KEEP THE CHAIN ALIVE: BRIDGE-SURVIVAL COMPRESSION FOR MULTI-HOP QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-hop question answering reads long retrieved contexts, so compression governs both the cost and the correctness of that reading. Current compressors, however, drop the bridge entities that connect one hop to the next, making the answer entity in the context unreachable. We propose a dual-paradigm theory of compression that organizes the scattered field into one design space. The quality paradigm attributes every compressor's losses to selection, generation, and arrangement, and identifies bridge survival as the fatal channel for multi-hop questions, one that obeys a product law and collapses exponentially with depth. The overhead paradigm prices compressor, reader, auxiliary, and amortized training, and yields a deployment criterion under which single-call compression is a necessity. We instantiate the theory as ChainPack, a single model trained to keep bridges alive through survival-filtered self-training. Across three full-scale multi-hop benchmarks, one call cuts 90% of the tokens and improves accuracy over the uncompressed context (+3.8 points, full-scale paired), stays within 4 points of the oracle bound on 3-4 hop chains, and on a stronger base model even surpasses this selection ceiling. The theory locates the common failure of existing compressors in bridge survival rather than selection, and the remaining headroom in readers rather than in better compression.

1 INTRODUCTION

Language models now support context windows approaching a million tokens (Gemini Team, 2024). Such windows are intended to hold retrieved documents (Lewis et al., 2020), and multi-hop question answering stresses them most, since the evidence for one question is spread across many documents that all enter the context at once. A large window, however, does not mean a well-used window. Reading cost grows with the amount of text put into the window, and retrieved documents are mostly not evidence for the question. The model therefore reads more text, incurs more computation, and faces more distraction. Context compression thus seeks to distill the retrieved context into a compact form before the model reads it, retaining the information the question needs and discarding the rest.

Extensive work on context compression has emerged along this line. Extractive methods such as LLMLingua select sentences by information content (Jiang et al., 2023; Pan et al., 2024). They filter noise effectively but can only retain sentences from the original context, and thus cannot reorganize evidence for the question. Abstractive methods such as BRIEF and RECOMP rewrite evidence with a generative model (Li et al., 2025; Xu et al., 2024). They adapt the evidence to the question, but a flat summary gives the reader no signal about which entity carries the reasoning from one hop to the next. We call this failure dangling. The answer remains in the compressed context, but the entity required to reach it does not, and the deeper the chain, the more likely this occurs (Figure 1, left). Iterative pipeline methods decompose the question and retrieve hop by hop (Wei et al., 2022; Zhou et al., 2023; Trivedi et al., 2023). They preserve the chain but cost an order of magnitude more LLM calls per question, and no computable criterion tells when such extra cost is worth paying.

These gaps share one root. Diagnosis exists but stays scattered. Prior studies locate individual failures such as broken references, yet no framework maps a compressor's losses to the operator that causes them, so methods are tuned end to end against reader scores, whatever is not optimized is

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.

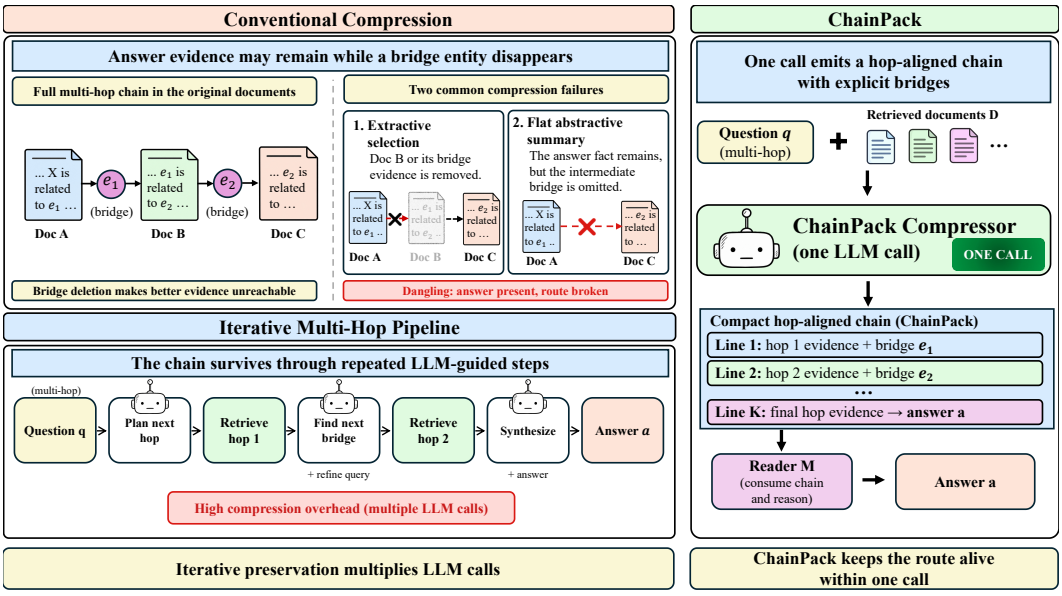

Figure 1: Motivation. Conventional compression (top left) may keep the answer while a bridge entity disappears, leaving the answer present but unreachable, and the iterative multi-hop pipeline (bottom left) preserves the chain only through repeated LLM-guided steps, multiplying the calls per question. ChainPack (right) emits a hop-aligned chain with explicit bridges in a single call, keeping the route alive.

silently given up, and cost reporting stays piecemeal, with latency and compression ratios appearing in isolation and nothing telling whether a compressor's own price is worth paying. The three failures trace back to these two absences. Extractive compression gives up the chain because information content never measures it. Abstractive compression gives it up because fluency does not say whether the reasoning survives. Pipelines incur unbounded costs because no criterion constrains them. Building a better compressor therefore requires precisely these two missing elements. Full-scale experiments confirm this conjecture and turn it into a *dual-paradigm theory of compression*. The quality paradigm attributes a compressor's failures to three operators, namely selection, generation, and arrangement, and identifies bridge survival as the fatal one for multi-hop questions. Bridge survival obeys a product law, so the deeper the chain, the faster accuracy collapses, unless survival itself is trained for. The overhead paradigm prices what a compressor may cost. It separates compressor, reader, auxiliary, and amortized training, and yields a deployment criterion under which single-call compression follows as a direct consequence (Figure 2, top).

Guided by both, we build *ChainPack*. Given a question and its retrieved documents, it generates a compact chain-structured summary where each line covers one hop, restating that step's evidence together with the bridge entity leading to the next. Training has two stages. The model first learns the chain format from labeled data, then improves on unlabeled questions by compressing, keeping only outputs in which every bridge survives, and retraining on them. The result is a single model that compresses in one call.

The single model delivers three findings at full scale. First, compression is not a tax. Under the same reader for every condition, ChainPack outperforms reading the full context, the first trained compressor to do so at this scale, and the gain concentrates exactly where the full context distracts the reader most, though the gain vanishes when interference is already low. Second, deep chains become tractable. A single call, trained for bridge survival, remains within 4 points of the oracle selection bound on 3-4 hop questions while cutting 90% of the tokens, and ablations confirm the gain rests on survival-filtered training data. Third, overhead determines deployability. One model, one call is Pareto-dominant across API, interactive, and edge settings, and on a stronger base model ChainPack even surpasses the standard oracle upper bound of reading comprehension, becoming the first compressor to exceed this selection ceiling. Across all evaluated compressors the frontier is saturated in the sense that every method sits within a few points of the oracle bound, which locates

the remaining headroom in the reader rather than in better compression. The method is robust across three architecture families and stable under full-scale hyperparameter sweeps.

In summary, we make four contributions:

- **A conjecture grounded in full-scale analysis.** Full-scale analysis maps compressor failures to three operators, identifies bridge survival as the fatal failure for deep chains, and shows that compression gains track the interference of the full context, namely the accuracy the reader loses to irrelevant documents.
- **A dual-paradigm theory of context compression.** The quality paradigm decomposes any compressor into selection, generation, and arrangement, with a product law explaining why deep chains fail. The overhead paradigm prices compression into four cost components and yields a deployment criterion favoring single-call compression.
- **ChainPack, a single-call compressor trained for bridge survival.** Survival-filtered self-training internalizes the chain into one model. The route is not obvious, because loss-level injections of the same objective through DPO and through a contrastive variant both fail, and only data-level filtering succeeds.
- **Full-scale findings and state-of-the-art results.** Compression outperforms the uncompressed context under the same reader, the compression frontier proves saturated across all evaluated methods, and ChainPack matches or outperforms every one of them while cutting 90% of the tokens. The results hold across three architecture families, compression rates, and hyperparameter sweeps.

2 Related Work

2.1 Context Compression Methods

Extractive compression selects sentences by information content. LLMLingua scores tokens by perplexity (Jiang et al., 2023), Selective Context by self-information (Li et al., 2023), and LongLLMLingua adds query-aware reordering (Jiang et al., 2024). LLMLingua-2 distills the pruning decision from a strong LLM into a compact token classifier (Pan et al., 2024). These methods are faithful to the source but need pruning labels or a proxy scorer, and cannot restate evidence for the question. Abstractive compression rewrites instead. RECOMP trains summarizers for retrieval-augmented reading (Xu et al., 2024) and BRIEF-Pro adds length-controlled question-aware summaries on a small model (Gu et al., 2026). Latent methods compress into embeddings or special tokens (Mu et al., 2023; Ge et al., 2024; Cheng et al., 2024) but bind to a specific reader and resist inspection, and summary-vector autoencoders extend the context window itself (Chevalier et al., 2023). Trained compressors distill query-relevant evidence into compact text (Yoon et al., 2024; Hwang et al., 2025). Pipelines decompose the question and retrieve hop by hop (Zhou et al., 2023; Trivedi et al., 2023; Press et al., 2023; Yun & Kim, 2026), preserving the chain at the cost of an order of magnitude more calls and error compounding across hops. Recent systems refine the selection operator itself with adaptive rates, attention guidance, or graph structure (Luo et al., 2025; Asante & Li, 2026; Yun & Kim, 2026). They remain training-free refinements of σ and do not train what the product law makes fatal, bridge survival. No existing method keeps the reasoning chain alive within a single call.

2.2 Context Utilization Analysis

A separate line studies how readers use long contexts. Lost in the Middle shows accuracy depends strongly on evidence position (Liu et al., 2024). Sufficient Context formalizes when a context holds enough information and reports failures even then (Joren et al., 2025). Recent work shows that independent selection can keep the answer while deleting the sentence defining the entity needed to reach it (Hu et al., 2026), that the gold answer may not survive into the packed context (Bala, 2026), and that compression can keep the answer correct while corrupting the reasoning chain (Ai et al., 2026). These studies establish that accuracy hides evidence-level failures, but they offer no design language. Our quality paradigm turns the observations into operators, and our full-scale survival analysis into measurable channels.

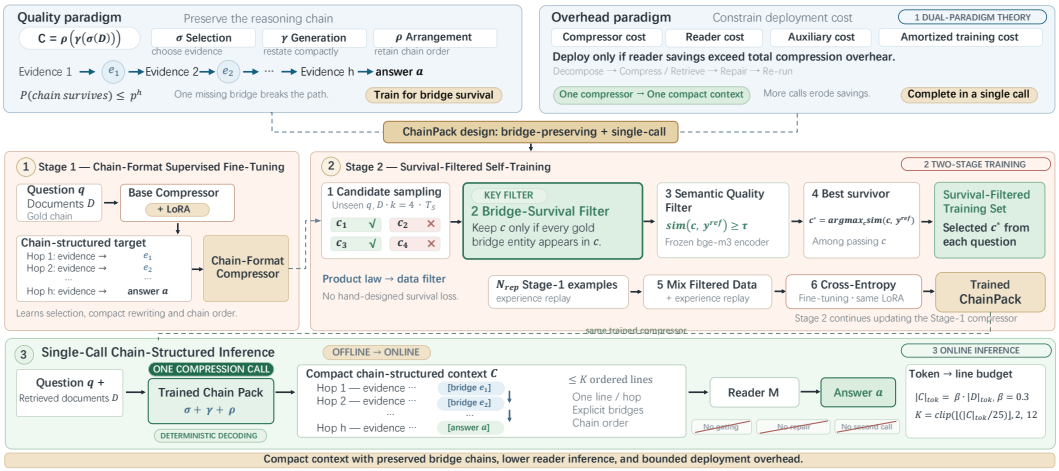

Figure 2: ChainPack overview. The dual-paradigm theory (top) requires training for bridge survival and staying within a single call. Two-stage training (middle): chain-format SFT, then survival-filtered self-training in which a candidate is kept only if every gold bridge survives and the semantic filter passes. Inference (bottom): one compression call turns the question and its retrieved documents into a hop-aligned chain that the reader answers directly.

2.3 Training Objectives and Cost Modeling

Training a compressor raises the question of what objective to optimize. The alignment literature warns that preference optimization can be gamed by its proxy reward (Rafailov et al., 2023), while rejection-sampling fine-tuning succeeds by filtering self-generated data (Yuan et al., 2023; Lightman et al., 2024; Zelikman et al., 2022). We import this lesson and find that loss-level injection of the chain-survival objective fails while data-level injection succeeds. Self-correction pipelines repair faults post hoc (Huang et al., 2024). Our threshold analysis shows why such repairs rarely pay off. On the cost side, KV-cache management (Kwon et al., 2023), heavy-hitter eviction (Zhang et al., 2023), and model routing (Ong et al., 2024) allocate or dispatch computation uniformly, orthogonal to deciding which tokens deserve to stay. A cost model for the compression pipeline itself remains missing. Our deployment criterion supplies it, and makes single-call compression a requirement rather than a preference. Our selection-generation- arrangement decomposition also echoes the classical content-selection and sentence-planning stages of natural-language generation systems (Reiter & Dale, 2000).

3 The Dual-Paradigm Theory and ChainPack

Task. Given a question q and a set of retrieved documents D, context compression produces a shorter text C from which a reader model answers q.

ChainPack is a single compressor that reads q and D and emits a compact chain-structured summary C in one call. Section 3.1 fixes notation and definitions. Section 3.2 develops the dual-paradigm theory behind the design. Sections 3.3 through 3.5 instantiate it as the compression format, the two-stage training procedure, and the inference protocol (Figure 2).

3.1 Preliminaries

Multi-hop question answering. The reader M receives the context D and must produce the answer a to q. For multi-hop questions, answering q requires composing evidence across multiple documents through bridge entities $e_1, \ldots, e_{h-1}$. The first evidence resolves e_1, which locates the next evidence, and so on until the answer is reached.

Chain survival. We call $e_1, \ldots, e_{h-1}$ the bridge entities of q and say the chain survives in a compressed context C if every e_i appears in C. Answering q from C requires the chain to survive.

Oracle upper bound. To calibrate any compressor, we construct C_{orc}, containing all gold evidence sentences followed by random filler sentences up to the 30% token budget. Reading C_{orc} never loses accuracy relative to D (Theorem B.2), so its reader accuracy is the selection ceiling for the given budget.

Low-rank adaptation. All trained components use LoRA, which inserts trainable low-rank matrices into the attention projections of a frozen model.

3.2 The Dual-Paradigm Theory

The quality paradigm: where accuracy is lost. Every compressor decomposes into three operators,

$$C = \rho\big(\gamma(\sigma(D))\big),$$

where σ selects content units from D, γ rewrites the selected units, and ρ determines the output order. Reader accuracy factors into signal extraction and resistance to noise,

$$\text{Acc}(X \mid q) = f(S_X)\, g(N_X),$$

where f is monotone nondecreasing in the signal and g is monotone nonincreasing in the noise (Assumption 1, Appendix A). Two consequences follow. First, keeping all signal never loses accuracy, and the context of all gold evidence plus random filler serves as an operational upper bound (Theorem B.2). Second, for multi-hop questions, signal is consumed through the bridge chain of Section 3.1. If each bridge survives independently with probability p, an h-hop chain survives with probability at most p^{h-1} (the product law, Theorem B.3). Answerability therefore collapses exponentially with depth unless bridge survival itself is trained for.

The overhead paradigm: what a compressor may cost. A compression system pays for its compressor, its reader, auxiliary work, and amortized training. It is worth deploying only when the reader saving exceeds these costs (Theorem B.5). Because reader cost is linear in tokens while pipeline cost is linear in calls, multi-call pipelines spend more on compression than they save at the reader. Single-call compression follows as a direct consequence. Post-hoc bridge repair faces the same arithmetic and pays only when detector precision exceeds a threshold our measurements show practical detectors do not reach (Theorem B.6).

These two paradigms jointly determine the design of ChainPack. The quality paradigm requires training for bridge survival. The overhead paradigm requires staying within a single call.

3.3 Chain-Structured Compression Format

Given a question q and documents D, ChainPack generates a summary of at most K lines. The compression ratio $\beta = 0.3$ sets the token budget $|C|_{\text{tok}} = \beta \cdot |D|_{\text{tok}}$, which is converted into a line budget as

$$K = \text{clip}(\lfloor |C|_{\text{tok}}/25 \rfloor\,,\ 2,\ 12)\,.$$

The forward pass is a single autoregressive decode conditioned on (q, D) under the chain instruction. Each output line is one hop. It restates the evidence needed for that step and names the bridge entity that carries the reasoning to the next line. This format realizes all three operators simultaneously: selection chooses the evidence per hop, generation restates it compactly, and the line order preserves the chain order.

3.4 Two-Stage Training

Stage 1: chain-format SFT. The model minimizes the token-level cross-entropy over target tokens on the labeled training set,

$$\mathcal{L}_{\text{sft}} = -\sum_{t} \log \pi_\theta(y_t \mid y_{<t},\ q,\ D),$$

where each training instance pairs a question and its retrieved documents with a gold chain constructed from the dataset annotations.

Stage 2: survival-filtered self-training. On questions unseen in Stage 1, the model samples $k = 4$ candidate compressions per question at temperature T_s. A candidate c enters the training set when

it passes two filters, and the procedure is parameterized by the sampling width k, the sampling temperature T_s, and the quality threshold τ. The survival filter requires every gold bridge entity b to appear verbatim in c, where matching is case-insensitive and ignores punctuation after whitespace normalization. The quality filter requires cosine similarity of at least τ between c and the reference summary under a frozen bge-m3 embedder. Among surviving candidates, the maximizer $c^* = \arg\max_c \mathrm{sim}(c, y^{\mathrm{ref}})$ is selected, where y^{ref} is the reference summary from the dataset annotations. The kept examples are mixed with N_{rep} Stage 1 examples as experience replay to prevent catastrophic forgetting of the chain format, and training continues with the same cross-entropy loss under the identical configuration.

This is the product law turned into a data filter. The objective maximizes bridge survival without any hand-designed loss term.

Training configuration. All stages fine-tune the same base model (Llama-3.2-3B-Instruct, from the Llama 3 family (Llama Team, AI @ Meta, 2024)) with LoRA ($r = 16$, $\alpha = 32$, attention projections only), batch 2×8 accumulation, learning rate 10^{-4} with cosine decay, for two epochs. LoRA is chosen over full fine-tuning because it preserves the base model's general language ability while adapting only the attention patterns needed for bridge survival, and because its low-rank structure matches the observation that only 0.28% of parameters are trainable yet sufficient for the task. The same pipeline, without modification, is used across all experiments.

Injection-route control. A controlled comparison holds data and hyperparameters fixed and varies only the injection route of the survival objective. Loss-level injections through DPO and through a contrastive variant both fail, while the data-level injection above succeeds.

3.5 Inference

At inference the model runs once per question with deterministic decoding (temperature T_d, nucleus p_n, at most $N_{\max}$ new tokens), which our sensitivity analysis favors, as sampling noise shows up as dropped bridges. The output length is about 10% of the input at the default budget. The compression pass adds roughly 0.5 seconds per question on a single GPU. There is no gating, no repair path, and no second call.

4 Experiments

4.1 Setup

We evaluate on three multi-hop benchmarks at full scale: 2WikiMultiHop (Ho et al., 2022) (11,281 questions), MuSiQue (Trivedi et al., 2022) (2,412), and HotpotQA validation (Yang et al., 2018) (6,947), with yes/no questions removed and no subsampling. Conditions are ChainPack-RFT (ours), LLMLingua-1/2 (official extractive compressors), LongLLMLingua (query-aware extraction), an oracle-selection upper bound (all gold sentences plus random filler at the same 30% budget, the operational form of the no-loss condition in Section 3), and the uncompressed context. RECOMP and EXIT (Xu et al., 2024; Hwang et al., 2025) target similar selection mechanisms and are represented by the LLMLingua family here. Baselines are chosen as the established, officially released methods that reproduce at our matched budget and reader. The 2025–2026 adaptive and iterative variants refine the selection operator (Section 2.1) rather than train bridge survival, and the iterative branch they represent is evaluated directly as the explicit decomposition pipeline in Section 4.3. Following RECOMP and BRIEF-Pro (Xu et al., 2024; Gu et al., 2026), whose main tables include untrained LLMs prompted to compress, among them the off-the-shelf Llama-3.2-3B-Instruct, we include the same off-the-shelf model under the identical chain instruction with no training, Vanilla Llama-3.2-3B-Instruct, which isolates the effect of fine-tuning. Because the untrained model overshoots the line budget, its outputs are additionally truncated to the length of ChainPack's output for the same question, so the comparison is budget-matched. Following standard multi-hop QA practice we score token-F1 and exact match. Every condition is read by the same reader (Qwen2.5-32B-Instruct (Qwen Team, 2024), temperature 0, locally hosted), a second reader (Gemma-2-9B (Gemma Team, 2024)) reproduces the direction of the main comparisons on a stratified subset, and significance is paired bootstrap over questions.

Table 1: Full-scale main results (F1 / EM).

Method	2Wiki	HotpotQA	MuSiQue	Avg F1
Uncompressed context	.666 / .560	**.739** / .569	.422 / .284	.609
LLMLingua-1 (Jiang et al., 2023)	.628 / .526	.737 / .568	.412 / .278	.593
LLMLingua-2 (Pan et al., 2024)	.527 / .440	.670 / .498	.364 / .233	.520
LongLLMLingua (Jiang et al., 2024)	.666 / .559	**.739** / .569	.414 / .280	.606
Vanilla Llama-3.2-3B-Instruct (budget-matched)	.484 / .397	.606 / .460	.332 / .214	.474
ChainPack (ours)	**.716 / .613**	.736 / .580	**.523 / .400**	**.658**
Oracle upper bound	.735 / .632	.773 / .608	.533 / .390	.680

Training uses only the public training splits of 2WikiMultiHopQA and MuSiQue. Stage 1 fine-tunes on 23,000 labeled examples whose gold chains are constructed from the dataset annotations. Stage 2 draws 4,500 questions unseen in Stage 1 (3,000 deep-chain MuSiQue of at least 3 hops and 1,500 from 2WikiMultiHopQA), samples four candidates per question, and keeps 2,628 after the survival and quality filters, about half of them from the deep-chain source. The kept examples are mixed with 6,000 replayed Stage 1 examples. Appendix C lists all hyperparameter values.

4.2 Main Results

Three observations. First, chain compression beats every extractive representative, including the query-aware one: ChainPack leads LLMLingua-2 by 18.9 points on 2Wiki and 15.9 on MuSiQue, the official LLMLingua-1 by 8.8 and 11.1, and query-aware LongLLMLingua by 5.0 and 10.9 (all $p < .0001$). Selecting by information content starves the bridges, with or without query awareness. Second, on two-hop benchmarks ChainPack sits within 2 points of the oracle upper bound. At this depth base-model compression is near-ceiling, which is why the deep-chain results below are where methods differentiate. Third, compression beats reading the uncompressed context. Pooling 20,640 paired questions, ChainPack wins by +3.8 F1 (paired bootstrap, $p < .0001$), to our knowledge the first trained compressor to beat the uncompressed context at this scale. The unconstrained vanilla variant clears the uncompressed context on MuSiQue and HotpotQA, which is itself direct evidence for the interference tax. The gain is not uniform, and its pattern is the subject of Section 4.3.

Rate-distortion. On a 120-question stratified probe we sweep the budget rate from 15% to 70% of the original tokens (Figure 3a). Accuracy stays at or above the uncompressed level everywhere (.534 uncompressed), with an interior optimum near 30–50% (.575–.579, against .541 at 15%): beyond 30% the line budget stops binding and the realized output stays around 93 tokens. The default operating point (30% budget, about 91% of tokens realized as cut) gains +4.1 points over the full context on this probe, on the rising edge of the gain condition of Section 3.

Vanilla baseline. At the matched budget the untrained base model collapses, reaching .484 on 2Wiki, .332 on MuSiQue, and .606 on HotpotQA, and ChainPack outperforms it by +23.2, +19.1, and +13.0 F1 respectively (paired bootstrap, 95% CIs well above zero). Survival training, not extra tokens, carries the gain. An unconstrained variant that ignores the line budget and writes 1.8 times as many tokens scores higher (.646 / .505 / .781) but still trails ChainPack on 2Wiki and MuSiQue, so the base model's conservative compression is no substitute for trained bridge survival.

4.3 Validating the Two Paradigms

Section 3.2 makes predictions for both paradigms, and we test them at full scale, one by one.

The interference tax. Define the tax as the accuracy gap between the oracle anchor and the uncompressed context: how much the reader loses to noise when reading everything. The tax is 11.0 points on MuSiQue, 6.9 on 2WikiMultiHop, and 3.4 on HotpotQA, and the compression gain tracks it almost exactly (+10.0, +4.9, -0.3): compression earns precisely the interference it removes. HotpotQA explains its own flat result: its reader already reads the full context within 3.4 points of the oracle, so there is no tax left to collect. This closes the mechanism behind Theorem B.1 and answers when compression pays.

Table 2: Cost structure per question (measured on one GPU, pipeline latency assumes 14 decomposition calls at the compression-call cost).

	Uncompressed	ChainPack	Explicit pipeline
LLM calls per question	1	2	15
Tokens read by the answering pass	992	94	94
Reader latency (s)	0.61	0.06	0.06
End-to-end latency (s)	0.61	0.56	≈7

Three failure channels. At the evidence level the channels of Section 3 behave as predicted: selection-only methods starve bridges and trail by the largest margins, flat abstractive summaries keep the answer sentence but lose the bridge (the dangling failure), and reordering against the causal chain costs 16 points in controlled comparison. Full-scale dangling rates (answer present, bridge missing) range from 8.5% to 22.1% depending on depth for conventional compression, and drop to near zero under survival-trained compression.

Deep chains. On hop-stratified MuSiQue, compression methods concentrate near the oracle upper bound: ChainPack reaches .479 on 3-hop and .449 on 4-hop (full scale), within 3.7 points of the oracle selection bound, while extractive compression stays more than 15 points below. ChainPack passes the explicit decomposition pipeline on 4-hop at a fraction of its cost (.449 vs. .399, 1 call vs 14).

The overhead paradigm: measured. The four cost components are read directly off the deployment. Reading the 992-token uncompressed context costs the reader 0.61 s, reading ChainPack's 94-token output costs 0.06 s, producing that output costs one compression call of about 0.5 s, and training is amortized to zero at inference. The deployment criterion of Theorem B.5 is therefore satisfied in this implementation, and Table 2 breaks the comparison down: against the uncompressed context the criterion is decided by the reader-side saving, which grows with context length while the compression call stays bounded, and against the explicit decomposition pipeline the single-call form dominates every setting while reaching higher quality on deep chains (.449 vs. .399 on 4-hop with 1 call instead of 14).

4.4 Ablation: Data Composition and Volume

The composition and volume of the self-distilled data are varied in Table 4 (Appendix D): removing the deep-chain source, replacing it entirely, or halving the data all move pooled F1 by at most 0.6 points, so the method is insensitive to self-distilled data composition and volume. The differences surface on hop-stratified 3-hop evaluation, where removing the deep-chain data costs accuracy relative to full RFT: deep-chain self-distillation is what the deep-chain gain rests on, and ablation granularity must respect depth.

Injection route. With data and hyperparameters fixed, DPO and contrastive injection of the survival objective both fail, while data-level injection succeeds (stratified protocol, Section 3.2). The objective must enter through the data.

4.5 Robustness and Scaling

Robustness is evaluated at full scale along the axes of base model, training hyperparameters, and decoding, with the compression-rate axis already covered by the rate-distortion analysis of Section 4.2 (Figure 3a). Figure 3b–d plots the remaining curves. The base model axis is the subject of the scaling analysis below.

No axis shows a cliff. Learning rate over two orders of magnitude and epoch counts 1–3 move pooled F1 by at most 1.1 points, and the operating point sits in the middle of each flat region rather than on a tuned peak. Deterministic decoding is best in absolute terms and preserves the most bridges (.855 vs. .836 at temperature 1.0), consistent with the product law: sampling noise is paid in bridges. The two method-level thresholds of Section 3.4, the sampling width k and the quality threshold τ, are fixed rather than swept; we return to this in the limitations.

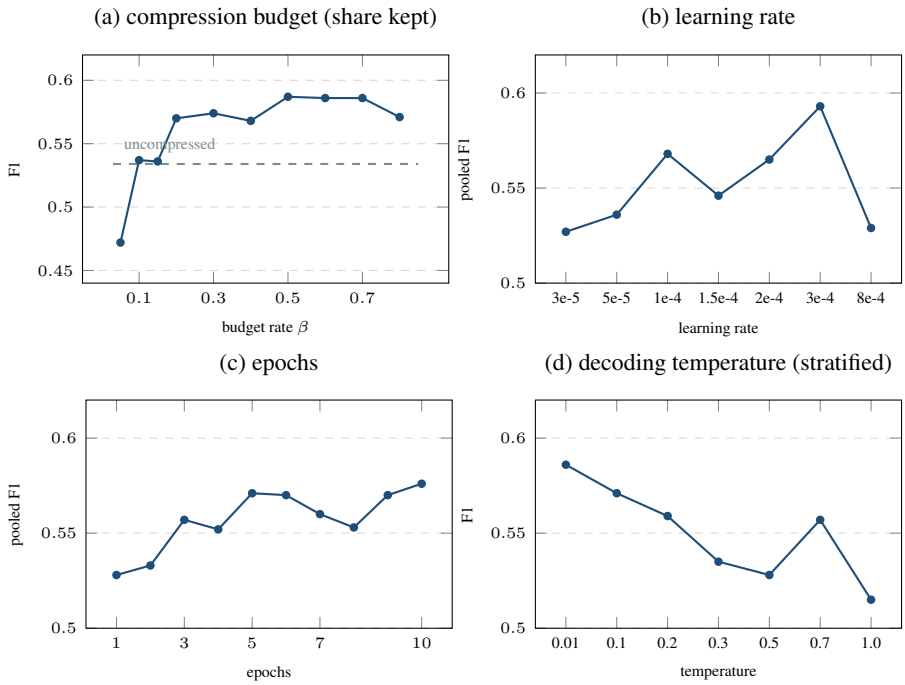

Figure 3: Sensitivity analysis. (a) Accuracy versus budget rate on the stratified probe. The dashed line is the uncompressed context, and the realized output length stays around 93 tokens beyond the 30% budget. (b)–(d) Learning rate, epochs, and decoding temperature: no axis shows a cliff.

Scaling. The same pipeline trains without modification on Qwen2.5-1.5B and Qwen3-4B. On the full 2WikiMultiHop, the Qwen3-4B instantiation reaches .745, surpassing the oracle anchor (.735) and every other condition: a stronger base turns chain compression from matching the selection ceiling into exceeding it, and the gain over the Llama-3.2-3B-Instruct instantiation (+2.9 points) indicates the method scales with the base.

5 Conclusion

We studied multi-hop context compression as a measurable design problem. The quality paradigm decomposes any compressor into selection, generation, and arrangement and locates where accuracy is lost, and the overhead paradigm prices what the compression itself may cost. Under this theory, one measurement program with three benchmarks and six systems, all local and API-free, produced three findings at full scale. Compression beats reading the full context by 3.8 points, and the gain tracks the interference the context imposes. Selection-based compression pays for its cheapness with evidence starvation. The compression frontier is saturated, locating the remaining headroom in the reader rather than in better compression. ChainPack is the theory's existence proof, a single model and one call trained under the product law for bridge survival that matches or outperforms every evaluated compressor while cutting 90% of the tokens. The theory also draws the boundary, in that post-hoc repair of broken chains cannot pay for its own detection. Limitations remain. The benchmarks are Wikipedia-style multi-hop corpora, the reader is one model family plus one replication, chain survival is an intermediate metric, the factorized-reader assumption (Appendix A) is an idealization, survival-filtered training requires bridge annotations, the sampling width k and quality threshold τ are fixed rather than swept, the 2025–2026 selection-refinement systems of Section 2 were not run at scale for lack of official implementations at our matched budget and reader, and each trained variant is a single run, so training-seed variance is unmeasured. If the reader is the bottleneck, the interesting question is no longer how to compress better but how to read better, namely compression that hands the reader a chain it can trust and readers trained to follow it.

Reproducibility Statement

All experiments run locally on a single GPU with no external API calls. The three benchmarks are public (2WikiMultiHopQA, MuSiQue, HotpotQA validation), no subsampling is applied beyond removing yes/no questions, and every condition is read by the same locally hosted reader. Appendix C lists all hyperparameters and experimental constants, Appendix D reports the data ablation, and the compression instruction is quoted in Section 3.3. The full pipeline (compression, training, and evaluation scripts), the trained adapters, and all per-question reader outputs will be released.

AI Usage Disclosure

AI coding and writing assistants were used to help develop the experimental code and to edit the manuscript text. All experimental results were produced by locally executed pipelines on public data, and the authors verified every reported number against the raw outputs. No AI system was used to generate scientific claims or to analyze the review process.

References

Mengting Ai, Jingrui He, and Yue Guo. Does accuracy equal evidence? reasoning faithfulness under kv cache compression. *arXiv preprint arXiv:2608.01631*, 2026.

Daniel Agyei Asante and Yang Li. Topocompress: Long context compression via graph-wired semantic trajectories. *arXiv preprint arXiv:2608.30811*, 2026.

Ananto Nayan Bala. Recall is not enough: A reader-context diagnostic for budget-constrained retrieval-augmented generation. *arXiv preprint arXiv:2607.00725*, 2026.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. In *Advances in Neural Information Processing Systems*, 2024.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of EMNLP*, 2023.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *Proceedings of ICLR*, 2024.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Jia-Chen Gu, Junyi Zhang, Di Wu, Yuankai Li, Kai-Wei Chang, and Nanyun Peng. Brief-pro: Universal context compression with short-to-long synthesis for fast and accurate multi-hop reasoning. In *Findings of ACL*, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Sora Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of COLING*, 2022.

Zhengpei Hu, Kai Li, Dapeng Fu, Xuechao Zou, Yuanhao Tang, Yue Li, Tengfei Cao, and Jianqiang Huang. Relevant but incomplete: Referential dangling as a paradigm-level failure mode in hard prompt compression. *arXiv preprint arXiv:2608.04569*, 2026.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *Proceedings of ICLR*, 2024.

Taeho Hwang, Soyeong Jeong, Sukmin Cho, and Jong C. Park. EXIT: Context-aware extractive compression for enhancing retrieval-augmented generation. In *Proceedings of NAACL*, 2025.

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 EMNLP*, 2023.

Huiqiang Jiang, Qianhui Wu, Chuang Pu, et al. Longllmlingua: Accelerating and enhancing large language models in long context scenarios via prompt compression. In *Proceedings of ACL*, 2024.

Hailey Joren, Jianyi Zhang, Chun-Sung Ferng, Da-Cheng Juan, Ashish Taly, et al. Sufficient context: A new lens on retrieval augmented generation systems. In *Proceedings of ICLR*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with Pagedattention. In *Proceedings of SOSP*, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020.

Yuankai Li, Jia-Chen Gu, Di Wu, Kai-Wei Chang, and Nanyun Peng. Brief: Bridging retrieval and inference for multi-hop reasoning via compression. In *Findings of NAACL*, 2025.

Yucheng Li, Bo Dong, Frank Guerin, Chenghua Lin, and Eric Cambria. Compressing context to enhance inference efficiency of large language models. In *Proceedings of EMNLP*, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *Proceedings of ICLR*, 2024.

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*, 2024.

Llama Team, AI @ Meta. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Lvzhou Luo, Yixuan Cao, and Ping Luo. Attncomp: Attention-guided adaptive context compression for retrieval-augmented generation. In *Findings of EMNLP*, 2025.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, 2023.

Isaac Ong, Amr Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph Gonzalez, Mohammad Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *arXiv preprint arXiv:2406.18665*, 2024.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Yonghui Wu, Lili Qiu, Dongmei Wang, Chin-Yew Yeh, and Reiichiro Nakano. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of ACL*, 2024.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Proceedings of ICLR*, 2023.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Pradeep Dasigi. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 2022.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Pradeep Dasigi. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of ACL*, 2023.

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

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented LMs with compressors and long-term memory. In *Proceedings of 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 EMNLP*, 2018.

Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. CompAct: Activating compact texts for question answering. In *Findings of EMNLP*, 2024.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

JungMin Yun and YoungBin Kim. Itercomp: Reasoning-aware adaptive prompt compression for multi-hop question answering. *arXiv preprint arXiv:2608.13588*, 2026.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, et al. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of ICLR*, 2023.

A Evidence Chain for the Factorized-Reader Assumption

Assumption 1 factors reader accuracy as $\mathrm{Acc}(X \mid q) = f(S_X)\, g(N_X)$ with f monotone nondecreasing in the signal and g monotone nonincreasing in the noise. It decomposes into four sub-claims of different evidential status, which we verify in turn.

Separability (strong). A 2×2 controlled experiment (gold-only versus gold-plus-noise, two readers) manipulates signal and noise independently: holding the signal fixed, adding noise alone produces a measurable accuracy change (-2.3 points for the strong reader, $+3.6$ for the weak one). Signal and noise are two independent experimental knobs whose effects can be read off separately, which is the direct operationalization of separability.

Product form (medium, with the limitation stated). Two supports. Mechanistically, answering correctly requires both extracting the signal and not being misled, and the joint probability of independent events is a product. An additive model has no such mechanism and would break the derivation of Theorem B.2. Predictively, the product form implies a monotone interference-tax relation and an interior optimal budget, whereas an additive form predicts no such interior optimum. Both predictions are confirmed at full scale. We have not run a formal model comparison (likelihood ratios) between product and additive forms. The adoption rests on mechanism and prediction discrimination, and we flag this as future work.

Monotone f (strong). The oracle anchor C_{orc} holds the signal fixed and only reduces noise. Its accuracy exceeds the full context on all three benchmarks and forms the ceiling (or near ceiling) of every compression condition. Signal never decreased and accuracy never dropped.

Monotone g (conditional, as qualified in Assumption 1). For the strong reader, adding noise costs -2.3 points (g decreasing). For the weak reader, adding noise gains $+3.6$ points, a counterexample

in which noise acts as a cue. Assumption 1 therefore conditions monotonicity on readers strong enough that noise acts as interference rather than as cues.

Independence for the product law. The bound direction (answerable implies chain survival) does not require independence. The exponential-decay direction requires weak dependence across hops. Measured survival rates (0.65–0.73 for flat compression, 0.841 for SFT, 0.865 for RFT) match the orders of magnitude predicted by p^{h-1} under weak dependence, supporting the assumption.

B Theorems and Proofs

Theorem B.1 (Gain condition). *Under Assumption 1, a compressed context C beats the original context D if and only if the relative reduction of interference exceeds the relative loss of signal, $f(S_C)/f(S_D) \geq g(N_D)/g(N_C)$.*

Proof. $\mathrm{Acc}(C) \geq \mathrm{Acc}(D)$ iff $f(S_C)\, g(N_C) \geq f(S_D)\, g(N_D)$; divide both sides by $f(S_D)\, g(N_C)$. □

Theorem B.2 (No-loss bound). *If C retains all signal of D ($S_C \supseteq S_D$) and carries no more misleading noise, then $\mathrm{Acc}(C) \geq \mathrm{Acc}(D)$.*

Proof. $S_C \supseteq S_D$ and monotone f give $f(S_C) \geq f(S_D)$; the noise condition gives $g(N_C) \geq g(N_D)$. Multiply. The oracle context of Section 3.1 instantiates C: all gold sentences, and random filler that is at most as misleading as the retrieved noise it replaces. □

Theorem B.3 (Product law). *If each of the $h-1$ bridge entities survives independently with probability p, the chain survives with probability p^{h-1}, and the fraction of h-hop questions answerable from C is at most p^{h-1}.*

Proof. Independence gives $\Pr[\text{all bridges survive}] = p^{h-1}$ directly. Chain survival is necessary for answerability, since a missing bridge leaves the answer entity unreachable by the definition of Section 3.1; the answerable fraction is therefore at most the survival probability. □

Theorem B.4 (Arrangement at zero cost). *Reordering that moves a hop's evidence across reader-attention regions changes the extraction probability of that evidence by the ratio of attended mass, at zero token cost. Measured attention is U-shaped (head-plus-tail $7.3\times$ the middle), and reordering against the causal chain costs up to 16 points.*

Proof sketch. Under a fixed attention profile, the extraction probability of a content unit scales with the attended mass at its position, so the ratio of masses bounds the change in f; the U-shape and the 16-point cost are measured. □

Theorem B.5 (Deployment condition). *A compressor is worth deploying iff the reader-side saving exceeds the sum of compressor, auxiliary, and amortized training cost: $\mathcal{S}_{\mathrm{read}} \geq C_{\mathrm{comp}} + C_{\mathrm{aux}} + C_{\mathrm{train}}/T$, where T is the number of queries served over the deployment lifetime.*

Proof. Total deployment cost is the sum of the four components by definition; deploy iff it does not exceed the reader cost it removes, which is the saving $\mathcal{S}_{\mathrm{read}}$. The call-count corollary follows since reader saving is linear in tokens while each extra compressor call adds a fixed cost linear in calls: deep multi-call pipelines add calls faster than they add saving. □

Theorem B.6 (Repair threshold). *Post-hoc bridge repair with detector precision a, applied to a fraction r of broken outputs, with repair cost c and per-repair gain G, pays only when $a \geq c/(rG)$.*

Proof. Expected gain per output is $a \cdot r \cdot G$ and expected cost is $c \cdot r$; the inequality is their difference taken nonnegative. Our measurements put practical detector precision below $c/(rG)$, so training survival at compression time dominates repairing it afterwards. □

C HYPERPARAMETERS

Table 3 lists every hyperparameter and experimental constant used anywhere in the pipeline, for training and inference alike. The values cover both training stages (chain-format SFT and survival-filtered self-training) and all robustness variants of Section 4.5, whose only differences are the learning rate, the epoch count, and the data composition swept there and in Appendix D. Inference reuses the training-time compression instruction with the fixed decoding configuration below, and every condition in the paper is evaluated with the reader configuration listed last.

Table 3: Hyperparameters and experimental constants.

Base model	Llama-3.2-3B-Instruct (Qwen2.5-1.5B / Qwen3-4B variants)
Adapter	LoRA $r = 16$, $\alpha = 32$, attention projections only
Batch	2×8 gradient accumulation
Learning rate	10^{-4}, cosine decay, 2 epochs
Stage 1 data	23,000 labeled (question, documents, gold chain)
Stage 2 pool	4,500 unseen questions (3,000 deep-chain MuSiQue + 1,500 2Wiki)
Sampling	$k = 4$ candidates, temperature $T_s = 0.8$
Quality threshold	$\tau = 0.75$ (frozen bge-m3 cosine)
Kept / replay	2,628 kept, 6,000 Stage-1 replay
Line budget	$K = \mathrm{clip}(\lfloor 0.3\,\lvert D\rvert_{\mathrm{tok}}/25 \rfloor, 2, 12)$
Decoding	temperature 0.01, nucleus 0.95, at most 420 new tokens
Reader	Qwen2.5-32B-Instruct, temperature 0 (replication: Gemma-2-9B)
Benchmarks	2Wiki 11,281 / MuSiQue 2,412 / HotpotQA 6,947 (yes/no removed)

D DATA ABLATION

The data ablation of Section 4.4 varies the composition and volume of the survival-filtered self-training set while holding everything else fixed: only-musdeep trains without the 2Wiki portion, no-musdeep without the deep-chain MuSiQue portion, and rft-half with half of the kept examples. All variants are trained with the same two-stage protocol and evaluated at full scale on the three benchmarks with the same reader as Table 1.

Table 4: Data ablation (full scale, pooled F1 over three benchmarks).

Variant	Self-distill data	MuSiQue	2Wiki	Hotpot	Pooled
RFT (full)	2wiki+mus	.523	.716	.736	.658
only-musdeep	mus only	.524	.709	.735	.656
no-musdeep	2wiki only	.512	.721	.723	.652
rft-half	half	.520	.712	.737	.656