneficial, providing guidance for where the mechanism should be applied. Together these give a working framework, a dependable correction mechanism, and an evaluation methodology that strengthens confidence in the reported results.

Contributions.

- EP-CoA, an Evidence and Position-aware Chain-of-Agents framework combining importance and positional-vulnerability scoring, capacity-constrained assignment, a worker summary index, and non-regression-constrained scoped escalation, competitive with strong baselines across three backbones and five long-context tasks (Sections 3–4).
- An Escalation Non-Regression (ENR) mechanism that accepts a corrective rewrite only when it does not reduce a quality proxy; it improves summarization quality at negligible cost and roughly halves the rate at which self-correction harms an answer, turning an unreliable rewrite step into a dependable one (Section 5).
- An analysis of evidence-to-agent assignment identifying the conditions under which position-aware placement helps, clarifying where the mechanism is best deployed (Section 5).

2 Related Work

Long-context reasoning and positional bias. The transformer architecture (Vaswani et al., 2017) underlies current long-context models. One line enlarges the sequence a transformer can process, through sparse attention (Beltagy et al., 2020; Zaheer et al., 2020), length extrapolation (Press et al., 2022), or streaming attention sinks (Xiao et al., 2024). Yet enlarging the window does not guarantee that a model uses its full input: An et al. (2025) show the effective context length falls well short of the nominal one, and models are easily distracted by irrelevant context (Shi et al., 2023). Independent of window size, Liu et al. (2024a) report a U-shaped dependence of accuracy on the position of relevant content. Subsequent work sharpens this: Wang et al. (2025b) give a mechanistic account of position bias and a method to eliminate it, and Wan et al. (2025) show a positional bias in the *faithfulness* of long-form summarization, not only in retrieval. Most directly relevant to our study, Byerly & Khashabi (2026) find that positional bias undermines self-consistency on long-context problems—aggregating multiple runs does not average the bias away. Evaluation of such abilities is standardized by long-context benchmarks (Bai et al., 2024). Our positional-vulnerability score operationalizes the position effect as an assignment signal rather than an attention correction, and our measurement analysis is consistent with Byerly & Khashabi (2026) in finding that repeated runs do not stabilize long-context scores.

Retrieval and evidence selection. Retrieval-augmented generation reduces input to scored passages, and recent systems make the selection itself multi-agent: MAIN-RAG (Chang et al., 2025) uses a set of agents to filter retrieved documents before generation. Such methods score evidence for *selection* and discard the rest; EP-CoA instead scores *retained* evidence for *placement* within a multi-agent chain and does not discard low-scoring segments, since in long-document QA the answer may depend on content no retriever ranked highly.

Multi-agent LLM systems. LLM agents collaborate through orchestrated conversation (Wu et al., 2024), role specialization for software and reasoning tasks (Hong et al., 2024; Qian et al., 2024), dynamic agent networks (Liu et al., 2024b), and, more simply, ensembling: Mixture-of-Agents (Wang et al., 2025a) aggregates several models' outputs in layers. For long context, CoA (Zhang et al., 2024) chains workers over chunks. EP-CoA extends this chain rather than replacing it, adding evidence-aware placement and gated correction to the same sequential backbone.

Communication topology and routing. A distinct line designs *how* agents are connected. G-Designer (Zhang et al., 2025) learns a task-adaptive communication topology over a fixed agent

set with a graph neural network, and Graph-of-Agents (Yun et al., 2026) organizes collaboration as a graph. AgentRouter (Zhang et al., 2026) routes a query to a weighted subset of agents via a knowledge-graph-guided router and aggregates by voting. These methods take a task and a set of agents and decide the communication or routing structure; they do not model where evidence sits in a long document, nor do they place long-context evidence segments. EP-CoA differs in aim and input: its unit is a positionally situated evidence segment placed into a chain, and its objective incorporates positional vulnerability. We include G-Designer and AgentRouter as baselines.

Reflection and self-correction. Reflexion (Shinn et al., 2023), Self-Refine (Madaan et al., 2023), and ReAct (Yao et al., 2023) let a model critique and revise its own outputs, and multi-agent variants use debate or consensus among agents to filter errors (Du et al., 2024; Chan et al., 2024; Chen et al., 2024). Self-consistency (Wang et al., 2023) aggregates multiple reasoning paths, and chain-of-thought prompting (Wei et al., 2022) elicits intermediate steps that such critics can inspect. Two observations motivate our gated design. First, Byerly & Khashabi (2026) show self-consistency is undermined by positional bias on long-context tasks, so aggregation alone does not remove the error we target. Second, in our data unconstrained self-correction is close to a coin flip. Our reflection checkpoint therefore accepts a revision only under a non-regression rule; calibrating the trigger with a coverage guarantee, for instance via conformal prediction (Vovk et al., 2005; Angelopoulos & Bates, 2023), is a natural extension we do not realize here. Table 1 summarizes the positioning of EP-CoA relative to these lines of work.

Table 1: Positioning of EP-CoA relative to representative multi-agent and long-context methods. "Evidence placement" = deciding where retained evidence is processed, distinct from selecting or routing it.

Method	Primary decision	Long-context evidence	Intermediate correction
Mixture-of-Agents (Wang et al., 2025a)	output aggregation	no	no
MAIN-RAG (Chang et al., 2025)	evidence filtering	selection	no
G-Designer (Zhang et al., 2025)	communication topology	no	no
Graph-of-Agents (Yun et al., 2026)	collaboration graph	no	no
AgentRouter (Zhang et al., 2026)	agent routing	no	voting
Chain of Agents (Zhang et al., 2024)	sequential chaining	document order	no
EP-CoA (ours)	evidence placement	scored assignment	ENR escalation

3 METHOD

EP-CoA augments the CoA chain with four components to improve long-context reasoning (Figure 1). These components incorporate evidence importance, positional awareness, agent capability, and adaptive reasoning mechanisms. Together, they help identify relevant evidence, account for positional vulnerability, and assign information to appropriate agents. The framework also supports reflection and selective escalation to correct errors while preserving useful intermediate results.

3.1 PRELIMINARIES

Let C be a long document and q a query. A chunker partitions C into an ordered set of evidence segments $E = \{e_1, e_2, \ldots, e_n\}$, where each segment e_i respects a per-agent token budget k, i.e. $\text{len}(e_i) \leq k$, and the index i preserves the segment's original position in C. Each segment is a contiguous span sized to fit one agent's context window; the chunker we use, Capacity-Aware Sentence Packing, is described in Appendix A. A chain of L worker agents $\{W_1, \ldots, W_L\}$ processes the segments and forwards communication units to a manager agent M, which produces the final answer y. In Figure 1, four segments are drawn for illustration; a document yields as many as its length requires. All segments are passed together to be scored, and at this stage no segment is yet judged more or less useful than another.

3.2 EVIDENCE AND POSITION-AWARE ASSIGNMENT

Evidence importance. Not all evidence a query matches is equally useful, and the distinctions matter for how a segment is scored. We separate three signals. *Relevance* $R(e_i, q)$ measures topical match between segment and query, computed as embedding cosine similarity (Reimers & Gurevych,

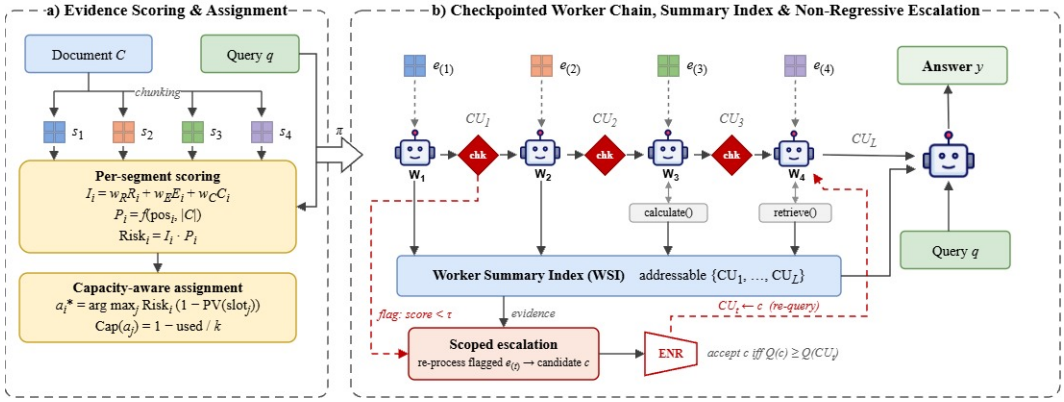


Figure 1: EP-CoA architecture. Segments $s_1 \ldots s_4$ are scored for importance I_i and positional vulnerability P_i and coupled into $\text{Risk}_i = I_i \cdot P_i$; a capacity-aware step assigns them to chain slots. Workers process in order with reflection checkpoints (*chk*); a worker summary index gives non-sequential access to earlier communication units; a scoped escalation worker, gated by the non-regression rule, re-processes flagged segments before the manager produces answer y.

2019); a segment can be relevant yet contain no answer. *Support* $E(e_i, q)$ measures whether the segment actually contains answer-bearing content, a graded utility distinct from topical overlap and related to the observation that models are distracted by relevant-looking but unhelpful context (Shi et al., 2023). *Contribution* $C(e_i, q)$ measures whether that content is non-redundant: a segment whose evidence is duplicated by others contributes little even if independently supportive, so C discounts a segment by its maximal similarity to other high-support segments. We combine them linearly:

$$I(e_i, q) = w_R\, R(e_i, q) + w_E\, E(e_i, q) + w_C\, C(e_i, q), \qquad w_R + w_E + w_C = 1. \tag{1}$$

Lexical and neural scoring (Robertson & Zaragoza, 2009; Karpukhin et al., 2020), re-ranking (Nogueira et al., 2020), and prompt compression (Jiang et al., 2023; Xu et al., 2024) offer alternative instantiations of these terms; our contribution is not a new scorer but the use of the combined signal to place, rather than select, evidence.

Positional vulnerability. Following the U-shaped position–accuracy dependence (Liu et al., 2024a; Wang et al., 2025b), each slot receives $\text{PV}(s) \in [0, 1]$, high for positions distant from the manager and low near it. With normalized slot index $p_s \in [0, 1]$, a symmetric placeholder is

$$\text{PV}(p_s) = \big(1 - (2p_s - 1)^2\big)^{\gamma}. \tag{2}$$

At the segment level, the vulnerability $P_i = f(\text{pos}_i, \text{ctx len})$ estimates how likely a segment is to be neglected because of where it sits, following the tendency of long-context models to under-attend to the middle of an input. In principle PV should be calibrated per backbone by controlled position-perturbation probing rather than assumed; we return to this in the conclusion.

Coupled assignment. The design hypothesis is that importance and vulnerability interact: a segment warrants protection only when both important and positionally endangered. We couple them multiplicatively,

$$\text{Risk}(e_i) = I(e_i, q) \cdot P_i, \tag{3}$$

which is large only when a segment is both important and positionally exposed, and assign segments so as to place high-risk evidence in low-vulnerability slots subject to each agent's capacity $\text{cap}(W_j) = 1 - \text{used}(W_j)/k$:

$$\max_x \sum_i \sum_s x_{is}\, I(e_i, q)\,\big(1 - \text{PV}(s)\big) \quad \text{s.t.} \quad \sum_s x_{is} = 1 \;\; \forall i, \qquad \sum_i x_{is}\, \text{len}(e_i) \le k \;\; \forall s. \tag{4}$$

In practice the assignment step selects, for each segment,

$$a_i^* = \arg\max_j \; \text{Risk}_i\,\big(1 - \text{PV}(\text{slot}_j)\big), \tag{5}$$

subject to each agent's remaining capacity $\mathrm{cap}(a_j) = 1 - \mathrm{used}/k$. In Figure 1, the dashed arrows descending from the assignment band to each worker are control signals: they carry the assignment decision, telling each worker which segment it will process, and are drawn dashed to distinguish this control flow from the solid data flow that passes evidence between workers. Equation 4 is the proposed objective; Appendix D reports that it does not improve over uniform assignment and explains why.

3.3 Non-Sequential Access and Non-Regressive Escalation

Worker agents and hand-offs. Each worker W_t holds its assigned evidence $e_{(t)}$ and processes it, emitting a communication unit CU_t. The chain hand-offs $W_1 \to W_2 \to \cdots \to W_L$ pass each worker's running summary forward to the next, and the rightmost worker's output flows into the manager. Between consecutive workers is a reflection checkpoint: before a hand-off propagates, a checker scores the worker's output for faithfulness, completeness, and grounding, and an output falling below threshold τ is flagged. Also, workers can call optional external tools (e.g. retrieve(), calculate()) on a segment while they are working on it and append the result to their communication unit before emitting it.

Worker Summary Index (WSI). The Worker Summary Index contains a record of each communication unit. The WSI maintains the units as individually addressable, so that a later worker or the manager can retrieve previous evidence by similarity to the query, instead of relying only on the single summary passed along the chain. This preserves evidence added earlier in the chain that a purely forward-passing summary would gradually wash out. The manager uses the entire index to write the answer.

Scoped escalation with Escalation Non-Regression (ENR). If an output is flagged by a reflection checkpoint, a scoped escalation worker will reprocess the flagged portion alone, it will not restart the pipeline. It might revisit a previous step via a re-query. The only exception is when the rewrite is allowed under the Escalation Non-Regression rule. Q is a quality proxy and c the candidate rewrite,

$$\mathrm{CU}_t \leftarrow \begin{cases} c, & \text{if } Q(c) \geq Q(\mathrm{CU}_t), \\ \mathrm{CU}_t, & \text{otherwise,} \end{cases} \tag{6}$$

so escalation can improve or leave the result unchanged but never worsen it with respect to Q.

Manager and answer. The manager agent receives as inputs the final worker's output along the chain, the query q and evidence retrieved from the WSI. Then it makes the final answer y from these. Important evidence has been directed to low-vulnerability slots, kept addressable through the WSI and shielded by non-regressive escalation. So the manager builds its answer from a more complete and reliable collection of intermediate results than a simple forward chain would give. The EP-CoA pseudocode is presented in Algorithm 1; its complexity is analyzed in Appendix B.

Algorithm 1 EP-CoA

Require: document C, query q, window k
$E \leftarrow \textsc{Chunk}(C, q, k)$ ▷ sentence-packed segments (Appendix A)
for each $e_i \in E$ **do**
  $I(e_i, q) \leftarrow w_R R + w_E E + w_C C$ ▷ Eq. 1
end for
assign E to chain slots via capacity-constrained objective; set reading order ▷ Eq. 4
$\mathrm{WSI} \leftarrow \emptyset$; $\mathrm{CU}_0 \leftarrow \emptyset$
for $t = 1$ **to** L **do**
  $\mathrm{CU}_t \leftarrow W_t(\mathrm{CU}_{t-1}, e_{(t)}, q)$; add CU_t to WSI
  if $\textsc{Reflect}(e_{(t)}, \mathrm{CU}_t, q) < \tau$ **then**
    $c \leftarrow \textsc{Escalate}(e_{(t)}, \mathrm{WSI}, q)$
    if $Q(c) \geq Q(\mathrm{CU}_t)$ **then**
      $\mathrm{CU}_t \leftarrow c$ ▷ ENR, Eq. 6
    end if
  end if
end for
return $M(\mathrm{CU}_L, \mathrm{WSI}.\textsc{Retrieve}(q), q)$

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Five long-document tasks from LongBench span answer regimes: extractive QA (HotpotQA (Yang et al., 2018), Qasper (Dasigi et al., 2021)), abstractive QA (NarrativeQA (Kočiský et al., 2018)), multiple choice (QuALITY (Pang et al., 2022)), and summarization (GovReport (Huang et al., 2021)). QA is scored by token-level F1 following standard reading-comprehension evaluation (Rajpurkar et al., 2016), QuALITY by accuracy, GovReport by ROUGE-L (Lin, 2004). Evaluation follows the long-context benchmark setting of Bai et al. (2024); median document lengths are listed in Appendix C.

Each task provides 200 examples, partitioned once: indices 0–119 form a pool for training and configuration selection, and indices 120–199 a held-out slice of 80 on which all methods are scored. All numbers are re-scored on the same held-out 80 from stored per-example predictions. Backbones are LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Qwen2.5-7B-Instruct , and Gemma-3-12B-IT (FP8).

Baselines are *Vanila* (whole document in one window, head-and-tail truncated when needed); *Chain of Agents* (Zhang et al., 2024); *Tree of Agents* (Yu et al., 2025); *G-Designer* (Zhang et al., 2025), which learns a communication topology over an agent set; and *AgentRouter* (Zhang et al., 2026), which routes to a weighted subset of agents. The latter two were designed for tasks that fit a single context and are adapted here by giving each agent the same chunker EP-CoA uses; Table 2 contrasts their design axes.

Table 2: Design axes of the compared methods. "Position aware" = models where evidence sits; "Evidence aware" = models which segments are answer-bearing.

Method	Long-context handling	Agent routing	Position aware	Evidence aware	Correction
Vanila	truncate/extend	–	no	no	no
Chain of Agents	chunk + chain	fixed order	no	no	no
Tree of Agents	chunk + tree	order search	no	partial	vote
G-Designer	graph topology	learned graph	no	no	no
AgentRouter	agent pool	learned route	no	no	no
EP-CoA (ours)	chunk + chain	scored assign	yes	yes	ENR escalation

Reliability threshold. Small score differences on long-context benchmarks are only meaningful relative to the variation a system exhibits when re-run, and we make this explicit so that every comparison in the paper rests on a stated threshold. Under batched, prefix-cached serving (Kwon et al., 2023)—the standard production configuration—the floating-point reduction order depends on which requests share a batch, so at temperature 0 a perturbed logit can flip a token at a near-tie, and a chained architecture can propagate one altered communication unit into a different final answer. This is a property of the serving stack rather than of any method, and it is consistent with recent evidence that repeated runs do not average away long-context effects (Byerly & Khashabi, 2026); a random seed does not remove it, since greedy decoding does not sample. We therefore measure it directly. Pooling runs that share a configuration and code build gives a per-task standard deviation (Table 3), and we adopt 2σ as a significance threshold, reporting a difference as a result only when it clears that task's threshold. Differences smaller than 2σ we describe as ties rather than wins. Adopting this discipline strengthens the paper's claims by construction and offers a template other long-context agent evaluations can reuse; we regard the protocol as a contribution in its own right.

Table 3: Run-to-run standard deviation of identical computations, used as a reliability threshold. The gap between tasks follows from the metric: ROUGE-L averages over a long summary, while F1 over a short answer is near-binary per example, so one changed answer shifts the aggregate by about 1.25 points.

Task	Samples	σ	2σ
HotpotQA	4	1.52	3.04
GovReport	2	0.04	0.07

4.2 RESULTS

4.2.1 MULTI-HOP AND EXTRACTIVE QUESTION ANSWERING

Table 4 reports all fifteen backbone–task settings; following the answer regimes of the benchmarks, we discuss them in three groups. On HotpotQA, EP-CoA attains the best score on LLaMA (59.28) and Gemma (63.92), ahead of the strongest baseline—the Vanila in both cases—by 4.66 and 6.75 points, both above the HotpotQA reliability threshold of Table 3. On Qwen, Chain of Agents leads (58.39) and EP-CoA is second (57.11); the 1.28-point gap is below the threshold, so we report it as a tie. On Qasper, EP-CoA sets the best LLaMA result (46.85, against 45.58 for G-Designer), while on Qwen and Gemma the Vanila leads (45.33 and 39.75). Qasper has the shortest documents of the five tasks, so the single agent rarely truncates there, in line with the observation in Section 4.2.3 that the multi-agent methods must earn their decomposition on inputs that fit one window.

Table 4: Performance comparison of EP-CoA with baseline methods on five long-context benchmarks across three backbone LLMs. Scores are F1 (HotpotQA, NarrativeQA, Qasper), accuracy (QuALITY), and ROUGE-L (GovReport).

Backbone	Method	HotpotQA	NarrativeQA	Qasper	QuALITY	GovReport
LLaMA-3.1-8B	Vanila	54.62	26.50	43.72	66.25	34.83
	Chain of Agents	50.49	23.91	41.06	71.25	29.77
	Tree of Agents	47.05	20.75	30.13	51.25	23.04
	G-Designer	54.39	21.18	45.58	52.50	36.64
	AgentRouter	37.88	9.87	15.30	48.75	5.85
	EP-CoA(ours)	59.28	23.08	46.85	68.75	34.87
Qwen2.5-7B	Vanila	52.25	18.91	45.33	73.75	32.21
	Chain of Agents	58.39	17.36	40.47	72.50	28.80
	Tree of Agents	53.50	21.24	30.23	58.75	21.61
	G-Designer	43.73	16.75	35.53	75.00	31.39
	AgentRouter	37.88	9.87	15.30	48.75	5.85
	EP-CoA(ours)	57.11	23.48	44.30	77.50	32.08
Gemma-3-12B	Vanila	57.17	26.41	39.75	66.25	28.83
	Chain of Agents	56.20	22.57	37.30	77.50	23.29
	Tree of Agents	50.57	18.96	31.98	63.75	24.35
	G-Designer	55.99	9.03	35.83	63.75	27.89
	AgentRouter	37.88	9.87	15.30	48.75	5.85
	EP-CoA(ours)	63.92	27.08	37.85	76.25	28.82

4.2.2 NARRATIVE COMPREHENSION AND MULTIPLE-CHOICE QUESTION ANSWERING

On NarrativeQA, the longest task in our suite, EP-CoA leads on Qwen (23.48, against 21.24 for Tree of Agents) and on Gemma (27.08, against 26.41 for the Vanila), making it the strongest multi-agent approach on abstractive QA under two of three backbones; on LLaMA the Vanila leads (26.50) and EP-CoA (23.08) is close to Chain of Agents (23.91). On QuALITY, EP-CoA sets the best Qwen result (77.50, against 75.00 for G-Designer), while Chain of Agents leads on LLaMA (71.25 vs. 68.75) and Gemma (77.50 vs. 76.25). Because the held-out slice has 80 examples, one question corresponds to 1.25 accuracy points, so the Gemma gap amounts to a single example. We did not measure run-to-run variation on these two tasks, so we do not apply a threshold to them.

4.2.3 LONG-DOCUMENT SUMMARIZATION

On GovReport, G-Designer leads on LLaMA (36.64), with EP-CoA (34.87) and the Vanila (34.83) next; on Qwen and Gemma the Vanila leads (32.21 and 28.83), with EP-CoA at 32.08 and 28.82. Against the GovReport threshold of Table 3 ($2\sigma = 0.07$), EP-CoA ties the Vanila on LLaMA and Gemma and trails it on Qwen. The principal summarization effect of EP-CoA is therefore the within-system gain of its non-regression rule, analyzed in Section 5.

Across all five tasks, EP-CoA attains the best result in six of fifteen settings (Table 4), more than any other single method, and is the strongest multi-agent approach on the extractive and abstractive

QA tasks under two of three backbones. Where it is not first, it is typically within close range of the leader, and it never collapses on a task the way routing- and tree-based baselines do on the hardest settings. These results indicate that augmenting a chain with evidence scoring, non-sequential access, and gated correction yields a competitive long-context reasoner across model families.

We contextualize the comparison with two observations that also inform where the gains come from. The single-agent baseline (Vanila) is a strong reference: on documents short enough to fit its window it loses no content, and on those tasks the multi-agent methods must earn their decomposition. Consistent with this, EP-CoA's largest margins over the single agent occur on settings where the Vanila is forced to truncate, indicating that a substantial part of the benefit is recovered content that a one-window reader would drop. Separately, the AgentRouter baseline is limited by a document-view truncation inherited from its original short-answer design; we report its row for completeness and do not draw routing conclusions from it. Neither observation diminishes EP-CoA's standing as the most frequent top performer, but both clarify the mechanism behind its gains.

5 Ablation Study

To assess the impact of each mechanism in EP-CoA, we decompose the system into a cumulative ladder on the LLaMA backbone, in which each rung adds one mechanism so that consecutive differences isolate it. The following variants are compared:

V0: Uniform. Segments are assigned to chain slots by uniform round-robin, with no importance or position scoring. This rung corresponds to a plain chain with the EP-CoA chunker.

V1: + Joint assignment. Adds the coupled importance/position assignment of Section 3.2. This setting tests whether scoring-based placement improves over uniform placement.

V2: + Reflect/escalate. Adds reflection checkpoints and scoped escalation at a fixed threshold, accepting every escalated rewrite. This setting measures unconstrained self-correction.

V3: + ENR (reported). Adds the Escalation Non-Regression rule of Equation 6. This is the reported system.

Table 5 reports the ladder on HotpotQA, NarrativeQA, and GovReport. The headline of the ablation is the effect of the non-regression rule (V2→V3), which we examine first; the analysis of when assignment-level scoring contributes is given in Appendix D.

Table 5: Cumulative ablation on LLaMA. Δ is the change from the rung above; "Fired" the number of escalations triggered; "Benefit/Regr." the fractions of fired escalations that improve/degrade the answer. V0→V1 call counts are identical within each task.

Task	Rung	Score	Δ	Calls	Fired	Benefit%	Regr.%
HotpotQA	V0 uniform	61.30	–	240	–	–	–
	V1 + joint assign.	58.31	−2.98	240	–	–	–
	V2 + reflect/escalate	57.55	−0.76	354	17/80	23.5	17.6
	V3 + ENR (reported)	59.28	+1.73	360	20/80	30.0	5.0
NarrativeQA	V0 uniform	23.50	–	350	–	–	–
	V1 + joint assign.	22.84	−0.66	350	–	–	–
	V2 + reflect/escalate	22.95	+0.11	567	28/80	35.7	21.4
	V3 + ENR (reported)	23.08	+0.13	567	30/80	26.7	26.7
GovReport	V0 uniform	34.19	–	240	–	–	–
	V1 + joint assign.	34.25	+0.05	240	–	–	–
	V2 + reflect/escalate	34.37	+0.12	394	37/80	43.2	48.6
	V3 + ENR (reported)	34.87	+0.50	404	42/80	64.3	31.0

The non-regression rule delivers a reliable gain. Adding ENR (V2→V3) improves GovReport by $+0.50$ ROUGE-L for only a 2.5% increase in calls, and it markedly improves the reliability of self-correction: the fraction of escalations that worsen the answer falls from 48.6% to 31.0% on GovReport and from 17.6% to 5.0% on HotpotQA. The effect is intuitive. An unconstrained rewrite

on GovReport improves the answer 43.2% of the time and degrades it 48.6%—barely better than a coin flip—so accepting every rewrite is unsafe. Gating on non-regression turns this into 64.3% improved against 31.0% degraded, so the accepted rewrites are net beneficial. The mechanism's value is that it makes an otherwise unreliable correction step dependable, and it does so at negligible cost, which is the property one wants from a safety mechanism in a multi-agent pipeline. Applied to this finding, the reliability threshold of Table 3 confirms the GovReport ENR gain ($+0.50$) as robust, since it clears the GovReport threshold.

Reflection and escalation. Adding reflection with scoped escalation (V1→V2) has a small effect on score on its own ($+0.12$, $+0.11$, -0.76 across tasks) while producing the candidate rewrites that ENR then filters. Reflection and escalation are coupled by design—escalation targets exactly the flagged chunks—so their contribution is best read together with the non-regression gate above, which converts the candidate rewrites they generate into a net-positive correction step.

Cost of each mechanism. Each mechanism EP-CoA adds is an inference cost, so its benefit must be weighed against its calls and tokens. Scoring and assignment (V0→V1) are free in agent calls: they reorder existing chunks and issue no additional model calls, so V0 and V1 have identical cost. The cost is concentrated in reflection and escalation (V1→V2), which raise calls by 48–62% and tokens by 49–74% by adding a checker pass at every hand-off and an escalation pass at every flagged one. The non-regression rule (V2→V3) is nearly free—it adds only the escalations that clear the reflection gate, a 2–3% increase. The non-regression gate produces the change that clears the GovReport threshold, while the more expensive reflection and escalation passes serve mainly to generate the candidate rewrites the gate then filters. For long-context agent pipelines, the useful lesson is not that correction is expensive but that most of its expense (the reflection and escalation passes) is spent producing candidates, while the value lies in the near-free step of deciding whether to keep them.

6 Conclusion

We introduced EP-CoA, an evidence and position-aware extension of Chain-of-Agents that scores long-context segments, provides non-sequential access to earlier evidence through a worker summary index, and corrects intermediate outputs through a non-regression-constrained escalation step. EP-CoA is competitive on all three backbones and five long-context benchmarks, and is the most frequent top performer among the compared methods. Its non-regression rule provides a reliable and low-cost mechanism to ensure the reliability of self-correction, and thus improves the quality of summarization. We further discuss the conditions under which evidence-to-agent assignment is beneficial, providing concrete guidance for deployment. Because all comparisons are reported against a measured per-task reliability threshold, the findings are robust to nondeterminism in serving by construction, and we are releasing this protocol for other long-context agent evaluations to reuse. Prospective directions are promising in the direction of decoupling evidence and placement from reading order and adding adaptive topology selection based on evidence distribution, calibrating the correction threshold, and further improving accuracy with stronger scoring and specialized agents.

AI Use Statement

In this work, we used generative AI tools for Literature Survey and related work. We have not used generative AI tools to develop the conceptual framework, formulate hypotheses, design the experiments, or interpret results, and the generation of synthetic data, translation, and mathematical proofs are not applicable to this work. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility Statement

All reported numbers are computed on the same held-out slice (indices 120–199 of each task) and rescored from stored per-example predictions, so every cell of Tables 4–7 is produced by an identical scoring pipeline. All backbones are open-weight models served through a single vLLM instance with prefix caching, a 32,768-token window, temperature 0, and client concurrency 16; generation is capped at LongBench's per-task limit for every method. The chunker is specified in Appendix A,

per-dataset evaluation details are given in Appendix C, and the run-to-run variation of the serving stack is quantified in Table 3.

REFERENCES

Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. Why does the effective context length of llms fall short? In *International Conference on Learning Representations (ICLR)*, pp. 54791–54810, 2025.

Anastasios N. Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Adam Byerly and Daniel Khashabi. Self-consistency falls short! the adverse effects of positional bias on long-context problems. *Transactions of the Association for Computational Linguistics (TACL)*, 14:292–317, 2026.

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

Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, and Na Zou. Main-rag: Multi-agent filtering retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2607–2622, 2025.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4599–4610, 2021.

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

Abhimanyu Dubey et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, 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 *International Conference on Learning Representations (ICLR)*, 2024.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics (TACL)*, 6:317–328, 2018.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out (ACL Workshop)*, 2004.

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

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

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. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pre-trained sequence-to-sequence model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 5336–5358, 2022.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations (ICLR)*, 2022.

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. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning (ICML)*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

David Wan, Jesse Vig, Mohit Bansal, and Shafiq Joty. On positional bias of faithfulness for long-form summarization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL)*, pp. 8791–8810, 2025.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y. Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations (ICLR)*, pp. 33944–33963, 2025a.

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

Ziqi Wang, Hanlin Zhang, Xiner Li, Kuan-Hao Huang, Chi Han, Shuiwang Ji, Sham M. Kakade, Hao Peng, and Heng Ji. Eliminating position bias of language models: A mechanistic approach. In *International Conference on Learning Representations (ICLR)*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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 conversation. In *Conference on Language Modeling (COLM)*, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

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

Song Yu, Xiaofei Xu, Ke Deng, Lin Li, and Lizhen Tian. Tree of agents: Improving long-context capabilities of large language models through multi-perspective reasoning. In *Findings of the Association for Computational Linguistics (EMNLP Findings)*, 2025.

Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Zou, Guohao Li, and Tianlong Chen. Graph-of-agents: A graph-based framework for multi-agent llm collaboration. In *International Conference on Learning Representations (ICLR)*, 2026.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

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

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan O. Arik. Chain of agents: Large language models collaborating on long-context tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zheyuan Zhang, Kaiwen Shi, Zhengqing Yuan, Zehong Wang, Tianyi Ma, Keerthiram Murugesan, Vincent Galassi, Chuxu Zhang, and Yanfang Ye. Agentrouter: A knowledge-graph-guided llm router for collaborative multi-agent question answering. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 788–809, 2026.

A Capacity-Aware Sentence Packing

Let x be a document, q a query, ι a worker instruction, and k the nominal agent window. Let $\tau(\cdot)$ denote the token count under the serving model's own tokenizer: LLaMA, Qwen and Gemma segment text differently, so a shared k only denotes a shared quantity if every method is measured with the tokenizer that will actually read the text.

Worker slot j holds document text only after reserving the query, the instruction, the unit it must emit, and—for every slot after the first—the unit arriving from its predecessor:

$$B_j = \max\big(256,\ k - \tau(q) - \tau(\iota) - c - c\,[\,j > 0\,]\big), \tag{7}$$

where c is the per-unit budget and $[\cdot]$ is the Iverson bracket. Segments are sized against B_1, the binding slot, rather than B_0: slot 0 alone receives no incoming unit, so sizing to it would produce segments that no later slot can accept.

Why emission is verified rather than assumed. Packing accumulates per-sentence token counts, but a worker reads the joined string. The two disagree: joining inserts separators, and a tokenizer merges across a sentence boundary differently than it does within a sentence. On NarrativeQA the assembled segment measured 0.03% to 2% larger than the sum of its parts. That margin is small and was destructive. A segment two tokens over B_1 fits no slot, so the assignment stage leaves it unplaced and it never reaches the aggregator. Before emission was verified, 42% of NarrativeQA segments exceeded budget, 111 were left unplaced across 80 examples, and 68 never reached the aggregator at all—roughly a third of each document absent from the answer, with every run completing normally and reporting a score. Lines 15–19 of Algorithm 2 therefore re-measure the joined text and backtrack until it genuinely fits, while the caller re-packs whatever was removed.

Complexity. One pass over $|S|$ sentences. EMIT backtracks at most $|P|$ times per call and each removal is permanent within that call, so the total number of re-measurements is $O(|S|)$ amortised. Tokenization dominates in practice.

Properties.

1. *No text is discarded.* Every sentence is emitted in some segment; a sentence longer than B_1 becomes its own segment rather than being truncated away.
2. *Segment size is honest.* Because B_1 is the binding budget, a segment that fits nowhere is a genuine signal—an oversized sentence—rather than an artefact of inconsistent accounting, and is routed to escalation instead of vanishing.
3. *Tokenizer-faithful.* Using the serving model's tokenizer keeps k comparable across the architecture and the baselines it is measured against.

Algorithm 2 Capacity-Aware Sentence Packing (CASP)

Require: document x, query q, instruction ι, window k, unit budget c
Ensure: segments $\mathcal{C} = \langle c_1, \ldots, c_l \rangle$ with $\tau(c_i) \leq B_1$

$B \leftarrow \max(256,\ k - \tau(q) - \tau(\iota) - 2c)$ ▷ Eq. 7, $j = 1$
$S \leftarrow \textsc{SentenceSplit}(x)$ ▷ rule-based; no entity model
$\mathcal{C} \leftarrow \langle\,\rangle$; $P \leftarrow \langle\,\rangle$ ▷ emitted segments; current pack
for all $s \in S$ **do**
  if $P \neq \langle\,\rangle$ **and** $\sum_{p \in P} \tau(p) + \tau(s) > B$ **then**
    $P \leftarrow \textsc{Emit}(P, B, \mathcal{C})$ ▷ returns the unemitted remainder
  end if
  append s to P
end for
while $P \neq \langle\,\rangle$ **do** ▷ drain; EMIT may leave a remainder
  $P \leftarrow \textsc{Emit}(P, B, \mathcal{C})$
end while
return $\mathcal{C}$
function $\textsc{Emit}(P, B, \mathcal{C})$
  while $|P| > 0$ **do**
    $t \leftarrow \textsc{Join}(P)$ ▷ measure what is assembled, not the parts
    if $\tau(t) \leq B$ **or** $|P| = 1$ **then**
      append t to $\mathcal{C}$ ▷ $|P| = 1$: an oversized sentence is kept, never dropped
      return $\langle\,\rangle$
    end if
    remove the last element of P ▷ backtrack and re-measure
  end while
  return remainder of P ▷ re-packed by the caller
end function

B Method Complexity and Efficiency

Segmentation and scoring are linear in document length; importance scoring adds one embedding pass and, for support, one short LLM call per segment. Assignment over n segments and L slots is solved greedily in $O(n \log n + nL)$. Chain execution issues L worker calls; WSI retrieval is a top-m similarity lookup. Escalation adds at most one call per flagged segment. Total agent calls are L plus the number of fired escalations, reported per task in Table 5. The empirical cost of each mechanism (calls and tokens per ablation rung) is discussed in Section 5.

C Evaluation Details

Table 6 summarizes the evaluation configuration per dataset. Median document lengths are measured with the LLaMA tokenizer. All tasks use the same 200-example split (indices 0–119 for configuration selection, 120–199 held out), the same serving configuration (Section 4.1), and LongBench's per-task generation limit for every method.

Table 6: Per-dataset evaluation configuration. Median length in LLaMA tokens.

Dataset	Answer regime	Metric	Held-out size	Median length
HotpotQA (Yang et al., 2018)	extractive QA	token F1	80	14,018
Qasper (Dasigi et al., 2021)	extractive QA	token F1	80	4,753
NarrativeQA (Kočiský et al., 2018)	abstractive QA	token F1	80	31,020
QuALITY (Pang et al., 2022)	multiple choice	accuracy	80	6,666
GovReport (Huang et al., 2021)	summarization	ROUGE-L	80	8,150

Table 7: Assignment-level scoring measured where it is active (smaller k, fragmented document). Scores are not comparable to Table 4; call counts are identical within each pair. In the current implementation, placement and reading order are coupled; decoupling them is left to future work.

Task	k	Chunks/ex	Uniform	Joint	Δ
HotpotQA	8,000	2.50	52.79	47.42	-5.37
GovReport	4,000	3.74	32.60	32.16	-0.44

D SENSITIVITY TO CHUNK BUDGET

Here *reading order* is the sequence in which segments are consumed when building the running communication unit; in the current implementation it is determined by the assignment, so placement and reading order are coupled. The coupled importance/position score reorders which segment each slot receives, and its effect depends on document fragmentation. At the reported configuration the document fits a single chunk on HotpotQA and GovReport (chunks $= 1.00$), so there is only one segment to place and V0 and V1 are identical by construction—assignment cannot and should not change a single-segment input. To probe the mechanism where it is active, we reduce the per-agent window k so that documents fragment (Table 7).

Here we find that coupling importance and position with reading order, in the current fixed-chain implementation, does not by itself improve over uniform placement, because in this implementation the assignment also determines the reading order and reordering by risk trades a positional benefit against a document-order cost. Both effects are larger than the corresponding reliability thresholds of Table 3. This localizes precisely where the mechanism should be deployed: assignment-level scoring is beneficial only when placement can be decoupled from reading order and when documents fragment into several chunks, and it should be disabled on inputs that fit a single window, where uniform placement is optimal. We view this as actionable design guidance rather than a property of the scoring signal itself, and separating placement from reading order is a concrete direction for realizing the intended benefit.