
Telegraph English: Semantic Prompt Compression via Structured Symbolic Rewriting

Anonymous Author(s)

Affiliation

Address

email

Abstract

We introduce Telegraph English (TE), a prompt-compression protocol that rewrites natural language into a symbol-rich, formally-structured dialect. Where token-deletion methods such as LLMingua-2 train a classifier to delete low-importance tokens at a fixed ratio, TE performs a full semantic rewrite: it decomposes the input into atomic fact lines, substitutes verbose phrases with ~ 40 logical and relational symbols, and lets the compression ratio adapt to each document’s information density. A consequence of the line-structure rule is that compression and semantic chunking become the same operation—each output line is an independently addressable fact, so the compressed representation is simultaneously a semantic index. We evaluate TE on 4,081 question-answer pairs from LongBench-v2 across five OpenAI models and two difficulty levels. At roughly 50% token reduction, TE preserves 99.1% accuracy on key facts with GPT-4.1 and outperforms LLMingua-2 at matched compression ratios on every model and task tested. The gap widens on smaller models—up to 11 percentage points on fine-detail tasks—suggesting that explicit relational structure compensates for limited model capacity. We release the grammar specification, compression prompt, benchmark data, and reference implementation.

1 Introduction

Large language models are increasingly embedded in retrieval-augmented generation (RAG), multi-agent orchestration, and long-context reasoning pipelines. Input cost scales linearly with token count, so prompt compression—feeding fewer tokens to the model while preserving the information it needs—has become a practical lever for controlling latency and cost.

Two families of approach exist. *Extractive* methods select a subset of tokens or sentences from the input [Jiang et al., 2023, Pan et al., 2024]; *abstractive* methods paraphrase or summarise it [Chevalier et al., 2023]. LLMingua-2 [Pan et al., 2024], currently the strongest published baseline, trains a GPT-4-distilled XLM-RoBERTa-large classifier to delete tokens below an importance threshold at a user-specified ratio.

Token deletion works, but it has structural limits that become visible once one looks past the compression ratio. The ratio is fixed regardless of input density. Deleting tokens can sever co-reference chains and destroy logical connectives, leaving the downstream model to hallucinate the relationships between surviving fragments. Token-deletion methods are input-only preprocessors—they compress the initial prompt, but generated output passes uncompressed to the next pipeline stage, so multi-step agent systems cannot compound the savings. Most consequentially, token deletion produces no structure: the output is a degraded copy of the input, unable to be indexed, selectively pruned, or dynamically updated.

36 We propose Telegraph English (TE), a different kind of compression. Rather than selecting which
37 tokens to keep, TE rewrites the passage into a compact, formally-structured dialect. The original
38 sentence

39 *“According to research by Johnson and colleagues (2023), the application of machine*
40 *learning techniques to medical diagnostics resulted in a 27.5% increase in early detection*
41 *rates while simultaneously reducing false positives by approximately 12% compared to*
42 *traditional methods.”*

43 becomes, under TE:

44 `ML→MEDICAL-DIAGNOSTICS: EARLY-DETECTION+27.5% ^ FALSE-POSITIVE-12%`
45 `[JOHNSON:2023]`

46 Sixty-eight tokens become fourteen. The causal relationship, both quantitative claims, and the citation
47 are each on record as separate, addressable units—and the phrase “application of...resulted in” has
48 collapsed into a single symbol.

49 What makes TE architecturally distinctive is a property that emerges from the grammar’s line-structure
50 rule: compression and semantic chunking are the same operation. Every TE output line contains
51 exactly one atomic fact—one claim, one relationship, one datum. This is not a post-processing step
52 but a consequence of how the grammar defines a legal output. The result is a representation that is
53 simultaneously compressed, retrieval-ready, and amenable to dynamic management: atomic lines are
54 individually embeddable; tagged sections support hierarchical context budgeting; and facts can be
55 updated, merged, or pruned without re-running the compressor.

56 **Contributions.** (1) A formal grammar specification for structured prompt compression (§3). (2) A
57 unified compression-and-chunking framework where semantic compression, retrieval-ready indexing,
58 and dynamic context management emerge from a single rewriting pass (§3, Appendix A). (3) A
59 large-scale empirical comparison against LLMingua-2 on 4,081 key-fact and 801 fine-detail QA
60 pairs across five models (§5). (4) Evidence that the advantage of semantic rewriting over token
61 deletion grows on smaller models and on detail-intensive tasks (§6). (5) A reference implementation
62 with CLI tools for compression, benchmarking, and error analysis.

63 2 Related Work

64 **Prompt compression.** LLMingua [Jiang et al., 2023] introduced budget-constrained prompt
65 compression using perplexity-based token selection. LLMingua-2 [Pan et al., 2024] improved on
66 this with a data-distillation approach: GPT-4 labels token importance on the MeetingBank corpus,
67 and an XLM-RoBERTa-large classifier learns to predict which tokens to delete. The compressor is
68 domain-agnostic in principle, though Pan et al. note effectiveness decreases on domains with different
69 token-importance distributions from the training data. The architectural constraint is that the output
70 remains a degraded subset of the input tokens—no new structure is introduced.

71 **Abstractive compression.** AutoCompressors [Chevalier et al., 2023] train summary tokens that
72 substitute for long contexts; RECOMP [Xu et al., 2023] generates abstractive summaries tailored to
73 retrieval queries. Both are effective but lossy by design—they discard information that cannot be
74 recovered, and neither produces a structured output that supports selective manipulation.

75 **Structured representations and agent context.** Chain-of-thought prompting [Wei et al., 2022] and
76 structured prompting [Hao et al., 2023] demonstrate that imposing structure on LLM inputs improves
77 reasoning. TE extends this insight to compression: explicit logical and relational operators help
78 downstream models reconstruct the intended meaning more reliably than degraded natural language.
79 For long-running agents, MemGPT [Packer et al., 2023] addresses context-window growth via virtual
80 memory hierarchies, and Reflexion [Shinn et al., 2023] maintains explicit self-reflection buffers;
81 both operate on natural-language representations. TE offers a complementary strategy of structured,
82 fact-level representations that can be selectively updated and pruned without further LLM calls.

83 **Semantic chunking for RAG.** Standard RAG pipelines split documents using fixed token windows
84 or sentence-boundary heuristics [Lewis et al., 2020, Gao et al., 2023]. TE sidesteps the chunking

question: compression produces atomic fact lines as a structural by-product, so no separate chunking stage is needed. Conceptually adjacent are controlled natural languages such as Attempto Controlled English [Fuchs et al., 2008], but those are designed for theorem-proving rather than compression, and consumed by formal reasoners rather than LLMs.

3 The Telegraph English Grammar

The grammar (version 5) lives in a 430-line specification document that doubles as the system prompt for the LLM-based compressor. We summarise its key design principles here; the full specification is supplementary material.

3.1 Foundations

Four principles govern the grammar, in strict priority order: (i) **fidelity over brevity**—no information may be dropped unless inferable from what remains; (ii) **atomic line structure**—each line contains exactly one claim, step, event, or question; (iii) **upper-case default**, except where case carries information (proper names, code, SI symbols); (iv) **target compression $\sim 5\times$** when feasible, but correctness, auditability, and reversibility take strict priority over token reduction.

3.2 Symbol vocabulary

TE defines a fixed vocabulary of relational and logical operators. The full set numbers roughly 40; Table 1 shows the core symbols that appear in most compressions. Each symbol has a single, non-interchangeable meaning. The grammar caps symbol density at three consecutive symbols per line—a readability constraint learned from early iterations where dense symbol chains became opaque even to GPT-4.

Table 1: Core relational and logical operators in the TE symbol vocabulary. The full vocabulary contains roughly 40 symbols organised by function (causal, logical, comparative, modal).

Symbol	Meaning	Example
=	Definition / equality	VELOCITY=DISTANCE/TIME
→	Causation / flow	HEAT→EXPANSION
⇒	Logical implication	RAIN⇒WETNESS
∴	Therefore / conclusion	X>Y ∧ Y>Z ∴ X>Z
∵	Because / reason	MOTOR-FAILURE ∵ OVERLOAD
↑/↓	Increase / decrease	TEMPERATURE↑
∧/∨/¬	And / or / not	A∧B, ¬EVIDENCE
≈/≠	Approximate / not equal	COST≈USD10M
VS	Contrast (never causal)	MODEL-A VS MODEL-B

3.3 Tags and domain conventions

Beyond the symbol vocabulary, the grammar provides a tagging system and a set of domain-specific formatting rules. Tags handle the framing that natural language carries through verbose syntactic constructions: temporal state (PAST:, NOW:, FUTURE:), modality (LIKELY:, POSSIBLE:, CONF=0.87), roles (AGENT:, PATIENT:, INSTRUMENT:), scope (CTX: for shared context), and structured content types (DEF:, Q:/A:). Each tag does double duty: it collapses verbose framing into a single token and provides the structural handle that downstream systems use for selective retrieval and context management.

Domain conventions standardise the surface forms that vary most across writers: quantities (VAR=VALUEUNIT), citations ([AUTH:YEAR], DOI:, ARXIV:), financial data (USD10.5M, Y/Y+5%, +2.5PT), and URLs. Locking these down at the grammar level removes a class of factual-error failure modes the compressor would otherwise need to handle case by case.

117 3.4 Compressor self-verification

118 Two mechanisms keep the compressor honest within a single LLM call: a quality gate and a prescribed
119 distillation sequence. The quality gate is a 12-point checklist covering formatting consistency, symbol
120 precision, abbreviation policy, number formatting, information preservation, and citation integrity.
121 It is embedded directly in the compression prompt, so the compressor self-verifies output before
122 returning it. The distillation sequence prescribes a six-pass reasoning order: (1) concept identification,
123 (2) claim extraction, (3) relation mapping, (4) redundancy elimination, (5) numerical verification,
124 (6) citation cross-checking. This is a chain-of-thought scaffold inside a single inference, not a
125 multi-call pipeline. Ordering matters: numerical verification before citation cross-checking, because
126 citations sometimes attach to numerical claims that must be confirmed first.

127 3.5 Compression as semantic chunking

128 Compression and semantic chunking are not separate stages—they are the same operation. Every
129 TE output line is an atomic fact, every section is tagged, and every CTX: block defines a scope. The
130 structure falls out of the grammar’s line-structure rule, not from any additional processing, and it
131 enables three things that token-deleted text cannot support: **selective retrieval** (a query about adverse
132 events retrieves exactly the relevant line and its scope, no chunking heuristic required); **graduated**
133 **compression-on-read** (a context-assembly system can keep the most relevant lines at full fidelity,
134 retain only heading tags for moderately relevant sections, and drop irrelevant sections entirely—no
135 LLM call needed); and **continuous state refinement** (facts can be updated in-place, merged, or
136 pruned during a session). We call this the *compress-once, manage-continuously* principle. A worked
137 example and a more detailed treatment appear in Appendix A.

138 4 Experimental Setup

139 4.1 Dataset

140 LongBench-v2 [Bai et al., 2024] supplies the source corpus: 503 long-context documents. We
141 filter to three categories suitable for factual QA—Single-Document QA, Multi-Document QA, and
142 Long-Dialogue History Understanding—which leaves 339 documents. NLTK sentence tokenisation
143 chunks each one into segments of at most 1,000 words, producing 4,081 chunk-level evaluation
144 units. The categories span technical reports, multi-source narrative synthesis, and conversational
145 history—three regimes where compression methods fail differently. The 1,000-word chunk cap
146 matches the practical input size for which prompt compression actually saves money.

147 4.2 Compression

148 Each chunk is compressed into TE using the v5 grammar prompt with OpenAI’s o4-mini model.
149 Token counts are measured with tiktoken (cl100k_base). The mean compression ratio is 0.585—a
150 41.5% token reduction—with a range from 0.13 to 1.57. The upper end deserves explanation: rare,
151 very short inputs that are already informationally dense occasionally *expand* under TE, because the
152 grammar’s fidelity-first principle prohibits dropping information even when doing so would reduce
153 token count. This is a feature, not a failure. The full distribution is shown in Figures 1 and 2.

154 For the LLMingua-2 baseline, the same chunks are compressed using the publicly available
155 llmLingua package at two retention rates: 0.50 (50% kept) and 0.33 (33% kept).

156 4.3 QA evaluation protocol

157 We design a multiple-choice protocol that isolates comprehension: can a model answer a factual
158 question correctly when reading compressed text instead of the original? GPT-4.1 generates a verbatim
159 QA pair from the original chunk, plus a semantically equivalent “modified answer” that prevents
160 simple string matching from inflating scores. GPT-4.1 (temperature 0.7) generates three plausible
161 distractors matched in style, length, and specificity. The modified answer and three distractors are
162 shuffled into a four-option question. The evaluation model sees the original, then the compressed text,
163 and selects an answer in each setting. Accuracy is the fraction of correct selections; an *error* is a case
164 where the model answered correctly on the original but incorrectly on the compressed version.

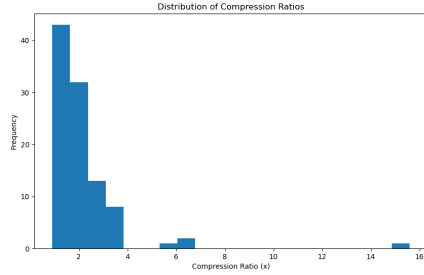


Figure 1: Distribution of compression ratios across 4,081 LongBench-v2 chunks compressed with TE (o4-mini, tiktoken cl100k_base). Mean 0.585, range 0.13–1.57. The right tail above 1.0 corresponds to short, dense inputs that expand under TE.

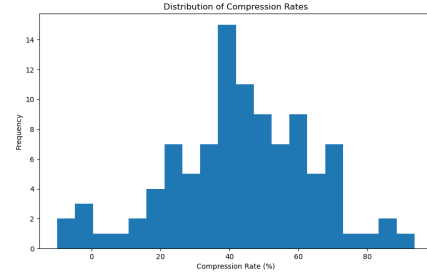


Figure 2: Distribution of per-chunk compression rate ($1 - \text{ratio}$) over the same corpus. The median chunk loses roughly 43% of its tokens; the bottom decile loses very little, reflecting TE’s adaptive behaviour on already-dense inputs.

4.4 Test suites and models

Two suites probe different levels of information preservation. **key_facts** (4,081 QA pairs) targets core concepts—headline findings, main claims, central arguments—with generically plausible distractors. **fine_facts** (801 QA pairs) is adversarially designed to target information that lossy compression is most likely to destroy: precise numerical qualifiers, conditional statements, boundary conditions, secondary details. Distractors are near-miss variants—e.g. changing 4.8% to 4.3%—that can only be distinguished with access to the exact original detail.

We evaluate five OpenAI models: GPT-4.1, GPT-4o, GPT-4o-mini, GPT-4.1-nano, and a fine-tuned GPT-4o variant. GPT-4.1 also generates the QA pairs and distractors. Different suites use different model subsets: **key_facts** is run on GPT-4.1, GPT-4o-mini, and GPT-4.1-nano; **fine_facts** on GPT-4o and GPT-4o-mini. The fine-tuned variant is reported in the cost analysis (§6.4) but is not used as a separate accuracy benchmark—it serves as a sanity check that fine-tuning on the original distribution does not change comparative behaviour at compression-decoded inputs.

5 Results

5.1 Key facts accuracy

Table 2: Accuracy on the **key_facts** suite (4,081 QA pairs). TE is Telegraph English at ~50% compression; LLML2-50 is LLMLingua-2 at 50% retention. Drop is in percentage points (pp) relative to original. **Bold** = best compressed.

Model	Original	TE	LLML2-50	TE Drop	LLML2-50 Drop
GPT-4.1	1.000	0.991	0.990	−0.9	−1.0
GPT-4o-mini	0.991	0.957	0.946	−3.4	−4.5
GPT-4.1-nano	0.980	0.950	0.949	−3.0	−3.1

On headline facts (Table 2), TE matches or edges out LLMLingua-2 across the board. The accuracy loss is negligible for GPT-4.1—less than a percentage point while halving the token count. The gap widens on smaller models: 1.1 pp on GPT-4o-mini, with the same direction at GPT-4.1-nano. Not dramatic. But consistent—the direction never reverses across configurations.

5.2 Fine facts accuracy

Fine details are harder (Table 3). Compression loss runs 3–4× higher than on key facts, regardless of method. TE holds an advantage of 3.2 pp over LLMLingua-2 on GPT-4o and 2.3 pp on GPT-4o-mini at matched 50% retention. Against more aggressive LLMLingua-2 at 33% retention (full numbers in Appendix C), TE’s lead grows to roughly 11 pp on GPT-4o-mini, where LLMLingua-2 drops a full

Table 3: Accuracy on the adversarial `fine_facts` suite (801 QA pairs). Fine-detail tasks expose larger compression effects; TE preserves more than LLMLingua-2 at matched retention.

Model	Original	TE	LLML2-50	TE Drop	LLML2-50 Drop
GPT-4o	0.996	0.965	0.933	−3.1	−6.3
GPT-4o-mini	0.938	0.843	0.820	−9.5	−11.8

21 pp from baseline. That configuration is where token deletion starts to break down: it is removing the very tokens the questions probe.

5.3 Accuracy hierarchy and compression statistics

Across all models and tasks the ranking holds without exception: original > TE > LLML2-50 > LLML2-33. TE’s mean compression ratio of 0.585 (std = 0.254) hides a wide spread: half the corpus sits between 0.41 and 0.74, with median 0.57. Documents dense with technical content or data tables resist compression; verbose narrative text yields ratios of 5:1 or better. Fidelity-first design means the ratio is an outcome, not a parameter.

5.4 Error analysis

Of the 4,081 `key_facts` items, 187 (4.6%) were correct on the original and incorrect on TE for GPT-4.1-nano. These error cases have a mean compression ratio of 0.531, slightly more compressed than the population mean—aggressive compression and error risk are correlated. Failures cluster around fine details: dates, units, conditional qualifications, and numerical relationships where TE either abbreviates a critical modifier or collapses a distinction the question specifically probes. One characteristic failure: a legal-document chunk where TE compressed “no later than 30 calendar days after receipt of written notice” into DEADLINE=30D-AFTER-NOTICE, and the question asked whether the deadline was in calendar or business days. The 30D abbreviation does not distinguish. This is a limitation of the symbol vocabulary, not a compressor error.

6 Analysis

6.1 Why semantic rewriting outperforms token deletion

Four mechanisms explain the pattern in the results. They are not ranked; different mechanisms dominate in different regimes. **Semantic-unit preservation:** token deletion operates at the token level and can split multi-word expressions, sever noun-modifier pairs, strand a number from its unit; TE works one level up, with related concepts grouped into hyphenated compounds and complete claims occupying single lines. **Explicit logical structure:** when LLMLingua-2 deletes a connective like “therefore” or “in contrast to,” the downstream model has to guess the relationship; TE refuses to offer the guess, with `∴`, `VS`, `→` each unambiguous and preserved regardless of what else is removed. **Co-reference stability:** TE’s one-claim-per-line discipline and upper-case entity naming eliminate pronoun resolution ambiguity; token deletion can strand a pronoun whose antecedent has been removed. **Adaptive compression:** a fixed-ratio method compresses dense and verbose passages identically; TE does not—dense passages emerge at ratios near 1.0, verbose ones below 0.2. The four mechanisms interlock, which is why LLMLingua-2 cannot match TE by adopting any single one of them.

6.2 The small-model effect

The TE advantage grows as model capacity shrinks. GPT-4.1 barely notices the difference between TE and LLMLingua-2 on key facts; GPT-4.1-nano and GPT-4o-mini show a wider gap, and on fine facts the divergence becomes substantial. The likely explanation is capacity-dependent. Smaller models have less ability to reconstruct implicit relationships from token-deleted fragments; TE compensates by offloading that reconstruction work to the compression stage—the evaluation model receives a representation where the relationships are already marked, rather than having to hallucinate

229 them from sparse clues. This has practical weight: smaller models are precisely the ones deployed in
230 cost-sensitive production pipelines, which is where prompt compression earns its keep.

231 6.3 The fine-facts gap

232 Key facts survive both compression methods reasonably well. Central claims are often redundantly
233 signalled, and even aggressive token deletion tends to preserve them. Fine details are stubborn in
234 a different way: precise numerical qualifiers, conditional caveats, and secondary attributions are
235 exactly the tokens an entropy-based classifier flags as low-importance in isolation. A number like
236 “4.8%” may look dispensable next to surrounding prose. But if the question asks whether the figure
237 was 4.8% or 4.3%, that token is the entire answer. TE’s claim-level decomposition and explicit
238 numerical formatting (+27.5%, CONF=0.87, Y/Y+12.3%) are designed to preserve these details:
239 numbers are never abbreviated, always attached to their units, and always placed in a structured
240 format the downstream model can parse unambiguously.

241 6.4 Pipeline-level cost

242 There is a structural difference between the two methods that the accuracy comparison alone obscures:
243 LLMingua-2 operates as an input-only preprocessor. It compresses the initial prompt; generated
244 output passes uncompressed to subsequent stages. TE can persist as a native format throughout a
245 pipeline. Consider a five-step agent pipeline with 2,000 tokens of initial context and five generation
246 steps averaging 400 tokens each, at \$10 per million tokens (Table 4). The savings compound because
247 each stage operates on TE-formatted text. LLMingua-2 compresses only the first stage’s input; the
248 remaining four stages process uncompressed output at full token cost. A more architectural treatment
249 of dynamic context management appears in Appendix B.

Table 4: Pipeline-level cost for a five-step agent pipeline (2,000-token initial context, five 400-token generations) at \$10 per million tokens. TE persists across stages; LLMingua-2 compresses only the first stage.

Method	Total tokens	Cost / 1K calls	Savings
Original	4,000	\$40	—
LLMingua-2	~3,300	\$33	\$7
Telegraph English	~1,600	\$16	\$24

250 7 Implementation

251 The reference implementation is a Python package with five pipeline stages: synchronous and
252 asynchronous (Batch API) compression of LongBench-v2 documents using the TE grammar prompt;
253 automated quality review via Claude (structured JSON scores 0–10 with strengths, weaknesses,
254 and example pairs); end-to-end QA benchmarking (generation, distractor creation, MC evaluation);
255 LLMingua-2 baseline evaluation against the same QA pairs; and error analysis with case-level
256 output. Each stage is accessible as both a CLI command and an importable library function.

257 8 Limitations

258 **LLM-dependent compression.** TE requires an LLM call per chunk, adding latency and cost at
259 compression time. This is amortised when compressed text is reused, but TE is poorly suited for
260 compressing ephemeral inputs that will be read once and discarded. **Proprietary evaluation models.**
261 Our benchmark relies on OpenAI models that are not open-weight, limiting reproducibility; future
262 work should extend evaluation to open models. **English only.** The grammar and benchmarks are
263 English; adapting the symbol vocabulary to other languages—particularly agglutinative or logographic
264 ones—is non-trivial. **Compressor model sensitivity.** TE quality depends on the model performing
265 the rewrite; we have not yet mapped this sensitivity curve. **QA generation bias.** Both QA pairs
266 and evaluations are produced by OpenAI models; an ideal evaluation would include human-written
267 questions or a diverse set of QA generators. **Dynamic context management is not yet benchmarked.**
268 The semantic chunking and dynamic state-management capabilities described in §3 and Appendix B

are architectural arguments, not empirical results from a multi-turn evaluation. We have demonstrated format-level feasibility; we have not measured downstream effects over extended sessions. This is the most important gap in the current evaluation. **Comparison scope.** We benchmark against LLMLingua-2 only—currently the strongest published baseline at our compression ratios. A broader comparison against AutoCompressors, RECOMP, and more recent methods would strengthen the claims.

9 Conclusion

Telegraph English demonstrates that structured semantic rewriting is a viable alternative to token deletion for prompt compression—and, on the evidence presented here, a better one. The advantage is largest where compression matters most practically: on smaller, cheaper models and on fine-grained details. The quantitative comparison may not be the most interesting part of this work. Token-deletion methods produce a smaller copy with no internal organisation; TE produces a representation where every line is an identified fact, every section is tagged, every relationship is marked with an explicit symbol. That structure makes the output not just smaller but more *useful*—more retrievable, more auditable, more maintainable over time. The compress-once, manage-continuously principle is, at this stage, an architectural argument rather than an empirical result; validating it in production agent systems is the obvious next step.

References

- Yushi Bai et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024.
- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of EMNLP*, 2023.
- Norbert E. Fuchs, Kaarel Kaljurand, and Tobias Kuhn. Attempto controlled english for knowledge representation. In *Reasoning Web*, volume 5224 of *Lecture Notes in Computer Science*, pages 104–124. 2008.
- Yunfan Gao et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.
- Yaru Hao et al. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713*, 2023.
- 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.
- Patrick Lewis et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.
- Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Ruhle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Chuanli Wang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of ACL*, 2024.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.
- Jason Wei et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.
- Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*, 2023.

A Compression as Semantic Chunking: Worked Example

To make §3 concrete, consider a multi-paragraph clinical-trial summary compressed into TE:

```
H1: CLINICAL-TRIAL OUTCOMES
CTX: PHASE-III RANDOMISED CONTROLLED-TRIAL (RCT); N=2400
PRIMARY-ENDPOINT: MORTALITY↓23% VS PLACEBO; p<0.001 [SMITH:2024]
SECONDARY-ENDPOINT: HOSPITALIZATION↓18%; p=0.003
ADVERSE-EVENTS: NAUSEA=12% ^ HEADACHE=8% ^ SERIOUS=2.1%
H1: SUBGROUP-ANALYSIS
AGE>65: MORTALITY↓31% (STRONGER-EFFECT)
AGE<65: MORTALITY↓14% (WEAKER-EFFECT)
CONF=0.92 FOR INTERACTION-EFFECT
H1: LIMITATIONS
FOLLOW-UP=18 MONTHS; LONG-TERM-EFFECTS UNKNOWN
EXCLUSION: PATIENTS WITH RENAL-IMPAIRMENT
```

Each line is a fact; each heading is a section boundary; each CTX: block defines a scope. The structure falls out of the grammar’s line-structure rule rather than from any additional processing, and it enables three things that token-deleted text cannot support.

Selective retrieval. A query about adverse events retrieves exactly the ADVERSE-EVENTS line and its CTX: scope. No sliding-window heuristic, no overlap parameter, no risk of splitting a relevant fact across chunk boundaries; the semantic boundaries are intrinsic to the format.

Graduated compression-on-read. When assembling a prompt under a tight token budget, an agent can apply different policies to different sections: keep the lines most relevant to the current query at full fidelity; retain only the heading tags (H1: LIMITATIONS) for moderately relevant sections, preserving topic structure at near-zero cost; drop irrelevant sections entirely. This second-stage compression is semantically principled—it operates on identified sections, not on token positions.

Continuous state refinement. During a conversation, facts from earlier turns can be revised without re-compressing the source: *update* (replace a corrected figure in place), *merge* (combine related facts when the distinction no longer matters), *prune* (remove claims that have moved past relevance), and *promote/demote* (expand a heading-collapsed section, or collapse a fully expanded one).

B Beyond Static Compression: Dynamic Context Architecture

The results in §5 measure TE as a static compression method—compress once, read once, evaluate. This is the fair comparison against LLMingua-2 and where the benchmark numbers live. But the more consequential property of TE may not be the compression ratio; it is the structure of the output.

Unifying compression and chunking. Conventional RAG systems run documents through two stages: chunking (splitting into fixed-size segments for embedding) and optional compression (reducing each chunk’s token count). These stages have different objectives and can interfere—a chunk boundary splits a sentence, then compression deletes the tokens needed to reconstruct it. TE collapses both stages into one. Each output line is a complete semantic unit; the chunking boundaries *are* the compression output. A TE-compressed document is immediately embeddable at the line level. The practical consequence for retrieval precision: fixed-window chunking inevitably includes irrelevant context within each chunk and risks splitting relevant information; TE surfaces exactly the facts a query matches, at the granularity of individual claims.

Hierarchical context budgeting. Because TE output is tagged with headings, context scopes, and role markers, a context-assembly system can make graduated decisions about inclusion. For a given token budget: full-fidelity inclusion of all atomic lines for the most relevant sections; heading-only retention for moderately relevant sections, preserving topic structure at near-zero cost; omission of irrelevant sections entirely. This graduated policy can achieve very high total compression (10–50×) when only a fraction of the document is relevant, while maintaining full detail where it matters. The policy operates on the TE output’s structure—no LLM call needed.

Dynamic state in agentic sessions. Long-running agent sessions accumulate context over many exchanges. The standard solutions are blunt: hard truncation drops the oldest tokens regardless of relevance; periodic summarisation requires an LLM call and is irreversible. TE enables something finer. Because context is already decomposed into tagged atomic facts, an agent can maintain a living state: *fact updates* replace the old line in place rather than appending alongside it; *redundancy pruning* removes facts whose information has been absorbed by later ones; *scope closure* collapses an entire CTX: block to a heading once a topic is resolved; *priority re-ranking* reorders facts by current relevance, placing the most important context where transformer attention is strongest. Context growth is controlled by continuously refining the active fact set, not by discarding the oldest tokens. This is cheap (string manipulation, no LLM calls) and semantically principled.

The cost profile is asymmetric by design: one expensive LLM rewrite per document, then indefinite cheap manipulation of the structured output.

C Full Results Tables

Table 5: Complete key_facts results with compression statistics.

Model	n	Original	TE	LLML2-50	TE Drop	LLML2-50 Drop	Mean ratio
GPT-4.1	4,081	1.000	0.991	0.990	−0.9	−1.0	0.585
GPT-4o-mini	4,081	0.991	0.957	0.946	−3.4	−4.5	0.585
GPT-4.1-nano	4,081	0.980	0.950	0.949	−3.0	−3.1	0.585

Table 6: Complete fine_facts results.

Model	n	Original	TE	LLML2-50	TE Drop	LLML2-50 Drop
GPT-4o	801	0.996	0.965	0.933	−3.1	−6.3
GPT-4o-mini	801	0.938	0.843	0.820	−9.5	−11.8

Table 7: Compression ratio statistics (tiktoken cl100k_base, $n = 4,081$ chunks).

Statistic	Value
Mean	0.585
Std	0.254
Min	0.000
25th percentile	0.407
Median	0.570
75th percentile	0.739
Max	1.567

Table 8: Error analysis: key_facts items correct on original, incorrect on TE (GPT-4.1-nano).

Statistic	Value
Total items	4,081
Error items	187
Error rate	4.58%
Mean compression ratio (errors)	0.531
Mean compression ratio (all)	0.585

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [Yes]

Justification: The abstract and §1 state four main claims: (i) TE matches or outperforms LLMingua-2 at matched compression ratios on every model and task tested; (ii) the gap widens on smaller models, up to ~11 pp on fine-detail tasks; (iii) the line-structure rule makes compression and semantic chunking the same operation; (iv) the reference implementation, grammar, and benchmark are released. Claims (i) and (ii) are supported by Tables 2 and 3 (§5) and Appendix C. Claim (iii) is the architectural argument supported in §3 and Appendix A; we mark it explicitly as architectural rather than empirical and reiterate this scoping in §Limitations. Claim (iv) is the artifact release listed in §Implementation.

Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: §Limitations enumerates seven specific limitations: LLM-dependent compression cost, reliance on proprietary OpenAI models for evaluation, English-only grammar, compressor-model sensitivity not yet characterised, QA-generation bias, no benchmark of dynamic context management (the most important gap), and comparison scope (LLMingua-2 only).

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate “Limitations” section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper is empirical and architectural; it contains no formal theorems or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: §4 specifies the source corpus (LongBench-v2), filtering criteria (three QA categories, 339 documents), chunking method (NLTK sentence tokenisation, 1,000-word cap), tokenizer (tiktoken cl100k_base), compressor model (OpenAI o4-mini), grammar prompt version (v5), baseline (LLMLingua-2 at 50% and 33% retention), evaluation models (GPT-4.1, GPT-4o, GPT-4o-mini, GPT-4.1-nano, fine-tuned GPT-4o), and the five-step QA evaluation protocol. The reference implementation reproduces all stages end-to-end (§Implementation).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We release the grammar specification (CC-BY-SA-4.0), the v5 compression prompt, the LongBench-v2-derived benchmark data (CC0), and a Python reference implementation (MIT License) covering all five pipeline stages. An anonymized snapshot of the code and benchmark data is provided with the supplementary material; the public repository link will be disclosed in the camera-ready version.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: §4 specifies the dataset split (3 LongBench-v2 categories, 339 documents, 4,081 chunks of $\leq 1,000$ words each); the QA generation protocol (GPT-4.1, temperature 0.7 for distractors); the LLMingua-2 retention rates (0.50, 0.33); and the multiple-choice evaluation protocol. No model training is performed, so there are no training-time hyperparameters or optimizer choices.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We report point estimates with sample size n (4,081 for key_facts, 801 for fine_facts) but do not report explicit error bars or confidence intervals. At these sample sizes the binomial standard error of a proportion is small (e.g. $\sigma \leq 0.005$ at $n = 4,081$, $p = 0.95$), and the gaps we report between TE and LLMingua-2 are several standard errors wide; evaluation calls use temperature 0 and are deterministic conditional on distractor placement. We did not run multiple seeds of distractor randomization to characterise that variance, which is a limitation we will address in revisions.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [No]

Justification: All experiments use OpenAI API endpoints (no local GPUs); the reference implementation runs on consumer CPUs with the OpenAI Batch API doing the heavy lifting. We discuss per-pipeline token cost in §6.4 but do not break out wall-clock time or total token volume per experiment in the main paper. These details will be added in the camera-ready version.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics. The work involves no human subjects, no privacy-sensitive data, no deployed system, and no scraping beyond the publicly released LongBench-v2 corpus (used under its license). Anonymity is preserved in this submission.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: TE is a foundational efficiency method for LLM input compression and does not involve a deployed system, generative content, or new data collection. Positive impact is reduced compute / cost / latency for retrieval-augmented and agent pipelines; negative impacts are no greater than those generic to any LLM efficiency improvement (e.g. enabling lower-cost downstream applications, both beneficial and harmful). We do not believe the method introduces a unique societal risk warranting a dedicated impact section.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We release no pre-trained models, no image-generation assets, and no scraped data. The released artifacts are a grammar specification, a compression prompt, and a benchmark derived from the publicly available LongBench-v2 corpus.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: LongBench-v2 [Bai et al., 2024] is cited and used under its release terms. LLMingua-2 [Pan et al., 2024] is used via the publicly available llmLingua Python package, with citation. OpenAI models (GPT-4.1, GPT-4o, GPT-4o-mini, GPT-4.1-nano, fine-tuned GPT-4o) are accessed through the public OpenAI API in compliance with its terms of service.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

685 **13. New assets**

686 Question: Are new assets introduced in the paper well documented and is the documentation
687 provided alongside the assets?

688 Answer: [Yes]

689 Justification: The grammar specification (CC-BY-SA-4.0), v5 compression prompt, derived
690 benchmark data (CC0), and reference implementation (MIT License) are documented in
691 §Implementation and provided in the supplementary materials with usage instructions,
692 license files, and per-stage CLI commands.

693 Guidelines:

- 694 • The answer [N/A] means that the paper does not release new assets.
- 695 • Researchers should communicate the details of the dataset/code/model as part of their
696 submissions via structured templates. This includes details about training, license,
697 limitations, etc.
- 698 • The paper should discuss whether and how consent was obtained from people whose
699 asset is used.
- 700 • At submission time, remember to anonymize your assets (if applicable). You can either
701 create an anonymized URL or include an anonymized zip file.

702 **14. Crowdsourcing and research with human subjects**

703 Question: For crowdsourcing experiments and research with human subjects, does the paper
704 include the full text of instructions given to participants and screenshots, if applicable, as
705 well as details about compensation (if any)?

706 Answer: [N/A]

707 Justification: The work involves no crowdsourcing and no human subjects. All evaluation is
708 automated through API calls to OpenAI models on the LongBench-v2 corpus.

709 Guidelines:

- 710 • The answer [N/A] means that the paper does not involve crowdsourcing nor research
711 with human subjects.
- 712 • Including this information in the supplemental material is fine, but if the main contribu-
713 tion of the paper involves human subjects, then as much detail as possible should be
714 included in the main paper.
- 715 • According to the NeurIPS Code of Ethics, workers involved in data collection, curation,
716 or other labor should be paid at least the minimum wage in the country of the data
717 collector.

718 **15. Institutional review board (IRB) approvals or equivalent for research with human
719 subjects**

720 Question: Does the paper describe potential risks incurred by study participants, whether
721 such risks were disclosed to the subjects, and whether Institutional Review Board (IRB)
722 approvals (or an equivalent approval/review based on the requirements of your country or
723 institution) were obtained?

724 Answer: [N/A]

725 Justification: No human subjects are involved; IRB review is not applicable.

726 Guidelines:

- 727 • The answer [N/A] means that the paper does not involve crowdsourcing nor research
728 with human subjects.
- 729 • Depending on the country in which research is conducted, IRB approval (or equivalent)
730 may be required for any human subjects research. If you obtained IRB approval, you
731 should clearly state this in the paper.
- 732 • We recognize that the procedures for this may vary significantly between institutions
733 and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the
734 guidelines for their institution.
- 735 • For initial submissions, do not include any information that would break anonymity (if
736 applicable), such as the institution conducting the review.

737 **16. Declaration of LLM usage**

738 Question: Does the paper describe the usage of LLMs if it is an important, original, or
739 non-standard component of the core methods in this research? Note that if the LLM is used
740 only for writing, editing, or formatting purposes and does *not* impact the core methodology,
741 scientific rigor, or originality of the research, declaration is not required.

742 Answer: [\[Yes\]](#)

743 Justification: LLMs are central to the methodology and are explicitly described in §3 and
744 §4: OpenAI’s o4-mini performs the TE compression rewrite using the v5 grammar prompt;
745 GPT-4.1 generates the QA pairs and distractors; GPT-4.1, GPT-4o, GPT-4o-mini, GPT-4.1-
746 nano, and a fine-tuned GPT-4o variant evaluate on compressed and uncompressed inputs.
747 All model identifiers, prompt versions, and roles are specified in the paper.

748 Guidelines:

- 749 • The answer [\[N/A\]](#) means that the core method development in this research does not
750 involve LLMs as any important, original, or non-standard components.
- 751 • Please refer to our LLM policy in the NeurIPS handbook for what should or should not
752 be described.