KG-SFT Contracts: Auditing Evidence Lineage and Loader Retention in Enterprise RAG Adaptation

Bogdan Roshchupkin[1] rosupkinbogdan6@gmail.com
Alexander Litvintsev[1] ailitvintsev@gmail.com
Alina Avanesyan[2] aavanesyan@hse.ru
Maria Khodorchenko[3] mkhodorchenko@itmo.ru
Andrei Zhukov[1] breballov@gmail.com
[1]Sber [2]Higher School of Economics, Moscow, Russia [3]ITMO University

Abstract

KG-assisted supervised fine-tuning depends on evidence selection, context compilation, and data loading. We present KG-SFT Contracts, a provenance-aware interface that audits these stages and links model comparisons to their data and scoring records. In a retrospective Russian enterprise case, a pool of 484 graph candidates contributed no graph-origin distractors to the compiled corpus, and exact loader replay discarded 40 of 946 intended examples before splitting. Removing annotated non-support context restored every row while preserving questions, targets, and verified support. Downstream comparisons remained statistically inconclusive. Controlled checks on a public graph-bearing fixture show how cross-record validation detects unresolved graph references and conflicting source revisions alongside role-aware repair. The installable implementation includes fixture-level CPU measurements, incremental adoption guidance, and an official RuBQ schema-import check. The case illustrates how lineage and retention audits make adaptation pipelines inspectable before model effects are attributed to their configured inputs.

1 Introduction

RAG adaptation trains with relevant and distracting documents (Zhang et al., 2024; Kachuee et al., 2025), while KG-guided SFT extracts subgraphs or builds graphs for synthetic supervision (Chen et al., 2025a,b). Yet enabling a graph-candidate channel does not show that graph-derived chunks entered the corpus, compiled rows survived the actual loader, or a later metric difference came from that channel. We audit this chain when graph records supply context candidates during *offline* SFT dataset construction.

Consider the synthetic Service Atlas: its v2 runbook requires OAuth2, while a graph-linked v1 runbook specifies an API key. The older passage is a useful hard negative only if the row preserves v2 support, labels v1 as a distractor, and survives loading. Lost provenance prevents source attribution; a dropped row removes the intended lesson.

Three silent failures separate configuration from realized supervision: **(F1)** no graph candidate is selected; **(F2)** rows disappear before splitting; and **(F3)** shared support documents break the independent-item assumption. The pilot exposes each: zero graph-origin selections from 484 candidates, 40 of 946 rows dropped, and 151 questions grouped into 22 support-document components. Confidentiality motivates machine-readable lineage, counts, hashes, and validation outcomes. We ask:

RQ1. Which evidence enters compiled examples, and which intended examples survive the actual loading path without losing protected content?

RQ2. What assurance do cross-boundary checks add to simple preprocessing, and which model conclusions remain defensible after accounting for score origin and shared support documents?

We contribute: (1) a canonical object carrying document and graph-record lineage through role-labelled SFT examples; (2) evidence, exposure, and claim contracts with caller-supplied exact encoding and protected-field repair; (3) a case audit that detects an unrealized graph-origin path, repairs loader attrition, and bounds model conclusions; and (4) an installable artifact with controlled comparisons, CPU measurements, a RuBQ adapter, JSON Schema, CLI, negative regression tests, and a paired ruMMLU capability diagnostic[1].

[1]Reproducibility artifact: source code and documentation.

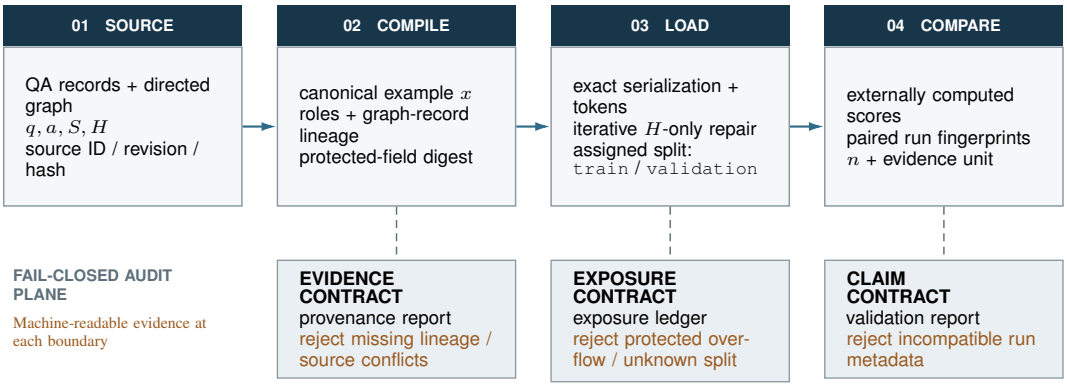


Figure 1: The compiler's data path (top) and audit plane (bottom). Each contract emits a reviewable artifact and stops on incompatible provenance, unrepairable protected content, invalid split assignment, or mismatched comparison metadata. Training, generation, and score computation are external to the library; their outputs enter the comparison stage.

2 Problem and Method

Concrete input and output. The library takes a directed graph JSON and QA JSONL with source revisions, support, and distractors. It emits validated rows, repair records, and a retention ledger; missing lineage or protected overflow fails closed. After external scoring, comparison records bind compatible runs to the same roster, score origin, inference fingerprints, denominator, and evidence unit.

2.1 Canonical compilation object

Let a source reference be $\pi = (i, u, r, h)$: stable source identifier, optional URI, revision, and content hash. A graph record is a node or directed typed relation (v_i, r, v_j) carrying one or more π values. A compiled example is

$$x = (q, a, S, H, \pi_x), \tag{1}$$

where q is the question, a the target, S the verified support chunks, and H non-support context. Each chunk has a stable ID, role, origin, source references, and, for graph-origin chunks, the producing graph-record IDs. An answer example requires non-empty S; refusal examples may have none. Parallel relations remain distinct and no reverse edge is inferred.

The schema separates an evidence object from its rendered text. This matters when several policies share wording or an entity merge obscures document identity. It also permits privacy-preserving publication: source text stays private while counts, aliases, hashes, and validation outcomes can be audited.

2.2 Three executable contracts

Figure 1 follows each object and each score origin.

Evidence contract. Every example, chunk, node, and edge must resolve to source provenance. Directed relations must declare direction; missing endpoints, duplicate IDs, unlabeled roles, and graph-origin chunks without graph-record IDs fail closed. Candidate generation may use graph adjacency, lexical similarity, retrieval, or manual rules, but the selected object's origin remains explicit. Thus a future audit can count graph-origin hard negatives directly rather than infer them from context text. For every source shared by a chunk and graph record, supplied revisions and content hashes must agree; a match on one shared source cannot hide a conflict on another. Support annotations are supplied upstream and preserved by the contract; their semantic validation remains the caller's responsibility.

Exposure contract. Define protected content $I(x) = (q, a, S, \pi_x)$ and mutable context H. A transformation f is admissible only if

$$\mathrm{hash}(I(f(x))) = \mathrm{hash}(I(x)). \tag{2}$$

The implementation also protects the example ID and task type. The caller supplies the serializer σ and tokenizer τ used by its training path. If $|\tau(\sigma(x))| > B$, repair removes the lowest-priority distractor, runs the same tokenizer again, and stops only when the row fits; it fails rather than truncate protected content. The loader adapter then records one ledger entry per intended example: token count, retained/dropped state, reason, and train/validation assignment. Only those

two exact split labels are accepted, and retained IDs must equal the disjoint union of the two exports. These checks establish unique-row retention through splitting. Establishing optimizer consumption additionally requires an independent training trace. The public adapter covers serialization, exact token IDs, budget retention, and split assignment; downstream collators and packing stages require separate checks. The case study replays the recovered closed loader and validates its final collated batches. Tail removal provides a simple, replaceable reference policy based on trusted support labels and caller-provided distractor priorities.

Claim contract. Every reported comparison binds a model artifact, fixed evaluation roster, prompt/decoding fingerprint, endpoint, denominator, missing-row policy, uncertainty method, and evidence unit. Questions are connected when their gold support resolves to a common source document; resampling and sign changes then operate on connected components. Token F1 is computed from the reference answer, the unsupported-claim proxy from sentence–retrieved-chunk NLI, and the corroborative correctness label from a blind external model judge. These are separate observables: context support, reference overlap, and judged answer correctness. Metrics are computed externally; the public validator checks claim-record completeness and run compatibility.

3 Enterprise Case Study

3.1 Data and controlled conditions

The confidential training-source inventory contains 25 technical architecture and operations documents. Evaluation uses 159 held-out QA records; the principal matched roster resolves to a separate 54-document gold-support inventory. Training and evaluation support inventories do not overlap. One training document occurs in the broader retrieved evaluation context only as non-support evidence. Retrieved contexts are long and code-heavy, usually with 15 chunks from a median of three documents.

The case is a retrospective audit of an archived offline adaptation pilot. It traces selected evidence, reconstructs the loading path, and produces a loader-safe corpus for subsequent training. Prospective integration follows the same interfaces: project records are normalized to the public schema and connected to a local loader adapter.

All conditions use a proprietary instruction-tuned checkpoint archived as GigaChat 2 Pro. Its public family report (Mamedov et al., 2025) does not identify our internal revision; architecture and parameter count are not inferred. LoRA conditions are typed context, no-hard-negative control, distractor-bearing adaptation, and loader-safe repair, compared with the unadapted generator. Appendix D records the training profile and missing trainer-state facts.

The typed source contains 1,079 intended dialogues; its distractor-bearing successor has 946 (833 answer-bearing and 113 refusal controls). Across matched keys, targets and support IDs are unchanged while 526 context signatures differ. Candidate rules admit at most two confusable, version-neighbor, or graph-derived hard negatives in six chunks. These selected additions are only one subset of mutable H, which also contains pre-existing retrieved chunks explicitly labelled non-support.

At the selection boundary, a server audit records a 484-chunk graph pool. Final metadata labels all 1,119 selected chunks in 573 rows as `existing_context` and none as graph-origin; independent aggregates agree. Approval was required, but candidate-level eligibility, approval, and rank traces are unavailable. The available selector identifies the gates whose outcomes would be needed to diagnose non-selection (Appendix H). With zero graph-origin chunks in compiled H, the model comparisons evaluate the realized distractor-bearing contexts.

The archived 83-node, 56-edge graph also lacks document provenance and declared direction despite direction-sensitive relations; 25/428 raw entity rows lack provenance. These records fail the public evidence contract's lineage and direction requirements.

3.2 Evaluation and inference

The main model comparison uses the exact 151-question intersection; questions, gold answers, contexts, prompts, and decoding fields are checked for equality. Available full-context outputs differ by condition (152–158 of 159), so every contrast reports its own denominator.

We retrospectively designate mean per-answer NLI-based unsupported-claim rate and macro token F1 as the two primary endpoints; exact definitions appear in Appendix A. The former estimates support using an automatic NLI model. Pre-

Corpus	Intended rows	Retained	Train / val.
Typed contexts	1,079	788	709 / 79
Distractor-bearing	946	906	815 / 91
Loader-safe repair	946	**946**	851 / 95

Table 1: Loader-retention funnel under exact recovered-loader replay. Counts are unique rows before epoch repetition.

Pipeline	Retained	Bad ID	Bad revision
Per-record checks	–	Accept	Accept
Exact length filter + replay	1/2	Accept	Accept
Role-aware repair + replay	2/2	Accept	Accept
Full compilation contracts	2/2	Reject	Reject

Table 2: Controlled public checks. Retention uses valid inputs; each faulty lineage input is tested separately. "–" means no loader check.

cision, recall, answer/gold length, and context ablations are diagnostics. A separate proprietary judge, blind to condition, sees the question, reference, candidate answer, and full context and accepts both correct and partial answers; its McNemar test is pair-level and corroborative.

The authoritative field-specific support resolver maps all 151 questions to 54 documents and 22 connected components; the largest contains 65 questions. We bootstrap and sign-flip whole components, then Holm-correct the two primary endpoints within each comparison. Appendix B specifies the estimand, resampling procedures, and support-resolution audit.

4 Results

4.1 The compiler exposes and repairs supervision loss

Table 1 is the process result. Exact replay of the recovered 1,024-token loader drops 291 typed-context rows and 40 distractor-bearing rows before splitting. For the latter, the compiler removes 104 annotated non-support chunks from the 40 over-budget rows and re-tokenizes after each edit. Only two removals are newly selected hard negatives; the rest are pre-existing mutable context. Questions, targets, support, roster, and order remain unchanged, and all 946 rows are retained: 851 train-assigned and 95 validation-assigned. The verified endpoint is unique-row retention through the recovered loading path.

4.2 Controlled checks on a public fixture

We compare four pipelines on the same public two-row Atlas fixture, budget, serializer, and tokenizer (Table 2). Per-record validation alone makes no retention claim; exact length filtering drops one row. Role-aware repair plus replay retains both, just as the full compiler does. The incremental assurance is cross-record lineage: only the full pipeline rejects both an unresolved graph-record reference and a conflicting source revision. Replay-based alternatives also reject invalid split labels. The injected faults isolate the checks provided by each pipeline.

The repaired fixture retains one graph-origin distractor row and removes one other distractor without changing protected content, exercising both lineage validation and repair on graph-bearing inputs.

4.3 Bounded software costs and public checks

Five warmed CPU runs on 100, 1,000, and 10,000 replicated Atlas rows take median .0098, .0904, and .9791 seconds including four exports. With a fixed four-node graph and whitespace tokens, these measurements characterize the controlled fixture workload (Appendix F). Official RuBQ import checks schema portability (Rybin et al., 2021); paired ruMMLU-1,000 provides a separate general-capability diagnostic (Fenogenova et al., 2024) (Appendix E).

4.4 Downstream model comparisons

Against the unadapted system, unsupported-claim rate changes by $-.026$ (component CI $[-.106, .024]$, $p = .543$); against no-hard-negative training, token F1 changes by $+.018$ (CI $[.004, .023]$, $p = .094$, Holm $p = .187$). None of eight component tests is Holm-significant. The blind judge accepts 140/151 unadapted versus 138/151 adapted answers ($p = .791$). Shorter answers, lower recall, zero-claim scoring, and differing claim-extraction coverage further limit interpretation. Complete effects, coverage, and descriptive context ablations are in Appendices A–C. Taken together, the comparisons remain inconclusive about a broad correctness improvement.

5 Related Work

Chen et al.'s KG-SFT extracts external reasoning subgraphs to generate and conflict-check explanatory supervision (Chen et al., 2025a); Graph-

Gen builds a fine-grained KG from source text and generates atomic, aggregated, and multi-hop QA (Chen et al., 2025b). These systems construct or enrich supervision; ours audits the subsequent evidence-to-loader-to-claim handoff. RAFT and PrismRAG provide the closest motivation for relevant-plus-distractor reader training (Zhang et al., 2024; Kachuee et al., 2025).

GraphRAG retrieves or organizes graph evidence at query time (Edge et al., 2024; Hu et al., 2025; Sharma et al., 2025); here graphs are offline construction inputs. DataDreamer contributes workflow fingerprints and traced cards, RAG Foundry integrates the adaptation workflow, and Bonito generates domain instructions (Patel et al., 2024; Fleischer et al., 2024; Nayak et al., 2024). Our interface jointly carries source-document and graph-record lineage through role-labelled SFT compilation, protects content under caller-supplied exact encoding and loader replay, and binds comparisons to score provenance and dependence components induced by shared gold-support documents. The integration combines established validation and pruning techniques into one auditable data path. Factuality work separates overlap, support, and correctness (Krishna et al., 2025; Fatahi Bayat et al., 2025); LoRA enables adaptation (Hu et al., 2022) but does not remove data-path confounds.

6 Adoption and Conclusion

Teams can start with lineage checks, add exact-token replay, then bind external scores. Without actual loader access, retention remains unknown. Appendix G details integration requirements and tradeoffs; deployment effort and production throughput are unmeasured.

In the enterprise case, lineage accounting identified an unrealized graph-origin channel, exact loader replay exposed silent row loss, and protected-field repair restored the intended corpus. Controlled public checks isolate the additional assurance from cross-record lineage validation, while staged adoption makes each interface available independently. Downstream model comparisons remained inconclusive.

For enterprise teams, the transferable order is simple: trace the selected evidence, verify the exact loading path, and only then attribute model effects. KG-SFT Contracts implements this order as executable evidence, exposure, and comparison contracts with inspectable outputs at each boundary.

Limitations of the Work

The study uses one proprietary checkpoint, one Russian enterprise QA set, and one checkpoint per condition; text, weights, exact revision, and training-seed variance are unavailable. Generation failures vary by condition. Inference has only 22 support-document components, one containing 43% of questions, and may miss other dependence. NLI and the model judge are not human evaluation; endpoints, component analysis, and repair were retrospective. The archive has no usable graph lineage and zero graph-origin hard distractors, so model contrasts cannot estimate a graph contribution. Loader retention is not optimizer exposure, and ruMMLU non-significance is not equivalence or non-inferiority. Public checks establish software behavior, not enterprise transfer or semantic support correctness; the redacted evidence cannot expose confidential text or emulate arbitrary trainer packing. The public cost study replicates two short examples against a fixed tiny graph and an inexpensive tokenizer; it does not measure production tokenization, large-graph behavior, integration effort, or incremental model benefit over simpler preprocessing. The missing selector trace prevents historical root-cause attribution.

Ethical Considerations

Only aggregates, hashes, synthetic text, and public RuBQ fields are released; no enterprise passage, output, credential, path, or weight is included. Automated evaluators may encode bias; source lineage supports deletion and license review.

Generative AI Declaration

Generative AI tools assisted manuscript drafting, language editing, and supporting code development. Model-based answer generation and evaluation are described in the experimental protocol. The authors are responsible for verifying the text, code, and results and for the final manuscript.

Acknowledgments

This article is an output of a research project (HSE-BR-2026-026) implemented as part of the Basic Research Program at HSE University. The article is also supported in part through computational resources of HPC facilities at HSE University. We gratefully acknowledge Sber for supporting this research and providing the computational resources and technical infrastructure used in the enterprise case study. We also thank ITMO University for its support.

References

Hanzhu Chen, Xu Shen, Jie Wang, Zehao Wang, Qitan Lv, Junjie He, Rong Wu, Feng Wu, and Jieping Ye. 2025a. Knowledge graph finetuning enhances knowledge manipulation in large language models. In *International Conference on Learning Representations*.

Zihong Chen, Wanli Jiang, Jinzhe Li, Zhonghang Yuan, Huanjun Kong, Wanli Ouyang, and Nanqing Dong. 2025b. GraphGen: Enhancing supervised fine-tuning for LLMs with knowledge-driven synthetic data generation. *arXiv preprint arXiv:2505.20416*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Farima Fatahi Bayat, Lechen Zhang, Sheza Munir, and Lu Wang. 2025. FactBench: A dynamic benchmark for in-the-wild language model factuality evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 33090–33110. Association for Computational Linguistics.

Alena Fenogenova, Artem Chervyakov, Nikita Martynov, Anastasia Kozlova, Maria Tikhonova, Albina Akhmetgareeva, Anton Emelyanov, Denis Shevelev, Pavel Lebedev, Leonid Sinev, Ulyana Isaeva, Katerina Kolomeytseva, Daniil Moskovskiy, Elizaveta Goncharova, Nikita Savushkin, Polina Mikhailova, Anastasia Minaeva, Denis Dimitrov, Alexander Panchenko, and Sergey Markov. 2024. MERA: A comprehensive LLM evaluation in Russian. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9920–9948, Bangkok, Thailand. Association for Computational Linguistics.

Daniel Fleischer, Moshe Berchansky, Moshe Wasserblat, and Peter Izsak. 2024. RAG Foundry: A framework for enhancing LLMs for retrieval augmented generation. *arXiv preprint arXiv:2408.02545*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2025. GRAG: Graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4145–4157. Association for Computational Linguistics.

Mohammad Kachuee, Teja Gollapudi, Minseok Kim, Yin Huang, Kai Sun, Xiao Yang, Jiaqi Wang, Nirav Shah, Yue Liu, Aaron Colak, Anuj Kumar, Wen-tau Yih, and Xin Luna Dong. 2025. PrismRAG: Boosting RAG factuality with distractor resilience and strategized reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 775–798. Association for Computational Linguistics.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. 2025. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4745–4759. Association for Computational Linguistics.

Valentin Mamedov, Evgenii Kosarev, Gregory Leleytner, Ilya Shchuckin, Valeriy Berezovskiy, Daniil Smirnov, Dmitry Kozlov, Sergei Averkiev, Lukyanenko Ivan, Aleksandr Proshunin, Ainur Israfilova, Ivan Baskov, Artem Chervyakov, Emil Shakirov, Mikhail Kolesov, Daria Khomich, Daria Latortseva, Sergei Porkhun, Yury Fedorov, and 14 others. 2025. GigaChat family: Efficient russian language modeling through mixture of experts architecture. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 93–106. Association for Computational Linguistics.

Nihal Nayak, Yiyang Nan, Avi Trost, and Stephen Bach. 2024. Learning to generate instruction tuning datasets for zero-shot task adaptation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12585–12611. Association for Computational Linguistics.

Ajay Patel, Colin Raffel, and Chris Callison-Burch. 2024. DataDreamer: A tool for synthetic data generation and reproducible LLM workflows. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3781–3799. Association for Computational Linguistics.

Ivan Rybin, Vladislav Korablinov, Pavel Efimov, and Pavel Braslavski. 2021. RuBQ 2.0: An innovated russian question answering dataset. In *The Semantic*

Web: 18th International Conference, ESWC 2021, pages 532–547. Springer International Publishing.

Kartik Sharma, Peeyush Kumar, and Yunqing Li. 2025. OG-RAG: Ontology-grounded retrieval-augmented generation for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32962–32981. Association for Computational Linguistics.

Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. 2024. RAFT: Adapting language model to domain specific RAG. In *Proceedings of the First Conference on Language Modeling*.

A Endpoint Definitions

Unsupported-claim proxy. Candidate answers are split after sentence-final `.`, `!`, or `?` followed by whitespace; stripped segments longer than 15 characters are treated as claims. Each claim is checked against every non-empty retrieved chunk with `MoritzLaurer/mDeBERTa-v3-base-mnli-xnli`. For chunks longer than 1,500 characters, a lexical-overlap heuristic selects the top three context sentences; the premise is then truncated so premise plus hypothesis is approximately 1,800 characters. A claim is supported if any chunk has `ENTAILMENT` as its argmax class; no confidence threshold is applied. The per-answer value is the fraction of claims without such support and is zero when no claim is extracted. If NLI is unavailable, the whole metric is missing rather than imputed. All primary pairs have zero missing endpoint values. The archived configuration and weight SHA-256 values are, respectively, `4e4430c95100d613df80fa01276f231931e1b535557cf62fbb3cc50323e50cca` and `65af59b1ff4450b09ecbf13ca35c840dbf038b26ff8e10e5ea89ca724828ed1e`.

Token F1. Reference and candidate are lowercased and tokenized with the Unicode-aware regular expression `\w+`. Multiset token overlap defines precision and recall, whose harmonic mean is F1; two empty strings score one and exactly one empty side scores zero. Per-answer values are rounded to four decimal places before the macro average.

Blind correctness judge. GigaChat 2 Max receives the question, reference, candidate answer, and complete retrieved context without truncation. Correct and partial labels are accepted; the paired binary endpoint rejects all other valid labels. Temperature and top-p are both .01, the response-token cap is unset, and at most three attempts are allowed. The stripped prompt SHA-256 is `662763567bf9253e2fc523ef4f15f5ece80f5c515956a26ae585c872e4bc3ea3`. All 151 matched pairs completed without judge/backend failure.

Claim-extraction coverage. Because a zero-claim answer receives unsupported-claim rate zero, Table 3 reports this edge case explicitly. The metric is defined over extracted claims rather than all answer text; condition-level answer counts differ because generation artifacts are incomplete. Differences in claim-extraction coverage may therefore contribute to unsupported-claim-rate contrasts; these unmatched records do not support causal attribution or exact decomposition of that contribution.

Condition	Answers	Zero claims	Claims
Unadapted	152	7 (4.6%)	598
Typed context	153	16 (10.5%)	392
No hard negatives	158	18 (11.4%)	387
Distractor-bearing	155	13 (8.4%)	492
Loader-safe repair	155	13 (8.4%)	448

Table 3: NLI claim-extraction coverage in available full-context outputs.

B Dependence-Aware Inference

For endpoint y, the estimand is the question-weighted paired mean difference $\hat{\Delta} = n^{-1} \sum_i (y_{i,2} - y_{i,1})$. Questions whose gold support documents are connected directly or transitively form one component. The four primary comparisons each contain 22 components: the 151-pair rosters have a largest component of 65, and the 155-pair rosters a largest component of 68.

For each of 20,000 bootstrap draws, we sample 22 whole components with replacement and divide their summed paired differences by their summed sampled question counts; the reported interval is the percentile 2.5–97.5 interval. The two-sided randomization test instead multiplies each component sum by one independent Rademacher sign and compares the absolute question-weighted mean against the observed statistic, retaining the fixed total denominator. With 22 components we use 50,000 Monte Carlo draws and the plus-one

correction. Both procedures concern the same paired mean contrast, but the percentile bootstrap uses a resampling distribution whereas the randomization test uses a null distribution that assumes component-level aggregate differences are sign-exchangeable (symmetric about zero). The interval is not an inversion of that test and the two may disagree in finite samples. Within each comparison, Holm correction covers exactly the two component sign-flip tests for unsupported-claim rate and token F1. Pair-level Wilcoxon tests are sensitivity analyses, not primary inference. Endpoints and procedures were selected retrospectively; no result is labelled confirmatory.

C Complete Primary Effects

Table 4 reports both retrospectively designated primary endpoints for every controlled comparison. Deltas are second minus first; negative unsupported-claim rate and positive F1 are favorable. Equal counts do not imply equal rosters.

Context-ablation diagnostics. With full context, the distractor-bearing generator reaches .312 token F1 and .210 unsupported-claim rate, versus .191/.956 without context, .160/.643 with distractors only, and .287/.364 with top-1 context. Unequal mode denominators (154–159) make these comparisons descriptive. The no-context NLI value partly follows from asking whether claims are supported by context available in that ablation; it is not an independent causal factuality estimate. The F1 pattern is consistent with use of answer-bearing context.

D Experimental Conditions and Traceability

The five paper-facing conditions are the unadapted generator, typed-context adaptation, no-hard-negative control, distractor-bearing adaptation, and loader-safe repair. Available full-context records are 152, 153, 158, 155, and 155, respectively. Historical small-checkpoint experiments remain registry-only and are never pooled with these estimates. Legacy metrics-only exports preserve question IDs and endpoint values but predate run/code fingerprints; paired reports therefore hash their exact inputs, rosters, support-component assignment, and analysis implementation.

The archived Pro profile specifies PEFT LoRA with rank 16, scaling 32, dropout .05, and q/k/v/o/gate/up/down projection targets. It records Adam, learning rate 2×10^{-5}, weight decay .001, 50 warm-up steps, three epochs, a 1,024-token training cap, fp16, seed 228, per-device batch one, and eight-step gradient accumulation. No trainer state survives for the exact scheduler, global step, process count, effective global batch, or selected checkpoint, so we do not reconstruct them.

The open diagnostic pins maintained MERA/ruMMLU revision `73cf223e40f1ef51ba74a048b584874a35c3d88b`, seed 1234, and a subject-round-robin 1,000-item manifest spanning all 57 subjects. Both systems reuse the same one-shot, dataset-rendered prompts and exact GigaChat SFT encoding; answers are scored by candidate-letter log probability. The paired report rejects partial rosters or any item, prompt, or protocol mismatch.

E Open Diagnostics

ruMMLU. On the frozen 1,000-item maintained-MERA roster, the unadapted and loader-safe systems score 78.0% and 77.7% (Wilson intervals [75.3, 80.5]% and [75.0, 80.2]%). The paired delta is -0.3 percentage points (bootstrap interval $[-1.2, +0.6]$ percentage points; McNemar $p = .678$). This detects no shift, but establishes neither equivalence nor non-inferiority and does not validate the private RAG endpoints.

RuBQ portability. The adapter maps public question, SPARQL, answer-bearing, and related-paragraph fields into the canonical schema. On the official RuBQ 2.0 test release, a frozen seed-228 preflight converted 100 of 1,694 eligible rows with 160 support and 2,108 distractor chunks; all passed provenance validation, and a rerun reproduced the output hashes. The converted rows and deterministic hashes provide a reproducible schema-portability check.

F Controlled Software Comparisons and Costs

The controlled software checks use the published synthetic Atlas fixture. It contains two QA rows, each with one support chunk and one graph-linked distractor. A 36-token budget under `FixtureLoaderAdapter` removes one distractor and retains both rows, including one graph-origin distractor row. All alternatives use the same inputs and encoding. “Per-record checks”

Contrast	Endpoint	n	Δ	Component 95% CI	p_{comp}	Holm p
Unadapted→distractor	Unsupported claims	151	-.026	[-.106, .024]	.543	1.000
Unadapted→distractor	Token F1	151	-.001	[-.028, .020]	.916	1.000
Typed→distractor	Unsupported claims	151	+.002	[-.012, .029]	.880	1.000
Typed→distractor	Token F1	151	+.002	[-.028, .012]	.861	1.000
No-hard-neg.→distractor	Unsupported claims	155	+.012	[-.031, .025]	.406	.406
No-hard-neg.→distractor	Token F1	155	+.018	[.004, .023]	.094	.187
Original→loader-safe	Unsupported claims	155	+.001	[-.027, .030]	.969	1.000
Original→loader-safe	Token F1	155	+.001	[-.009, .008]	.773	1.000

Table 4: Complete source-component primary-effect estimates. Pair-level tests are sensitivity analyses.

runs `ContractExample.from_dict` independently on each row. "Length filter" replays the exact fixture budget without editing; "repair + replay" uses the identical tail-removal policy as the full compiler, but omits graph-lineage validation and compilation exports. This shared policy isolates the contribution of cross-record checks and audit exports.

Two independently injected faults retain valid per-record structure but (a) reference an absent graph relation or (b) change the chunk source revision without changing its graph record. Only full compilation rejects these two cross-record faults. A third fault makes the adapter return an unknown split; all replay-based alternatives reject it, whereas per-record checks do not inspect it. In a fourth, support alone exceeds the budget: the length filter drops the row, and both repair pipelines fail explicitly instead of truncating support. No fault-detection percentage or production prevalence is estimated from these hand-constructed cases.

For timing, we replicate the pair with unique example IDs to 100, 1,000, and 10,000 rows, leaving source text and the four-node, three-relation graph fixed. All rows are retained; half lose one distractor and half retain their graph-origin distractor. The measurement environment is Darwin arm64, Python 3.12.9, and NetworkX 3.6.1, matching the package lockfile. Each workload is run five times after one 100-row warm-up; imports and example construction are untimed. Stage probes and the full compilation probe run separately, so their times must not be summed. Export time includes JSON/JSONL encoding and filesystem writes, not durable-storage synchronization. Raw timings, environment, and source hashes are in `public_validation/revision_software_checks.json`.

In the uncached reference implementation, a row with r_i distractor removals uses $r_i + 3$ token-count calls during repair (initial, loop checks, final), plus one during replay. The total is $4n + \sum_i r_i$, giving the exact call counts in Table 5. A filter-only replay uses n calls but accepts row loss. Real tokenizers, long contexts, packing, and larger graphs can change runtime substantially. Production overhead, graph-size scaling, integration labor, and model benefits require separate measurements.

To reproduce from the supplemental source package:

```
uv run --frozen python \
scripts/run_revision_software_checks.py
uv run --frozen python -m unittest \
tests.test_revision_software_checks
```

G Incremental Adoption

The three interfaces can be adopted separately; passing an earlier stage does not certify a later one. The library accepts an existing graph and does not perform graph extraction, retrieval, training, or scoring.

Stage 1: evidence-only integration. Normalize stable example/chunk/graph-record IDs, source revisions or hashes, support/distractor roles, and directed relations. Run per-record and cross-record checks before any model work, and count selected origins in the compiled output. This needs no tokenizer, but does require existing metadata or upstream changes to capture it. It establishes recorded lineage consistency, not semantic support correctness, selection quality, or loader survival.

Stage 2: exact encoding and retention. Implement `serialize`, `token_ids`, and `partition` with the actual training template, tokenizer revision, budget, and split policy. First compare replay counts and IDs with the real loader on a small sample; only then enable distractor-only repair. If collators or packing filter rows, check those stages separately. Without that access,

Rows	Record ms	Lineage ms	Repair ms	Replay ms	Compile s [min, max]	Token calls
100	1.21	.096	3.72	.254	.0098 [.0098, .0103]	450
1,000	11.80	.783	36.96	2.575	.0904 [.0893, .0910]	4,500
10,000	151.43	7.716	369.21	26.653	.9791 [.9640, .9914]	45,000

Table 5: Controlled CPU software timings: medians over five runs after a 100-row warm-up. Separate stage probes are not additive. Compilation includes graph checks, repair, replay, hashing, and four file exports, but excludes record parsing, imports, and workload construction.

publish retention as unknown rather than substitute the fixture adapter. A filter-only pipeline may be adequate if dropping over-budget rows is intentional; repair is useful when the intended corpus must be conserved and trusted role labels exist.

Stage 3: comparison integration. Attach externally generated scores to immutable run and roster fingerprints, endpoint definitions, missingness, and support-document IDs. Bind uncertainty to the declared dependence unit; the library does not compute the scores or prove statistical assumptions. Source linking and production adapters are the main project-specific work, for which no staff-time estimate is available.

The private audit illustrates two integration outcomes: detecting 40 unintended row losses and identifying which evidence channels reached the corpus. Teams needing only row-format checks may prefer a simpler validator. The staged workflow makes provenance checks available before exact trainer integration; its requirements and trade-offs are also documented in the package's adoption guide.

H Diagnosing an Unrealized Candidate Channel

The archive establishes an inventory of 484 graph candidates, approval-required selection, and zero graph-origin selections among 1,119 selected chunks. It does not establish how many graph candidates passed each intermediate gate. Inspecting the available selector reveals support/duplicate exclusions, relevance rules, graph-source-document eligibility, fingerprint-matched approval, and ranked top-k selection. Existing-context candidates receive a score bonus, but without eligible-candidate scores this does not explain the observed outcome. A global pool count is not a count of eligible candidates per question, and missing approval traces do not prove that every candidate was unapproved. This code inspection is not a version-pinned historical replay.

For a future prospective run, record one event per example–candidate pair: stable candidate/graph/source IDs and content hash; origin; proposed role; relevance score and rule; provenance and approval outcomes; deduplication outcome; rank, cutoff, and selected/rejected status; and a first-rejection reason. Aggregate this trace by origin through proposed, eligible, approved, ranked, selected, and loader-retained stages. Use explicit reasons such as missing source, approval absent, content-hash mismatch, relevance below threshold, duplicate, or rank beyond budget. Persist the selector/configuration hash and reconcile final selected counts with compiled-row lineage and the retention ledger. This proposed trace extends the public compiler's boundary checks with upstream selector instrumentation. Recovering the historical cause of zero graph-origin selections would require such candidate-level records.

I Corpus, Support, and Leakage Audits

The typed and distractor-bearing corpora contain 1,282 and 1,125 unique context chunk IDs across the same 25-document inventory. Their 946 matched rows have zero target or support-ID changes and 526 context-signature changes. Normalized training/evaluation question, target, and evidence-text hashes are disjoint; support chunk and support-document ID intersections are zero. One training document appears among all retrieved evaluation documents but not among evaluation support documents, so the narrow claim is zero verified support leakage rather than zero document overlap.

The specialized support audit resolves all 151 principal rows from the explicit gold-support field. It recovers 54 source documents and 22 components. A generic five-document count is rejected because it omits the field-specific resolution rule. Forty-three rows have multiple support documents; their semantic interpretation requires blinded review before future confirmatory analysis.

J Reproducibility Artifact

Artifact availability. The public implementation is available through the source code and documentation, with the baseline fixed at commit `089398a10d31`. The repository and supplementary source package expose the canonical dataclasses, JSON Schema, directed `MultiDiGraph` builder, invariant-aware transforms, exact-token interface, replay ledger, comparison-claim validator, source-component helper, RuBQ adapter, synthetic Atlas fixture, and deterministic tests, including invalid-split, conflicting-lineage, and null-document-ID regressions. The source supplement additionally includes controlled alternative checks, their tests, timing records, and incremental adoption documentation. The minimal public smoke test is:

```
uv run python \
  scripts/run_public_smoke.py
```

It compiles the fixture twice and checks output hashes, protected fields, directed relation counts, repair records, split assignment, and a compatible claim record. The fixture tokenizer is explicitly a smoke-test adapter. Reproducing the enterprise replay requires the archived production serializer/tokenizer and private data, which are not distributable. A separate supplementary evidence bundle contains redacted selection and retention audits, aliased primary endpoint rows, a recalculation script, and the strict aggregate ruMMLU report; it contains no enterprise text, raw identifiers, model output, server path, or weight.