

WHAT SURVIVES LEARNED SYMBOLIC COMPRESSION?

Anonymous authors

Paper under double-blind review

ABSTRACT

Lossy text compression for language-model pipelines is judged by reconstruction distance or by downstream task accuracy, and neither says whether the compressed representation still states what the source stated. We measure agreement with a constructed reference for grammar-constrained symbolic codes. The reference represents source facts as equations, and a fixed decoder recovers the equations a code encodes. An exact rule system compares their consequences independently of the code’s well-formedness checker. On controlled arithmetic micro-worlds, we evaluate learned translators from 70M to 2.8B parameters against structured controls and learned baselines at matched byte and token budgets. Passing the checker is not preserving the source: a self-consistent mistranslation passes it, and so do sampled learned-translation errors. Across byte budgets, the best learned trace translator trails the strongest compact structured control, with area under the closure-agreement curve of 0.67 against 0.86. Increasing translator size does not close this aggregate gap; the largest Pythia translator matches the best smaller translator’s saved byte-axis agreement scores. Source fidelity relative to the constructed reference, internal validity, and what a consumer model recovers are three different quantities and should be measured separately.

1 INTRODUCTION

A compressed representation becomes a substitute for its source. Its usefulness depends on what survives that substitution, including facts that a downstream task may never ask about. Prompt pruning and learned summaries reduce the cost of long contexts, but successful downstream answers test only the information those answers require (Jiang et al., 2023; Pan et al., 2024; Xu et al., 2024). We study a complementary question: how much of a specified set of source facts and consequences survives in the representation itself?

Symbolic codes make this question concrete because their statements can be decoded and checked. Yet a checker can accept a code that describes the wrong world. Suppose the source states $x = 2$, $y = x + 3$, and $z = 2y$. A code that changes the offset to $y = x + 4$ can consistently compute and check $z = 12$. Its internal checks succeed even though the source entails $z = 10$. Distinguishing these outcomes requires a comparison with the source, independent of the code’s own checks.

We make that comparison exact on controlled affine arithmetic. A **reference projection** records source facts as canonical rational equations, and a fixed decoder maps the code into the same equation language. Exact rules form finite sets of explicit equations, entailed values, and pairwise differences. Their Jaccard overlap is **closure agreement**; a trace accepted independently by the code’s checker is **linter-valid**. A separate consumer task measures whether another model recovers the named values from the representation. The construction connects source-grounded compression with closure-based distortion for deductive sources (Trukhina & Vashkelis, 2026a; Xu, 2026c;a).

These instruments expose different properties of learned compression. Sampled learned outputs contain valid codes with source errors. Across byte budgets, the strongest compact structured control retains more measured content than the best trace translator; larger translators leave this aggregate gap open in the evaluated training setup. Token accounting changes the leading structured control. Consumer recovery then tests the representations with a model that must extract their values, under fixed response conditions.

Reviewers: please read the [Reviewer Guidelines](https://iclr.cc/Conferences/2027/ReviewerGuidelines) (iclr.cc/Conferences/2027/ReviewerGuidelines) and the [AI Policy for Reviewers](https://iclr.cc/Conferences/2027/AIPolicyForReviewers) (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM, Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 MEASURING WHAT SURVIVES THE SUBSTITUTION

The measurement compares two descriptions of the same world: the source-side reference and the facts extracted from a compressed code. Figure 1 follows this comparison through the running example. The checker has a separate role: it tests the code’s grammar and supported internal constraints using only the code.

From text to comparable facts. The reference projection is determined before any compressed code exists. It contains the explicit equations associated with the rendered source, including deliberately redundant statements. Canonicalization combines coefficients, reduces rational values, and normalizes equation order and sign. Thus $z = 2y$ and $2y - z = 0$ identify the same explicit equation. The encoder receives the rendered text; the construction record supplies the reference used for scoring.

Each supported code format has a fixed decoder into this equation language. For trace code, the decoder reads supported GIVEN bindings and affine EQ equalities. Free-text reasoning and other trace fields remain outside the extracted fact set, while unsupported material remains visible in parsing diagnostics. Canonical JSON, compact structured codes, and template prose have their own fixed parsers. This shared output language permits comparisons across formats without requiring a common surface syntax.

Constructed source

x is 2; y is x plus 3; z is 2 times y.

Reference projection

x=2
y=x+3
z=2*y

Apply fixed consequence rules



Reference closure

2*y-z=0; x-y=-3
x-z=-8; x=2
y-z=-5; y=5; z=10

Self-consistent mistranslation

GIVEN:
x := 2
GOAL: z
EQ[e1]: y = x + 4
EQ[e2]: z = 2*y
CHECK[c1]: arith: e2.value == 12
ANS: 12

Code-only checker: accepts

Decode; apply the same rules

Decoded-code closure

2*y-z=0; x-y=-4
x-z=-10; x=2
y-z=-6; y=6; z=12

Source agreement: 2/12 = 0.167

Exact-code fixture: checker accepts; closure agreement 1.000

Figure 1: Can valid code change its source? This constructed fixture changes $x + 3$ to $x + 4$ and passes its code-only checker, but closure agreement falls from the exact-code fixture’s 1.000 to 0.167.

A finite consequence signature. Write $C(S)$ for the set containing explicit canonical equations, uniquely entailed variable values, and uniquely entailed pairwise differences from facts S . Exact rational row reduction tests these consequences. For source facts R and decoded code facts Z , agreement is

$$J(R, Z) = \frac{|C(R) \cap C(Z)|}{|C(R) \cup C(Z)|}.$$

An empty decoded set or an inconsistent side receives zero agreement. The variable domain includes variables on either side, making both missing and added facts visible. Retaining explicit equations also means that equivalent affine bases can receive different scores when their explicit members

fall outside the value and difference families. This finite signature defines the preservation target throughout the paper. Using a set makes repeated copies of the same consequence count once. Using the union in the denominator also distinguishes agreement from recall alone: extra decoded facts can reduce the score even when the reference facts remain recoverable. Within this signature, each distinct equation, value, or difference contributes equally. The resulting score measures overlap in the chosen consequences, rather than the number of source sentences copied or the importance of a fact to a particular downstream question.

What the example measures. The reference in Figure 1 has seven distinct consequences under this rule. They include the values $x = 2$, $y = 5$, and $z = 10$, their pairwise differences, and the relation $2y - z = 0$. An exact code preserves all seven and scores one. The consistent mistranslation preserves $x = 2$ and $2y - z = 0$, but changes the other values and differences. Two shared consequences among twelve in the union give agreement $1/6$. Checking its answer against its own equations accepts the changed world; comparison with the reference identifies the change.

The same example distinguishes useful omission from lost information. One supplied fixture adds the explicit source statement $z = 10$. Omitting that statement from the code preserves agreement one because $x = 2$, $y = x + 3$, and $z = 2y$ still entail it. In another fixture, omitting $z = 2y$ removes the connection to z and leaves only three of the seven reference consequences, giving agreement $3/7$. These are constructed fixtures that explain the instrument; learned-output evidence follows in Section 4.

The comparison also accounts for additions. A further supplied fixture keeps every reference equation and adds $w = 9$, a value absent from the source. The original seven consequences remain, but the decoded signature grows to eleven, giving agreement $7/11$. The added value and its differences with the existing quantities enter the denominator because the variable domain includes both descriptions. Recovering every source fact therefore suffices for full recall but not full agreement.

Content under an output allowance. We evaluate nominal fractions 0.30, 0.45, 0.60, and 0.80 of each world’s nonredundant source rendering. Absolute allowances have floors of 128 bytes or 48 reference tokens. Using the same nonredundant source to set allowances prevents added redundant text from granting extra output space. Byte and token conditions generate separate outputs under their respective allowances. Raw output accounting includes whitespace, fences, explanations, and malformed material. An over-budget code stays in the evaluation denominator and receives zero matched-budget agreement.

For mean agreement $\bar{F}_j$ at ordered allowances r_j , closure-AUC is

$$\text{AUC} = \frac{1}{0.80 - 0.30} \sum_{j=1}^3 (r_{j+1} - r_j) \frac{\bar{F}_j + \bar{F}_{j+1}}{2}.$$

Every budget point contributes to this normalized area; the curve retains the operating points that the aggregate combines. Appendix A specifies world-level aggregation and resampling. Bytes and reference tokens are separate cost units, so we retain both axes when comparing formats. The redundancy analysis makes a further controlled comparison: it adds already entailed values and differences while preserving the reference consequence set. Code sizes are compared only for pairs meeting the registered agreement and checker requirements on both renderings.

3 EXPERIMENTAL DESIGN AND THE ROLES OF THE CONTROLS

The sources describe controlled micro-worlds whose named quantities are linked by exact affine relations: sums, constant multiples, and offsets. An anchored, connected construction determines rational values, expressed through varied sentences and statement order. All renderings of a world stay together in one split. Training uses 24,000 worlds; held-out reporting uses 200 worlds with 800 renderings, and consumer evaluation uses a fixed subset of 60 worlds with 240 renderings. Separate development and selection worlds support implementation choices and checkpoint selection. Appendix A.7 specifies the splits, adapters, selection rule, and available provenance.

The construction supplies both training targets and evaluation references. Every fidelity score compares the decoded equations with that constructed reference. Exact affine equalities provide a

controlled setting for tracing information through compression. The reference language excludes negation, modality, quantifiers, time, and uncertainty.

Learned encoders with fixed decoders. Trace translators use low-rank adapters on Pythia checkpoints from 70M to 2.8B parameters, with three seeds labelled A, B, and C (Biderman et al., 2023). Additional Qwen translators provide a second-family comparison (Yang et al., 2025). The completed grid contains 39 training cells across translators and learned controls. Adapter rank 16 and learning rate 0.0002 were chosen on development data. All cells use an effective batch size of 128 and at most three epochs; checkpoint selection maximizes closure-AUC on the separate selection worlds, choosing the earliest checkpoint in a tie. Thus the size comparison evaluates the codes produced by a common training and selection procedure. The structured controls—canonical JSON, CCL-Core, CCL-Min, and fixed-template prose—use trained Pythia 1.4B encoders and deterministic decoders. CCL-Core and CCL-Min adapt source-grounded formats to this domain (Trukhina & Vashkelis, 2026a). These comparisons evaluate complete encoding systems, including learning errors and representation cost. The structured controls vary the target format; the scale ladder keeps the trace format fixed while varying the translator. The release supports aggregate comparisons but omits raw learned outputs, weights, and the imported construction and decoding implementations, including the adapted CCL serialization (Appendix A.7).

Construction and task controls. The irredundant-core oracle reads the construction directly and serializes its correct core facts. It shows how a complete reference code performs under the same output allowances. LLMingua-2 pruning and SemanticZip-style learned or strong encoders supply task-oriented comparisons (Pan et al., 2024; Trukhina & Vashkelis, 2026b). These arithmetic-task adaptations receive cost and consumer scores; their formats have no supplied deterministic equation decoder for primary closure scoring. Appendix H records the separate unconstrained-prose construction attempt.

Recovering values from representations. Consumers receive the representation and canonical entity identifiers, then return a value map of integers or reduced fractions. The consumers are Qwen3-Next-80B and GPT-OSS-20B, with a registered 256-token response cap (Qwen Team, 2025). A secondary GPT-OSS condition keeps its prompt and changes decoding settings. Each output scores the fraction of requested values recovered exactly, with parsing failures counted as incorrect. The reported score averages these fractions. Source agreement, validity, and recovery thus use separate instruments, with the saved statistical procedures specified in Appendix A.

4 PASSING THE CHECKER LEAVES SOURCE ERRORS UNDETECTED

Learned translators also produce internally valid codes that change their source. Some sampled codes satisfy both their checker and their output allowance while losing source agreement.

Table 1 summarizes the saved diagnostic samples. For each translator, the analysis selects 40 failing output records by a deterministic ordering of item hashes. Failure here means imperfect budgeted agreement, which includes both source disagreement and exceeding the allowance. The table therefore retains the over-budget counts alongside checker acceptance.

Table 1: Checker outcomes among 40 sampled failing output records per translator. Failures include source disagreement and budget violations; the two columns can overlap.

Translator	Lint-valid	Over budget
Pythia 410M	8	7
Pythia 1B	11	0
Pythia 1.4B	12	5
Pythia 2.8B	14	0
Qwen 0.6B	16	0
Qwen 1.7B	15	1

The samples contain 8–16 linter-valid outputs per model. The Pythia 1B, Pythia 2.8B, and Qwen 0.6B samples contain no over-budget records; their accepted failures therefore have imperfect source agreement. These rows establish the source-error case directly. For rows with budget violations, the marginal counts leave their overlap with checker acceptance unresolved.

The diagnostic samples identify failure modes rather than estimate their prevalence across all outputs. Their selection conditions on failure, and an item can contribute output records under different evaluation conditions. Appendix F gives the sampling rule and complete taxonomy.

We next apply that source-conditioned comparison to every output, including failures, to measure retention across budgets.

5 A COMPACT STRUCTURED CONTROL RETAINS MORE CONTENT PER BYTE

The strongest trace translator has lower byte-axis closure-AUC than the strongest structured control: 0.672 against 0.860 for CCL-Min. Figure 2 shows how the aggregate difference arises across the available budgets. The comparison asks how much measured content each complete encoding system delivers within a shared allowance. It includes the translator’s choice of facts, the cost of expressing them, and whether the resulting code fits.

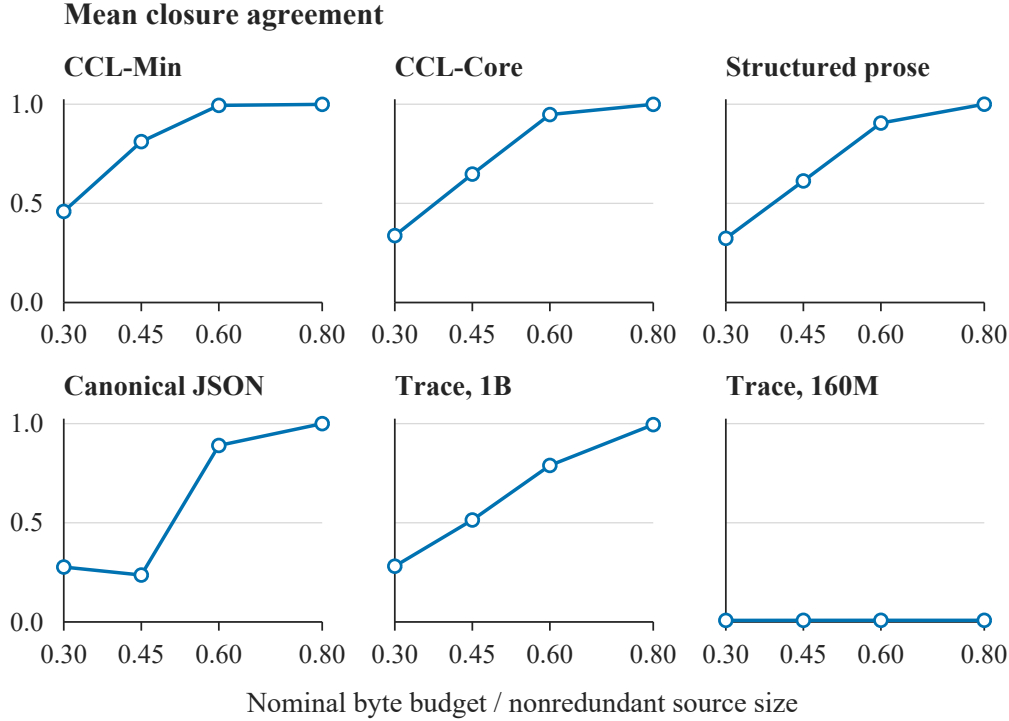


Figure 2: How much closure agreement fits each byte budget? Shared axes show each system’s four saved measurements, including JSON’s initial decline and the low 160M trace scores.

Where the byte advantage appears. The displayed 1B trace system retains less measured content than CCL-Min under the tighter allowances and approaches it at the largest allowance. The displayed trace series is the best aggregate performer; individual budget points can favor other translators.

The saved trace scores rise from 0.281 and 0.514 at the two tighter byte allowances to 0.789 and 0.995 at the larger ones. CCL-Min gives 0.459, 0.811, 0.994, and 0.999 at the same operating points. At allowance 0.60, CCL-Min already approaches complete agreement while the trace system retains less of the measured content.

The comparison also distinguishes structured formats from one another. Canonical JSON reaches byte-axis closure-AUC 0.624, below the best trace result. CCL-Min’s aggregate advantage therefore concerns the strongest compact format, rather than every structured representation.

The saved JSON curve also declines between its first two allowances before rising. Each operating point uses a separate generated output, rather than extending the same code from the preceding point. Accordingly, the curves report observed retention under each allowance; they need not rise monotonically.

Changing the unit changes the comparison. Token accounting changes the ordering among structured controls. Canonical JSON leads that axis at 0.514, compared with 0.426 for CCL-Min; trace translators reach at most 0.369. Table 2 places the structured controls’ two aggregates together.

Table 2: The leading structured control changes with the cost unit. Each column aggregates outputs generated for that budget axis; it is not a recount of one shared set of codes.

Structured control	Byte closure-AUC	Token closure-AUC
Canonical JSON	0.624	0.514
CCL-Core	0.776	0.471
CCL-Min	0.860	0.426

Appendix B reports both axes for all systems. A byte-efficient representation need not lead under the reference tokenizer: the budget unit changes both the generated outputs and the observed ranking.

Correct facts still have a representation cost. The irredundant-core oracle separates exact source access from compact representation. Its byte-axis agreement is zero at 0.30 and perfect from 0.60, when its complete code fits. This code reads the construction record directly and retains its irredundant equations. Even with direct access to correct facts, their chosen serialization must fit the allowance.

These budget comparisons establish an aggregate advantage for the compact structured control. The next question is whether increasing the trace translator’s capacity closes that advantage under the same measurement.

6 LARGER TRANSLATORS REACH A MEASURED PLATEAU

Increasing translator size improves the weakest trace systems but does not close the aggregate gap to CCL-Min. Figure 3 places each saved Pythia seed and the Qwen summaries against the structured reference. The comparison keeps the trace representation and evaluation target fixed while varying the translator within the model-size ladder.

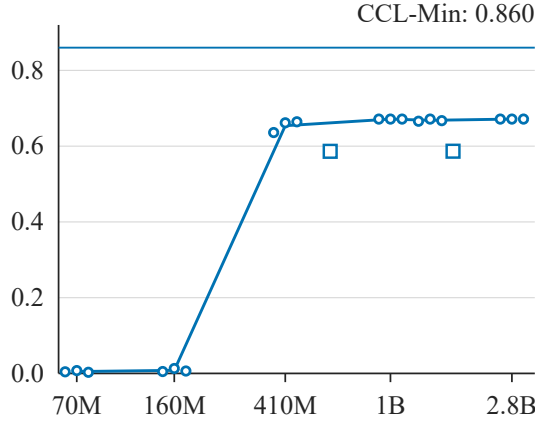
The 70M and 160M systems remain near zero, while the 410M translator reaches 0.654. The 1B and 2.8B models both reach 0.672, and the intervening 1.4B model reaches 0.668. The Qwen summaries are 0.586 and 0.587.

The larger Pythia systems share a plateau in byte-axis closure agreement. Moving beyond the weakest models brings a large change in measured retention; subsequent capacity increases yield a much narrower range of byte-axis scores. The recorded byte-axis curves for 1B and 2.8B coincide, while token-axis and validity measurements differ. The plateau is therefore a result about saved agreement scores under this training setup.

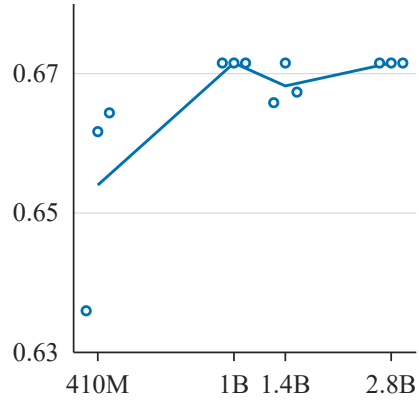
A plateau and an equivalence decision ask different questions. The scale decision combines an AUC equivalence margin of 0.02 with a bound on the difference in checker pass rates. No smaller Pythia model meets that combined criterion. The 1B and 1.4B comparisons satisfy the AUC interval condition, but their checker-rate differences exceed the joint rule’s allowance. For 410M, the AUC interval extends past the equivalence boundary. Appendix D reports both quantities.

Byte closure-AUC

All translators



Larger Pythia: detail



Translator parameters (log scale in each panel)

— Pythia mean □ Qwen3 mean
 ○ Seeds A / B / C: -4 / 0 / +4 pt horizontal offsets

Figure 3: Does scale close the measured byte-axis gap? Scores stay below CCL-Min; the larger-Pythia detail resolves seed variation, with circles showing seeds A/B/C from left to right.

Table 3: Why similar byte-axis AUC does not satisfy the joint scale rule. Differences are from Pythia 2.8B. AUC intervals must lie inside $(-0.02, 0.02)$; checker pass-rate differences must have magnitude at most 0.02.

Pythia model	AUC difference: 95% interval	Checker-rate difference
410M	$[-0.021, -0.015]$	-0.084
1B	same AUC	-0.049
1.4B	$[-0.005, -0.002]$	-0.042

Removing redundant wording is a separate property. The redundancy test reports a zero mean log-size change in each evaluated translator-seed comparison. This aggregate stability is consistent with the specified training target, a canonical representation shared by redundant renderings. The test compares each rendering at its own first qualifying budget cell, where its source agreement and checker result meet the registered requirements. For nonredundant and redundant sources x_0, x_3 with qualifying codes z_0^*, z_3^* , it measures

$$\Delta_{\log B} = \log \frac{B(z_3^*)}{B(z_0^*)}, \quad \Delta_{\log r} = \log \frac{B(z_3^*)/B(x_3)}{B(z_0^*)/B(x_0)}.$$

The first quantity tracks absolute code size; the second tracks size relative to the expanded source. Evaluated comparisons contain 3–27 qualifying world pairs. Systems with too few qualifying pairs have undefined tests, as retained in Appendix C. The result characterizes stability against redundant source wording among qualifying pairs. It can coexist with the remaining gap in agreement across the full budget range, where other outputs lose content or fail to fit.

The budget and scale results measure content through a fixed decoder. The final comparison replaces that decoder-based question with a practical one: can another model use the representation to recover the named values?

7 CONSUMER RECOVERY UNDER FIXED RESPONSE CONDITIONS

Qwen3-Next-80B recovers values more accurately from canonical JSON than from CCL-Min in the consumer experiment. Recovery spans both budget axes on a report subset; the headline closure-AUC uses the byte axis on the full report set. Table 4 compares exact value recovery under the supplied consumer protocol. For each output, the score is the fraction of requested values returned correctly; the table averages these fractions.

Table 4: Exact value recovery by Qwen3-Next-80B. Source and oracle inputs provide reference points; structured and task-oriented encodings show recovery from compressed representations.

Input	Exact recovery
Source text	0.869
Irredundant-core oracle	0.907
Canonical JSON	0.703
Trace, 1B translator	0.586
CCL-Min	0.471
SemanticZip-style adapter	0.463
Strong-encoder SemanticZip-style	0.787
LLMLingua-2	0.169

Canonical JSON and the displayed 1B trace code exceed CCL-Min’s recovery point estimate for this consumer. This is a comparison of recovery under the consumer protocol; relating its ordering to closure agreement would require both measurements on the same codes and worlds.

The consumer’s recovery point estimate is higher for the oracle’s explicit equations than for the rendered source. The source score is thus a reference-input measurement, not an upper bound on recovery. The strong-encoder SemanticZip-style encoding provides the highest displayed recovery among those task-oriented encodings.

Primary GPT-OSS-20B recovery rounds to zero for all but the strong-encoder input, with generation frequently reaching the 256-token cap. The secondary condition keeps the prompt and token cap fixed and changes the reasoning-effort setting. It recovers some values across systems, showing that the consumer measurement also depends on decoding conditions. Appendix E gives every condition and its recorded stop reasons.

These condition-specific point estimates describe what each consumer recovers; Appendix A documents the separate pooled comparison procedure.

8 RELATED WORK

Compression objectives. Prompt compression reduces the material presented to a language model. Selective Context and the LLMLingua family remove tokens, with objectives ranging from information selection to question relevance and task-agnostic retention (Li et al., 2023; Jiang et al., 2023; 2024; Pan et al., 2024). RECOMP learns summaries of retrieved passages; gist tokens and AutoCompressors learn continuous representations for subsequent model use (Xu et al., 2024; Mu et al., 2023; Chevalier et al., 2023). Telegraph English uses compact symbolic fact lines, and related experiments compare re-expression with summaries and token-matched controls (Arbuzov et al., 2026; Bei et al., 2026). These approaches motivate measuring both representation cost and consumer behavior. Our fixed decoder supplies an additional comparison against specified source consequences.

Preservation beyond task success. Context Codec evaluates typed facts and their source support, while SemanticZip distinguishes protected from lossy content and measures recovery by another model (Trukhina & Vashkelis, 2026a;b). The controls here adapt these ideas to affine sources. Closure-based rate–distortion theory supplies the precedent for comparing consequence sets, including deductive sources and reversible logging (Xu, 2026c;a;b). We instantiate that pattern with a finite equation signature and learned text-to-code translators. Semantic communication also studies meaning

preservation under communication constraints, using learned encoders and decoders (Xie et al., 2021). Here exact decoding makes the scored fact sets inspectable.

Faithfulness and executable meaning. Summarization studies identify unsupported generated content; FactCC predicts source-conditioned consistency, and FActScore evaluates support at the level of atomic facts (Maynez et al., 2020; Kryściński et al., 2020; Min et al., 2023). The constructed arithmetic reference permits an exact comparison of both missing and added consequences. Semantic parsing links language to executable representations, while constrained decoding enforces structural requirements (Liang et al., 2013; Scholak et al., 2021). Our separate checker and reference comparison distinguish these structural properties from preservation of the source.

Learned representations and scale. CommNet, DIAL, referential games, and GLC study messages through the behavior they enable, including connections to interpretable symbols (Sukhbaatar et al., 2016; Foerster et al., 2016; Lazaridou et al., 2017; Du et al., 2026). Consumer recovery retains that behavioral question alongside source agreement. Language-model scaling studies and the Pythia family motivate systematic size comparisons (Kaplan et al., 2020; Hoffmann et al., 2022; Biderman et al., 2023). Our size ladder evaluates the resulting codes under fixed preservation rules and output allowances, connecting translator capacity with the content retained by a complete encoding system.

9 DISCUSSION

The learned-output errors and budget curves identify two requirements for a useful substitute: preserving the specified source consequences and expressing them within the output allowance.

Choose what must survive. The reference projection turns preservation into an explicit set comparison. The fixtures distinguish dropping a redundant statement from losing a relation, and a self-consistent mistranslation exposes the information missing from a code-only check. Learned outputs exhibit the same separation between internal acceptance and source agreement. For a pipeline that replaces its source with code, the preservation target therefore belongs in the evaluation design alongside the code grammar.

Choose the cost that matters. The byte and token results lead to different choices among structured controls. CCL-Min leads aggregate agreement per byte, while JSON leads under token accounting. These are properties of complete encoding systems: a trained translator must recover the source facts and express them within the allowance. The oracle makes the latter requirement visible even with direct access to correct facts. An encoding comparison becomes useful when its budget matches the resource the intended pipeline must conserve.

Evaluate the substitute where it will be used. Translator size, checker acceptance, and consumer recovery provide complementary evidence about a code. The Pythia byte-axis plateau coexists with differences in checker pass rates, so replacing a translator on the basis of one score can change another property. The consumer experiment adds the model that must interpret the representation and the decoding procedure that produces its answer. Its descriptive ordering supplies a separate view of the compressed inputs, with the evaluation population and response conditions stated alongside the scores.

The exact arithmetic domain makes this separation inspectable. Extending it to open text requires a source representation and consequence family appropriate to that domain; the finite signature already makes those choices part of the present score. The source-to-code comparison identifies changed content; the consumer test measures what a particular reader can recover from the resulting representation.

REPRODUCIBILITY STATEMENT

The materials include measurement contracts, score tables, display scripts and data, configuration and checkpoint hashes, and the analysis implementation. Seeds are labelled A, B, and C in the paper; their integer values remain in the supplied configuration. Timestamps are withheld from the anonymous version and restored at camera-ready; the artifact chain fixes the order. Appendix A.7 lists the additional artifacts required for complete replay.

ETHICS STATEMENT

The study uses constructed arithmetic sources and a restricted exact-value consumer task. Its central safety implication is that a code’s internal acceptance should remain distinct from evidence of faithful translation.

AI USE STATEMENT

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

REFERENCES

- Mikhail L. Arbutov, Sisong Bei, Ziwei Dong, Dmitri Kalaev, and Alexey A. Shvets. Telegraph English: Semantic prompt compression via structured symbolic rewriting, 2026. URL <https://arxiv.org/abs/2605.04426v1>. Version 1.
- Sisong Bei, Mikhail L. Arbutov, Ziwei Dong, Dmitri Kalaev, and Alexey Shvets. Context compression is not one thing: Readable symbolic re-expression vs. coherent summary at matched budget, 2026. URL <https://arxiv.org/abs/2606.14875v1>. Version 1.
- Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usven Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2397–2430. PMLR, 2023. URL <https://proceedings.mlr.press/v202/biderman23a.html>.
- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.232. URL <https://aclanthology.org/2023.emnlp-main.232/>.
- Wei Du, Benyu Wu, Yuqing Sun, Wei Guo, Yuntao Du, Zhongmin Yan, Guoxian Yu, and Lizhen Cui. Learning efficient and interpretable multi-agent communication. In *International Conference on Learning Representations*, 2026. URL <https://openreview.net/forum?id=a3CUE06G5Y>.
- B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi: 10.1214/aos/1176344552. URL <https://www.jstor.org/stable/2958830>.
- Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/c7635bfd99248a2cdef8249ef7bfbe4-Paper.pdf.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom

540 Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy,
541 Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre.
542 Training compute-optimal large language models, 2022. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2203.15556)
543 2203.15556.

544 Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of*
545 *Statistics*, 6(2):65–70, 1979. URL <https://www.jstor.org/stable/4615733>.

547 Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compress-
548 ing prompts for accelerated inference of large language models. In *Proceedings of the 2023*
549 *Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376. Asso-
550 ciation for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.825. URL
551 <https://aclanthology.org/2023.emnlp-main.825/>.

552 Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili
553 Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt
554 compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational*
555 *Linguistics (Volume 1: Long Papers)*, pp. 1658–1677. Association for Computational Linguistics,
556 2024. doi: 10.18653/v1/2024.acl-long.91. URL [https://aclanthology.org/2024.](https://aclanthology.org/2024.acl-long.91/)
557 [acl-long.91/](https://aclanthology.org/2024.acl-long.91/).

558 Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child,
559 Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models,
560 2020. URL <https://arxiv.org/abs/2001.08361>.

561 Wojciech Kryściński, Bryan McCann, Caiming Xiong, and Richard Socher. Evaluating the fac-
562 tual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference*
563 *on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9332–9346. Associ-
564 ation for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.750. URL
565 <https://aclanthology.org/2020.emnlp-main.750/>.

566 Daniël Lakens. Equivalence tests: A practical primer for t tests, correlations, and meta-analyses. *So-*
567 *cial Psychological and Personality Science*, 8(4):355–362, 2017. doi: 10.1177/1948550617697177.
568 URL <https://journals.sagepub.com/doi/10.1177/1948550617697177>.

569 Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the
570 emergence of (natural) language. In *International Conference on Learning Representations*, 2017.
571 URL <https://arxiv.org/abs/1612.07182v2>.

572 Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference
573 efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods*
574 *in Natural Language Processing*, pp. 6342–6353. Association for Computational Linguistics,
575 2023. doi: 10.18653/v1/2023.emnlp-main.391. URL [https://aclanthology.org/2023.](https://aclanthology.org/2023.emnlp-main.391/)
576 [emnlp-main.391/](https://aclanthology.org/2023.emnlp-main.391/).

577 Percy Liang, Michael I. Jordan, and Dan Klein. Learning dependency-based compositional semantics.
578 *Computational Linguistics*, 39(2):389–446, 2013. doi: 10.1162/COLI_a_00127. URL <https://aclanthology.org/J13-2005/>.

579 Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality
580 in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for*
581 *Computational Linguistics*, pp. 1906–1919. Association for Computational Linguistics, 2020. doi:
582 10.18653/v1/2020.acl-main.173. URL [https://aclanthology.org/2020.acl-main.](https://aclanthology.org/2020.acl-main.173/)
583 [173/](https://aclanthology.org/2020.acl-main.173/).

584 Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer,
585 Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual
586 precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical*
587 *Methods in Natural Language Processing*, pp. 12076–12100. Association for Computational
588 Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.741. URL [https://aclanthology.](https://aclanthology.org/2023.emnlp-main.741/)
589 [org/2023.emnlp-main.741/](https://aclanthology.org/2023.emnlp-main.741/).

- Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0848. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/3d77c6dcc7f143aa2154e7f4d5e22d68-Abstract.html.
- Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606, 2018. doi: 10.1073/pnas.1708274114. URL <https://www.pnas.org/doi/10.1073/pnas.1708274114>.
- 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. LLMingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.57. URL <https://aclanthology.org/2024.findings-acl.57/>.
- Qwen Team. Qwen3-Next-80B-A3B-Instruct. Model card, 2025. URL <https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Instruct>.
- Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9895–9901. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.779. URL <https://aclanthology.org/2021.emnlp-main.779/>.
- Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://papers.nips.cc/paper_files/paper/2016/hash/55b1927fdafef39c48e5b73b5d61ea60-Abstract.html.
- Natalia Trukhina and Vadim Vashkelis. Compress the context, keep the commitments: A formal framework for verifiable LLM context compression, 2026a. URL <https://arxiv.org/abs/2605.17304v1>. Version 1.
- Natalia Trukhina and Vadim Vashkelis. SemanticZip: A pilot framework for lossy text compression with LLMs as semantic decompressors, 2026b. URL <https://arxiv.org/abs/2605.24541v1>. Version 1.
- Huiqiang Xie, Zhijin Qin, Geoffrey Ye Li, and Biing-Hwang Juang. Deep learning enabled semantic communication systems. *IEEE Transactions on Signal Processing*, 2021. doi: 10.1109/TSP.2021.3071210. URL <https://arxiv.org/abs/2006.10685v3>.
- Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/bda88ed2892f5e61c9a9bf215c566913-Paper-Conference.pdf.
- Jianfeng Xu. Rate-distortion theory for deductive sources under closure fidelity, 2026a. URL <https://arxiv.org/abs/2604.15698v4>. Version 4.
- Jianfeng Xu. Closure-preserving rate-distortion for reversible logging, 2026b. URL <https://arxiv.org/abs/2606.16592v2>. Version 2.
- Jianfeng Xu. Semantic rate-distortion theory: Deductive compression and closure fidelity, 2026c. URL <https://arxiv.org/abs/2604.11204v1>. Version 1.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang

648 Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger
649 Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan
650 Qiu. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701

A STATISTICAL PROCEDURE AND IMPLEMENTATION

A.1 REFERENCE FACTS AND FINITE CLOSURE

Canonical equations combine repeated variables, sort identifiers, clear denominators, divide by the positive greatest common divisor, and orient the first nonzero coefficient positively. Exact rational row reduction tests entailment over the union of source and decoded variables.

For a consistent equation set S , the scored set contains its explicit canonical equations, all uniquely entailed entity values, and all uniquely entailed pairwise differences. Explicit affine equations remain members even when they fall outside the value and difference templates. Consequently, logically equivalent affine bases can have different scored sets.

Redundant source additions are restricted to entailed values and pairwise differences, keeping the scored reference set fixed across the four source renderings. An inconsistent side, or an empty decoded set, receives zero agreement. Unsupported text remains visible in parsing diagnostics.

A.2 RATES AND AGGREGATION

Let B count UTF-8 bytes and T count reference tokens. Raw outputs retain whitespace, fences, explanations, and malformed material. The reference tokenizer is Qwen/Qwen3.8-27B, revision 1d4bf0f2ff6012fd82039f2fa52739d0dd7c60c0, with special tokens disabled. For the rendering x_0 without added redundancy, the absolute allowances are

$$b_B(r) = \max\{128, \lfloor rB(x_0) \rfloor\}, \quad b_T(r) = \max\{48, \lfloor rT(x_0) \rfloor\}, \quad (1)$$

$$r \in \{0.30, 0.45, 0.60, 0.80\}. \quad (2)$$

Realized rates use the corresponding source rendering in the denominator. An over-budget output receives zero matched-budget agreement while its raw output and diagnostics remain recorded. Each budget cell contributes an item-level agreement score.

For mean scores $\bar{F}_j$ at ordered budgets r_j , normalized closure-AUC is

$$\text{AUC} = \frac{1}{0.80 - 0.30} \sum_{j=1}^3 (r_{j+1} - r_j) \frac{\bar{F}_j + \bar{F}_{j+1}}{2}. \quad (3)$$

The analysis requires every budget point. Scale summaries retain separate seed results and their equal-weight mean.

A.3 WORLD RESAMPLING

The resampling unit is the latent world. All associated renderings, budgets, systems, and seeds move together. The percentile interval uses the ordered replicate positions at 0.025 and 0.975, with linear interpolation between neighboring positions (Efron, 1979).

Scale contrasts and the designated Pythia 1.4B redundancy analysis use 10,000 resamples. The all-model redundancy summaries and baseline contrasts use 1,000 resamples. The numerical bootstrap seed is retained in the configuration.

Scale and redundancy computations retain the multiplicity of sampled worlds. Baseline contrasts first collapse records by item identifier, so repeated selections of a world contribute once per replicate. These baseline intervals therefore use the implemented deduplicated-world procedure.

A.4 REDUNDANCY COMPARISON

The first passing budget requires closure precision and recall of at least 0.95 and a full-linter pass. A rendering without a passing cell receives ordinal code 5. Size and rate comparisons use only worlds

passing in both the R0 and R3 renderings. Each rendering uses its own first passing cell:

$$\Delta_{\log B} = \log \frac{B(z_3^*)}{B(z_0^*)}, \quad (4)$$

$$\Delta_{\log r} = \log \frac{B(z_3^*)/B(x_3)}{B(z_0^*)/B(x_0)}, \quad (5)$$

$$\Delta_{\text{pass}} = \Pr(\text{pass at } 0.80 \mid R3) - \Pr(\text{pass at } 0.80 \mid R0). \quad (6)$$

The rule requires the upper interval bound for $\Delta_{\log B}$ to be at most $\log(1.05)$ and the upper bound for $\Delta_{\log r}$ to be negative. It also requires the lower bound for Δ_{pass} to be at least -0.02 . Maximum-budget success is evaluated specifically at 0.80, independently of any earlier success.

A.5 SCALE COMPARISON

The reference is Pythia 2.8B. A smaller model qualifies only when its paired AUC interval lies strictly inside $(-0.02, 0.02)$ and its full-linter pass-rate difference has magnitude at most 0.02. An adjacent larger-minus-smaller contrast with upper interval bound below -0.02 prevents a scale threshold. An equivalence margin expresses a chosen tolerance for differences (Lakens, 2017). The Qwen models remain separate diagnostic points.

The reported scale analysis consists of seed means, paired reference contrasts, and adjacent contrasts. The proposed mixed-model slope, secondary token-scale contrasts, and redundancy sensitivity regression have no corresponding outputs in the analysis record.

A.6 BASELINE AND CONSUMER COMPARISONS

The comparison adapter is Pythia 1.4B, as specified before evaluation. Pythia 1B supplies the separate best-byte-AUC descriptive comparison. Each baseline contrast averages paired item differences after averaging repeated records for the same item.

The implemented task contrast first averages every stored consumer score per output, including the secondary decode control. The consumer tables retain the three conditions separately. Thus the pooled contrast and each named consumer’s recovery score are different summaries.

The implementation assigns each baseline an indicator of 1.0 if any paired interval includes zero, and 0.01 otherwise. Holm adjustment operates across baselines within each metric family, pooling the byte and token cells for that indicator (Holm, 1979). These inputs are interval-derived indicators, rather than calibrated hypothesis-test p -values.

The recorded dominance flag requires nonnegative lower bounds in every cell, positive lower bounds in at least two cells, and rejection after the indicator adjustment. The archived analysis retains these flags; they are not used as hypothesis-test evidence in this paper. The original protocol instead specified separate axes and consumers, with adjustment across operating points.

A.7 DATA AND TRAINING SETTINGS

Training uses 24,000 worlds; development and selection each use 80 worlds and 320 renderings. The held-out report set contains 200 worlds and 800 renderings. The consumer subset contains the 60 smallest-hash report worlds and all 240 renderings. Every rendering of a world remains in its assigned split.

LoRA rank 16 and learning rate 0.0002 were selected on development data before checkpoint selection. Training uses alpha twice the rank, dropout 0.05, bfloat16, effective batch size 128, and at most three epochs. AdamW settings are $(\beta_1, \beta_2) = (0.9, 0.95)$, epsilon 10^{-8} , and weight decay 0.1. The schedule is cosine, with warmup ratio 0.03 and gradient clipping 1.0.

Attention and MLP projections receive adapters. Checkpoint selection maximizes selection closure-AUC, choosing the earliest checkpoint within 10^{-6} ties. Training seeds are A, B, and C; the release configuration maps these labels to integers.

The checkpoint record includes matched Pythia 1.4B adapters for JSON, CCL-Core, CCL-Min, structured prose, and SemanticZip-style packets. The structured formats have deterministic decoders. The aggregate tables identify the output format of each control.

Each translator produces one greedy completion with temperature zero, repetition penalty one, and EOS enabled. The generation allowance adds 32 tokens for byte-controlled cells and 8 for token-controlled cells. All returned tokens remain in rate accounting.

Available artifacts comprise aggregate score tables, the analysis driver, and configuration and checkpoint hashes. Raw scorecards, learned output texts, fitted weights, and the imported estimator, decoder, target-builder, and construction implementations are absent from this record. The exact adapted CCL serialization is therefore unavailable for independent inspection. Aggregate results can be inspected, while complete row-to-checkpoint attribution and raw-output equality remain unverified.

B RATE-DISTORTION RESULTS

The complete scored-format results retain both budget axes. The source, pruning, and packet controls receive consumer scores because they lack deterministic equation decoders.

Table 5: Mean matched-budget closure agreement and normalized AUC on the bytes axis.

System	0.30	0.45	0.60	0.80	AUC
TE, 1.4B	0.276	0.509	0.788	0.992	0.668
TE, 160M	0.008	0.008	0.008	0.008	0.008
TE, 1B	0.281	0.514	0.789	0.995	0.672
TE, 2.8B	0.281	0.514	0.789	0.995	0.672
TE, 410M	0.276	0.510	0.771	0.949	0.654
TE, 70M	0.005	0.005	0.005	0.005	0.005
TE, Qwen3 0.6B	0.233	0.417	0.648	0.999	0.586
TE, Qwen3 1.7B	0.233	0.417	0.648	1.000	0.587
Canonical JSON	0.277	0.236	0.890	1.000	0.624
CCL-Core	0.337	0.648	0.948	0.999	0.776
CCL-Min	0.459	0.811	0.994	0.999	0.860
Oracle core	0.000	0.725	1.000	1.000	0.768
Structured prose	0.324	0.613	0.905	1.000	0.749

Table 6: Mean matched-budget closure agreement and normalized AUC on the tokens axis.

System	0.30	0.45	0.60	0.80	AUC
TE, 1.4B	0.068	0.146	0.269	0.539	0.256
TE, 160M	0.001	0.003	0.005	0.007	0.004
TE, 1B	0.090	0.207	0.383	0.775	0.365
TE, 2.8B	0.090	0.210	0.393	0.774	0.369
TE, 410M	0.071	0.142	0.270	0.566	0.261
TE, 70M	0.002	0.003	0.004	0.005	0.004
TE, Qwen3 0.6B	0.088	0.201	0.386	0.772	0.363
TE, Qwen3 1.7B	0.090	0.202	0.384	0.769	0.362
Canonical JSON	0.150	0.325	0.562	0.988	0.514
CCL-Core	0.132	0.290	0.524	0.902	0.471

Continued from previous page

System	0.30	0.45	0.60	0.80	AUC
CCL-Min	0.113	0.230	0.403	0.994	0.426
Oracle core	0.000	0.000	0.005	1.000	0.202
Structured prose	0.143	0.279	0.453	0.781	0.420

C REDUNDANCY RESULTS

The table reports log ratios of output size and realized rate, followed by the maximum-budget success difference. Undefined rows remain visible. Paired counts refer to worlds passing in both renderings; success differences use all 200 worlds.

Table 7: Redundancy comparison by Pythia model and seed. Undefined means fewer than two world pairs passed both renderings.

Model	Seed	Paired	Status	$\Delta_{\log B}$	$\Delta_{\log r}$	Δ_{pass}	Rule
1.4B	A	18	evaluated	0.0	-1.229	0.530	passes
1.4B	B	17	evaluated	0.0	-1.228	0.520	passes
1.4B	C	21	evaluated	0.0	-1.228	0.520	passes
160M	A	0	undefined	n/a	n/a	0.000	n/a
160M	B	0	undefined	n/a	n/a	0.000	n/a
160M	C	0	undefined	n/a	n/a	0.000	n/a
1B	A	16	evaluated	0.0	-1.230	0.490	passes
1B	B	16	evaluated	0.0	-1.229	0.470	passes
1B	C	17	evaluated	0.0	-1.227	0.490	passes
2.8B	A	27	evaluated	0.0	-1.231	0.565	passes
2.8B	B	25	evaluated	0.0	-1.228	0.570	passes
2.8B	C	26	evaluated	0.0	-1.229	0.555	passes
410M	A	1	undefined	n/a	n/a	0.310	n/a
410M	B	6	evaluated	0.0	-1.224	0.490	passes
410M	C	3	evaluated	0.0	-1.241	0.505	passes
70M	A	0	undefined	n/a	n/a	0.000	n/a
70M	B	0	undefined	n/a	n/a	0.000	n/a
70M	C	0	undefined	n/a	n/a	0.000	n/a

The zero output-size log ratios describe the measured size statistic. Equality of output lengths alone leaves serialized content unspecified.

D TRANSLATOR SCALE

The scale table preserves the individual seed estimates and paired intervals. Pythia 1B and 2.8B have equal byte-axis AUC estimates; their consumer, token-axis, and linter results remain separate measurements.

Table 8: Byte-axis closure-AUC by seed and paired difference from Pythia 2.8B.

Model	A	B	C	Mean	Δ to 2.8B	95% interval
70M	0.004	0.007	0.003	0.005	-0.667	[-0.678, -0.655]
160M	0.005	0.013	0.006	0.008	-0.663	[-0.675, -0.652]

Continued from previous page

Model	A	B	C	Mean	Δ to 2.8B	95% interval
410M	0.636	0.662	0.664	0.654	-0.018	[-0.021, -0.015]
1B	0.672	0.672	0.672	0.672	same AUC	same AUC
1.4B	0.666	0.672	0.667	0.668	-0.003	[-0.005, -0.002]
2.8B	0.672	0.672	0.672	0.672	reference	n/a
Qwen3 0.6B	0.587	0.586	0.586	0.586	sentinel	n/a
Qwen3 1.7B	0.587	0.587	0.587	0.587	sentinel	n/a

Table 9: Paired AUC differences for adjacent Pythia sizes, larger minus smaller.

Adjacent models	Difference	95% interval
160m minus 70m	0.003	[0.003, 0.004]
410m minus 160m	0.646	[0.635, 0.656]
1b minus 410m	0.018	[0.015, 0.021]
1.4b minus 1b	-0.003	[-0.005, -0.002]
2.8b minus 1.4b	0.003	[0.002, 0.005]

No model meets the combined AUC and linter equivalence rule. The 410M AUC interval extends past the negative margin. The 1B and 1.4B linter-rate differences exceed the permitted magnitude.

Table 10: Full-linter pass-rate differences from the 2.8B reference on the byte axis.

Pythia model	Linter pass-rate difference
70m	-0.029
160m	0.029
410m	-0.084
1b	-0.049
1.4b	-0.042

E CONSUMER RECOVERY AND STOP REASONS

Each consumer receives a code and all canonical entity identifiers, then returns an exact JSON value map. Parser failure scores every entity wrong. The primary output cap is 256 tokens. The secondary GPT-OSS condition changes reasoning effort to low, retaining the prompt and output cap.

Recovery is the mean of recorded per-output exact-value fractions across budgets, axes, renderings, and available seeds. The counts include repeated renderings and budget cells, rather than independent worlds. End-turn and token-limit counts expose completion behavior directly.

Table 11: Qwen3 Next 80B A3B: exact recovery, parser acceptance, and stop counts.

System	Recovery	Parser	Calls	End turn	At cap
TE, 1.4B	0.555	0.952	5760	5760	0
TE, 160M	0.036	0.922	5760	5760	0
TE, 1B	0.586	0.964	5760	5760	0
TE, 2.8B	0.594	0.965	5760	5760	0

Continued from previous page

System	Recovery	Parser	Calls	End turn	At cap
TE, 410M	0.546	0.952	5760	5760	0
TE, 70M	0.025	0.947	5760	5760	0
TE, Qwen3 0.6B	0.555	0.936	5760	5760	0
TE, Qwen3 1.7B	0.563	0.935	5760	5760	0
Canonical JSON	0.703	0.981	5760	5760	0
CCL-Core	0.639	0.953	5760	5760	0
CCL-Min	0.471	0.966	5760	5760	0
LLMLingua-2	0.169	0.995	1920	1920	0
Oracle core	0.907	1.000	1920	1920	0
SemanticZip-style	0.463	0.968	5760	5760	0
Source text	0.869	0.993	1920	1920	0
Strong SemanticZip-style	0.787	1.000	1920	1920	0
Structured prose	0.650	0.952	5760	5760	0

Table 12: GPT-OSS 20B, primary: exact recovery, parser acceptance, and stop counts.

System	Recovery	Parser	Calls	End turn	At cap
TE, 1.4B	0.000	0.000	5760	1	5759
TE, 160M	0.000	0.001	5760	4	5756
TE, 1B	0.000	0.001	5760	3	5757
TE, 2.8B	0.000	0.001	5760	3	5757
TE, 410M	0.000	0.000	5760	2	5758
TE, 70M	0.000	0.000	5760	0	5760
TE, Qwen3 0.6B	0.000	0.000	5760	0	5760
TE, Qwen3 1.7B	0.000	0.000	5760	0	5760
Canonical JSON	0.000	0.000	5760	0	5760
CCL-Core	0.000	0.000	5760	0	5760
CCL-Min	0.000	0.000	5760	0	5760
LLMLingua-2	0.000	0.000	1920	0	1920
Oracle core	0.000	0.000	1920	0	1920
SemanticZip-style	0.000	0.000	5760	0	5760
Source text	0.000	0.000	1920	0	1920
Strong SemanticZip-style	0.449	0.576	1920	1120	800
Structured prose	0.000	0.000	5760	0	5760

Table 13: GPT-OSS 20B, secondary decode control: exact recovery, parser acceptance, and stop counts.

System	Recovery	Parser	Calls	End turn	At cap
TE, 1.4B	0.097	0.126	5760	2269	3491
TE, 160M	0.005	0.149	5760	2323	3437
TE, 1B	0.092	0.116	5760	2009	3751
TE, 2.8B	0.109	0.135	5760	2300	3460
TE, 410M	0.072	0.094	5760	1951	3809
TE, 70M	0.004	0.194	5760	2348	3412

Continued from previous page

System	Recovery	Parser	Calls	End turn	At cap
TE, Qwen3 0.6B	0.083	0.106	5760	2333	3427
TE, Qwen3 1.7B	0.089	0.116	5760	2503	3257
Canonical JSON	0.125	0.159	5760	1862	3898
CCL-Core	0.095	0.143	5760	1679	4081
CCL-Min	0.043	0.056	5760	651	5109
LLMLingua-2	0.008	0.156	1920	334	1586
Oracle core	0.206	0.235	1920	452	1468
SemanticZip-style	0.021	0.032	5760	566	5194
Source text	0.072	0.073	1920	141	1779
Strong SemanticZip-style	0.743	0.944	1920	1865	55
Structured prose	0.134	0.196	5760	1975	3785

The primary GPT-OSS condition has extensive truncation and an exception for the strong encoder. The secondary condition also retains token-limit stops. These completion patterns accompany the recovery scores in each condition. Values displayed as 0.000 retain the supplied three-decimal precision and can include nonzero recovery.

F ERROR CENSUS

The census selects up to 40 error records per series, sorting by the hash of the item identifier. A single item can occur in multiple records through budgets or seeds. An error is a matched-budget closure score below one. Omission and unsupported-fact labels use the penalized explicit precision and recall fields. Budget violations can therefore trigger these labels without identifying the underlying translation error. The linter-valid error label also uses this matched-budget score, so it includes accepted outputs penalized for exceeding their budget. For samples containing budget overruns, marginal counts leave their overlap with accepted outputs unresolved.

Five categories require manual inspection and remain unassessed: entity substitution, sign or direction, coefficient or offset, redundancy copying, and unsupported TE fragments. Task-only systems have no closure census.

Table 14: Basic score-derived counts. Labels are multi-label counts over the selected records.

System	Errors	Sampled	Omission	Unsupported fact	Over budget
TE, 1.4B	16253	40	40	14	5
TE, 160M	19200	40	40	40	10
TE, 1B	16224	40	40	9	0
TE, 2.8B	16224	40	40	9	0
TE, 410M	16793	40	40	16	7
TE, 70M	19200	40	40	40	7
TE, Qwen3 0.6B	16814	40	40	3	0
TE, Qwen3 1.7B	16806	40	40	4	1
Canonical JSON	12243	40	40	8	3
CCL-Core	13064	40	40	0	0
CCL-Min	11270	40	40	19	19
Oracle core	3416	40	40	40	39
Structured prose	15411	40	40	2	2

Table 15: Parsing and validity counts. Labels are multi-label counts over the selected records.

System	Malformed	Inconsistent	Invalid	Valid error	Decode fail
TE, 1.4B	0	0	28	12	0
TE, 160M	0	0	35	5	0
TE, 1B	0	0	29	11	0
TE, 2.8B	0	0	26	14	0
TE, 410M	0	0	32	8	0
TE, 70M	2	2	36	4	0
TE, Qwen3 0.6B	0	0	24	16	0
TE, Qwen3 1.7B	0	0	25	15	0
Canonical JSON	5	0	0	0	0
CCL-Core	0	0	0	0	0
CCL-Min	0	0	0	0	0
Oracle core	0	0	0	0	12
Structured prose	0	0	0	0	0

G MACHINE CHECKS OF THE CONSTRUCTED REFERENCE

The machine checks compare expected fixture results and serialization stability.

Table 16: Machine validation of the reference comparison.

Criterion	Evidence	Status
Machine fixtures	11/11	PASS
Canonical serialization byte-stable (two independent runs of the pilot)	identical	PASS
Identical closures across the four redundancy variants (six pilot worlds)	6/6 worlds	PASS
Lint-er-valid mistranslation fails the closure comparator	closure agreement 0.167	PASS
Omission and hallucination move the source-conditioned metrics	fixtures 3, 4, 6	PASS

H ADDITIONAL CONTROL CONSTRUCTION

Teacher supervision required a budget-compliant Nova Pro summary whose exact values both consumers recovered. Only 282 of 24,000 assignments passed, leaving too little supervision to train the matched unconstrained-prose control; the three planned cells were removed before grid training. Failures included arithmetic errors, budget overruns, consumer recovery errors and truncation at the 256-token cap, identifier leaks, and transport failures. The frozen records retain the construction results and a separate 24-context natural-text task evaluation, which supplies no closure measurement for the affine comparison.

I PROTOCOL DECISIONS AFFECTING THE COMPARISONS

The configuration was fixed using development data before selection or held-out reporting (Nosek et al., 2018). Rank 16 and learning rate 0.0002 were selected in the development sweep. The accelerator choices and removal of the unsuccessful prose control preceded grid training. The secondary GPT-OSS decoding condition was adopted after checkpoint selection and before the report set was opened. The original chronological record and all analysis outputs remain in the reproducibility archive.

1134 J REUSED INFRASTRUCTURE

1136 The TE grammar, structural checker, full checker, and training-harness patterns are reused infrastruc-
1137 ture. This study adds source-conditioned closure agreement, redundancy response, and translator-
1138 scale measurements; related benchmark and consumer-adaptation results contribute no observations
1139 to these tables. The independent closure comparator uses decoded equations and exact rational
1140 arithmetic to compare source content, including in the linter-valid mistranslation fixture.

1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187