Every Time I Hire a Linguist, Inference Costs Go Down: On Linguistic Rules as Effective Prompt Compressors

Anonymous ACL submission

Abstract

Prompt compression shortens LLM input to reduce inference cost, yet existing methods score token importance through LM forward passes. It remains questionable whether such nuanced, costly token selection is necessary. Compression requires identifying informative content, which linguistics has long explored via various inference-free linguistic rules. We therefore ask: can **linguistic rules alone** serve as effective prompt compressors, without LM-based scoring at compression time?

To address this, we conduct offline evolutionary search over lexical, syntactic, semantic, and discourse seeds to discover competitive rule combinations for a linguistic rule-based compressor. At deployment, it requires no LM forward pass and lightweight CPU-side processing. We evaluate it via a dual-path protocol balancing compression quality and reconstruction fidelity.

Across datasets spanning short passages, multi-document reasoning, and dialogue memory, evolved compressors match the performance of strong LM-based baselines on most settings, especially on long datasets. Evolutionary analysis reveals that effective compression fuses signals across linguistic levels and, as the compression ratio increases, rules shift from token-level pruning to sentence-level extraction. Code and data are available at: `https://anonymous.4open.science/r/LingComp2-0C80`

1 Introduction

Large language models (LLMs) are increasingly deployed in long-context applications like retrieval-augmented generation (RAG) (Lewis et al., 2020), multi-agent communication (Li et al., 2023a), and long-term dialogue (Wu et al., 2025), where massive contextual data often overshadows the core task instruction. This drives up inference costs, increases latency, and degrades model comprehension by diluting key information. Hard prompt compression (Li et al., 2025a), which shortens the input by pruning tokens while preserving natural language readability, has emerged as a standard remedy. Yet the dominant paradigm relies on language model (LM) based scoring, whether via perplexity or trained classifiers, to assess token importance (Jiang et al., 2023; Li et al., 2023b; Pan et al., 2024; Raiyan et al., 2026). This raises two critical concerns: (1) such scoring is computationally expensive, forward pass-dependent, and inherently opaque; (2) token-level granularity risks severing syntactic dependencies and semantic coherence. Given these drawbacks, a natural question arises: is fine-grained, per-token model scoring truly the only path to effective compression?

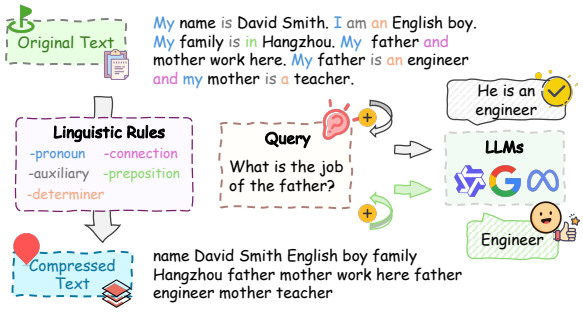

Figure 1: Illustration of linguistic rule-based prompt compression. By following specific linguistic rules, the original text is transformed into a compressed form. LLMs can still correctly answer the same query from both the original and compressed inputs.

Linguistics has long studied these signals: content-versus-function word distinctions at the lexical level (Shannon, 1948; Zipf, 2016), argument structure at the syntactic level (Tesnière, 2015), negation and quantifier scope at the semantic level (Horn, 1989; Barwise and Cooper, 1981), and topic sentence placement and entity coherence at the discourse level (Halliday and Hasan, 2014). These can be extracted by standard NLP tools at low cost, and each is an explicit, inspectable property rather than a vague score directly gener-

ated by models. Furthermore, by operating over structural units such as phrases and clauses, linguistic analysis can better preserve contextual integrity. Yet linguistic rules have never been systematically explored for prompt compression. Even PartPrompt (Mao et al., 2026), which builds compression on syntactic parse trees, operates at a single linguistic level. We therefore pose a fundamental question: *can linguistic rules alone serve as effective prompt compressors?*

However, these rules originate from distinct subfields and research traditions. No unified inventory exists, nor a principled approach to their composition, and the cross-level combinatorial space is prohibitively large for manual search.

We address the issue via an offline evolutionary program search. It starts from 42 seeds that encode explicit lexical, syntactic, semantic, and discourse rules, alongside composite pipelines chaining these rules. Powered by AlphaEvolve[1] (Novikov et al., 2025), LLMs act as mutatiors to iteratively refine rules and discover novel cross-level combinations. The search converges on an evolved linguistic rule-based compressor, implemented as a Python program. During deployment, it relies exclusively on the standard spaCy pipeline (Honnibal et al., 2020) and requires no LM forward pass.

We evaluate across four datasets encompassing short passages, multi-document reasoning, and dialogue memory, via a dual-path protocol testing both directly compressed and reconstruction-based QA. On medium-to-long documents, our evolved compressors match strong LM-based baselines on most settings without LM forward pass at deployment and substantially lower computational costs. They also achieve the lowest cross-model variance, suggesting more consistent transfer across three tested receiver LLMs. Additionally, our evolutionary analysis shows that the best compressors found by our search consistently fuse signals across multiple linguistic levels, while the dominant compression unit shifts from token-level pruning to sentence-level extraction as budgets tighten. In summary, our work makes three contributions:

- **Framework.** To our knowledge, we introduce one of the first prompt compression frameworks centered on explicit linguistic rules and evolved via LLM-guided search.
- **Empirical findings.** Empirical evidence that rule-based compressors can match strong LM-based baselines on most longer-document settings while offering superior cross-model stability and computational efficiency.
- **Analytical insights.** Effective compression fuses multiple linguistic levels, and the optimal granularity shifts from tokens to sentences as compression deepens.

2 Related Work

2.1 Prompt compression

Prompt compression reduces input length to lower LLM inference cost while preserving task-relevant information (Nagle et al., 2024; Zheng et al., 2025; Li et al., 2025a; Kummer et al., 2026). *Soft prompt methods* (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024; Li et al., 2025b; Cheng et al., 2024) embed prompts into continuous embeddings, yet their outputs are confined to latent spaces, being neither human-readable nor reusable across LLMs.

Hard prompt methods produce readable compressed text by pruning or rewriting tokens. Selective Context (Li et al., 2023b) pioneered this direction by estimating token self-information via a causal LM to drop low-information units. Building on this, LLMLingua (Jiang et al., 2023) added coarse-to-fine budget control with iterative token-level pruning, and LongLLMLingua (Jiang et al., 2024) extended it to long contexts. LLMLingua-2 (Pan et al., 2024) further reframed compression as token classification with a distilled encoder. Selection-p (Chung et al., 2024) replaced distillation with self-supervised continual pre-training for transferability. FrugalPrompt (Raiyan et al., 2026) and Lin et al. (2025) computed token saliency via train-free attention-based attribution. A parallel line explores query-conditioned compression, in which retained content adapts to the downstream task (Xu et al., 2024; Jung and Kim, 2024; Chuang et al., 2024; Łajewska et al., 2025; Liskavets et al., 2025). Despite this diversity, they still need an LM forward pass to judge token importance.

Notable exceptions are PartPrompt (Mao et al., 2026) and Telegraph English (Arbuzov et al., 2026), which respectively leverage syntactic parse trees and symbolic replacement to conduct compression. However, the former still relies on LM-computed scores for node selection. The latter assumes the LLM can accurately restore special symbols and requires design of the symbol-word mapping. To avoid dependence on LM scoring, massive manual

[1]We employ the evolutionary algorithm sourced from OpenEvolve, an open-source implementation of AlphaEvolve.

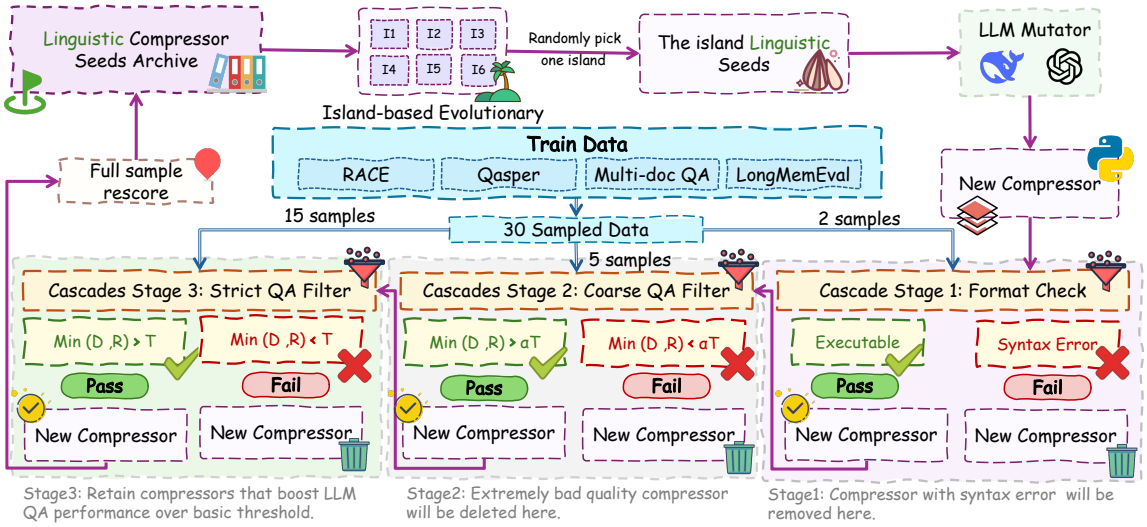

Figure 2: Overview of our evolutionary framework for discovering a linguistic prompt compressor. An LLM mutator edits seeds drawn from six islands, and each new compressor is screened by a three-stage cascade: format check (Stage 1), coarse QA filter (Stage 2), and strict QA filter (Stage 3). Survivors are rescored on the full sample and returned to the archive. T is the adaptive acceptance threshold (3.2), $\alpha < 1$ relaxes it at Stage 2, and $\min(D, R)$ is the smaller value of the direct and reconstruction QA accuracies (3.3).

effort and utilize linguistic priors, we apply evolutionary search to navigate the multi-level linguistic rules space and construct efficient compressors that requires only lightweight NLP pipelines.

2.2 Linguistic Rules for Compact Expression

Linguistic theories capture language regularities through compact rules: for example, grammar reduces clauses to head-dependent relations (Tesnière, 2015; de Marneffe et al., 2021); and rhetorical structure theory abstracts discourse into a small set of relations (Mann and Thompson, 1988). By separating essential structure from content, they align naturally with the goals of text compression.

Traditional NLP studies have validated this across distinct levels. At the **lexical** level, foundational research demonstrates that content words carry propositional meaning while functional elements can be safely discarded for summarization (Shannon, 1948; Zipf, 2016; Luhn, 1958; Edmundson, 1969). At the **syntactic** and **semantic** levels, compression techniques successfully preserve essential propositions by retaining core argument structures (Knight and Marcu, 2002; Filippova and Strube, 2008; Etzioni et al., 2008; Yoshikawa et al., 2012; Cheng et al., 2022). Furthermore, **discourse** connectives and entity chains serve as vital anchors for maintaining coherence across sentences (Prasad et al., 2008; Halliday and Hasan, 2014). These established principles motivate the hypothesis that explicit linguistic rules may support an effective compressor framework.

2.3 Evolutionary Prompt Optimization

LLM-guided evolutionary search has proven highly effective for navigating vast discrete search spaces. While current methods like EvoPrompt (Guo et al., 2024), OPRO (Fernando et al., 2024), and PromptBreeder (Yang et al., 2024) successfully apply this capability to refine natural language instructions, they leave its potential for structural program synthesis unexplored. Constructing a standalone prompt compressor from multidimensional linguistic rules fundamentally presents an equally massive discrete search challenge. Recognizing this alignment, we therefore apply a similar LLM-guided evolutionary paradigm to navigate the linguistic rule space, automatically discovering a competitive and highly efficient compression pipeline.

3 Methodology

To investigate whether deterministic linguistic rules can serve as effective prompt compressors, we design an offline evolutionary search framework based on AlphaEvolve. The framework proceeds in three stages: (1) we construct a set of linguistically motivated seeds as the initial population. (2) LLMs serving as offline mutation operators iteratively refine the compressor source code through a population-based search guided by compression ratio and downstream QA accuracy. (3) After evo-

lution, we evaluate the top-performing compressors on test sets through a dual-path evaluation protocol. Figure 2 shows the overall pipeline.

3.1 Seed Construction

To establish the foundational population for our evolutionary search, we design 42 seeds. Guided by the linguistic literature in §2.2, these seeds comprehensively span lexical, syntactic, semantic, and discourse levels, while also including several composite pipelines. Each seed encodes explicit, deterministic rules for identifying informative content. Full specifications are detailed in Appendix B.

10 lexical seeds operate at the word level, selecting or removing tokens based on part-of-speech categories, named entity types, and word frequency. Their core strategy is to retain content words and named entities while dropping predictable function words, with additional operations such as short-synonym substitution and symbolic abbreviation.

12 syntactic seeds operate within the sentence, exploiting dependency structure to extract core argument triples and prune dispensable sub-trees such as relative clauses, coordinated conjuncts, and adjectival or adverbial modifiers.

6 semantic seeds preserve meaning-critical markers regardless of compression budget, including negation markers whose removal reverses polarity, modal verbs that encode certainty or obligation, quantifiers that constrain proposition scope, and comparative constructions.

9 discourse seeds operate across sentences, using signals such as sentence position, discourse connectives, entity chains, and graph-based sentence ranking to select informative sentences and remove near-duplicate content.

5 composite pipelines cascade rules from multiple levels, chaining sentence-level ranking, dependency-core extraction, SVO triple construction, and symbolic substitution into multi-stage compression sequences.

Regarding implementation, all seeds rely exclusively on spaCy[2] for tokenization, POS tagging, lemmatization, dependency parsing, named entity recognition, and noun chunking. We additionally utilize NLTK (Bird and Loper, 2004) with WordNet (Miller, 1992) for synonym lookup, wordfreq for lexical frequency, and standard Python libraries (`re`, `math`, `collections`, `numpy`). No LM inference is involved at compression time.

[2]We only allow the (`en_core_web_sm`) to conduct linguistic analysis

3.2 Evolutionary Search

Framework. AlphaEvolve (Novikov et al., 2025) is an evolutionary coding agent that iteratively improves programs via LLM mutation and automated evaluation. Originally built for algorithmic tasks with explicit, stable metrics, we adapt it for prompt compression, as illustrated below.

Population and Archive. Every candidate in our population is a Python script exposing a `compress(text: str) -> str` function. We divide the population into six independent islands. We distribute the 42 seed compressors evenly across these islands. Each island receives strategies from all four linguistic levels to guarantee initial diversity. In addition, each island manages a local MAP-Elites (Mouret and Clune, 2015) archive that categorizes candidates into 48 discrete cells based on code complexity and compression ratio. Every cell keeps only the single highest-scoring compressor. A new candidate can replace the current occupant only if it achieves a higher fitness.

Mutation. At each iteration, a parent compressor is selected from the archive along with its evaluation feedback, including QA accuracy, compression ratio, and failure examples. An LLM reads the source code and context, then proposes edits to designated mutable regions, which are marked in the source code as evolution blocks. Edits are expressed as search-and-replace pairs, allowing the LLM to adjust scoring weights and filtering conditions, or restructure the compression logic. We adopt a three-tier LLM ensemble as mutators.

Cascaded evaluation. Full-sample evaluation at every iteration is computationally prohibitive. We therefore design a cascaded pipeline to filter candidates, using increasingly larger sample sizes and stricter thresholds. Only the most promising candidates reach the final full-sample evaluation.

- **Stage 1** The compressor runs on 2 samples with no LLM calls. Candidates are rejected if the output is empty, non-string, or outside the target compression window.
- **Stage 2.** Surviving candidates are evaluated on $1/6$ of the samples through both direct and reconstruction QA accuracy. A candidate is eliminated if the smaller of its two path accuracies falls below a threshold, defined as the average no-compression accuracy divided by the target compression ratio and scaled by a constant coefficient smaller than 1.
- **Stage 3.** The evaluation sample size is ex-

panded to $1/2$ of the sample pool, and the screening threshold is tightened to the average no-compression accuracy divided by the target compression ratio.

Candidates passing all three stages are re-evaluated on the full training set to obtain reliable direct and reconstruction accuracies, which are used to compute the fitness function and rank compressors within the population.

Island Migration. The six islands run their own mutation and selection loops in parallel, exploring different regions of the compressor space, and are connected only through periodic migration under a ring topology (Whitley et al., 1999). At fixed intervals, the top-fitness fraction of each island is copied to its two adjacent islands. Each program migrates at most once per lineage, which prevents a single strong compressor from repeatedly cloning itself across the population.

3.3 Dual-Path Evaluation and Fitness

We evaluate compression quality from two complementary perspectives. Direct accuracy, denoted as D, measures task performance from the compressed text. We send the compressed text and the question to a receiver LLM and score the accuracy of its answers using the dataset-specific metric defined in §4.3. Reconstruction accuracy, denoted as R, measures task performance after a reconstruction LLM first rebuilds the original passage from the compressed text and a receiver LLM then answers from the reconstructed text. A compressor might retain enough information for direct question answering yet lose details required by a different model for reconstruction, or vice versa.

A qualified compressor should perform well on both paths. We therefore define the fitness score as:

$$f = \min(D, R) \tag{1}$$

The $\min(D, R)$ acts as a conservative choice focusing evolutionary pressure strictly on the weaker path. A compressor cannot achieve high fitness unless both accuracies are reasonably high, making the improvement of the lower score the most direct method to increase the minimum. Following evolution, we select the top-performing candidate that meets the target compression ratio and run the selected compressor on the test set.

4 Experimental Setup

4.1 Datasets

We evaluate our evolved compressors across four datasets covering medium- to long-text scenarios. Specifically, **RACE** (Lai et al., 2017) provides medium-length reading comprehension challenges from English exams. To assess long documents, we use the LongBench subset of **Qasper** (Dasigi et al., 2021; Bai et al., 2024), which features queries whose answers are distributed across NLP research papers. For reasoning across multiple texts, **Multi-doc QA** concatenates supporting documents from HotpotQA (Yang et al., 2018) and 2WikiMultihopQA (Ho et al., 2020) to enforce complex logical connections. Finally, **LongMemEval** (Wu et al., 2025) tests retention in loosely structured dialogue histories, where highly uneven information density introduces a distinct robustness challenge.

For each dataset, we draw a balanced sample and split it by a random shuffle, assigning 100 and 200 samples to the training and test sets, respectively. The four datasets contribute equally, with 400 training and 800 test examples in total. Detailed information on the datasets, including lengths and question examples, is provided in Appendix C.

4.2 Baselines

We compare against seven hard prompt compression baselines, covering perplexity-based, classification-based, and hybrid scoring strategies.

- **Selective Context** (Li et al., 2023b) removes low-self-information units scored by GPT-2 (Radford et al., 2019).
- **LLMLingua** (Jiang et al., 2023) performs coarse-to-fine budget allocation with iterative token pruning using Llama-2-7B (Touvron et al., 2023).
- **LongLLMLingua** (Jiang et al., 2024) extends LLMLingua to long contexts via contrastive perplexity and document reordering using the same model, Llama-2-7B.
- **LLMLingua-2** (Pan et al., 2024) casts compression as token classification using an XLM-RoBERTa-large (Conneau et al., 2020) trained on GPT-4 (OpenAI et al., 2024) distilled labels to predict preservation for tokens.
- **Selection-p** (Chung et al., 2024) introduces a small number of additional parameters on Llama-2-7B backbone and trains with self-supervised continual pre-training, producing a

preserve-or-discard probability for each token.

- **FrugalPrompt** (Raiyan et al., 2026) ranks tokens by GlobEnc (Modarressi et al., 2022) attention-rollout saliency over an Electra-large encoder (Clark et al., 2020).
- **PartPrompt** (Mao et al., 2026) builds a syntactic parse tree for each sentence, computes node-level information entropy with Llama-2-7B, organizes the local trees into a global hierarchical tree, and prunes it recursively to produce the compressed prompt.

4.3 Evaluation Metrics

Direct accuracy and Reconstruction accuracy in §3.3 are the primary metrics. For each dataset, the two paths are evaluated with the same metric.

Accuracy. For RACE with multiple-choice QA pairs, we report exact match rate. For Qasper, Multi-doc QA, and LongMemEval with open-ended QA pairs, Direct accuracy and Reconstruction accuracy refer to LLM-as-judge accuracy, while token-level F1 is act as a reference metric for analysis. During evolutionary search, accuracy on the three open-ended datasets is based solely on the LLM-as-judge score.

Compression ratio. Defined as the original word count divided by the compressed word count to be reported to show the compression level.

4.4 Implementation Details

Models. Three open-source LLMs of comparable scale are used as both receiver LLMs and reconstruction LLMs: Qwen-2.5-7B-Instruct (Qwen et al., 2025), LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024), and Gemma-3-12B-IT (Team et al., 2025). For each sample, one model serves as the receiver LLM and a different model serves as the reconstruction LLM; after reconstruction, a receiver LLM answers the QA prompt from the reconstructed text. GPT-5.4-mini (OpenAI, 2026a) serves as the LLM-as-judge model for the three open-ended datasets. During evolution, mutations are proposed by a three-tier ensemble (GPT-5.4-mini 40%, DeepSeek-V4-Pro (DeepSeek, 2026) 35%, GPT-5.5 (OpenAI, 2026b) 25%).

Evolutionary search. The search runs for 30 generations. Each island maintains a 6×8 MAP-Elites grid. Migration occurs every 3 generations with a migration rate of 15%.

Compression ratios. We evaluate both the baselines and our evolved compressors at four target compression ratios ($1.5\times$, $2.5\times$, $4\times$, and $6\times$). An evolved compressor does not always hit its target ratio exactly, so we associate each target with an acceptance window of ± 0.25, ± 0.5, ± 0.75, and ± 1.0 respectively. During the search, an evolved compressor is stored in the archive only when its compression ratio falls within this window and it passes the cascade filtering.

5 Results and Analysis

5.1 Main Results

The experimental results are listed as five findings.

Linguistic rules match strong LM baselines on lengthy documents. On the three longer datasets, Table 1 shows our compressor attaining the best Direct accuracy on 3 of the 12 dataset-ratio cells. RACE is the relative weak spot. Its multiple-choice passages are markedly shorter, and our method trails the baselines there. One plausible explanation is the evolutionary setup: because the training pool draws three quarters of its signal from the longer datasets, the evolved compressor may be biased toward lengthy-text patterns. A bootstrap significance analysis (Appendix E) sharpens this contrast. Against the strongest baseline at each cell, our compressor is significantly weaker on RACE at $2.5\times$, $4\times$, and $6\times$. Yet, on the primary LLM-as-judge metric, it remains statistically comparable on the three longer datasets in all 12 cells, while the reference token-level F1 results remain statistically comparable in 11 of 12 cells (Tables 15 and 16).

Linguistic compression stays competitive across all compression ratios. Within Table 1, the compressor's rank traces a clear arc. Competitiveness peaks at 2.5×, where it ranks first or second on all three longer datasets, and at 1.5× it ranks first on Qasper and Multi-doc QA. Performance then tapers at 4× and is weakest at 6×, slipping to third or fourth place. Even so, a bootstrap analysis finds Direct accuracy statistically comparable to the strongest baseline on Qasper and Multi-doc QA at all four ratios, and on LongMemEval except at 6× (Appendix E). One plausible reason owes to the categorical nature of linguistic rules at extreme ratios. Because these rules make an all-or-nothing decision on each token based on fixed properties such as POS and dependency role, our method cannot rank tokens by fine gradations like a continuous LM score. This granularity gap matters most at the tightest budget, where only one token in six is kept. Finally, the Direct and Reconstruction paths

Method	RACE				Qasper				Multi-doc QA				LongMemEval			
	1.5×	2.5×	4×	6×	1.5×	2.5×	4×	6×	1.5×	2.5×	4×	6×	1.5×	2.5×	4×	6×
Selective-Ctx	**90.51**	81.02	74.02	69.48	61.50	49.01	43.00	24.50	57.07	56.58	49.54	45.00	44.04	37.52	43.52	45.00
LLMLingua	88.48	81.98	**78.99**	77.00	31.47	24.04	19.52	19.00	55.55	44.50	39.96	41.49	42.51	39.05	39.47	41.04
LongLLMLingua	89.97	81.48	76.97	**77.99**	33.98	22.00	18.98	20.50	58.04	49.00	43.96	42.99	43.02	39.08	42.48	44.01
LLMLingua-2	88.52	**87.00**	75.98	67.97	60.98	**57.48**	**50.51**	**47.05**	60.03	55.03	**54.53**	48.56	45.02	42.05	39.57	41.08
FrugalPrompt	90.03	76.47	69.46	69.48	60.97	46.46	41.93	42.97	58.53	**60.05**	51.58	**50.09**	37.01	46.54	49.04	**47.04**
Selection-p	78.97	73.48	69.46	69.48	51.96	47.04	34.52	36.99	52.04	44.01	39.03	31.53	**46.01**	49.03	44.57	44.08
PartPrompt	80.00	78.47	74.54	69.99	61.99	49.46	40.49	36.01	52.55	47.57	41.53	37.52	45.05	48.03	**51.03**	42.52
Ours	86.01	69.52	66.00	59.48	**62.51**	54.00	46.48	39.49	**62.51**	57.00	51.02	43.02	42.94	**49.48**	48.48	44.47

Table 1: Main results across four datasets and compression ratios. Each cell reports Direct accuracy (%) averaged over three receiver LLMs. RACE uses exact match; the three open-ended datasets use LLM-as-judge accuracy as the primary metric. Full per-model results, including Reconstruction accuracy and reference token-level F1, are in Appendix D. **Bold** = best; underline = second; dashed = third.

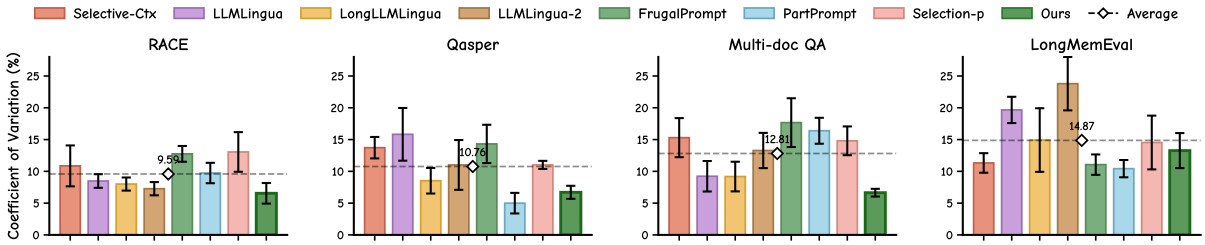

Figure 3: Cross-model stability measured by Coefficient of Variation (CV, %) of Direct accuracy across three receiver LLMs. Bars show the mean CV over four compression ratios. Lower is better.

diverge across datasets (Table 10 in Appendix D). Significant shortfalls fall mainly on the Direct path for RACE and mainly on the Reconstruction path for Qasper, while Multi-doc QA shows no shortfalls on either path and LongMemEval reveals one solely at 6× on the Direct path.

Linguistically compressed text shows we have most stable cross-model behavior. To quantify cross-model stability, we measure the Coefficient of Variation (CV) of Direct accuracy. For each (method, dataset, ratio) triple, CV is defined as:

$$\mathrm{CV} = \frac{\sigma}{\mu} \tag{2}$$

where μ and σ are the mean and standard deviation of accuracy over the three receiver models. We then average CV across the four compression ratios to obtain a single stability measure per method and dataset. As Figure 3 demonstrates, our method achieves the lowest overall CV among all baselines and ranks among the most stable on each dataset.

Linguistic compression keeps the compression ratio on target, whereas LM-based baselines drift. Figure 4 plots the deviation between actual and target compression ratios averaged across datasets. Our method achieves near-zero deviation

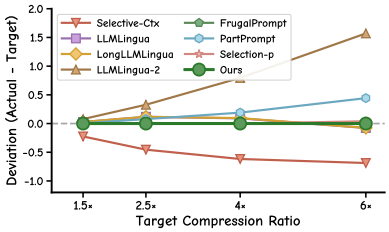

Figure 4: Deviation of actual from target compression ratio (actual − target), averaged over all datasets. Positive values denote over-compression.

at all target ratios, while several LM-based baselines exhibit drift that grows with the compression level. This controllability arises because our compressor computes an explicit target token count from the desired ratio and enforces it through a hard budget, which keeps ratio deviation small.

Linguistic compression requires no LM forward pass at deployment and only lightweight CPU-side processing. Because compression relies solely on the spaCy pipeline, our method can run on a single CPU without an LM forward pass, whereas all baselines require GPU-backed neural scoring. Under our reported hardware setup, our compressor is faster than every baseline at all tested

compression ratios (details in Appendix F).

5.2 Evolutionary Analysis

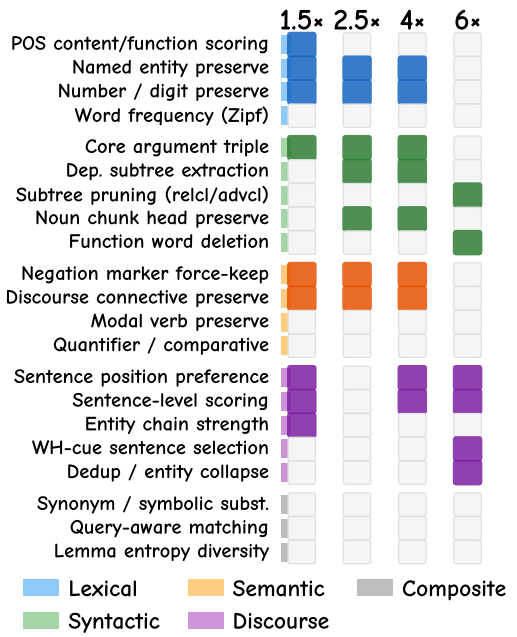

Figure 5: Survival of linguistic rules in the best evolved compressor at each compression ratio. Filled cells denote presence; rows are grouped by linguistic level. Bottom group: signals eliminated across all ratios.

Evolution converges on a compact rule core, discarding several theoretically motivated rules. Figure 5 summarizes linguistic rule survival during the search at each compression ratio (details are shown in Appendix B). At ratios from $1.5\times$ to $4\times$, the best compressors draw on rules from all four linguistic levels, so *the best compressors found by our search consistently rely on cross-level fusion rather than any single level.* Within this range, named entity preservation, negation force-keeping, core argument triples, and discourse connective preservation persist. Each covers a distinct facet of QA-relevant information. At $6\times$, the best compressor is dominated by discourse and syntactic operations and functions as a skeleton that preserves the most critical entities while discarding finer lexical and semantic detail. Conversely, several theoretically motivated rules are eliminated at every ratio, including word frequency scoring, synonym substitution, lemma entropy diversity, and, notably, query-aware matching. The absence of query-aware rules may indicate that preserving the intrinsic structural integrity of the text is more useful in our search setup.

The dominant unit of compression shifts from tokens to sentences as ratio increases. This shift is visible across the four best evolved compressors (details in Appendix G). At $1.5\times$ the best compressor uses flat token-level scoring; at $2.5\times$ it shifts to tiered filling based on dependency subtree skeletons; at $4\times$ a two-stage cascade emerges that first selects sentences and then prunes tokens within them; and at $6\times$ the architecture becomes sentence selection followed by subtree-level deletion. Discourse-level rules track this progression, evolving from an auxiliary bonus at low ratios to the primary selection mechanism at high.

Evolutionary search generates refined mechanisms beyond any single seed. Beyond recombining seeds, the search produces mechanisms with no direct equivalent among the initial individual seeds: dynamic entity chain scoring based on cross-sentence frequency ($1.5\times$), a hybrid truncation strategy balancing priority selection with evenly spaced document coverage ($2.5\times$), sentence-then-token composition that combines discourse and syntactic seeds into a tighter pipeline ($4\times$), and subtree-level deletion paired with a post-processing chain of entity collapse and enumeration compression ($6\times$). These innovations show that evolutionary search does not merely select predefined strategies, but can also refine and recombine them into new compositional structures through mutation.

6 Conclusion

We have shown that linguistic rule-based compressors, discovered through offline LLM-guided evolutionary search, can perform effectively without any LM forward pass at deployment. From 42 seeds spanning lexical, syntactic, semantic, and discourse levels, evolution converges on multi-level rule combinations that can match strong LM-based baselines on most datasets, particularly the longer ones, while RACE remains more challenging.

Evolutionary analysis finds effective compression fuses four linguistic levels' signals, and the dominant unit of compression shifts from tokens to sentences as the compression ratio increases. A stable core including named entity preservation, negation force-keeping, core argument triples, and discourse connective retention persists across moderate compression ratios. We hope this work can offer insights for interpretable, model-agnostic, and computationally lightweight prompt compression.

Limitations

Language and parser dependence. This work focuses on English only and highly relies on spaCy's English pipeline for tokenization, parsing, and named entity recognition. Extending the framework to other languages would require language-specific NLP tools and likely a new round of evolutionary search. More broadly, the method is only as reliable as the underlying linguistic analyses: changes in parser quality may change absolute compression performance and the specific rule combinations that evolution prefers.

Input format. Our compressors are designed for standard prose. They have not been validated on inputs with non-standard structure, such as source code, nested JSON, mathematical notation, tables, or fragmented raw data, where conventional syntactic and discourse parsing may degrade.

Search-space coverage. The 42 seed compressors span different rules, but they do not exhaust the space of linguistically motivated compression strategies. The current search is also shaped by the available linguistic rules, the mutation space, and the fixed wrapper and fitness design. Stronger compressors may emerge under a broader rule inventory or a different search formulation.

Evaluation scope and metrics. We evaluate only question-answer question on the evaluation, and we do so on balanced subsets of four datasets rather than the full benchmarks. For the three open-ended datasets, the primary metric is LLM-as-judge accuracy, with token-level F1 reported as a reference metric. Although this combination helps capture semantic correctness, the main conclusions for open-ended QA might be influenced by the chosen judge model and judging prompt.

Receiver drift. Our dual-path protocol uses three small LLMs of comparable scale. The evolved rule core may not transfer unchanged to future models with substantially different sizes, training data, or architectures, and re-optimization may be needed as receiver models evolve.

Offline optimization cost. The final compressor runs on CPU without an LM forward pass, but discovering it requires an offline evolutionary search with multiple LLM calls over generations. This upfront development cost is amortized when a compressor is reused, but it may still be non-trivial in low-budget settings.

Search configuration. We report results from one search configuration. We do not exhaustively tune the evolutionary hyperparameters or ablate the individual contribution of each mutator model, so the current results should not be interpreted as an upper bound for the framework.

Ethics Statement

We do not foresee any immediate negative ethical consequences of our research.

References

Mikhail L. Arbuzov, Sisong Bei, Ziwei Dong, Dmitri Kalaev, and Alexey A. Shvets. 2026. Telegraph english: Semantic prompt compression via structured symbolic rewriting. *Preprint*, arXiv:2605.04426.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. In *Philosophy, language, and artificial intelligence: Resources for processing natural language*, pages 241–301. Springer.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Huangfei Cheng, Jiawei Wu, Tiantian Li, Bin Cao, and Jing Fan. 2022. Improving abstractive multi-document summarization withnbsp;predicate-argument structure extraction. In *PRICAI 2022: Trends in Artificial Intelligence: 19th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2022, Shanghai, China, November 10–13, 2022, Proceedings, Part II*, page 268–282, Berlin, Heidelberg. Springer-Verlag.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. xRAG: Extreme context compression for retrieval-augmented generation with one token. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846, Singapore. Association for Computational Linguistics.

Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. 2024. Learning to compress prompt in natural language formats. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7756–7767, Mexico City, Mexico. Association for Computational Linguistics.

Tsz Ting Chung, Leyang Cui, Lemao Liu, Xinting Huang, Shuming Shi, and Dit-Yan Yeung. 2024. Selection-p: Self-supervised task-agnostic prompt compression for faithfulness and transferability. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11057–11070, Miami, Florida, USA. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *Preprint*, arXiv:2003.10555.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. 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*, pages 4599–4610, Online. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

DeepSeek. 2026. deepseek-V4-model-card. `https://fe-static.deepseek.com/chat/transparency/deepseek-V4-model-card-EN.pdf`. [Accessed 22-05-2026].

Harold P Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open information extraction from the web. *Commun. ACM*, 51(12):68–74.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 25–32, Salt Fork, Ohio, USA. Association for Computational Linguistics.

Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*.

Michael Alexander Kirkwood Halliday and Ruqaiya Hasan. 2014. *Cohesion in english*. Routledge.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in python.

Laurence R Horn. 1989. A natural history of negation.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore. Association for Computational Linguistics.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand. Association for Computational Linguistics.

Hoyoun Jung and Kyung-Joong Kim. 2024. Discrete prompt compression with reinforcement learning. *IEEE Access*, 12:72578–72587.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Cornelius Kummer, Lena Jurkschat, Michael Färber, and Sahar Vahdati. 2026. Prompt compression innbsp;thenbsp;wild: Measuring latency, rate adherence, andnbsp;quality fornbsp;faster llm inference. In *Advances in Information Retrieval: 48th European Conference on Information Retrieval, ECIR 2026, Delft, The Netherlands, March 29 – April 2, 2026, Proceedings, Part I*, page 257–271, Berlin, Heidelberg. Springer-Verlag.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Weronika Łajewska, Momchil Hardalov, Laura Aina, Neha Anna John, Hang Su, and Lluis Marquez. 2025. Understanding and improving information preservation in prompt compression for LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17520–17541, Suzhou, China. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. volume 33, pages 9459–9474.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023b. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, Singapore. Association for Computational Linguistics.

Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. 2025a. Prompt compression for large language models: A survey. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7182–7195, Albuquerque, New Mexico. Association for Computational Linguistics.

Zongqian Li, Yixuan Su, and Nigel Collier. 2025b. 500xCompressor: Generalized prompt compression for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25081–25091, Vienna, Austria. Association for Computational Linguistics.

Yuhao Lin, Wenya Guo, Ying Zhang, Chengyi Yang, and Zengxiang Li. 2025. Prompt compression based on key-information density. *Expert Systems with Applications*, 284:127738.

Barys Liskavets, Maxim Ushakov, Shuvendu Roy, Mark Klibanov, Ali Etemad, and Shane K. Luke. 2025. Prompt compression with context-aware sentence encoding for fast and improved llm inference. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text - Interdiscip. J. Study Discourse*, 8(3).

Wenhao Mao, Chengbin Hou, Tianyu Zhang, Xinyu Lin, Ke Tang, and Hairong Lv. 2026. Parse trees guided LLM prompt compression. *IEEE Trans. Pattern Anal. Mach. Intell.*, 48(1):421–435.

George A. Miller. 1992. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2022. GlobEnc: Quantifying global token attribution by incorporating the whole encoder layer in transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 258–271, Seattle, United States. Association for Computational Linguistics.

Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *Preprint*, arXiv:1504.04909.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Alliot Nagle, Adway Girish, Marco Bondaschi, Michael Gastpar, Ashok Vardhan Makkuva, and Hyeji Kim. 2024. Fundamental limits of prompt compression: A rate-distortion framework for black-box language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *Preprint*, arXiv:2506.13131.

OpenAI. 2026a. Introducing GPT-5.4. https://openai.com/index/introducing-gpt-5-4/. [Accessed 22-05-2026].

OpenAI. 2026b. Introducing GPT-5.5. https://openai.com/index/introducing-gpt-5-5/. [Accessed 22-05-2026].

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981, Bangkok, Thailand. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Syed Rifat Raiyan, Md Farhan Ishmam, Abdullah Al Imran, and Mohammad Ali Moni. 2026. Frugalprompt: Reducing contextual overhead in large language models via token attribution. *Preprint*, arXiv:2510.16439.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Lucien Tesnière. 2015. *Elements of structural syntax*. John Benjamins Publishing Company.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Darrell Whitley, Soraya Rana, and Robert B Heckendorn. 1999. The island model genetic algorithm: On separability, population size and convergence. *Journal of computing and information technology*, 7(1):33–47.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Katsumasa Yoshikawa, Ryu Iida, Tsutomu Hirao, and Manabu Okumura. 2012. Sentence compression with semantic role constraints. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 349–353, Jeju Island, Korea. Association for Computational Linguistics.

Zhang Zheng, Jinyi Li, Yihuai Lan, Xiang Wang, and Hao Wang. 2025. An empirical study on prompt compression for large language models. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

George Kingsley Zipf. 2016. *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio books.

A Implementation Details

All experiments adopt random seed 42. Baseline compression methods run on a single NVIDIA A100-40GB GPU; our evolved compressors run fully on CPU.

All decoding uses temperature 0 (greedy, no top-k or top-p) except the mutator, whose temperature anneals from 1.0 (early exploration) to 0.0 (late refinement). The mutator selects from a mixture of three model tiers driven by the current temperature: `gpt-5.4-mini` as the weak tier, `deepseek-v4-pro` as the medium tier, and `gpt-5.5` as the strong tier; higher temperatures favor the weak tier for diversity and lower temperatures favor the strong tier for incremental refinement. The mutator's max-token budget is 4096 for the weak and medium tiers and 8192 for the strong.

The three receiver LLMs are queried with temperature 0 and max_tokens 64 for the RACE dataset and 256 for the open-ended datasets Qasper, Multi-doc QA, and LongMemEval. The reconstruction LLM uses the same three-model pool with temperature 0 and max_tokens 4096. Receiver LLMs and mutators are accessed through OpenRouter.

The LLM-as-judge calls use `gpt-5.4-mini` throughout, temperature 0, max_tokens 4. The mutator and judge calls route through OpenRouter.

Per-call timeouts are 120 s for receivers, 250 s for the mutator's weak/medium tiers and the judge, and 500 s for the strong tier. Retry budgets are 5 for receivers and 3 for the mutator and judge.

B Linguistic Seeds

This appendix provides the full specification of the 42 seeds that form the initial population for evolutionary search. Each seed is a self-contained Python function `compress(text: str) -> str` encoding one linguistically motivated compression hypothesis. All seeds share a target-ratio wrapper that enforces the desired compression ratio by truncating or back-filling the output to $\lfloor n_{\text{src}}/r \rceil$ words. The seed itself determines *how to rank or select* tokens, not how many to keep.

Superscripts in Tables 2–6 denote operation types: ^{S} Scoring + top-k, ^{T} Tiered fill, ^{D} Deletion, ^{E} Extraction, ^{R} Substitution, ^{L} Sentence selection, ^{C} Cascade, ^{P} Post-processing.

#	Seed Name	Motivation
1	Content word retain^{S}	Keep content words, drop function words (Shannon, 1948)
2	NER-first^{T}	QA answers are often named entities
3	Stopword deletion^{D}	Closed-class words carry minimal info
4	Content-func. gap^{S}	Maximize content/function keep-rate gap
5	Zipf rare boost^{S}	Rare words carry more information
6	Acronym preserve^{S}	Acronyms are key identifiers
7	Lemma dedup^{S}	One instance per lemma
8	Synonym subst.^{R}	Replace with shorter synonyms
9	Symbolic abbrev.^{R}	Replace phrases with symbols
10	Canonical POS^{S}	POS hierarchy

Table 2: Lexical seeds (10).

B.1 Design Principles

The 42 seeds are designed to provide the evolutionary search with a diverse initial population spanning the compression–accuracy trade-off space. Several principles guide their construction:

Theoretical grounding. Each seed encodes a hypothesis rooted in linguistic theory or empirical findings. Lexical seeds draw on the content–function word distinction (Shannon, 1948; Zipf, 2016); syntactic seeds on dependency grammar and sentence compression (Tesnière, 2015; Filippova and Strube, 2008); semantic seeds on negation scope (Horn, 1989) and quantifier semantics (Barwise and Cooper, 1981); discourse seeds on coherence theory (Halliday and Hasan, 2014) and rhetorical structure (Mann and Thompson, 1988).

Intentional overlap. Several seeds share machinery (e.g., SVO extraction and entailment extraction are nearly identical; the four discourse-marker seeds all reduce to sentence-level predicates). This redundancy is intentional: it provides the evolutionary search with overlapping starting points that can be recombined through crossover and mutation.

Level diversity. Seeds are distributed across all four linguistic levels plus composite pipelines, ensuring that no single level dominates the initial population. This diversity is critical for the evolutionary search to discover cross-level combinations rather than converging on a single-level strategy.

#	Seed Name	Motivation
11	Arg. triple only^{E}	Keep (nsubj, ROOT, dobj) (Etzioni et al., 2008)
12	Verb skeleton^{T}	ROOT subtrees strongest at light comp.
13	Dep. role priority^{S}	Core arguments above modifiers
14	Relcl pruning^{D}	Relative clauses dispensable (Filippova and Strube, 2008)
15	Amod pruning^{D}	Adj. modifiers rarely carry answers
16	Advmod pruning^{D}	Adv. modifiers highly droppable
17	Conjunct pruning^{D}	Beyond-first conjuncts dispensable
18	Appos pruning^{D}	Appositives restate head noun
19	SVO extraction^{E}	Minimal propositional content
20	Chunk head retain^{T}	Chunk head most informative
21	Depth pruning^{D}	Deeper tokens more dispensable
22	Parenthetical del.^{D}	Parentheticals non-essential

Table 3: Syntactic seeds (12).

#	Seed Name	Motivation
23	Negation keep^{S}	Removal reverses polarity (Horn, 1989)
24	Modal preserve^{S}	Encodes certainty / obligation
25	Quantifier preserve^{S}	Constrains scope (Barwise and Cooper, 1981)
26	Comparative preserve^{S}	Encodes relational meaning
27	Reasoning anchor^{T}	Modal+ROOT+connective skeleton
28	WH/causal boost^{S}	Reasoning anchors for QA

Table 4: Semantic seeds (6).

#	Seed Name	Motivation
29	First-sent. anchor^{T}	First-sent. kept at 2–5× rate
30	Position decay^{S}	Geometric budget from first sent.
31	Entity chain^{S}	Entities in ≥2 sents are anchors (Halliday and Hasan, 2014)
32	TextRank^{L}	Centrality by word overlap
33	Dedup removal^{L}	High-overlap sents redundant
34	Causal conn.^{S}	Reasoning coherence (Prasad et al., 2008)
35	Temporal chain^{S}	Anchors event sequences
36	Doc-initial bias^{S}	First 20 tokens get bonus
37	Entity collapse^{P}	Collapse repeated mentions

Table 5: Discourse seeds (9).

#	Seed Name	Motivation
38	Rank+dep-core^{C}	TextRank then dep-core extraction
39	Rank+SVO^{C}	Sent. selection then SVO triples
40	Entropy diversity^{S}	Diminishing returns for repeated lemmas
41	Query-aware^{S}	Boost tokens matching question
42	Full cascade^{C}	Rank→dep→SVO→symbols

Table 6: Composite pipeline seeds (5).

B.2 Signal Survival Across Compression Ratios

Tables 7 and 8 track which seed rules survived in the best evolved compressor at each target ratio. Rules that persist across all tested ratios represent the irreducible core of linguistic compression; those eliminated across all ratios were not beneficial for downstream QA despite their theoretical motivation.

C Dataset Information

For each of the four datasets, we use 100 training examples and 200 test examples. Table 9 reports context length statistics (whitespace-separated word count). The four datasets span a wide range of input lengths, from short multiple-choice passages on RACE (∼300 words) to long multi-document and dialogue contexts (∼4,000–7,000 words). Below we also show one representative example per dataset; for long contexts the passage is truncated with "…" for readability.

C.1 Example: RACE

Context (305 words): Do you start your work by making a list of all you have to complete, from walking the dog and washing the clothes, to phoning a friend or checking your emails? Do you always take a list with you to the supermarket? Do you write down all the steps you need to see a big project through? If you're a natural list maker you can consider this habit as a good thing to do. …A list makes it possible for us to spend too much time on unimportant items. …

Question: The author introduces the topic to be discussed by ___.

Options: (A) asking a series of questions; (B) presenting some simple facts; (C) stating his own point of view; (D) listing the benefits of a to-do list.

Answer: A

C.2 Example: Qasper

Context (3,697 words): Ancient Chinese is the writing

#	Seed	1.5×	2.5×	4×	6×
Lexical (10)					
1	Content word retain	✓			
2	NER-first	✓	✓	✓	
3	Stopword deletion	✓			✓
4	Content-func. gap	✓			
5	Zipf rare boost				
6	Acronym preserve		✓		
7	Lemma dedup				
8	Synonym subst.				
9	Symbolic abbrev.				
10	Canonical POS	✓			
Syntactic (12)					
11	Arg. triple only	✓	✓	✓	
12	Verb skeleton	✓	✓	✓	
13	Dep. role priority	✓	✓	✓	
14	Relcl pruning				✓
15	Amod pruning				
16	Advmod pruning				
17	Conjunct pruning				
18	Appos pruning				✓
19	SVO extraction				
20	Chunk head retain		✓	✓	
21	Depth pruning				
22	Parenthetical del.				

Table 7: Survival of lexical and syntactic seeds. ✓ = present in best evolved compressor.

#	Seed	1.5×	2.5×	4×	6×
Semantic (6)					
23	Negation keep	✓	✓	✓	
24	Modal preserve				
25	Quantifier preserve				
26	Comparative preserve				
27	Reasoning anchor	✓	✓	✓	
28	WH/causal boost	✓	✓	✓	✓
Discourse (9)					
29	First-sent. anchor	✓			✓
30	Position decay	✓			
31	Entity chain	✓			
32	TextRank				
33	Dedup removal				
34	Causal conn.	✓	✓	✓	
35	Temporal chain	✓	✓	✓	
36	Doc-initial bias	✓			✓
37	Entity collapse				✓
Composite (5)					
38	Rank+dep-core			✓	
39	Rank+SVO				
40	Entropy diversity				
41	Query-aware				
42	Full cascade				

Table 8: Survival of semantic, discourse, and composite seeds. ✓ = present in best evolved compressor.

language in ancient China. It is a treasure of Chinese culture which brings together the wisdom and ideas of the Chinese nation . . . Compared with modern Chinese, ancient Chinese is more concise and shorter. . . . We apply two NMT models, an RNN-based encoder–decoder and a Transformer-based model, to translate ancient Chinese into modern Chinese, and evaluate against several baselines. . . .

Question: What NMT models did they compare with?

Answer: RNN-based NMT model, Transformer-NMT.

C.3 Example: Multi-Doc QA

Context (5,804 words, multi-passage): *Passage 1: Ian Barry (director).* Ian Barry is an Australian director of film and TV. . . . *Passage 2: Dana Blankstein.* Dana Blankstein-Cohen (born March 3, 1981) is the executive director of . . . *[8 supporting passages in total, including the one identifying the director of* Scarecrow (1984)*.]* . . .

Question: Where does the director of film *Scarecrow (1984 Film)* work at?

Answer: High Courses for Scriptwriters and Film Directors.

C.4 Example: LongMemEval

Context (4,414 words, multi-turn dialogue history): `[Session 1]` **user:** I'm looking for some book recommendations. . . . I just finished a 416-page novel, but before that, I read "The Power" by Naomi Alderman in December, which had 341 pages . . . **assistant:** You're a reader who enjoys immersive, thought-provoking novels . . . *[In a later session, the user reports finishing "The Nightingale" by Kristin Hannah,* 440 *pages, and asks for similar recommendations.]* . . .

Question: What was the page count of the two novels I finished in January and March?

Answer: 856 (i.e., 416 + 440).

Dataset	Train words	Test words
RACE	292 ± 131	325 ± 148
Qasper	$4{,}070 \pm 2{,}273$	$3{,}933 \pm 2{,}280$
Multi-doc QA	$7{,}020 \pm 3{,}350$	$6{,}582 \pm 3{,}401$
LongMemEval	$3{,}972 \pm 2{,}192$	$4{,}361 \pm 2{,}459$

Table 9: Context length for each dataset's train ($n = 100$ per dataset) and test ($n = 200$ per dataset) split, reported as mean $\pm$ standard deviation.

D Detailed Results

This appendix presents per-model, per-ratio results for all compression methods across the four evaluation datasets. For the three open-ended datasets, LLM-as-judge accuracy is the primary metric, while token-level F1 is reported as a reference metric.

Method	RACE				Qasper				Multi-doc QA				LongMemEval			
	1.5×	2.5×	4×	6×	1.5×	2.5×	4×	6×	1.5×	2.5×	4×	6×	1.5×	2.5×	4×	6×
Selective-Ctx	81.54	73.03	**70.03**	58.49	35.51	32.99	33.48	22.49	37.50	33.52	32.03	33.49	45.00	42.53	49.49	45.51
LLMLingua	**87.03**	69.01	66.03	**64.04**	26.01	19.52	14.99	13.50	28.05	28.04	30.02	29.49	44.02	46.03	46.02	45.55
LongLLMLingua	84.02	69.50	64.01	63.04	24.51	17.50	15.00	18.52	31.01	28.52	31.02	28.98	48.53	41.49	43.02	44.57
LLMLingua-2	85.04	**75.53**	65.54	60.54	37.00	41.51	40.49	37.54	37.04	40.55	**39.03**	35.52	41.02	**54.00**	48.53	48.50
FrugalPrompt	80.51	64.99	64.02	57.05	**40.02**	38.99	37.98	**39.47**	39.54	**47.53**	37.54	**40.06**	40.03	45.02	49.04	47.01
Selection-p	69.97	63.01	61.51	61.99	40.02	**42.49**	**40.98**	30.98	33.02	32.00	30.04	30.54	45.56	46.54	48.53	**53.08**
PartPrompt	74.05	63.54	65.07	57.00	36.01	36.53	34.97	28.46	**40.56**	39.49	33.03	40.03	50.01	49.50	**50.52**	50.54
Ours	82.51	72.00	56.56	56.49	33.04	28.03	32.50	26.51	33.94	38.49	34.53	36.48	**54.02**	52.00	49.54	47.00

Table 10: Reconstruction accuracy (%) averaged over three receiver LLMs. RACE uses exact match; the three open-ended datasets use LLM-as-judge accuracy as the primary metric. Reference token-level F1 results are reported in the detailed tables below. **Bold** = best; underline = second; dashed = third.

D.1 RACE

Table 11 reports the per-model exact-match accuracy on RACE across all compression ratios. Since RACE is a multiple-choice task, LLM-as-judge evaluation is not applicable; the corresponding columns (DJ, RJ) are left blank.

D.2 Qasper

Table 12 reports the per-model reference token-level F1 and primary LLM-as-judge accuracy on Qasper across all compression ratios.

D.3 Multi-doc QA

Table 13 reports the per-model reference token-level F1 and primary LLM-as-judge accuracy on Multi-doc QA across all compression ratios.

D.4 LongMemEval

Table 14 reports the per-model reference token-level F1 and primary LLM-as-judge accuracy on LongMemEval across all compression ratios.

E Statistical Significance Analysis

We assess whether the gap between **Ours** and the strongest baseline at each (dataset, ratio, metric) cell is statistically significant. We run a two-sided bootstrap with 10,000 resamples (fixed seed 42); the two-sided p is $2\min(\Pr[\Delta \leq 0], \Pr[\Delta \geq 0])$ over the bootstrap distribution. p-values are corrected with the Benjamini–Hochberg procedure within each dataset across all tests. Tables 15 and 16 report Ours versus the strongest baseline. Across both tables, Ours is comparable to the strongest baseline in 83.9% of comparisons across reference token-level F1 and primary LLM-as-judge accuracy. By metric, the not-significantly-worse rate is 87.5% on token-level F1, 92% on LLM-as-judge, and 1/2 on RACE exact match. Direct-path LLM-as-judge accuracy on Qasper, Multi-doc QA, and LongMemEval is comparable to all optimal baselines.

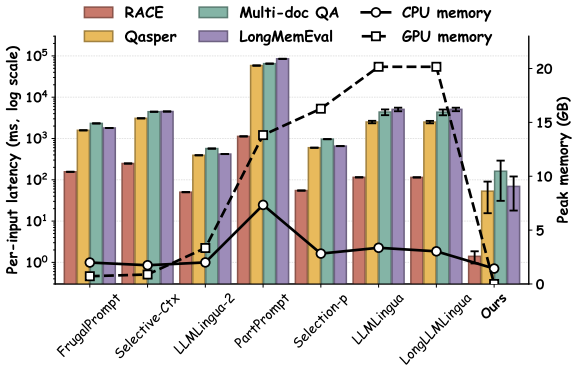

Figure 6: This figure shows the tendency of memory usage and the latency of the various compression methods

F Compressors' Cost Analysis

This appendix reports the latency and memory footprint of our evolved compressor at each of the four target ratios (1.5×, 2.5×, 4×, 6×). All measurements use the same 40-input subset of the test set (10 inputs per dataset × 4 datasets). Per-input latency is the median over three repeated calls. Peak CPU memory is the maximum process resident set size observed during the run. Our compressor runs on CPU with only spaCy. This is in contrast to all seven transformer baselines, whose runtime cost is reported in Table 18, 19, 20, 21, 22, and Figure 6 measured on a single NVIDIA A100.

Baseline measurements. Latency, GPU memory, and CPU memory for the seven transformer baselines were produced by the same harness on a single NVIDIA A100 (40 GB), one Colab notebook per

method, with the same 40-input benchmark set described above. Table 18 reports the cost at the $2.5\times$ target ratio; the ratio-dependent numbers vary by less than $\pm 5\%$ for transformer baselines because the per-input cost is dominated by the input encoding (model size and input length), not by the target ratio.

Per-dataset breakdown. The overall numbers in Table 18 aggregate across the 40-input benchmark set. Because input length differs by an order of magnitude across the four datasets (RACE ~350 words; Qasper ~3K; Multi-doc QA ~5–6K; LongMemEval up to ~10K+), Tables 19–22 report the per-input latency broken down by dataset, one table per target ratio. The dataset axis dominates: RACE inputs are processed by every baseline in < 300 ms, while LongMemEval inputs cost the same model 4–6 seconds each. Ours is the only method whose latency is dominated more by the per-ratio program architecture than by input length. * marks the four PartPrompt entries at $4\times$, which are imputed as the mean of the three completed ratios (1.5/2.5/$6\times$) because the $4\times$ run was interrupted after only five LongMemEval inputs; the imputation is consistent with the overall PartPrompt $4\times$ value in Table 18 and justified because PartPrompt's per-input cost is dominated by Stanford CoreNLP parsing, which is largely independent of target ratio.

G Best Evolved Compressors

This appendix presents the pseudocode of the best evolved compressor at each target compression ratio. All compressors share a common wrapper that enforces the target token count $n_{\text{target}} = \lfloor n_{\text{src}}/r \rceil$ by truncating or back-filling the output. The inner logic, which determines *how* tokens are ranked or selected, differs substantially across ratios, reflecting the architectural shift discussed in §5.2.

G.1 Best Compressor at $1.5\times$

Architecture: **Flat token-level scoring.** Each token receives a numeric score combining POS category, dependency role, entity chain strength, negation/causal keyword membership, and sentence-level bonuses. The top-n_{target} tokens are retained in document order.

```
function compress(text, ratio=1.5):
  tokens ← spaCy(text)
  target_n ← round(len(text.split()) /
ratio)
  must ← causal_connectives(text) ∪
temporal_markers(text)

  # Sentence-level bonuses
  for each sentence s:
    bonus ← 0
    if s is first sentence: bonus += 20
    if s contains digits: bonus += 18
    if s contains entities: bonus += 22
    assign bonus to all tokens in s

  # Entity chain strength (dynamic)
  for each entity lemma:
    strength ← 40 + min(45, 10 ×
occurrence_count)

  # Token scoring
  for each token t:
    score ← 0
    if t ∈ must or t.dep ∈ {ROOT, nsubj,
dobj, pobj, attr}: score += 55
    if t.dep = neg or t ∈ keywords: score +=
95
    if t.ent_type: score +=
chain_strength[t.lemma]
    score += POS_score(t) # NUM:88, PROPN:72,
NOUN:60, VERB:48/40, ...
    score += sentence_bonus[t]

  keep top target_n tokens by score
  return kept tokens in document order
```

G.2 Best Compressor at $2.5\times$

Architecture: **Tiered fill with hybrid truncation.** The compressor first extracts ROOT verb subtrees as structural skeletons, then fills remaining budget with answer-bearing cues (entities, numbers, negation, connectives) and noun chunk heads. When the collected set exceeds the budget, a hybrid strategy balances priority-based selection with evenly spaced sampling for document coverage.

```
function compress(text, ratio=2.5):
```

```
tokens ← spaCy(text)
target_n ← round(len(text.split()) / ratio)
keep ← {}

# Tier 1: ROOT subtree skeletons
for each ROOT verb r:
for each t in subtree(r):
if t.dep ∈ {ROOT, nsubj, dobj, attr, ccomp,
xcomp, neg}
or t.ent_type: add t to keep

# Tier 2: answer-bearing cues
for each token t:
if t.ent_type or t.pos = NUM or t.dep = neg
or t ∈ cue_words: add t to keep
add causal_connectives(text)
add temporal_markers(text)

# Tier 3: noun chunk heads
for each noun chunk: add last NOUN/PROPN to
keep

# Hybrid truncation if over budget
if |keep| > target_n:
priority ← tokens with entities, capitals,
or digits
chosen ← top half from priority
fill remaining slots by even spacing across
keep

return kept tokens in document order
```

G.3 Best Compressor at $4\times$

Architecture: **Sentence-then-token cascade.** A two-stage pipeline first scores and selects informative sentences, then applies token-level pruning within the selected sentences if the output still exceeds the budget.

```
function compress(text, ratio=4.0):
tokens ← spaCy(text)
sents ← segment(text)
target_n ← round(len(text.split()) / ratio)

# Stage 1: Sentence scoring and selection
for each sentence s:
score ← 0
if s contains entities: score += 2.5
if s contains numbers: score += 2.0
if s contains negation: score += 1.5
if s contains causal connectives: score +=
1.0
if s contains temporal markers: score += 1.0
if s contains ROOT verb: score += 1.0
select sentences by score until budget filled

# Stage 2: Token pruning (if still over
budget)
if output > target_n:
keep ← {entities, PROPN, NUM}
add causal & temporal connectives
for each ROOT verb: add subtree core roles
for each noun chunk: add head NOUN/PROPN

return kept tokens in document order
```

G.4 Best Compressor at $6\times$

Architecture: **Sentence selection + subtree deletion + post-processing.** The compressor first selects sentences by surface cues (first sentence, WH-question words, question marks, colons), then removes entire dispensable subtrees and function words within each selected sentence, and finally applies a post-processing chain.

```
function compress(text, ratio=6.0):
sents ← segment(text)
target_n ← round(len(text.split()) / ratio)

# Stage 1: Sentence selection
kept_sents ← {}
for each sentence s at index i:
if i = 0: add s # first sentence always kept
if s contains
{who,what,when,where,why,how,which}: add
s
if s contains "?" or ":": add s

# Stage 2: Subtree deletion + function
word removal
for each kept sentence s:
tokens ← spaCy(s)
for each token t:
if t.dep ∈ {advcl, ccomp, xcomp, relcl, acl,
appos}:
drop entire subtree(t)
if t.dep ∈ {det, aux, cop, mark, case, punct}:
drop t
if t.pos = ADP and t.lemma ∉ common_preps:
drop t

# Stage 3: Post-processing
output ← collapse_repeated_entities(output)
output ← compress_enumeration_blocks(output)
output ← merge_short_adjacent_sentences(output)

return output
```

H Experimental Prompts

This appendix collects, verbatim, every LLM prompt used in the pipeline: (i) the system / user prompts that drive the evolutionary mutator, (ii) the receiver-LM QA prompts (multiple-choice and open-ended), (iii) the reconstruction-model prompt that expands a compressed string back into prose, and (iv) the LLM-as-judge prompt used for the binary semantic-correctness grading.

Placeholders in curly braces (`{text}`, `{q}`, `{compressed}`, etc.) are substituted at runtime; everything else is sent to the LLM as written.

H.1 Mutator Prompts

The mutators are large LLMs that propose edits to a parent compressor's source code via SEARCH/REPLACE diffs. We also use a

challenger-round user template that forces the mutator to pursue a specific structural strategy when the population becomes homogeneous. The detailed system prompts are shown in Tables 23 and 24, and the challenger prompt is shown in Table 25.

H.2 Receiver LLM Prompts

The receiver LLM (`Qwen2.5-7B`, `Llama-3.1-8B`, `Gemma-3-12B`) is queried with one of two prompt templates depending on the dataset's answer format. In the Direct path, the receiver LLM reads the compressed text. In the Reconstruction path, a reconstruction LLM first expands the compressed text into prose, after which the receiver LLM reads the reconstructed prose. In both cases, the same receiver template is applied, with `{text}` bound to either the compressed string or the reconstructed string in Table 26.

H.2.1 Multiple-choice QA Prompts

`{o}` in Table 27 is the option block (e.g. `A. ...`, `B. ...`, ...), and `{L}` is the allowed-letter list (e.g. `A/B/C/D`) for the multiple-choice questions in the RACE dataset.

H.2.2 Open-ended QA Prompts

The remaining three datasets use open-ended answers, so we use the template in Table 28 as the prompt for receiver LLMs.

H.3 Reconstruction LLM Prompts

The reconstruction LLM, drawn from the same three-model pool as the receiver LLMs but rotated to a different model than the per-sample QA receiver, takes a compressed string and expands it into coherent prose. The reconstructed prose is then fed to a receiver LLM on the Reconstruction path. The detailed prompt is shown in Table 29.

H.4 LLM-as-Judge Prompts

The judge LLM grades each receiver response against the gold answer on a binary scale. An explicitly instructed `1/0` prompt, where reliability dominates and the wrapper preamble suppresses chain-of-thought leakage, is shown in Table 30.

Method	*Gemma-3-12B*					*LLaMA-3.1-8B*					*Qwen-2.5-7B*				
	r	DA	RA	DJ	RJ	r	DA	RA	DJ	RJ	r	DA	RA	DJ	RJ
Target ratio $1.5\times$															
Selective-Ctx	1.31	92.42	89.39	–	–	1.30	79.10	74.63	–	–	1.30	**100.00**	80.60	–	–
LLMLingua	1.43	84.85	**92.42**	–	–	1.37	80.60	**82.09**	–	–	1.44	**100.00**	**86.57**	–	–
LongLLMLingua	1.43	84.85	87.88	–	–	1.37	**85.07**	79.10	–	–	1.44	**100.00**	85.07	–	–
LLMLingua-2	1.57	92.42	**92.42**	–	–	1.56	80.60	82.09	–	–	1.59	92.54	80.60	–	–
FrugalPrompt	1.50	**95.45**	83.33	–	–	1.50	77.61	74.63	–	–	1.50	97.01	83.58	–	–
PartPrompt	1.49	80.30	83.33	–	–	1.57	67.16	64.18	–	–	1.58	92.54	74.63	–	–
Selection-p	1.49	72.73	63.64	–	–	1.50	79.10	79.10	–	–	1.49	85.07	67.16	–	–
Ours	1.50	87.88	84.85	–	–	1.50	77.61	80.60	–	–	1.50	92.54	82.09	–	–
Rank		4	5	–	–		7	3	–	–		7	4	–	–
Target ratio $2.5\times$															
Selective-Ctx	2.10	84.85	78.79	–	–	2.10	76.12	70.15	–	–	2.10	82.09	**70.15**	–	–
LLMLingua	2.11	77.27	71.21	–	–	2.11	**82.09**	71.64	–	–	2.11	86.57	64.18	–	–
LongLLMLingua	2.11	77.27	69.70	–	–	2.11	80.60	68.66	–	–	2.11	86.57	**70.15**	–	–
LLMLingua-2	2.94	**87.88**	**81.82**	–	–	2.94	82.09	74.63	–	–	2.94	**91.04**	**70.15**	–	–
FrugalPrompt	2.50	69.70	63.64	–	–	2.50	70.15	62.69	–	–	2.50	89.55	68.66	–	–
PartPrompt	2.58	72.73	71.21	–	–	2.58	73.13	52.24	–	–	2.58	89.55	67.16	–	–
Selection-p	2.48	69.70	65.15	–	–	2.48	68.66	56.72	–	–	2.48	82.09	67.16	–	–
Ours	2.50	74.24	72.73	–	–	2.50	64.18	**79.10**	–	–	2.50	70.15	64.18	–	–
Rank		5	3	–	–		8	1	–	–		8	8	–	–
Target ratio $4.0\times$															
Selective-Ctx	3.48	77.27	75.76	–	–	3.48	65.67	62.69	–	–	3.48	79.10	**71.64**	–	–
LLMLingua	2.87	77.27	71.21	–	–	2.87	70.15	**67.16**	–	–	2.87	**89.55**	59.70	–	–
LongLLMLingua	2.87	71.21	65.15	–	–	2.87	**73.13**	65.67	–	–	2.87	86.57	61.19	–	–
LLMLingua-2	4.95	72.73	72.73	–	–	4.95	71.64	**67.16**	–	–	4.95	83.58	56.72	–	–
FrugalPrompt	4.00	62.12	68.18	–	–	4.00	61.19	59.70	–	–	4.00	85.07	64.18	–	–
PartPrompt	4.12	**83.33**	**78.79**	–	–	4.12	62.69	55.22	–	–	4.12	77.61	61.19	–	–
Selection-p	4.00	62.12	63.64	–	–	4.00	61.19	62.69	–	–	4.00	85.07	58.21	–	–
Ours	4.00	66.67	68.18	–	–	4.00	64.18	58.21	–	–	4.00	67.16	43.28	–	–
Rank		6	6	–	–		5	7	–	–		8	8	–	–
Target ratio $6.0\times$															
Selective-Ctx	5.43	65.15	56.06	–	–	5.43	53.73	62.69	–	–	5.43	**89.55**	56.72	–	–
LLMLingua	3.62	**77.27**	**71.21**	–	–	3.62	67.16	**67.16**	–	–	3.62	86.57	53.73	–	–
LongLLMLingua	3.62	75.76	71.21	–	–	3.62	**68.66**	65.67	–	–	3.62	**89.55**	52.24	–	–
LLMLingua-2	7.74	62.12	68.18	–	–	7.74	64.18	58.21	–	–	7.74	77.61	55.22	–	–
FrugalPrompt	5.99	65.15	66.67	–	–	5.99	61.19	47.76	–	–	5.99	82.09	56.72	–	–
PartPrompt	6.21	68.18	57.58	–	–	6.21	67.16	59.70	–	–	6.21	74.63	53.73	–	–
Selection-p	6.09	65.15	59.09	–	–	6.09	53.73	62.69	–	–	6.09	**89.55**	**64.18**	–	–
Ours	6.00	56.06	54.55	–	–	6.00	53.73	55.22	–	–	6.00	68.66	59.70	–	–
Rank		8	8	–	–		8	7	–	–		8	2	–	–

Table 11: Detailed results on RACE across compression ratios. r = actual compression ratio; DA = direct accuracy (exact-match accuracy, %); RA = reconstruction accuracy (exact-match accuracy, %). RACE is a multiple-choice task; LLM-as-judge evaluation (DJ/RJ) is not applicable. **Bold** indicates the best result in each column. The Rank row shows the ranking of our method among all methods for each metric.

Method	*Gemma-3-12B*					*LLaMA-3.1-8B*					*Qwen-2.5-7B*				
	r	DA	RA	DJ	RJ	r	DA	RA	DJ	RJ	r	DA	RA	DJ	RJ
Target ratio $1.5\times$															
Selective-Ctx	1.26	**35.40**	23.59	62.12	37.88	1.26	45.53	29.57	71.64	31.34	1.26	30.65	17.34	50.75	37.31
LLMLingua	1.82	21.54	10.92	25.76	27.27	1.89	27.57	20.93	41.79	23.88	1.89	17.23	11.36	26.87	26.87
LongLLMLingua	1.82	20.35	12.64	30.30	27.27	1.89	28.16	24.70	38.81	22.39	1.89	17.68	11.10	32.84	23.88
LLMLingua-2	1.58	33.32	20.60	56.06	37.88	1.58	42.50	28.16	**74.63**	34.33	1.58	30.29	**21.93**	52.24	38.81
FrugalPrompt	1.50	31.74	20.78	54.55	**43.94**	1.50	**46.51**	**30.69**	67.16	34.33	1.50	**35.18**	20.29	61.19	**41.79**
PartPrompt	1.50	29.01	24.26	59.09	37.88	1.49	44.16	29.77	61.19	38.81	1.50	33.14	15.27	**65.67**	31.34
Selection-p	1.51	32.57	24.48	43.94	**43.94**	1.49	39.25	28.60	53.73	**44.78**	1.51	28.84	18.45	58.21	31.34
Ours	1.50	35.00	**25.20**	**65.15**	40.91	1.50	44.40	18.90	64.18	32.84	1.50	29.70	13.20	58.21	25.37
Rank		2	1	1	3		3	8	4	5		5	6	4	7
Target ratio $2.5\times$															
Selective-Ctx	2.02	26.78	16.71	51.52	30.30	2.02	38.29	26.31	52.24	32.84	2.02	29.77	17.46	43.28	35.82
LLMLingua	3.60	19.51	10.91	31.82	22.73	3.60	19.51	20.18	19.40	20.90	3.60	10.37	6.24	20.90	14.93
LongLLMLingua	3.60	17.40	8.87	22.73	18.18	3.60	21.43	19.81	20.90	19.40	3.60	10.33	9.18	22.39	14.93
LLMLingua-2	2.78	**35.09**	22.93	53.03	**43.94**	2.78	**42.52**	31.86	**59.70**	41.79	2.78	28.44	22.24	**59.70**	38.81
FrugalPrompt	2.50	28.08	**26.75**	37.88	37.88	2.50	39.55	**35.20**	55.22	**52.24**	2.50	28.21	17.99	46.27	26.87
PartPrompt	2.58	23.50	26.31	42.42	42.42	2.58	34.53	26.46	52.24	31.34	2.58	30.85	**22.90**	53.73	35.82
Selection-p	2.51	32.70	23.39	54.55	40.91	2.51	33.45	28.02	40.30	38.81	2.51	26.88	19.92	46.27	**47.76**
Ours	2.50	30.20	17.40	**54.55**	33.33	2.50	34.90	25.40	50.75	28.36	2.50	**39.60**	14.80	56.72	22.39
Rank		3	5	2	5		4	6	5	6		1	6	2	6
Target ratio $4.0\times$															
Selective-Ctx	3.35	27.09	20.80	42.42	28.79	3.35	32.68	23.66	**52.24**	35.82	3.35	24.56	22.43	34.33	35.82
LLMLingua	5.61	21.36	8.88	22.73	13.64	5.61	17.72	17.08	16.42	13.43	5.61	9.70	11.67	19.40	17.91
LongLLMLingua	5.63	16.70	10.85	15.15	15.15	5.63	15.82	14.89	19.40	17.91	5.63	10.28	10.04	22.39	11.94
LLMLingua-2	4.71	29.06	21.08	**51.52**	**39.39**	4.71	35.80	30.25	49.25	38.81	4.71	26.55	22.59	50.75	43.28
FrugalPrompt	4.00	21.70	26.39	28.79	34.85	4.00	**37.74**	**35.47**	46.27	**46.27**	4.00	27.68	20.07	50.75	32.84
PartPrompt	4.16	26.16	25.26	39.39	28.79	4.16	31.41	24.27	40.30	34.33	4.16	28.64	**26.17**	41.79	41.79
Selection-p	4.01	**32.19**	**28.15**	37.88	37.88	4.01	28.26	26.92	32.84	38.81	4.01	22.46	22.16	32.84	**46.27**
Ours	4.00	24.60	22.00	42.42	31.82	4.00	27.60	23.80	46.27	34.33	4.00	**36.40**	20.20	**50.75**	31.34
Rank		5	4	3	4		6	5	4	6		1	5	1	6
Target ratio $6.0\times$															
Selective-Ctx	5.38	20.48	13.44	24.24	21.21	5.38	17.41	16.73	19.40	14.93	5.38	20.06	19.83	29.85	31.34
LLMLingua	7.93	18.41	10.75	19.70	13.64	7.93	18.08	14.25	17.91	14.93	7.93	12.03	7.99	19.40	11.94
LongLLMLingua	7.92	17.97	14.99	19.70	22.73	7.92	17.51	15.37	19.40	14.93	7.92	13.43	7.65	22.39	17.91
LLMLingua-2	7.47	**35.91**	24.80	**56.06**	**45.45**	7.47	32.87	28.81	32.84	34.33	7.47	28.45	21.84	**52.24**	32.84
FrugalPrompt	6.00	25.78	**25.21**	37.88	33.33	6.00	**36.64**	**38.91**	43.28	**46.27**	6.00	28.14	18.63	47.76	**38.81**
PartPrompt	6.35	27.34	20.37	37.88	21.21	6.35	27.11	24.52	34.33	25.37	6.35	21.39	**22.95**	35.82	**38.81**
Selection-p	6.05	19.07	15.95	34.85	27.27	6.05	26.67	20.65	32.84	32.84	6.05	22.46	18.91	43.28	32.84
Ours	6.00	25.10	16.50	37.88	28.79	6.00	26.10	22.80	**44.78**	23.88	6.00	**29.70**	20.60	35.82	26.87
Rank		4	4	4	3		5	4	1	5		1	3	5	6

Table 12: Detailed results on Qasper across compression ratios. r = actual compression ratio; DA = reference direct token-level F1 (%); RA = reference reconstruction token-level F1 (%); DJ = direct LLM-as-judge accuracy (%); RJ = reconstruction LLM-as-judge accuracy (%). **Bold** indicates the best result in each column. The Rank row shows the ranking of our method among all methods for each metric.

Method	*Gemma-3-12B*					*LLaMA-3.1-8B*					*Qwen-2.5-7B*				
	r	DA	RA	DJ	RJ	r	DA	RA	DJ	RJ	r	DA	RA	DJ	RJ
Target ratio $1.5\times$															
Selective-Ctx	1.26	**54.87**	34.73	**71.21**	37.88	1.26	30.51	34.85	49.25	38.81	1.26	47.85	29.07	50.75	35.82
LLMLingua	1.44	53.91	34.11	65.15	37.88	1.42	46.36	19.93	55.22	20.90	1.42	44.88	29.93	46.27	25.37
LongLLMLingua	1.45	54.37	32.96	65.15	33.33	1.42	44.16	23.39	61.19	31.34	1.42	44.44	30.49	47.76	28.36
LLMLingua-2	1.59	54.39	41.81	65.15	45.45	1.59	44.10	30.26	59.70	35.82	1.60	**54.61**	30.87	**55.22**	29.85
FrugalPrompt	1.50	49.58	45.64	65.15	48.48	1.50	44.81	**36.13**	58.21	**38.81**	1.50	49.21	26.36	52.24	31.34
PartPrompt	1.52	47.69	**46.57**	62.12	**53.03**	1.53	36.19	27.10	49.25	32.84	1.53	47.12	34.28	46.27	35.82
Selection-p	1.51	46.31	35.74	60.61	37.88	1.50	41.03	32.04	53.73	35.82	1.52	43.64	26.49	41.79	25.37
Ours	1.50	48.10	22.00	65.15	22.73	1.50	**47.20**	29.50	**67.16**	32.84	1.50	47.50	**41.60**	**55.22**	**46.27**
Rank		6	8	6	8		1	5	1	6		4	1	2	1
Target ratio $2.5\times$															
Selective-Ctx	2.03	53.01	33.31	**72.73**	37.88	2.03	24.88	23.91	41.79	31.34	2.03	47.67	24.68	55.22	31.34
LLMLingua	2.40	35.86	33.55	43.94	36.36	2.40	28.82	21.92	46.27	28.36	2.40	38.39	22.48	43.28	19.40
LongLLMLingua	2.40	40.35	30.61	50.00	33.33	2.40	27.08	24.00	50.75	28.36	2.40	42.81	25.16	46.27	23.88
LLMLingua-2	2.86	43.46	39.22	60.61	50.00	2.86	38.83	30.22	**56.72**	35.82	2.86	43.85	32.79	47.76	35.82
FrugalPrompt	2.50	**55.90**	**47.42**	69.70	**53.03**	2.50	**43.23**	**42.92**	55.22	**46.27**	2.50	**55.05**	35.62	55.22	**43.28**
PartPrompt	2.63	50.24	39.14	62.12	37.88	2.63	27.12	31.48	41.79	37.31	2.63	41.75	**37.37**	38.81	**43.28**
Selection-p	2.51	35.00	25.86	45.45	31.82	2.51	27.84	34.41	38.81	37.31	2.51	47.93	25.92	47.76	26.87
Ours	2.50	46.60	30.60	56.06	36.36	2.50	38.90	35.40	53.73	43.28	2.50	51.50	34.60	**61.19**	35.82
Rank		4	7	5	6		2	2	3	2		2	3	1	4
Target ratio $4.0\times$															
Selective-Ctx	3.42	43.01	33.96	57.58	37.88	3.42	28.48	20.36	38.81	25.37	3.42	**49.61**	27.22	**52.24**	32.84
LLMLingua	3.98	30.38	30.40	31.82	33.33	3.98	23.92	22.94	43.28	26.87	3.98	40.40	29.36	44.78	29.85
LongLLMLingua	3.98	29.28	35.23	36.36	34.85	3.98	25.54	20.56	43.28	25.37	3.98	47.07	**34.89**	**52.24**	32.84
LLMLingua-2	4.94	43.97	40.77	60.61	45.45	4.94	**41.59**	29.71	**61.19**	**35.82**	4.94	38.99	29.43	41.79	35.82
FrugalPrompt	4.00	**56.36**	**45.44**	**68.18**	**45.45**	4.00	33.21	**32.19**	38.81	34.33	4.00	44.39	27.35	47.76	32.84
PartPrompt	4.38	42.45	41.24	48.48	39.39	4.38	20.67	19.09	29.85	22.39	4.38	43.80	32.45	46.27	**37.31**
Selection-p	4.03	42.47	36.01	45.45	37.88	4.03	30.63	22.96	38.81	26.87	4.03	33.38	26.27	32.84	25.37
Ours	4.00	45.30	36.00	54.55	40.91	4.00	30.20	22.20	47.76	28.36	4.00	41.60	27.80	50.75	34.33
Rank		2	5	4	3		4	5	2	3		5	5	3	3
Target ratio $6.0\times$															
Selective-Ctx	5.41	44.02	32.18	45.45	31.82	5.41	29.47	24.07	41.79	29.85	5.41	42.92	32.29	**47.76**	**38.81**
LLMLingua	6.11	27.78	27.91	39.39	27.27	6.11	31.49	28.85	46.27	35.82	6.11	36.55	27.52	38.81	25.37
LongLLMLingua	6.13	27.93	26.81	40.91	24.24	6.13	34.07	30.21	46.27	37.31	6.13	37.30	27.29	41.79	25.37
LLMLingua-2	7.92	45.96	34.52	60.61	39.39	7.92	**39.85**	34.69	**49.25**	38.81	7.92	32.63	28.88	35.82	28.36
FrugalPrompt	6.00	**58.67**	**47.85**	**68.18**	**51.52**	6.00	36.30	28.51	37.31	34.33	6.00	42.69	28.31	44.78	34.33
PartPrompt	6.82	41.86	39.11	40.91	45.45	6.82	23.65	32.70	31.34	38.81	6.82	41.78	**33.18**	40.30	35.82
Selection-p	6.06	34.34	34.88	37.88	37.88	6.06	26.06	26.69	22.39	28.36	6.06	34.02	31.07	34.33	25.37
Ours	6.00	36.30	25.20	46.97	31.82	6.00	30.80	**40.30**	38.81	**47.76**	6.00	**43.00**	25.40	43.28	29.85
Rank		5	8	3	6		5	1	5	1		1	8	3	4

Table 13: Detailed results on Multi-doc QA across compression ratios. r = actual compression ratio; DA = reference direct token-level F1 (%); RA = reference reconstruction token-level F1 (%); DJ = direct LLM-as-judge accuracy (%); RJ = reconstruction LLM-as-judge accuracy (%). **Bold** indicates the best result in each column. The Rank row shows the ranking of our method among all methods for each metric.

Method	*Gemma-3-12B*					*LLaMA-3.1-8B*					*Qwen-2.5-7B*				
	r	DA	RA	DJ	RJ	r	DA	RA	DJ	RJ	r	DA	RA	DJ	RJ
Target ratio $1.5\times$															
Selective-Ctx	1.26	2.35	2.37	53.03	45.45	1.27	1.82	1.21	43.28	43.28	1.28	1.50	**2.72**	35.82	46.27
LLMLingua	1.41	**6.01**	2.87	45.45	48.48	1.43	0.84	2.25	**50.75**	43.28	1.42	2.21	1.37	31.34	40.30
LongLLMLingua	1.41	**6.01**	2.90	46.97	54.55	1.43	0.81	2.22	49.25	52.24	1.42	1.50	0.94	32.84	38.81
LLMLingua-2	1.57	2.35	2.40	50.00	45.45	1.57	0.88	0.00	46.27	43.28	1.58	0.70	0.39	38.81	34.33
FrugalPrompt	1.50	4.24	2.49	39.39	45.45	1.50	0.20	1.03	34.33	32.84	1.50	4.06	2.15	37.31	41.79
PartPrompt	1.50	5.66	3.24	**54.55**	51.52	1.50	2.54	1.07	40.30	53.73	1.51	**5.74**	1.25	40.30	44.78
Selection-p	1.50	0.55	2.96	48.48	**57.58**	1.50	0.00	**2.40**	46.27	38.81	1.50	2.61	1.02	43.28	40.30
Ours	1.50	2.30	**3.60**	30.30	**57.58**	1.50	**3.50**	0.20	50.75	**53.73**	1.50	2.40	0.80	**47.76**	**50.75**
Rank		7	1	8	2		1	7	2	2		4	7	1	1
Target ratio $2.5\times$															
Selective-Ctx	2.03	2.26	2.25	42.42	48.48	2.03	0.27	0.45	35.82	31.34	2.03	0.82	1.11	34.33	47.76
LLMLingua	2.36	4.91	2.89	48.48	51.52	2.36	0.56	1.63	43.28	46.27	2.36	1.92	1.91	25.37	40.30
LongLLMLingua	2.36	4.91	2.58	**54.55**	39.39	2.36	0.43	0.39	37.31	46.27	2.36	2.92	2.24	25.37	38.81
LLMLingua-2	2.73	1.71	**3.10**	51.52	53.03	2.73	0.65	1.89	44.78	**52.24**	2.73	1.69	**2.83**	29.85	56.72
FrugalPrompt	2.50	4.24	2.52	**54.55**	48.48	2.50	1.81	0.55	38.81	32.84	2.50	**3.66**	1.04	46.27	53.73
PartPrompt	2.52	**5.56**	2.90	53.03	50.00	2.52	**2.69**	0.59	41.79	46.27	2.52	3.38	1.83	**49.25**	52.24
Selection-p	2.47	1.71	2.87	**54.55**	**54.55**	2.47	2.26	1.73	46.27	38.81	2.47	2.61	0.89	46.27	46.27
Ours	2.50	0.50	2.30	45.45	51.52	2.50	2.30	**2.20**	**55.22**	46.27	2.50	1.30	2.10	47.76	**58.21**
Rank		8	7	7	4		3	1	1	5		7	3	2	1
Target ratio $4.0\times$															
Selective-Ctx	3.28	2.30	2.68	48.48	46.97	3.28	0.10	1.55	46.27	47.76	3.28	0.00	1.80	35.82	**53.73**
LLMLingua	3.90	2.35	**3.00**	33.33	50.00	3.90	1.69	0.67	37.31	50.75	3.90	2.32	1.51	47.76	37.31
LongLLMLingua	3.90	2.35	2.19	37.88	46.97	3.90	1.77	1.47	41.79	50.75	3.90	2.32	1.51	47.76	31.34
LLMLingua-2	4.58	2.58	2.72	53.03	54.55	4.58	0.85	**2.67**	43.28	41.79	4.58	0.00	**2.75**	22.39	49.25
FrugalPrompt	4.00	4.24	2.58	**57.58**	**57.58**	4.00	1.42	0.80	46.27	43.28	4.00	2.52	0.89	43.28	46.27
PartPrompt	4.10	**5.87**	2.94	**57.58**	54.55	4.10	**2.63**	1.38	**49.25**	44.78	4.10	**5.13**	1.46	46.27	52.24
Selection-p	3.96	4.24	2.72	**57.58**	54.55	3.96	1.76	1.73	34.33	38.81	3.96	2.53	1.11	41.79	52.24
Ours	4.00	2.20	1.40	43.94	57.58	4.00	0.80	1.20	47.76	**53.73**	4.00	1.90	0.30	**53.73**	37.31
Rank		8	8	6	2		7	6	2	1		6	8	1	7
Target ratio $6.0\times$															
Selective-Ctx	5.03	3.52	3.10	45.45	46.97	5.03	2.06	1.78	49.25	38.81	5.03	1.23	0.89	40.30	50.75
LLMLingua	6.03	1.71	**3.33**	48.48	54.55	6.03	0.81	1.58	41.79	41.79	6.03	1.50	**2.30**	32.84	40.30
LongLLMLingua	6.03	1.71	2.90	45.45	59.09	6.03	1.48	2.00	41.79	41.79	6.03	1.50	2.11	**44.78**	32.84
LLMLingua-2	7.15	**5.37**	2.63	57.58	48.48	7.15	1.95	**2.30**	38.81	**46.27**	7.15	0.00	0.69	26.87	50.75
FrugalPrompt	6.00	4.24	3.06	54.55	50.00	6.00	1.46	1.81	43.28	40.30	6.00	2.88	1.07	43.28	50.75
PartPrompt	6.39	4.61	2.84	46.97	57.58	6.39	**2.15**	1.12	41.79	40.30	6.39	**4.27**	1.07	38.81	53.73
Selection-p	5.96	3.08	3.19	**59.09**	**68.18**	5.96	1.30	2.08	37.31	38.81	5.96	1.79	0.89	35.82	52.24
Ours	6.00	0.70	1.20	39.39	46.97	6.00	1.60	1.70	**53.73**	40.30	6.00	0.90	1.50	40.30	**53.73**
Rank		8	8	8	8		4	6	1	6		7	3	4	2

Table 14: Detailed results on LongMemEval across compression ratios. r = actual compression ratio; DA = reference direct token-level F1 (%); RA = reference reconstruction token-level F1 (%); DJ = direct LLM-as-judge accuracy (%); RJ = reconstruction LLM-as-judge accuracy (%). **Bold** indicates the best result in each column. The Rank row shows the ranking of our method among all methods for each metric.

Dataset	Ratio	Best baseline	Best	Ours	Δ	95% CI	p_{corr}	Sig.
DA: Direct reference accuracy (%; RACE = exact match, others = token-level F1)								
RACE	1.5×	Selective-Ctx	90.51	86.01	−4.50	[−10.3, +1.4]	.285	≈
RACE	2.5×	LLMLingua-2	87.00	69.52	−17.48	[−25.2, −10.0]	< .001	↓
RACE	4.0×	LLMLingua	78.99	66.00	−12.99	[−21.5, −4.7]	.027	↓
RACE	6.0×	LongLLMLingua	77.99	59.48	−18.51	[−26.9, −10.1]	< .001	↓
Qasper	1.5×	FrugalPrompt	37.81	36.37	−1.44	[−7.4, +4.7]	.722	≈
Qasper	2.5×	LLMLingua-2	35.35	34.90	−0.45	[−6.2, +5.3]	.893	≈
Qasper	4.0×	LLMLingua-2	30.47	29.53	−0.94	[−6.6, +4.5]	.817	≈
Qasper	6.0×	LLMLingua-2	32.41	26.97	−5.44	[−11.0, +0.1]	.131	≈
Multi-doc QA	1.5×	LLMLingua-2	51.03	47.60	−3.43	[−11.5, +4.7]	.711	≈
Multi-doc QA	2.5×	FrugalPrompt	51.39	45.67	−5.73	[−14.0, +2.5]	.578	≈
Multi-doc QA	4.0×	FrugalPrompt	44.65	39.03	−5.62	[−13.6, +2.4]	.578	≈
Multi-doc QA	6.0×	FrugalPrompt	45.89	36.70	−9.19	[−17.3, −1.2]	.276	≈
LongMemEval	1.5×	PartPrompt	4.65	2.73	−1.91	[−3.6, −0.3]	.187	≈
LongMemEval	2.5×	PartPrompt	3.88	1.37	−2.51	[−4.1, −0.9]	.052	≈
LongMemEval	4.0×	PartPrompt	4.54	1.63	−2.91	[−4.7, −1.1]	.052	≈
LongMemEval	6.0×	PartPrompt	3.68	1.07	−2.61	[−4.2, −1.0]	.045	↓
RA: Reconstruction reference accuracy (%; RACE = exact match, others = token-level F1)								
RACE	1.5×	LLMLingua	87.03	82.51	−4.52	[−11.1, +1.8]	.285	≈
RACE	2.5×	LLMLingua-2	75.53	72.00	−3.53	[−11.8, +4.7]	.552	≈
RACE	4.0×	Selective-Ctx	70.03	56.56	−13.47	[−22.1, −4.5]	.027	↓
RACE	6.0×	LLMLingua	64.04	56.49	−7.55	[−16.4, +1.7]	.196	≈
Qasper	1.5×	FrugalPrompt	23.92	19.10	−4.82	[−9.8, −0.0]	.116	≈
Qasper	2.5×	FrugalPrompt	26.65	19.20	−7.45	[−13.0, −2.3]	.024	↓
Qasper	4.0×	FrugalPrompt	27.31	22.00	−5.31	[−10.6, +0.2]	.139	≈
Qasper	6.0×	FrugalPrompt	27.58	19.97	−7.62	[−13.1, −2.3]	.019	↓
Multi-doc QA	1.5×	FrugalPrompt	36.04	31.03	−5.01	[−12.5, +2.6]	.600	≈
Multi-doc QA	2.5×	FrugalPrompt	41.99	33.53	−8.45	[−16.4, −0.4]	.314	≈
Multi-doc QA	4.0×	FrugalPrompt	34.99	28.67	−6.33	[−14.0, +1.4]	.490	≈
Multi-doc QA	6.0×	PartPrompt	35.00	30.30	−4.70	[−12.5, +3.0]	.600	≈
LongMemEval	1.5×	LLMLingua	2.16	1.53	−0.63	[−1.9, +0.7]	.746	≈
LongMemEval	2.5×	LLMLingua-2	2.61	2.20	−0.41	[−1.8, +1.1]	.851	≈
LongMemEval	4.0×	LLMLingua-2	2.71	0.97	−1.75	[−3.1, −0.4]	.122	≈
LongMemEval	6.0×	LLMLingua	2.40	1.47	−0.94	[−2.2, +0.3]	.475	≈

Table 15: Statistical significance of **Ours** versus the strongest baseline at each (dataset, ratio) cell on **reference non-judge accuracy**. Δ = Ours − Best. 95% CI from a two-sided bootstrap with 10,000 resamples (seed 42). p_{corr} is Benjamini–Hochberg FDR corrected within each dataset across all (ratio, metric, baseline) tests. Sig.: ↑ = Ours significantly better; ≈ = comparable (not significantly worse); ↓ = Ours significantly worse ($\alpha = 0.05$).

Dataset	Ratio	Best baseline	Best	Ours	Δ	95% CI	p_{corr}	Sig.
DJ: Direct judge accuracy (%)								
Qasper	1.5×	PartPrompt	61.99	62.51	+0.53	[−8.2, +9.3]	.953	≈
Qasper	2.5×	LLMLingua-2	57.48	54.00	−3.48	[−12.8, +5.7]	.650	≈
Qasper	4.0×	LLMLingua-2	50.51	46.48	−4.03	[−13.2, +5.2]	.548	≈
Qasper	6.0×	LLMLingua-2	47.05	39.49	−7.55	[−16.7, +1.6]	.210	≈
Multi-doc QA	1.5×	LLMLingua-2	60.03	62.51	+2.49	[−6.2, +11.2]	.786	≈
Multi-doc QA	2.5×	FrugalPrompt	60.05	57.00	−3.05	[−12.4, +6.2]	.750	≈
Multi-doc QA	4.0×	LLMLingua-2	54.53	51.02	−3.51	[−13.0, +5.9]	.731	≈
Multi-doc QA	6.0×	FrugalPrompt	50.09	43.02	−7.07	[−16.5, +2.2]	.578	≈
LongMemEval	1.5×	Selection-p	46.01	42.94	−3.08	[−12.4, +6.2]	.750	≈
LongMemEval	2.5×	Selection-p	49.03	49.48	+0.45	[−8.9, +9.7]	.977	≈
LongMemEval	4.0×	PartPrompt	51.03	48.48	−2.56	[−12.0, +6.5]	.835	≈
LongMemEval	6.0×	FrugalPrompt	47.04	44.47	−2.56	[−11.8, +6.7]	.816	≈
RJ: Reconstruction judge accuracy (%)								
Qasper	1.5×	FrugalPrompt	40.02	33.04	−6.98	[−15.7, +1.8]	.223	≈
Qasper	2.5×	Selection-p	42.49	28.03	−14.47	[−23.5, −5.6]	.005	↓
Qasper	4.0×	Selection-p	40.98	32.50	−8.49	[−17.6, +0.6]	.139	≈
Qasper	6.0×	FrugalPrompt	39.47	26.51	−12.96	[−21.7, −4.2]	.010	↓
Multi-doc QA	1.5×	PartPrompt	40.56	33.94	−6.62	[−15.4, +2.1]	.578	≈
Multi-doc QA	2.5×	FrugalPrompt	47.53	38.49	−9.04	[−18.1, +0.4]	.424	≈
Multi-doc QA	4.0×	LLMLingua-2	39.03	34.53	−4.50	[−13.6, +4.4]	.708	≈
Multi-doc QA	6.0×	FrugalPrompt	40.06	36.48	−3.58	[−12.7, +5.6]	.711	≈
LongMemEval	1.5×	PartPrompt	50.01	54.02	+4.01	[−5.4, +13.1]	.750	≈
LongMemEval	2.5×	LLMLingua-2	54.00	52.00	−2.00	[−11.5, +7.5]	.906	≈
LongMemEval	4.0×	PartPrompt	50.52	49.54	−0.98	[−9.4, +9.7]	.978	≈
LongMemEval	6.0×	Selection-p	53.08	47.00	−6.08	[−15.4, +3.4]	.579	≈

Table 16: Statistical significance of **Ours** versus the strongest baseline at each (dataset, ratio) cell on **LLM-as-judge accuracy**. Δ = Ours − Best. 95% CI from a two-sided bootstrap with 10,000 resamples (seed 42). p_{corr} is Benjamini–Hochberg FDR corrected within each dataset across all (ratio, metric, baseline) tests. Sig.: ↑ = Ours significantly better; ≈ = comparable (not significantly worse); ↓ = Ours significantly worse ($\alpha = 0.05$).

Dataset	**1.5×**	**2.5×**	**4×**	**6×**
RACE	2.5 ±1.9	0.7 ±0.5	1.8 ±1.2	0.6 ±0.4
Qasper	164.5 ±259.1	9.2 ±7.8	28.8 ±31.9	9.4 ±7.7
Multi-doc QA	549.5 ±498.2	13.5 ±8.5	67.6 ±53.7	13.5 ±8.6
LongMemEval	221.6 ±322.5	10.5 ±8.9	34.2 ±36.4	10.2 ±8.3
Overall mean	**234.5** ±370.5	**8.4** ±8.5	**33.1** ±42.0	**8.4** ±8.4

Table 17: Per-input compression latency (ms, median per input) of our evolved compressor across the four target ratios. Each cell is mean ± standard deviation over 10 inputs per dataset (3 reps each, median taken). Run on Apple M-series CPU; peak process memory is 1.4 GB at every ratio (spaCy `en_core_web_sm` ~12M parameters).

Method	**Backbone (params)**	**Load (s)**	**GPU (GB)**	**CPU (GB)**	**Latency (ms)**			
					1.5×	2.5×	4×	6×
Selective-Ctx	GPT-2 (124M)	20	0.88	1.74	3087.6	3066.6	3071.3	3049.1
LLMLingua	Llama-2-7B	67	20.16	3.36	3946.1	3073.9	2630.8	2405.2
LongLLMLingua	Llama-2-7B	59	20.16	3.03	3923.6	3067.3	2628.7	2406.1
LLMLingua-2	XLM-R-large (560M)	32	3.34	2.00	359.1	357.6	357.8	356.7
FrugalPrompt	Electra-large (335M)	15	0.72	1.99	1465.0	1460.8	1459.6	1459.8
Selection-p	Llama-2-7B + adapter	111	16.27	2.82	566.7	567.6	567.0	566.8
PartPrompt	Llama-2-7B + CoreNLP	65	13.82	7.34	51784.2	53538.8	*52323.0	51646.5
Ours	**spaCy (12M, CPU)**	**0.6**	**0.0**	**1.43**	**234.5**	**8.4**	**33.1**	**8.4**

Table 18: Compression cost across methods at all four target ratios. Baselines are executated based on A100 GPU, but ours are on CPU-only device.

Method	RACE	Qasper	Multi-doc QA	LongMemEval
Selective-Ctx	250.0	3088.8	4475.3	4536.4
LLMLingua	114.1	3016.4	6229.3	6424.6
LongLLMLingua	114.3	2998.1	6189.4	6392.5
LLMLingua-2	51.2	391.6	573.6	420.1
FrugalPrompt	156.1	1582.7	2317.8	1803.5
Selection-p	54.5	595.7	962.1	654.3
PartPrompt	1125.5	57246.2	64097.0	84667.9
Ours	**2.5**	**164.5**	**549.5**	**221.6**

Table 19: Per-input compression latency (ms) at the $1.5\times$ target ratio, broken down by dataset. Mean over 10 inputs per dataset $\times$ 3 reps (median per input).

Method	RACE	Qasper	Multi-doc QA	LongMemEval
Selective-Ctx	247.7	3082.5	4439.0	4497.1
LLMLingua	115.1	2416.1	4544.8	5219.7
LongLLMLingua	114.2	2410.5	4518.3	5226.3
LLMLingua-2	49.4	391.9	568.6	420.7
FrugalPrompt	157.7	1574.1	2316.9	1794.6
Selection-p	55.0	599.1	962.7	653.5
PartPrompt	1134.9	60262.6	66352.5	86405.3
Ours	**0.7**	**9.2**	**13.5**	**10.5**

Table 20: Per-input compression latency (ms) at the $2.5\times$ target ratio, broken down by dataset.

Method	RACE	Qasper	Multi-doc QA	LongMemEval
Selective-Ctx	248.0	3074.0	4466.5	4496.7
LLMLingua	114.7	2303.6	3605.1	4499.6
LongLLMLingua	114.3	2301.3	3575.7	4523.3
LLMLingua-2	50.6	393.1	569.1	418.3
FrugalPrompt	155.6	1571.0	2314.7	1796.9
Selection-p	54.8	598.6	961.9	652.7
PartPrompt	1123.8	58359.0	64891.7	84918.0
Ours	**1.8**	**28.8**	**67.6**	**34.2**

Table 21: Per-input compression latency (ms) at the $4\times$ target ratio, broken down by dataset.

Method	RACE	Qasper	Multi-doc QA	LongMemEval
Selective-Ctx	245.4	3061.8	4412.2	4477.0
LLMLingua	114.5	2292.6	3075.1	4138.8
LongLLMLingua	114.0	2285.6	3064.9	4159.9
LLMLingua-2	49.3	391.0	568.9	417.6
FrugalPrompt	155.3	1575.0	2316.1	1792.8
Selection-p	55.1	598.7	960.9	652.4
PartPrompt	1111.1	57568.2	64225.7	83680.8
Ours	**0.6**	**9.4**	**13.5**	**10.2**

Table 22: Per-input compression latency (ms) at the $6\times$ target ratio, broken down by dataset.

```
You are an expert evolutionary search agent helping discover small Python
`compress(text: str) -> str` functions that reduce document length while
preserving the information a small receiver LM (7-12B) needs to answer questions.

## Research context

This work studies **linguistic-knowledge-based text compression** -- NOT
general "LLM-as-compressor". The paper contribution is demonstrating that
rule-based + structural-linguistic compressors, optimized via evolution,
can match or beat prior LLM-compression baselines (LLMLingua, partprompt,
selective_context) on held-out QA. Therefore:

### LM usage constraints (HARD RULES)

1. **Allowed**: structural linguistic analyzers that produce deterministic
   outputs:
   - `spacy` -- tokens / POS / dependency tree / NER (core tooling)

2. **Allowed (non-LM)**: frequency-based heuristics
   - `wordfreq.zipf_frequency(token)` -- word-frequency table for UID-inspired
     surprisal proxies. NO neural LM.
   - NLTK WordNet -- lexical hierarchies (hypernyms)

3. **FORBIDDEN**:
   - Do NOT prompt an LLM to "compress this text" or "summarize this" --
     that is LLM-as-compressor, which defeats the paper's
     linguistic-knowledge thesis.
   - Do NOT use neural LM surprisal estimators (minicons, GPT-2 perplexity) --
     these are "LM-based scoring" and blur the contribution. Use wordfreq
     instead.
   - Do NOT call the receiver or reconstruction model inside `compress()`.
```

Table 23: System prompt for the LLM mutator during evolution

```
## Contract

- You ONLY rewrite code inside `# EVOLVE-BLOCK-START` / `# EVOLVE-BLOCK-END`
  markers. Leave everything outside them byte-identical.
- `compress` must always return a non-empty string.
- The compression ratio `len(text.split()) / len(compress(text).split())`
  must stay in [1.3, 7.0]. Outside this range -> rejected.
- The output can be prose, JSON, PENMAN graph, triples, bullet list -- the
  reconstruction model will expand any format back into prose for QA.
- Fitness = `min(direct_qa_acc, reconstruct_qa_acc)` (after floor gates).
  Floor = 80% x baseline on both paths. Admissibility additionally requires
  n-gram overlap with source in [0.15, 0.75] (prevents fabrication or
  near-identity outputs).

## What good evolution looks like

- Small, targeted edits usually beat full rewrites -- start with the existing
  structure, change one layer at a time.
- Wrap risky logic in `try/except` with a conservative fallback (e.g.,
  first N/3 tokens). A crash is a hard zero on fitness.
- Diversity matters: the archive uses MAP-Elites + behavior fingerprint
  dedup. If your output on 8 fixed probes exactly matches an existing
  archive member, your candidate is rejected as a clone at stage1 (before
  any LLM calls). Tweaking variable names or whitespace doesn't change
  behavior; make real strategy changes.

## Metrics you see above

Every prompt includes the parent's current `comp_ratio`, `direct_qa_acc`,
`reconstruct_qa_acc`, `combined_score`, and which MAP-Elites bin it
occupies. These describe the parent -- you decide how to modify.
```

Table 24: System prompt for the LLM mutator during evolution (continued)

```
## CHALLENGER ROUND -- iteration {evolution_round}

The archive contains many similar compressors and the last few iterations
have produced incremental variations. Your task this round is to produce a
**STRUCTURALLY DIFFERENT** compressor using this specific strategy:

**Required strategy:** {forced_strategy}

## Rules

1. Do NOT tweak the parent. Rewrite the entire compress() body.
2. The output behavior MUST match the strategy. Examples:
   - "structured JSON output" -> compress returns `json.dumps({...})`
   - "OpenIE triples" -> compress returns `"subj1|verb1|obj1 | subj2|verb2|obj2"`
   - "AMR PENMAN notation" -> compress returns a PENMAN-formatted graph string
   - "coref collapse" -> compress uses fastcoref cached clusters to
     substitute pronouns with canonical mentions, then drops stopwords
   - "wordfreq filter" -> compress uses `wordfreq.zipf_frequency(token)` to
     drop the most predictable (high-zipf) tokens, keeping informational ones
3. **LM usage constraints** (hard rule, do not violate):
   - You MAY call structural analyzers (spacy, amrlib, fastcoref,
     frame_semantic_transformer, stanza) -- these output deterministic
     linguistic structures
   - You MAY read precomputed caches (`PRECOMPUTED_CACHE`)
   - You MUST NOT directly prompt an LLM to "compress this text" -- that would
     make your approach identical to LLMLingua / LLM-as-compressor
   - You MUST NOT use neural LM surprisal estimators (minicons, GPT-2) -- use
     wordfreq for frequency-based heuristics instead
4. Target compression ratio hint: {target_comp_ratio:.2f}x (but ratio is
   enforced by the V11 wrapper; focus on the strategy, not the ratio)

## Parent context (what NOT to imitate)

Parent compress() body:
{parent_code}

Parent metrics:
{parent_metrics}

## Your task

Write a valid Python function body for `compress(text: str) -> str` that:
(a) implements the required strategy,
(b) produces observably different compressed text than the parent, and
(c) preserves information that allows a small receiver LM to answer
    questions about the original text.

## Output format -- STRICT SEARCH/REPLACE DIFF

You MUST output exactly the SEARCH/REPLACE block format below. The harness
parses this format with a regex; any deviation = your edit is dropped.

<<<<<<< SEARCH
<paste the exact lines of the parent compress() that you want to replace>
=======
<paste the new lines that replace them>
>>>>>>> REPLACE
```

Table 25: Challenger prompt for LLM mutators

Table 26: Prompt for receiver LLMs

```
IMPORTANT: Output ONLY the requested answer -- no explanations, no chain of
thought, no commentary.

[receiver template follows below]
```

```
Read the following text carefully.

TEXT:
{text}

QUESTION: {q}
OPTIONS:
{o}

Answer with only the letter ({L}).
```

Table 27: RACE dataset QA template

```
Read the following passage and answer the question.

PASSAGE:
{text}

QUESTION: {q}

Answer concisely (one phrase or sentence).
```

Table 28: Prompt template for open-ended questions

```
You are a reconstruction model. The compressed input may be in ANY of these
formats: (a) prose text, (b) AMR in PENMAN notation with parentheses and
roles like :ARG0 :ARG1 :location, (c) OpenIE triples formatted 'subj|verb|obj'
separated by '||' or newlines, (d) a bullet list or '- ' fragments, or
(e) telegraphic tokens with connective words dropped. Identify the format,
parse it, and expand into coherent English prose that preserves ALL entities,
numbers, and relationships. Do not add information not in the input.
Output ONLY the reconstructed text, no preamble.

Compressed: {compressed}

Reconstructed:
```

Table 29: Prompt for reconstruction LLMs

```
You are an expert evaluator. Given a question, a model response, and gold
answer(s), decide whether the response correctly answers the question.

QUESTION: {question}

MODEL RESPONSE: {response}

GOLD ANSWER(S): {golds}

Output ONLY "1" if the response substantively matches at least one gold
answer (paraphrasing OK), or "0" otherwise. No explanation.
```

Table 30: LLM-as-judge prompt for GPT-5.4-mini