

EVALUATING RELATIONAL CONTEXT COMPRESSION AT REALIZED TOKEN BUDGETS

Anonymous authors

Paper under double-blind review

ABSTRACT

Context compression must preserve answerable information within a token budget, yet shorter text alone cannot isolate the effect of its representation. We study Telegraph English, which rewrites retrieved passages into relational statements, using a pre-registered encoder-consumer matrix and an open compiler trained by distillation. Six of seven encoders violated the registered output schema on most or all inputs. Only 33 of 9,600 budgeted outputs fell within the registered band: at or below the requested budget and no more than two percent short. These findings motivate evaluating answer F1 at realized token ratios. The matrix was collected, but all registered confirmatory comparisons were unavailable because their comparator or calibration conditions were missing or ineligible. The remaining answer-F1 comparisons with full prose are descriptive and score construction failures as empty answers. The second compiler failed its pre-registered acceptance test against a budget-targeted pruning baseline. Its out-of-distribution component was unscorable because answer keys were absent, and citation-reach judging was not reached. Context-compression comparisons require measured token budgets, explicit construction failures, and available comparator evidence.

1 INTRODUCTION

Context compression changes both the amount of evidence a language model receives and the form in which it receives it. Deletion retains selected parts of a passage; rewriting can express the same relationships in a different syntax. The text actually delivered determines both the reader’s evidence and its token cost.

Telegraph English (TE) rewrites source passages into compact relational statements (Arbuzov et al., 2026; Bei et al., 2026). For illustration, “Orin’s adviser is Sana, and Sana works at North Lab” can become “Orin | adviser | Sana; Sana | workplace | North Lab.” The intermediate name connects the relations needed to answer where Orin’s adviser works. An encoder constructs the relations, and a separate consumer reads them to answer the question.

Token budgets introduce a second dependency into this comparison. An instruction to produce a short text specifies what the encoder should deliver; tokenization measures what it actually delivered. Two encoders given the same requested budget can therefore present different amounts of context to a reader. A study of relational form needs to follow the produced text through construction, measurement, and answering.

Only 33 of 9,600 outputs in our budgeted generation study fall within the registered 98–100% target band. The two anchors’ median lengths exceed their requests throughout the measured grid. A pre-registered encoder-consumer matrix places reading outcomes at the delivered lengths and exposes different encoder orderings across readers. Its intended matched rewrite comparison is unavailable, so these observations describe the constructed inputs relative to full prose.

A separate distilled compiler supplies the complementary case. Its fixed serializer enforces a token ceiling, but mechanically valid plans still yield low downstream answer scores: the requested-budget area is 0.0658 for the compiler against 0.3936 for the pruning baseline. Together, the studies follow compression from a requested allowance, through the text supplied, to the reader’s answer. This sequence distinguishes requesting a short input, delivering one, and preserving useful evidence within it.

2 EVALUATION DESIGN

One source, several ways to supply it to a reader. An encoder rewrites a supplied context, and a consumer answers a question using the resulting text. The matrix contains 1,200 fixed examples from MuSiQue, 2WikiMultiHopQA, and HotpotQA (Trivedi et al., 2022; Ho et al., 2020; Yang et al., 2018). Full prose is that original context, providing a common reference for the information available before rewriting. Changing the representation changes the evidence presented to the reader while preserving the question being asked.

An encoder can omit a useful fact, express it in a different form, or fail to supply text; a consumer can also fail to use information that remains. Answer accuracy measures the resulting procedure on the fixed question. The representation and its construction status identify which input produced that answer, allowing the later analysis to relate the score to delivered length and text availability.

Both studies use a modified token-overlap F1, denoted F1 below. For normalized prediction and answer token lists P, G , let $u = |\text{set}(P) \cap \text{set}(G)|$. For nonempty lists the score is

$$\text{F1}(P, G) = \frac{2u}{|P| + |G|}.$$

The numerator counts distinct shared tokens; the denominators retain repetitions, so this differs from multiset token F1. The scorer takes the highest score over the supplied answer aliases. Appendix A specifies normalization and empty-answer handling.

Encoders and consumers. The **cross-family matrix** is the pre-registered grid of fixed encoder and consumer models evaluated on these examples. Its five encoder families span four provider lineages: GPT, Claude Sonnet, Llama, Magistral, and Ministral; the last two share the Mistral lineage. The encoder instances are GPT-5.6-sol, Sonnet 5, Llama 4 Maverick, Magistral Small, and Ministral 8B. The consumers are Qwen3.5-9B, Llama3.1-8B-Instruct, and Mistral-7B-Instruct-v0.3. These models are fixed design points. Crossing encoders with consumers shows how the same construction procedure is read by different model families. Appendix B records model and construction details.

Requested and realized lengths. The **realized ratio** is the token count of the supplied representation divided by that of full prose, using the same consumer tokenizer. This context-only measure excludes the question and answer instructions. Natural-length Telegraph English receives no requested compression ratio. Budgeted Telegraph English receives a target between 0.25 and 0.70 of the full-prose context, followed by at most one token-count feedback revision. The matrix prompt is question-agnostic, task-informed, and tokenizer-neutral; the budget wrapper supplies the numerical target. Question-agnostic construction sees the source without the particular evaluation question. It must therefore select a representation for subsequent reading without using that question to choose which facts to keep. The prompt requests flat, readable relations and preservation of factual content within a fixed output envelope. Readable relations define the requested surface form; the consumer evaluation measures answer quality from the produced text.

Section 3 traces budget requests through to measured outputs.

What a representation comparison controls. Full prose and an encoder-produced prose rewrite serve different roles. Full prose anchors answerability to the original context, while a prose rewrite can control for passing the source through an encoder and imposing a length constraint. The registered direct-prose condition uses the same encoder and source as TE and targets its per-row consumer-token budget. It would hold the source, encoder, reader, and delivered token budget constant while comparing two rewriting procedures; their retained facts could still differ. This condition did not produce an eligible comparison, so the available matrix describes reader accuracy for the delivered TE inputs relative to full prose.

Failure and comparison rules. The cross-family matrix retains construction failures as empty answers with F1 zero. This measures the complete construction-and-reading procedure, including its inability to supply usable text. The natural-length table pools examples within each encoder-consumer pair; the descriptive paired contrast instead gives equal weight to each dataset, encoder,

and consumer combination. The shared full-prose observation is joined to every relevant comparison by question identifier, retaining pairing through the statistical analysis.

A separate compiler study. The open compiler tests a different route to constructing TE: a trained model emits a source-linked plan, and a fixed serializer supplies the text for reading. It has its own training data, construction rules, and acceptance test. It additionally includes QASPER research-paper questions (Dasigi et al., 2021). Its frozen score integrates F1 over the *requested* budget grid, whereas the matrix frontier uses measured ratios with requested positions assigned to construction failures. The compiler result therefore evaluates the learned construction procedure as it responds to budget requests. Its acceptance decision uses the compiler’s own paired observations; the matrix supplies a separate study of encoder-produced representations.

3 FROM BUDGET REQUESTS TO DELIVERED CONTEXT

Recovering text and satisfying the schema are separate checks. The encoder interface requests an envelope containing a relational-text field and an issues list. Six of seven encoders violate this schema on most or all inputs in the broader generation census. This census includes the matrix encoders and the smaller and larger Ministral variants. GPT-5.6-sol is the sole encoder that supplies every requested envelope correctly. The difficulty is therefore visible before a consumer answers a question.

A mechanical recovery pass extracts readable bodies from some malformed envelopes. This produces two useful counts: outputs that obey the interface and outputs that supply text for a reader. For example, Llama’s raw envelopes all violate the schema, while recovery supplies nonempty text for all its inputs (Appendix C). An available body can still omit or change source information; recovery checks availability rather than semantic fidelity. The downstream evaluation uses recovered text while retaining the original construction status.

The registered budget band tests the text, not the instruction. For a source with N context tokens, a requested ratio r sets the integer target $T = \lfloor rN \rfloor$. Let R be the token count of the decoded relational text under the tokenizer used for that request. The registered acceptance band is

$$0.98T \leq R \leq T. \quad (1)$$

The upper bound enforces the ceiling; the lower bound tests whether generation fills the requested allowance. These conditions distinguish controlling the amount of delivered text from merely staying below a maximum. The count excludes envelope syntax, question text, and reader instructions. After an initial attempt, the wrapper permits one revision using token-count feedback.

Only 33 of 9,600 budgeted outputs satisfy this band. Figure 1 shows where the parsed lengths fall across the requested grid. Both anchors produce median lengths above their requests throughout that grid. The response to a budget instruction thus preserves a substantial gap between the intended allowance and the delivered context.

The plotted lengths use the Qwen tokenizer and the last parsed attempt. Bars describe variation across outputs, rather than uncertainty about a mean. Unparsed outputs remain in the closure denominator even though they supply no length to the plot. Likewise, recovered bodies can contribute lengths while their raw envelopes remain schema violations.

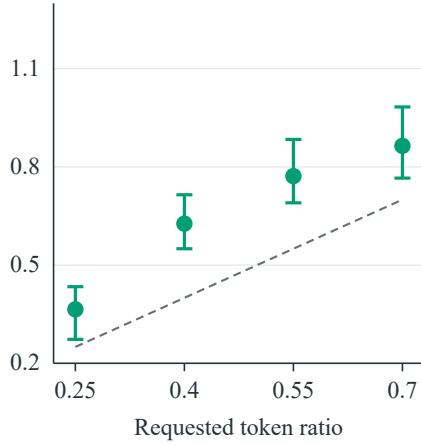
Delivered tokens determine the reading comparison. The realized ratio associates the returned text with its measured context cost under each consumer’s tokenizer. A plot against requested ratios instead describes how the procedure responds to different instructions. The next section uses measured ratios to place the matrix’s answer scores alongside delivered context length.

4 READING ACCURACY AT DELIVERED LENGTHS

The natural-length matrix resolves encoder and reader choices. Table 1 reports answer scores for each encoder–consumer pair, with full prose as the common reference.

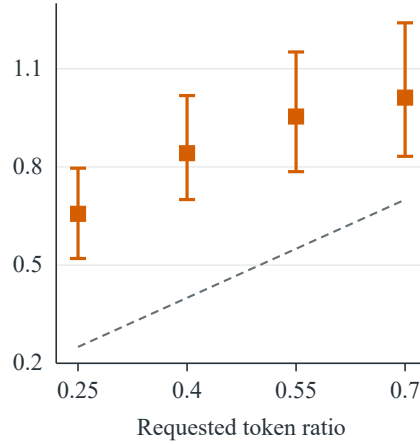
Realized token ratio

GPT-5.6 Sol



Parsed n: 1184, 1198, 1200, 1200

Sonnet 5



Parsed n: 1200, 1198, 1200, 1198

--- identity Last parsed outputs; bars show 10th–90th percentiles.

Figure 1: Do requested budgets predict realized lengths? Among parsed outputs, medians and 10th–90th percentiles exceed the identity line on the Qwen tokenizer; 33 of all 9,600 cells close within 98–100% of target.

Table 1: Descriptive natural-length modified token-overlap F1 by encoder and consumer, with full prose as the reference. Entries pool examples; construction failures receive zero F1.

Encoder	Llama F1	Mistral F1	Qwen F1
GPT-5.6-sol	0.3954	0.3909	0.5378
Llama 4 Maverick	0.3501	0.3741	0.4252
Magistral Small	0.4024	0.3754	0.5220
Ministral 8B	0.4009	0.3781	0.4897
Sonnet 5	0.3796	0.3834	0.4824
Full prose	0.4066	0.3679	0.5621

The direction relative to full prose differs across consumers. Every relational-text entry in the Mistral column lies above its full-prose reference; the entries in the Llama and Qwen columns lie below theirs. The encoder ordering also changes with the consumer. These observations make the pair, rather than either model in isolation, the unit of the descriptive comparison.

The dataset-balanced paired contrast is -0.0251 , with a 95% interval from -0.0356 to -0.0146 . This summary first compares methods on the same questions, then equally weights datasets, encoders, and consumers. The table instead pools examples within each pair, preserving the composition of the fixed sample. Appendix A gives the paired calculation and resampling procedure.

A separate consumer-input measure gives a pooled TE-to-prose token ratio of 0.8875. It includes the question and reader template and equally weights datasets, encoders, and consumers. The context-only realized ratio isolates the rewritten part of the input; the complete-prompt ratio describes its contribution to the full reader input.

Construction outcomes are part of the measured procedure. Sonnet’s natural-length condition lacks usable text on 87 of 1,200 rows, and one GPT budgeted condition fails on 16 of 1,200 rows.

Both fall below the registered construction threshold. The analysis retains these failures as empty answers and restricts the comparisons to descriptive reporting.

Realized lengths locate the available accuracy–cost observations. Figure 2 places answer F1 against measured context length for the two budgeted anchors, with a separate panel for each consumer. Each point equally weights the dataset means, unlike the example pooling in Table 1. Lines follow the sequence of budget requests and show how the delivered inputs change along that sequence. Successful constructions contribute their measured context-token ratios. Failures have no realized readable input, so the registered placement uses the requested position and F1 zero. The resulting aggregates include both outcomes under this explicit placement rule.

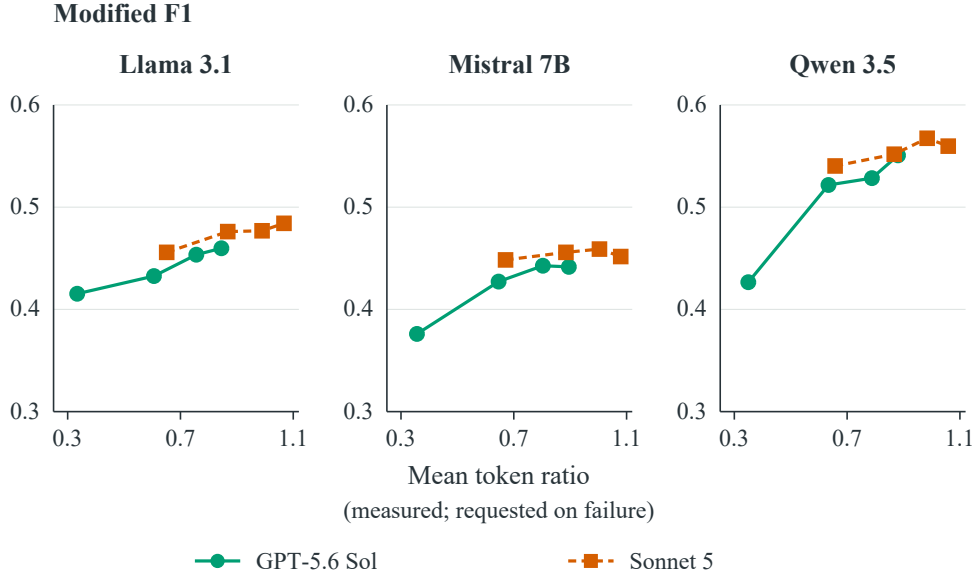


Figure 2: Dataset-balanced modified token-overlap F1 across budget requests. Horizontal means use measured context-token ratios for successful constructions and requested ratios for failures; failures also receive zero F1. Lines connect the recorded points for each encoder. Comparator curves are unavailable.

An equal-cost comparison would require comparator outcomes at corresponding delivered lengths.

Comparator purpose determines the inference. Full prose answers how a constructed input compares with the original supplied context. The registered direct-prose rewrite instead controls for encoding the same source under a length constraint, making it the required counterpart for the planned form comparison. Its development requirement failed; the pruning baseline had no matrix consumer pass; a second baseline had unresolved usage rights. The planned targeted-prompt calibration was also absent. Consequently, every registered confirmatory comparison is unavailable. Appendix D links each intended comparison to its missing prerequisite.

5 FROM TEACHER TARGETS TO AN OPEN COMPILER

The compiler study asks whether a trained open model can supply the relational representation through an explicit plan. Construction proceeds from source units to propositions with source references, then from that plan to readable TE. These stages provide distinct observations: the plan can be checked mechanically, its serialized text can be counted, and the consumer can be evaluated on that text.

Learning from grouped source units. The open Qwen3-Next-80B teacher produces plans for individual source units, which are assembled into groups before a separate call supplies dependency links. The training set contains 27,284 accepted group examples. A Qwen3-8B student learns

question-agnostic and question-conditioned plan construction; the acceptance test uses the question-agnostic output (TE-Cache). The compiler supplies relations and source references, and a fixed serializer turns them into readable Telegraph English.

The plan represents propositions and their grounding separately. A span table names intervals in the source units; proposition rows carry predicates, arguments, qualifiers, and references to those spans. Dependency rows link propositions through supported relations such as coreference, temporal precedence, or qualification. The prompt asks for preservation of negation, modality, attribution, and scope because these features can change the content a reader receives even when the main entities remain. It also keeps answer labels and downstream scores outside the plan-construction input.

Accepted groups are filtered by source and sequence limits. The training construction caps the source at 7,500 characters and 20 units, and excludes sequences beyond the training limit. Evaluation sources extend to 24,000 characters, making longer contexts a test beyond that construction envelope. The full-parameter student is trained on the accepted groups with both plan modes; Appendix F records target assembly, the earlier target diagnosis, training, and filtering rules. Training runs for two epochs with an 8,192-token maximum sequence length. Only output tokens receive labels, and question-agnostic and question-conditioned modes have equal target-token weight. The two modes share one fitted model; the acceptance comparison uses the question-agnostic compilation from that model.

A plan allowance and a text budget control different stages. The proposition allowance grows with source-unit count and character length under a fixed rule shared by training and evaluation. That rule also determines the visible-token allowance for the emitted JSON plan. It gives longer and more divided sources room to express more propositions, up to the registered cap. The consumer’s requested compression ratio enters afterward, when the completed plan is rendered as TE.

For K source units and L source characters, the implemented rule is

$$M = \text{clamp}(\max(K + 2, \text{round}(L/350)), 4, 32), \\ m = \min(K, M), \quad T_{\text{plan}} = 150M + 40K.$$

The prompt uses m , M for its proposition allowance and T_{plan} for its target-token allowance. This rule depends on source size, whereas the later reader-text budget depends on the requested ratio and consumer tokenizer.

The serializer treats the requested token budget as a ceiling. The registered rule selects the longest fitting rendering among fixed-content punctuation variants and records failure when none fits. It neither deletes propositions nor pads a short representation to meet the target. This construction can satisfy a ceiling while leaving substantial budget unused. For a fixed plan, increasing the ceiling expands the eligible renderings without rerunning content selection. The resulting text is therefore determined jointly by what the compiler put in the plan and which registered rendering fits the consumer tokenizer. Writing $\mathcal{V}(p)$ for the fixed-content renderings of plan p and τ_c for the consumer’s token count expresses the registered selection rule:

$$z^* \in \arg \max_{z \in \mathcal{V}(p): \tau_c(z) \leq B} \tau_c(z).$$

The serializer fails if no rendering fits the ceiling B . Increasing B changes this feasible set, while the propositions in p remain fixed. Thus a sweep over reader budgets measures how the existing plan is delivered; it does not ask the compiler for progressively more source facts. The comparison uses a *budget-targeted* LongLLMLingua baseline, with realized-budget equivalence unverified in the final record.

Plan validity is measurable before answer accuracy. Table 2 reports valid compilations across the evaluation sources. Each question is compiled in both training modes, so the denominator counts plans rather than distinct questions. The runtime check requires a JSON root object with the expected schema, source hash, and mode, nonempty proposition and span fields, and a proposition count within the cap. These checks establish basic construction integrity; complete span-boundary and semantic validation are separate requirements.

The in-domain sources yield mostly valid plans, while the longer QASPER inputs frequently exhaust the output ceiling. After compilation, the serializer can still reject a malformed plan row or a plan

Table 2: Valid compiler plans by dataset, pooling question-agnostic and question-conditioned modes. The out-of-envelope QASPER inputs have substantially fewer valid plans.

Dataset	Valid plans	Total plans
2WikiMultiHopQA	599	600
HotpotQA	593	600
MuSiQue	1,157	1,200
QASPER	488	1,000

whose text does not fit the requested ceiling. Answer evaluation measures the text that reaches the consumer after these stages, retaining construction failures on questions with answers as zero F1.

The compiler fails the measured acceptance criteria. Figure 3 shows the compiler and pruning baseline across the requested budget grid. The width-normalized area is 0.0658 for the compiler and 0.3936 for LongLLMLingua on the in-domain datasets. The difference is -0.328 , with a 95% interval from -0.349 to -0.307 . None of the three consumer families meets the required positive-contrast criterion. This measured contrast fails the registered acceptance rule.

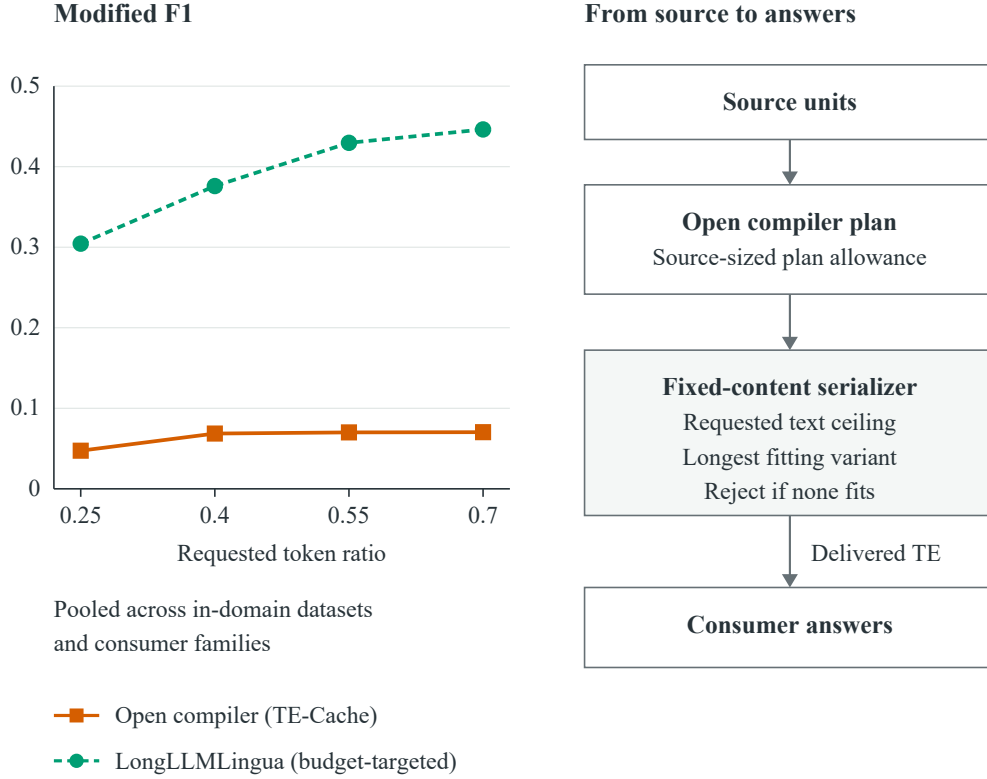


Figure 3: Modified token-overlap F1 across requested budgets for the question-agnostic compiler and the budget-targeted pruning baseline. The schematic locates the plan allowance and consumer-text ceiling in the compression procedure. The requests do not establish equal realized token counts. Appendix G reports the complete acceptance decision and unmeasured components.

The curve evaluates responses to the registered budget requests. At each request, its score averages the available in-domain datasets and consumer families; the width-normalized area summarizes those points under the frozen scoring rule. For requested ratios $r_1 < \dots < r_4$ and those mean scores $f(r_j)$,

the compiler area is

$$A = \frac{1}{r_4 - r_1} \sum_{j=1}^3 \frac{f(r_j) + f(r_{j+1})}{2} (r_{j+1} - r_j).$$

Its horizontal intervals come from the requested grid, whereas the matrix’s raw area uses measured-ratio positions with the stated failure convention. The baseline rises across the displayed requests, while the compiler’s scores remain low. Plan construction, the fixed-content serializer, and the reader all lie on the path measured by this contrast.

Two components remain unmeasured. The terminal answer file has no answers for the 500 QASPER questions, a pre-existing data defect that makes the out-of-distribution accuracy component unscorable. These cases are excluded and counted separately from construction failures on questions with answers, which receive F1 zero. Citation-reach judging was prepared but never executed. The acceptance summary in Appendix G distinguishes these missing evaluations from the measured failures.

The cost component first requires a requested-budget point where compiler accuracy matches the comparator. No point meets that prerequisite, so the study has no eligible accuracy-matched point for its reuse-cost comparison.

The retained observations connect mostly valid in-domain plans to poor answer scores after serialization and reading. Locating the lost evidence within that path requires the generated plan, delivered text and measured length, and paired reader outcome.

6 RELATED WORK

What reaches the reader. Readable compression changes the text supplied to a language model. Selective Context removes lexical units using self-information, while LLMingua allocates budgets and prunes tokens iteratively (Li et al., 2023; Jiang et al., 2023a). LongLLMingua incorporates question relevance and information positioning; LLMingua-2 learns token selection from distilled data (Jiang et al., 2024; Pan et al., 2024). RECOMP trains extractive and abstractive compressors for downstream task performance (Xu et al., 2024). Gist tokens, AutoCompressors, and the In-context Autoencoder compress context into learned activations or memory slots (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024). Token selection can directly change how much source material is retained, whereas rewriting generates a new string whose length depends on its wording. These different construction operations motivate observing each compressor’s output before treating a requested rate as a shared condition. Our setting uses inspectable text between a distinct encoder and consumer, making the delivered string and its token count observable parts of the interface.

Relational form and its construction. Telegraph English rewrites prose as compact relational statements; subsequent work compares this form with token-matched controls and post-truncated prose summaries (Arbuzov et al., 2026; Bei et al., 2026). StructGPT supplies linearized evidence from structured sources, while PICARD enforces output syntax during decoding (Jiang et al., 2023b; Scholak et al., 2021). These approaches connect a representational choice to a procedure that produces usable inputs. The present evaluation follows that connection through schema compliance, mechanically recovered text, measured length, and consumer answers. Its matrix tests several encoder–consumer pairings under fixed construction rules.

Learning a compressor from targets. Sequence-level distillation trains a student on teacher outputs, and rationale distillation adds intermediate explanations (Kim & Rush, 2016; Hsieh et al., 2023). RECOMP likewise trains a smaller compressor from generated summaries selected using downstream signals (Xu et al., 2024). The targets determine which output structures the student is taught to produce. Our compiler study examines source-unit grouping and output allowances in those targets, then evaluates the learned plan after fixed serialization. Plan validity and downstream answer quality provide separate measurements of this construction-and-reading procedure.

Which comparison a score answers. Performance-versus-budget reporting exposes resource dependence (Dodge et al., 2019); long-context studies distinguish available information from its effective use (Liu et al., 2024). LLMingua discusses generated-length control and reports actual token use alongside accuracy (Jiang et al., 2023a). Pre-registration fixes intended comparisons before

outcomes are observed (van Miltenburg et al., 2021). Building on these practices, we distinguish requested budgets, realized inputs, and the evidence needed for a representation comparison. Full prose anchors the original context; an encoder-produced prose rewrite supplies a different control. Their separate roles determine what the available answer scores can establish.

7 DISCUSSION

The matrix demonstrates that a common requested ratio does not place encoders at a common delivered length. The compiler shows the complementary limit: enforcing a text ceiling is compatible with poor answer quality.

Measure the input the consumer receives. A budget instruction specifies a desired output; the delivered representation determines the reader’s task. Relational text can be recovered from a malformed wrapper while exceeding its requested length. A serializer can respect a ceiling while leaving much of the allowance unused. These cases explain why schema compliance, text availability, and token use need separate measurements. Retaining the actual string and its consumer-specific token count makes answer quality interpretable at the input that produced it. The tokenizer and inclusion of prompt overhead also belong with the reported ratio.

Choose the control for the scientific question. Full prose asks how a constructed representation performs relative to the original supplied context. A prose rewrite from the same encoder at a comparable delivered length addresses the additional question of representational form. The matrix provides the former comparison descriptively; its registered controlled comparison remains unavailable. The compiler supplies a measured failure of its acceptance rule over requested budgets. That outcome evaluates the trained compiler, serializer, and consumers as a procedure responding to budget requests. Comparison at equal realized lengths requires the corresponding measured inputs from both arms.

Connect construction quality to answer quality. At evaluation, a valid plan establishes the implemented syntactic and reference checks; consumer answers measure the usefulness of its serialized text. Keeping construction failures in the scored denominator evaluates the complete procedure, including its ability to deliver an input. Questions without answer keys instead define an unavailable evaluation component.

Across the fixed English question-answering data and tested model versions, the studies make the path from compression request to answer score explicit. Their practical consequence is a comparison record that retains the request, returned text, measured length, construction outcome, and paired alternative. That record makes delivered evidence, rather than the requested allowance alone, the basis of the comparison.

REPRODUCIBILITY STATEMENT

The supplement contains the available numerical exports, protocols, recovered analysis sources, figure scripts, and an inventory of missing replay inputs. It supports rebuilding the reported displays; complete model-level replay additionally requires the unavailable raw examples, predictions, and trained weights. Timestamps are withheld from the anonymous version for restoration at camera-ready; the artifact chain records their order.

ETHICS STATEMENT

The study evaluates context compression on existing question-answering datasets. Its principal resource consideration is the compute used for teacher generation, compiler training, and consumer inference. Cost reporting covers the recorded component measurements.

AI USE STATEMENT

Generative models also produced teacher targets for compiler distillation, as described in Section 5. In this work, generative AI tools were used to polish the manuscript prose and assist with preparing explanatory illustrations. Generative AI tools were not used to design the model. All AI-assisted text and visual materials were reviewed and edited by the authors. The authors take responsibility for the final content of this paper, including all text, claims, and artifacts produced with AI assistance.

REFERENCES

- Mikhail L. Arbuzov, Sisong Bei, Ziwei Dong, Dmitri Kalaev, and Alexey A. Shvets. Telegraph English: Semantic prompt compression via structured symbolic rewriting, 2026. URL <https://arxiv.org/abs/2605.04426v1>. Version 1.
- Sisong Bei, Mikhail L. Arbuzov, Ziwei Dong, Dmitri Kalaev, and Alexey Shvets. Context compression is not one thing: Readable symbolic re-expression vs. coherent summary at matched budget, 2026. URL <https://arxiv.org/abs/2606.14875v1>. Version 1.
- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.232. URL <https://aclanthology.org/2023.emnlp-main.232/>.
- 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: Human Language Technologies*, pp. 4599–4610. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.365. URL <https://aclanthology.org/2021.naacl-main.365/>.
- Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2185–2194. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1224. URL <https://aclanthology.org/D19-1224/>.
- 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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/0b276510ec2d3f6613a8b60c41ff0438-Abstract-Conference.html.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th*

540 *International Conference on Computational Linguistics*, pp. 6609–6625. International Committee
541 on Computational Linguistics, 2020. doi: 10.18653/v1/2020.coling-main.580. URL <https://aclanthology.org/2020.coling-main.580/>.
542
543
544 Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner,
545 Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger
546 language models with less training data and smaller model sizes. In *Findings of the Association for*
547 *Computational Linguistics: ACL 2023*, pp. 8003–8017. Association for Computational Linguistics,
548 2023. doi: 10.18653/v1/2023.findings-acl.507. URL <https://aclanthology.org/2023.findings-acl.507/>.
549
550 Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compress-
551 ing prompts for accelerated inference of large language models. In *Proceedings of the 2023*
552 *Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376. Associ-
553 ation for Computational Linguistics, 2023a. doi: 10.18653/v1/2023.emnlp-main.825. URL
554 <https://aclanthology.org/2023.emnlp-main.825/>.
555
556 Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili
557 Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt
558 compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational*
559 *Linguistics (Volume 1: Long Papers)*, pp. 1658–1677. Association for Computational Linguistics,
560 2024. doi: 10.18653/v1/2024.acl-long.91. URL <https://aclanthology.org/2024.acl-long.91/>.
561
562 Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. StructGPT: A
563 general framework for large language model to reason over structured data. In *Proceedings of*
564 *the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9237–9251.
565 Association for Computational Linguistics, 2023b. doi: 10.18653/v1/2023.emnlp-main.574. URL
566 <https://aclanthology.org/2023.emnlp-main.574/>.
567
568 Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of*
569 *the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327.
570 Association for Computational Linguistics, 2016. doi: 10.18653/v1/D16-1139. URL <https://aclanthology.org/D16-1139/>.
571
572 Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference
573 efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods*
574 *in Natural Language Processing*, pp. 6342–6353. Association for Computational Linguistics, 2023.
575 doi: 10.18653/v1/2023.emnlp-main.391. URL <https://aclanthology.org/2023.emnlp-main.391/>.
576
577 Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and
578 Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the*
579 *Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL
580 <https://aclanthology.org/2024.tacl-1.9/>.
581
582 Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In
583 *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0848.
584 URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/3d77c6dcc7f143aa2154e7f4d5e22d68-Abstract.html.
585
586 Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor
587 Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-
588 2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings*
589 *of the Association for Computational Linguistics: ACL 2024*, pp. 963–981. Association for
590 Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.57. URL <https://aclanthology.org/2024.findings-acl.57/>.
591
592 Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally
593 for constrained auto-regressive decoding from language models. In *Proceedings of the 2021*
Conference on Empirical Methods in Natural Language Processing, pp. 9895–9901. Associ-
 ation for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.779. URL
<https://aclanthology.org/2021.emnlp-main.779/>.

594 Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Mul-
595 tihop Questions via Single-hop Question Composition. *Transactions of the Association for*
596 *Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL [https:](https://aclanthology.org/2022.tacl-1.31/)
597 [//aclanthology.org/2022.tacl-1.31/](https://aclanthology.org/2022.tacl-1.31/).
598
599 Emiel van Miltenburg, Chris van der Lee, and Emiel Krahmer. Preregistering NLP research.
600 In *Proceedings of the 2021 Conference of the North American Chapter of the Association*
601 *for Computational Linguistics: Human Language Technologies*, pp. 613–623. Association for
602 Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.51. URL [https:](https://aclanthology.org/2021.naacl-main.51/)
603 [//aclanthology.org/2021.naacl-main.51/](https://aclanthology.org/2021.naacl-main.51/).
604 Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with
605 compression and selective augmentation. In *International Conference on Learning Representations*,
606 2024. URL [https://proceedings.iclr.cc/paper_files/paper/2024/file/](https://proceedings.iclr.cc/paper_files/paper/2024/file/bda88ed2892f5e61c9a9bf215c566913-Paper-Conference.pdf)
607 [bda88ed2892f5e61c9a9bf215c566913-Paper-Conference.pdf](https://proceedings.iclr.cc/paper_files/paper/2024/file/bda88ed2892f5e61c9a9bf215c566913-Paper-Conference.pdf).
608 Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and
609 Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question
610 Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language*
611 *Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259. URL [https://aclantholo](https://aclanthology.org/D18-1259/)
612 [gy.org/D18-1259/](https://aclanthology.org/D18-1259/).
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647

A STATISTICAL PROCEDURE AND SCORE DEFINITIONS

A.1 FROZEN ANSWER SCORE

The recorded scorer normalizes prediction and answer strings by lowercasing, removing articles, removing non-word punctuation, and collapsing whitespace. Let P and G be the resulting token lists, and let $u = |\text{set}(P) \cap \text{set}(G)|$. Precision is $u/|P|$ and recall is $u/|G|$; their harmonic mean is the recorded answer F1. The maximum score over the supplied answer aliases is used. An empty normalized prediction receives zero against a nonempty normalized gold answer; the primitive returns one when both normalized lists are empty. The compiler wrapper separately forces empty or non-complete predictions to zero when gold answers exist.

This implementation uses distinct overlapping tokens in the numerator and full list lengths in the denominators. It therefore differs from ordinary multiset token F1 when a word repeats. The same frozen function is used by both studies. The reported scores preserve that implementation; no benchmark rescoring was performed for this manuscript.

A.2 MATRIX STATISTICS

The report sample consists of 600 MuSiQue, 300 2WikiMultiHopQA, and 300 HotpotQA questions. Table 1 averages all 1,200 rows separately for each encoder–consumer pair. The shared full-prose observation is evaluated once per question and consumer and joined by identifier when forming paired contrasts. This join preserves the question as the resampling unit.

For each dataset, encoder, and consumer, the descriptive contrast averages the paired TE-minus-full-prose score differences. The pooled contrast equally averages those 45 cell means. The frozen bootstrap draws 10,000 multinomial question-count vectors within each dataset using NumPy PCG64 and seed zero. Each draw applies the same counts to all paired methods, encoders, and consumers. Intervals use linearly interpolated 2.5th and 97.5th percentiles. The descriptive interval is unadjusted and carries no confirmatory test interpretation.

The pre-registered confirmatory procedure included family-level testing and Benjamini–Hochberg correction. Every registered test is unavailable because a required condition is absent or ineligible. The export assigns these tests a bookkeeping value of $p = 1.0$; this value is a status convention rather than an observed test result. Appendix D reports the statuses directly.

A.3 TOKEN ACCOUNTING AND MATRIX FRONTIERS

For question i and consumer c , let N_{ic} be the token count of its full-prose context. The requested budget is $T_{icr} = \lfloor rN_{ic} \rfloor$, with $r \in \{0.25, 0.40, 0.55, 0.70\}$. The generation wrapper accepts counts in $[0.98T_{icr}, T_{icr}]$ and permits one count-feedback revision. Figure 1 uses saved medians and percentiles of the last parsed attempts under the Qwen tokenizer. Unparsed outputs are absent from those quantiles but remain in the closure denominator.

The matrix frontier counts the decoded representation string with the named consumer tokenizer and no added special tokens. JSON syntax, question text, and reader instructions are excluded. Successful constructions contribute their measured representation-to-prose ratio; failures with no realized count contribute their requested position and F1 zero. Each Figure 2 point equally averages three dataset means from the supplied pooled display. Lines connect observed aggregates; no intermediate budget outcomes were generated.

The registered matrix area is the raw trapezoidal integral over mean token-ratio positions at the compressed grid points, using measured lengths for successes and requested positions for failures. It is computed per dataset, anchor, and consumer before the registered aggregation. The full-prose reference is excluded from this integral. Comparator areas are unavailable, so this paper displays the available curves without a comparator-ranking claim.

The separate consumer-input ratio includes the complete rendered prompt, including question and reader template. Its supplied pooled value, 0.8875, averages the 45 dataset–encoder–consumer means. The exported per-cell dictionary drops the dataset key and retains only MuSiQue entries after overwriting. Accordingly, the manuscript uses the code-verified pooled scalar and omits that

misleading per-cell table. The original export and implementation are preserved in the supplement with this defect documented.

A.4 COMPILER STATISTICS

The compiler scorer shares the answer-F1 and paired-bootstrap primitives above. At each requested ratio, it equally averages F1 over datasets with available answers and over consumers. If $f(r)$ denotes this mean, its area is

$$A = \frac{1}{r_{\max} - r_{\min}} \sum_j \frac{f(r_j) + f(r_{j+1})}{2} (r_{j+1} - r_j).$$

Here the abscissae are the requested ratios, not achieved token counts. This width-normalized score differs from the matrix’s raw token-ratio integral.

The contrast is the question-agnostic compiler area minus the available LongLLMLingua area. The same within-dataset question resample applies across arms, consumers, and budgets. Only records marked complete supply predictions; other record states receive zero when gold answers exist. QASPER records have empty answer lists and are excluded and counted separately. Their absence leaves the out-of-domain acceptance clause unscorable.

B MODELS, PROMPTS, AND CONSTRUCTION

B.1 MATRIX IMPLEMENTATION

The primary TE prompt requests one fact per line, flat readable relations, factual preservation, unambiguous abbreviations, and a fixed JSON envelope. It is question-agnostic, task-informed, and tokenizer-neutral. Its output schema is `{"curated_te": "...", "issues": []}`. The token count covers the decoded relational text, excluding envelope keys and escapes. Mechanical recovery extracts text from recognized surface forms without using answer outcomes. The primary analysis retains recovery version 2; a later recovery revision is outside the primary result.

Direct budget-conditioned prose was intended to use the same encoder and source, targeting the TE output’s consumer-token budget. It allowed count-only revisions without truncation or answer-based selection. Its development failure prevents this paired representation comparison. LongLLMLingua was intended to receive source and question; BRIEF-Pro required development-only calibration of its native sentence-count control. The required comparison outputs were unavailable for the matrix.

The five encoder families are fixed model points across four provider lineages, with Magistral and Ministral sharing a provider. The secondary Ministral size panel expands the schema census but not the primary family average. Consumer weights and tokenizers are identified by immutable revisions in the supplement. The frozen manifest records the shared prose and matrix readers’ matching code and prompt hashes.

The recovered reader implementation uses each model’s chat template, greedy generation, and a maximum of 48 new tokens. It disables optional thinking, checks for an open reasoning block, and records over-window inputs instead of truncating them. The exact external reader prompt remains hash-identified but absent from the supplied files. Recovered worker versions are preserved as archival implementation evidence rather than asserted to be identical to every earlier executed reader binary.

B.2 SOURCE SAMPLING

The fixed report sample uses deterministic hash ordering within the registered hop strata. MuSiQue contains 311 two-hop, 189 three-hop, and 100 four-hop questions. 2WikiMultiHopQA contains 100 questions in each of those strata; HotpotQA contains 300 two-hop questions. Development examples are separate and serve engineering and calibration decisions. The compiler evaluation adds 500 QASPER contexts to these 1,200 in-domain examples. Raw selected contexts and answer files are not contained in the supplied archive.

C SCHEMA COMPLIANCE AND RECOVERED TEXT

The broader generation census separates correctly formed outputs from text recovered mechanically after a schema violation. Table 3 copies the generated census counts. The correctly formed column requires a nonempty relational-text field; Sonnet also has eight well-formed but empty outputs. Recovery availability is a text-construction measure, not a validation of semantic fidelity.

Table 3: Generation census across 1,200 inputs per encoder. Correctly formed nonempty outputs and mechanically recovered nonempty text measure different construction properties.

Encoder	Correct form	Usable text
GPT-5.6-sol	1,200	1,200
Llama 4 Maverick	0	1,200
Magistral Small	2	1,200
Ministral 14B	0	1,200
Ministral 3B	0	1,182
Ministral 8B	0	1,200
Sonnet 5	202	1,113

The matrix nevertheless contains all 50,400 expected reading records across its 42 condition cells. Record presence and usable construction are separate checks. Sonnet’s natural condition has construction completeness 0.9275, with a dataset minimum of 0.8983. The GPT budgeted condition identified in the main text has completeness 0.9867. The primary analysis retains failed constructions as empty answers.

D UNAVAILABLE MATRIX COMPARISONS

The registered matrix analysis contains eleven unavailable comparisons. Table 4 groups them by the missing prerequisite while retaining the registered identifiers. Every listed comparison has status unavailable; no entry represents an executed null test.

Table 4: Registered matrix comparisons and their missing prerequisites. All listed tests are unavailable.

Registered comparison	Missing prerequisite
H1; H2a; H2b; H4a	Direct-prose construction failed development; dependent form comparisons unavailable.
H3 vs. BRIEF-Pro	Usage rights unresolved.
H3 vs. LongLLMLingua	No matrix consumer pass.
H4b; H4c	Targeted-prompt calibration not generated.
Frontier vs. direct prose	Direct-prose comparator unavailable.
Frontier vs. BRIEF-Pro	BRIEF-Pro comparator unavailable.
Frontier vs. LongLLMLingua	Pruning comparator unavailable.

H1 concerns the pooled representation contrast; H2a and H2b concern the specified encoder conditions and their contrast. H4a measures form, H4b prompt adaptation, and H4c the consumer interaction in adaptation.

E AVAILABLE FRONTIER POINTS

Table 5 reproduces the supplied dataset-balanced means used in Figure 2. The underlying per-dataset table and source summaries accompany the figure script. Failure placement and pooling follow Appendix A.

Table 5: Dataset-balanced frontier points by encoder, consumer, and requested ratio. The token-ratio column uses measured lengths for successes and requested positions for construction failures; failures receive zero modified token-overlap F1.

Encoder	Consumer	Requested	Token ratio	F1
GPT-5.6-sol	Llama	0.25	0.3338	0.4153
GPT-5.6-sol	Llama	0.4	0.6060	0.4326
GPT-5.6-sol	Llama	0.55	0.7562	0.4535
GPT-5.6-sol	Llama	0.7	0.8452	0.4598
GPT-5.6-sol	Mistral	0.25	0.3566	0.3760
GPT-5.6-sol	Mistral	0.4	0.6462	0.4272
GPT-5.6-sol	Mistral	0.55	0.8031	0.4427
GPT-5.6-sol	Mistral	0.7	0.8952	0.4416
GPT-5.6-sol	Qwen	0.25	0.3496	0.4266
GPT-5.6-sol	Qwen	0.4	0.6339	0.5217
GPT-5.6-sol	Qwen	0.55	0.7880	0.5284
GPT-5.6-sol	Qwen	0.7	0.8805	0.5507
Sonnet 5	Llama	0.25	0.6512	0.4558
Sonnet 5	Llama	0.4	0.8677	0.4761
Sonnet 5	Llama	0.55	0.9894	0.4769
Sonnet 5	Llama	0.7	1.0671	0.4842
Sonnet 5	Mistral	0.25	0.6708	0.4484
Sonnet 5	Mistral	0.4	0.8849	0.4558
Sonnet 5	Mistral	0.55	1.0038	0.4590
Sonnet 5	Mistral	0.7	1.0792	0.4517
Sonnet 5	Qwen	0.25	0.6574	0.5403
Sonnet 5	Qwen	0.4	0.8670	0.5518
Sonnet 5	Qwen	0.55	0.9838	0.5674
Sonnet 5	Qwen	0.7	1.0585	0.5597

F COMPILER TARGETS, TRAINING, AND VALIDATION

F.1 FIRST-COMPILER DIAGNOSIS

The primary diagnosis artifact measures 29,634 teacher targets, of which 14,634 are question-agnostic and 15,000 are question-conditioned. Every target has one source unit; 84.32% reach the seven-proposition cap. The Spearman correlation between proposition count and source length is 0.2988. Only 827 targets carry a dependency, and 73.26% place all cited spans in the first half of the source. The median maximum cited-span endpoint, divided by source length, is 0.432. These quantities describe training targets.

The second compiler changes target grouping, prompt and schema, assembly, and the training length distribution together. These joint changes leave the contribution of each target property unresolved.

F.2 SECOND-COMPILER CONSTRUCTION

The open teacher is Qwen3-Next-80B and the student is Qwen3-8B. The accepted training set contains 27,284 groups, comprising 15,000 question-agnostic and 12,284 question-conditioned examples. Per-unit teacher outputs are validated, assembled with a one-to-one unit mapping, linked through a separate dependency call, and validated again at group level. Dependency absence is recorded rather than rejected. Assembly relabels span and proposition identifiers while preserving source-unit mappings and remapping within-unit links. The separate dependency call receives proposition identifiers, predicates, argument surfaces, and information about units from the same document; it can add links across unit boundaries. The student receives the grouped source and produces its corresponding plan in one compilation.

Let K be the source-unit count and L its character count. The output allowance is set by

$$M = \text{clamp}(\max(K + 2, \text{round}(L/350)), 4, 32), \\ m = \min(K, M), \quad T = 150M + 40K.$$

Here M and m specify the proposition allowance and T the target-token allowance used by the prompt construction. The same rule is applied to evaluation inputs, and the supplement includes the exact implementation.

Training construction enforces a source ceiling of 7,500 characters and at most 20 units. Evaluation sources extend to 24,000 characters. The registered group-drop allowance is at most 25%; sequence-over-limit examples are excluded and logged.

The student uses full-parameter bfloat16 training with full-shard FSDP, activation checkpointing, and no packing. Maximum sequence length is 8,192; AdamW uses learning rate 2×10^{-5} , weight decay 0.1, and gradient clipping at 1.0. Training lasts two epochs with cosine scheduling and 3% warmup. Only output tokens receive labels, and the two modes have equal target-token weight. The effective batch allowance is 131,072 tokens across eight GPUs, with micro-batch one and accumulation two. The configuration records separate data-order, initialization, and dropout seeds; these are components of one fitted model, not independent training replicates. The archived recipe notes approximately $2.5\times$ more target tokens per row than for the first compiler at the same learning rate.

F.3 WHAT A VALID COMPILATION CHECKS

The recovered worker checks a JSON root object, source hash, mode, nonempty propositions, proposition cap, and a nonempty span table. Its valid-plan count does not check every span boundary or entailment relation. The counts in Table 2 measure this implemented validation over both compilation modes. The 2Wiki summary is recovered verbatim from an embedded audit input; the other dataset summaries are standalone files.

MuSiQue has 41 JSON-invalid plans and two source-hash failures. QASPER has 496 JSON-invalid plans at the output ceiling, eight excessive-proposition plans, and eight prompt-over-context failures. These leave 488 valid plans among its 1,000 mode-specific attempts. The archived HotpotQA coverage field is 1.0002, which exceeds a bounded fraction; it is not used as a coverage result here. QASPER’s separate coverage field is conditional on valid plans and is not a whole-sample success rate.

G COMPILER ACCEPTANCE AND RECORD COMPLETENESS

The pre-registered acceptance rule requires all components to pass. Table 6 states their final dispositions under the terminal scorer. The measured failures determine the overall failed decision despite the unavailable components.

Table 6: Compiler acceptance components, final status, and the supporting observation. Incomplete and unavailable components remain distinct from measured failures.

Component	Status	Observation
Pooled area	Failed	Compiler area is below the budget-targeted comparator.
Paired interval	Failed	The required positive lower endpoint is absent.
Families and transfer	Incomplete	Zero of three families pass; QASPER answers are missing.
Citation reach	Unavailable	Judge tasks were prepared but never submitted.
Open compiler	Failed	The compiler does not pass the pooled-area and interval requirements.
Matched-accuracy cost	Failed	No requested-budget point meets the prerequisite accuracy comparison.

The family component requires a positive lower endpoint of the 95% interval for at least two of three consumers and for the out-of-domain contrast. The missing QASPER answers prevent evaluation of its latter clause. The cost component tests measured reuse costs only at accuracy-matched points; its failure identifies the missing accuracy prerequisite rather than a measured cost disadvantage.

The final consumer ledger contains 66,300 records: 57,910 complete, 8,375 construction failures, 14 refusals, and one over-context record. These counts span full prose, both compiler modes, the pruning baseline, and all datasets. The 19,500 QASPER records are excluded from F1 because their answer lists are empty. All other failed record states remain in the paired scored set with zero F1.

The historical protocol requested a comparator-crash sensitivity analysis, but that output is absent from the supplied final artifacts. The terminal paired score retains comparator failures under its zero-score rule. Available component timing and cost records are provided without an amortized cost-saving conclusion.

H PROTOCOL ORDER AND ARTIFACT AVAILABILITY

The supplied protocols distinguish decisions made before answer outcomes from the terminal analysis.

Matrix order. First, the roster, sampling, neutral prompt, comparisons, and analysis contract were fixed. Engineering checks then determined comparator eligibility and mechanical text recovery. After generation, completeness was checked while answer outcomes remained sealed. A recorded decision authorized descriptive reading with construction failures retained. The terminal analysis then reported the descriptive outcomes and unavailable confirmatory tests.

Compiler order. The protocol first specified its open-compiler construction and acceptance conditions. An open-teacher amendment fixed the training-source boundary. A serializer amendment changed exact equality to a token ceiling after a check on training targets, before method-arm evaluation outputs existed. The second training recipe and grouped-target construction were fixed before that training run. Compilation, serialization, and consumer records preceded the terminal acceptance score.

Rebuildable artifacts. The reviewer supplement contains final numerical summaries, figure input extracts and scripts, protocol copies, receipt-pinned analysis sources, and recovered compiler and reader workers. It includes an extraction manifest for source files embedded in audit requests.

972 Declared release transforms remove calendar timestamps and infrastructure identifiers from re-
973 viewer copies while preserving scientific values and code behavior. Model revisions and scientific
974 configuration identifiers remain available where needed to identify an implementation.
975

976 **Missing replay inputs.** Raw question contexts, answer files, individual predictions, full teacher
977 targets, and fitted compiler weights are absent from the supplied handoff. Some exact prompt files
978 and imported rendering or serialization modules are also missing. Consequently, the supplement
979 rebuilds figures and permits inspection of the saved statistics and available implementations; it cannot
980 rerun the complete experiments from this archive alone.
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025