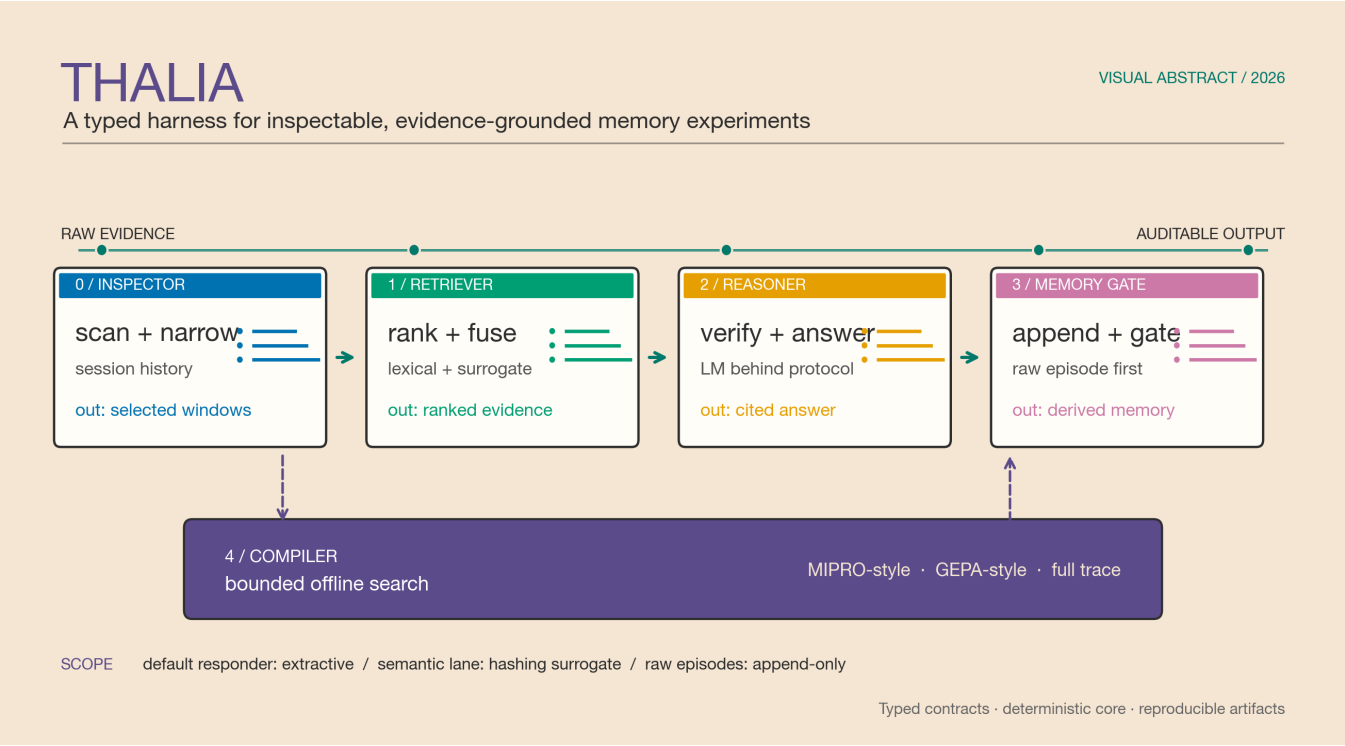
THALIA: A Typed Agentic Harness for Reproducible Long-Context Memory Experiments

Typed Pipelines, Lexical-Anchored Retrieval, Episodic-First Memory, and Bounded Compiler Traces



Daniel Ari Friedman
Active Inference Institute
FractAI
daniel@activeinference.institute
ORCID: 0000-0001-6232-9096
DOI: 10.5281/zenodo.21763245

August 2, 2026

Contents

1 Abstract

THALIA — Typed Harness with Analytical Lexical-Integrated Architecture — is an executable research harness for long-context memory systems. We implement a 5-stage pipeline that makes retrieval, context narrowing, answer generation, memory mutation, and configuration search explicit. Every stage has a typed input/output contract, a runnable implementation, an evidence trace, and a declared scope. The Inspector selects line-addressed context; the Retriever ranks and fuses lexical and surrogate-semantic evidence; the Reasoner produces a cited answer; the Memory Gate appends raw episodes before gated consolidation; and the Compiler searches a bounded configuration surface.

The core is deterministic and offline: BM25/grep, hashing-based retrieval, weighted reciprocal-rank fusion, real SQLite state, and bounded MIPRO/GEPA-style search run without network calls or mocks. A one-method LM protocol and a stable embedding interface provide explicit replacement lanes for genuine DSPy/neural backends and learned retrieval. Source-bound variables, a claim ledger, formal anchors, figure registries, and rendering checks connect implementation evidence to the manuscript.

We evaluate the harness in separate evidence tiers. On the fixed 6-example LongMemEval-style diagnostic set, the default configuration reaches composite 0.621 (answer token-F1 0.477, citation-support proxy 0.830, efficiency 0.854). The 8-candidate MIPRO-style search ties on this easy set, so its primary result is a complete, auditable trace and tie rule. A genuine `dspy.Predict`/`dspy.BootstrapFewShot` path is exercised. In the retained 24-example local comparison, replacing the extractive Reasoner with gemma3:4b changes mean token-F1 from 0.587 to 0.540 (paired difference -0.046, 95% descriptive interval [-0.161, 0.067]). On the external LongMemEval_S transfer slice, extractive and neural answer token-F1 are 0.048 and 0.110, respectively, with 95% descriptive bootstrap intervals. In the 40-question bottleneck slice, ranking recall is 1.00 while Inspector recall is 0.60; an adaptive context policy changes neural answer token-F1 from 0.086 to 0.171 while compact-context outputs remain unchanged.

The paper makes three design contributions: a typed and inspectable contract graph; an evidence-preserving runtime that localizes retrieval, narrowing, and memory failures; and a source-bound evaluation layer that can be regenerated and extended across models, embeddings, and context policies. The results support method integrity and diagnostic localization within the declared tiers. Token-F1, citation-support overlap, source-index alignment, and reachability remain operational diagnostics. Sentence-level factuality, usefulness, calibration, causal robustness, general model competence, and production reliability require the separate expert-rating and operational protocols specified in the repository.

Keywords: declarative pipelines, recursive language models, agentic search, reciprocal rank fusion, episodic memory, gated consolidation, compiler optimization, reproducible research.

2 Introduction: Research Question and Contributions

Long-context agents combine a language model with retrieval, context selection, memory, and tool orchestration. That composition creates a measurement problem: an answer can fail when the system fails to surface evidence, the model fails to use surfaced evidence, memory is rewritten, or the metric rewards the wrong answer shape. Retrieval-augmented generation already frames external memory as a complement to parametric knowledge [Lewis et al., 2020], while long-context evaluation shows that even models with large windows can use relevant information unevenly by position [Liu et al., 2024]. A harness that reports only the final answer therefore obscures the mechanism that failed.

THALIA treats the harness as a research instrument. It is **declarative** (typed stages with explicit contracts), **inspectable** (context is external state that can be queried), **evidence-preserving** (raw episodes are retained), **lexically anchored** (literal witnesses are retrieved before semantic fallback), and **compiler-searchable** (bounded configuration choices leave an auditable trace). The scope is a measurable route from evidence to answer. Universal claims about agent design require broader comparative evidence.

This manuscript synthesises four research lenses and one reproducible-research substrate into a single executable method. It then evaluates the method in separate evidence tiers so deterministic contract validation and model performance remain distinct claims.

2.1 Research Questions and Claim Ceilings

The study separates questions about the measurement instrument from questions about the systems measured by that instrument. This separation is a claim boundary: a passing contract or artifact check cannot by itself promote an answer-quality, security, or production claim.

Question	Primary evidence	Observation unit	Claim ceiling
RQ1. Does the contract execute reproducibly?	Typed tests, invariants, ledgers, and artifact hashes	Run and trace	Implementation integrity and reproducibility
RQ2. Where does a failure enter the pipeline?	Seeded synthetic ablations and bottleneck diagnostics	Example and context window	Failure localization in the generated task family
RQ3. What changes under backend or context replacement?	Recorded neural, learned-retrieval, and external-benchmark rows	Paired question or probe; model-dependent	Host- and configuration-specific transfer diagnostics
RQ4. Are stronger claims warranted?	P0 provenance, expert quality, and production acceptance gates	Adjudicated or operational row	Promotion only after an accepted certificate

The results in this manuscript address RQ1–RQ3. RQ4 is a promotion boundary, not a completed result: the corresponding protocols are implemented, but their independent acceptance records are not yet present.

The central methodological move is to make every handoff explicit. The Inspector emits line-addressed evidence windows with grep patterns and confidence metadata. The Retriever exposes lexical and surrogate-semantic rankings, fuses them by a documented score, and records whether evidence is delivered inline or by file. The Reasoner returns an answer, a task class, a trace, and source citations. The Memory Gate appends the raw turn before evaluating a derived memory delta. The Compiler searches a bounded configuration surface and records the full trace.

2.2 Four Research Foundations and the Reproducible Substrate

1. **DSPy** [Khattab et al., 2023] — prompts as parameterized code; pipelines as typed computational graphs that a compiler optimizes against a metric.
2. **Recursive Language Models** [Zhang et al., 2025] — place a long prompt in an external REPL variable that the model programmatically inspects before answer generation.
3. **Agentic search** [Sen et al., 2026] — grep/BM25 generally outperforms vector retrieval in the cited inline comparisons, with the magnitude conditioned by harness and tool style; retrieval-plus-orchestration is the correct unit of analysis.
4. **Memory consolidation faults** [Zhang et al., 2026] — continuously rewriting a memory bank degrades utility below the no-memory baseline; raw episodes must be preserved and consolidation explicitly gated.

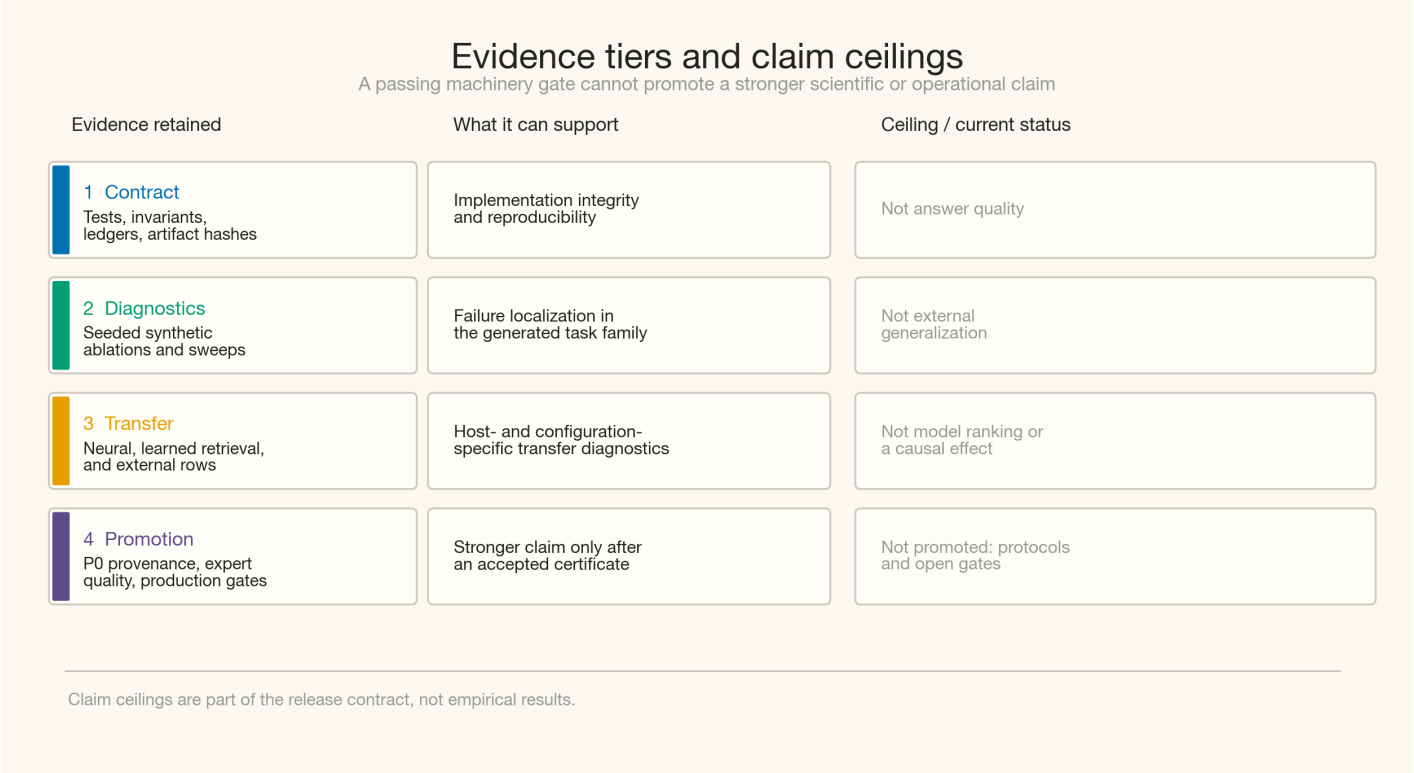


Figure 1: Claim-ceiling map for the four evidence tiers. The schematic distinguishes the observation retained by each tier, the claim it can support, and the stronger interpretation it cannot support without a separate acceptance record. It is a design and release boundary, not an empirical result.

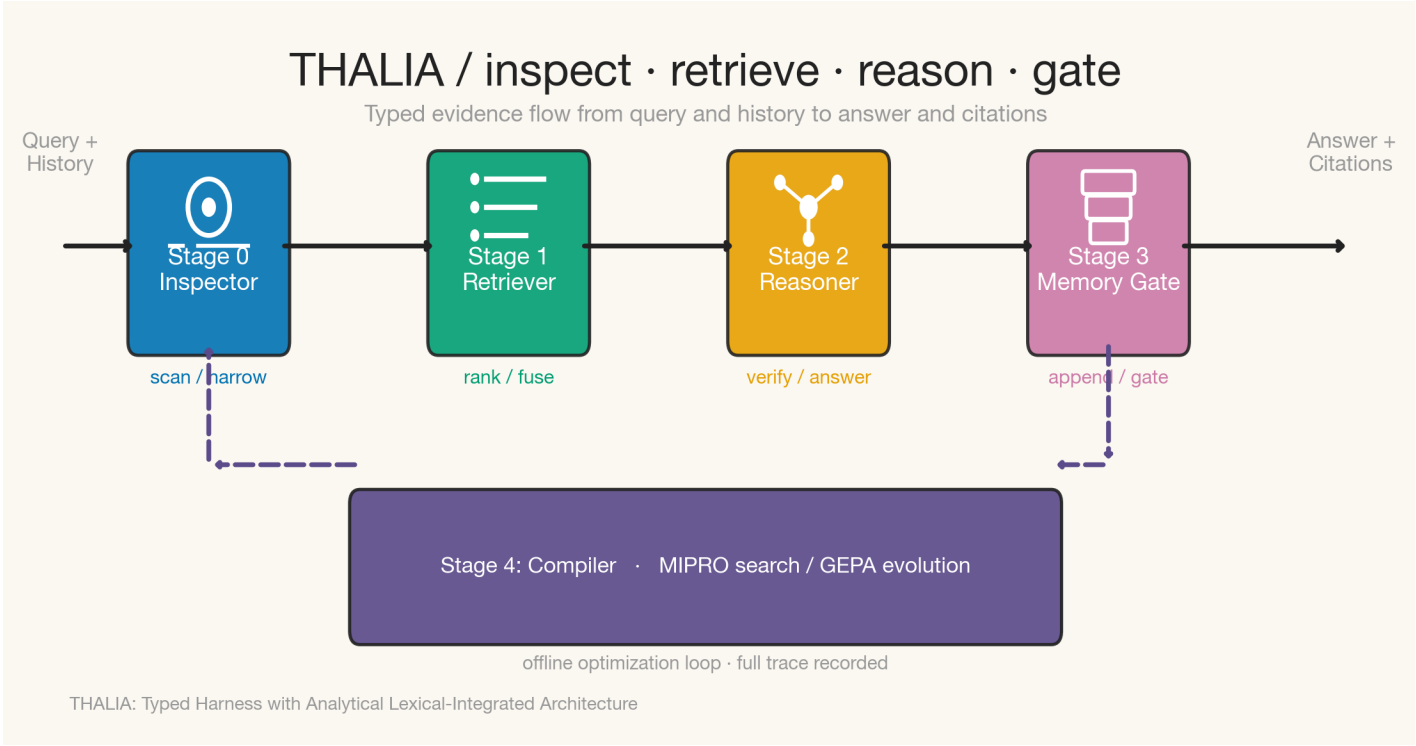


Figure 2: Schematic architecture map with no empirical unit, sample size, uncertainty interval, or model dependence. Query and history pass through inspection, retrieval, reasoning, and memory gating; arrows indicate typed data flow. The Compiler is a bounded offline loop with a recorded trace.

5. **`docxology/template`** [Friedman, 2026] — a two-layer reproducible-research substrate (Zero-Mock testing, coverage gates, cryptographic provenance, `AGENTS.md`/`SKILL.md` documentation duality) that THALIA is built inside.

The synthesis also sits within the broader retrieval-and-evaluation literature. RAG establishes the value of coupling a generator to non-parametric evidence [Lewis et al., 2020]; Lost in the Middle demonstrates the gap between evidence exposure and evidence use [Liu et al., 2024]; and FActScore shows that lexical overlap requires a separate factual-precision assessment [Min et al., 2023]. THALIA uses these findings as separable measurement surfaces. Its retrieval, answer, citation, and memory outputs can be inspected independently.

Citation-grounded generation research sharpens this separation. ALCE evaluates citation recall and precision at the statement/evidence interface and reports them alongside answer correctness [Gao et al., 2023]. ARES similarly separates context relevance, answer faithfulness, and answer relevance in RAG evaluation [Saad-Falcon et al., 2024]. THALIA adopts the measurement principle while keeping its offline core deliberately narrower: source-index precision and recall are audits of declared evidence alignment, and the lexical support proxy remains a mechanical diagnostic. Sentence-level entailment and human judgments remain future evidence lanes.

This separation is also important for the planned quality study. LongMemEval organises long-term memory around information extraction, multi-session and temporal reasoning, knowledge updates, and abstention [Wu et al., 2025]. LoCoMo adds very long-term question answering, event summarization, and multimodal dialogue evaluation [Maharana et al., 2024]. These benchmark families cover complementary memory abilities; their automatic scores cover only part of factuality and usefulness. TruthfulQA demonstrates that truthfulness is a distinct failure surface rather than a synonym for fluent answer generation [Lin et al., 2022]. For generated answers, FActScore's atomic-claim decomposition supplies a more defensible unit for factual precision than token overlap [Min et al., 2023], while SelfCheckGPT illustrates a separate, sampling-based route for detecting inconsistency in black-box generations [Manakul et al., 2023]. THALIA uses these works as construct boundaries: evidence reachability, answer correctness, evidence faithfulness, usefulness, abstention, confidence calibration, and operational readiness are recorded as related but non-interchangeable questions.

The planned confidence and abstention measurements likewise have a statistical interpretation. Calibration asks whether a reported

probability tracks empirical correctness. Fluency and apparent certainty are separate observables; the calibration literature shows why that distinction matters [Guo et al., 2017]. Selective prediction frames abstention as a risk–coverage trade-off [Geifman and El-Yaniv, 2017], and inter-rater agreement measures label reliability. Construct validity requires separate evidence that the rubric measures the intended construct [Artstein and Poesio, 2008]. These distinctions motivate the manuscript's refusal to promote the operational campaign or automatic proxies as a model-quality result.

The Hermes Agent ecosystem [Nous Research, 2026a,b] supplies the concrete engineering pattern that ties these together: a strictly-ordered prompt stack with a *frozen* memory snapshot, an episodic SQLite/FTS5 store [SQLite Consortium, 2026] distinct from the consolidated `MEMORY.md`, and a DSPy + GEPA-style evolution loop [Agrawal et al., 2025] that mutates skills against execution traces under hard constraints.

2.3 Contributions: What We Built, What We Found, and Why It Matters

THALIA is an executable research instrument for long-context memory systems. Model architecture and state-of-the-art ranking are outside its scope. The method contribution is a composable measurement boundary that makes evidence flow, memory mutation, model substitution, and publication artifacts inspectable.

2.3.1 What we built

Four design contributions define that boundary:

- **A typed contract graph.** The Inspector, Retriever, Reasoner, Memory Gate, and Compiler expose explicit inputs, outputs, evidence traces, and scope boundaries. Each handoff is testable, and each stage can be inspected without inferring its behavior from a final answer (sec. 5).
- **Evidence-preserving state management.** The runtime appends raw episodes before evaluating gated consolidation, keeps the transcript outside the model prompt, and returns cited evidence with the answer. This preserves the source material needed to audit retrieval, memory, and answer failures.
- **A deterministic core with replacement lanes.** BM25/grep, hashing-based retrieval, weighted reciprocal-rank fusion, SQLite/FTS5 state, and bounded MIPRO/GEPA-style search run offline. A one-method LM protocol and a stable embedding interface allow genuine neural/DSPy backends and learned retrieval to be evaluated under the same stage contracts. The default extractive LM and hashing semantic lane remain explicitly labelled as bounded evidence tiers.
- **Source-bound research infrastructure.** Generated variables, formal anchors, claim-ledger bindings, figure metadata, artifact manifests, and rendering checks connect code, evidence, prose, and publication outputs. The resulting package is a reproducible method specification with inspectable evidence and build receipts (sec. 13).

2.3.2 What we found

The results answer RQ1–RQ3; RQ4 remains a release decision boundary:

- **The contract is executable.** The deterministic tier reaches composite 0.621 on the fixed 6-example set, with answer token-F1 0.477, citation-support proxy 0.830, and efficiency 0.854. The 8-candidate compiler search ties on this easy set. Its primary finding is a complete, auditable trace with a deterministic tie rule; configuration improvement is unobserved on this distribution.
- **Failure surfaces are measurable before generation.** Tight-budget retrieval, evidence funnels, source-index diagnostics, noise probes, and bottleneck tests expose how context selection changes the evidence delivered to the Reasoner. In the retained bottleneck slice, ranking recall is 1.00 and Inspector recall is 0.60. The adaptive policy changes neural answer token-F1 from 0.086 to 0.171 on that slice, providing a targeted context-policy result with no general performance estimate.
- **Model substitution changes the result.** In the retained local comparison, gemma3:4b changes mean token-F1 from 0.587 to 0.540, with paired difference -0.046 and descriptive interval [-0.161, 0.067]. On the external LongMemEval_S slice, extractive and neural answer token-F1 are 0.048 and 0.110. These are model- and host-specific transfer measurements; they are evidence about the replacement lane and its recorded configuration.
- **The clearest observed difference is a lexical cited-excerpt/source-exposure proxy.** Under the recorded injection probes, THALIA's cited excerpts contain the supplied gold value on 30 of 30 probes, with direction-normalized effect +1.000 and paired interval [1.000, 1.000]. This proxy is not a true-source record, recoverable-source test, or claim-level auditability determination. The same probes show a poisoning effect of -0.000. The composite baseline lift is +0.334 because citation support and evidence efficiency increase while raw answer token-F1 ties. These observations are narrow and preserve the negative controls needed for further testing.

2.3.3 Why the method matters

The important contribution is a reusable measurement boundary for agentic memory systems. The harness reports where evidence is selected, how memory is mutated, which source indices are cited, and when a model-dependent backend changes the answer. The current evidence identifies operational provenance as the clearest advantage while showing equality on poisoning and no general answer-quality advantage. Researchers can therefore vary context policies, retrieval representations, generators, or memory gates while preserving the surrounding contracts and provenance. That separation makes a failure diagnosable and an intervention reproducible.

The most promising research direction is comparative evaluation under fixed contracts. The current package supplies the executable scaffold for expert-rated factuality, completeness, faithfulness, usefulness, abstention, calibration, safety, and production readiness. Those constructs require the separate protocols and acceptance gates defined in the repository. The present evidence establishes method integrity and failure localization within the declared tiers; general model-quality ranking and production certification remain open studies.

2.4 Reading the Architecture and Evidence Tiers

- **sec. 3** analyses the four foundational works and the substrate.
- **sec. 4** describes the Hermes prompt-assembly and self-evolution patterns THALIA adopts.
- **sec. 5** specifies the 5 stages — the methodology.
- **sec. 9** maps THALIA onto the `template/` build/validation lifecycle.
- **sec. 10** relates the architecture to Active Inference.
- **sec. 11** reports the deterministic local evaluation, with an explicit scope boundary.
- **sec. 12** consolidates design principles and the file structure.
- **sec. 13** records the provenance and how to regenerate every artifact.
- **sec. 14** covers the operable surface — configuration, the LM protocol, sessions, and the CLI.
- **sec. 15** reports the executable invariants, fuzz testing, and negative results.

3 Foundations: Declarative Pipelines, Long-Context Access, Memory, Search, and Reproducibility

This section analyses the four foundational works and the reproducible-research substrate, extracting from each the specific design implication THALIA implements. The works are complementary lenses with distinct evidence roles. DSPy addresses program structure and optimization, RLMs address context access, the memory-fault study addresses state mutation, agentic search addresses retrieval policy, and the template supplies research infrastructure. THALIA's synthesis is an executable design hypothesis. The sources provide complementary evidence; together they motivate the executable synthesis without constituting a unified theory of agency.

3.1 DSPy: Typed Program Structure and Compiler Optimization

DSPy [Khattab et al., 2023] abstracts LM pipelines as imperative computational graphs in which LMs are invoked through declarative modules. DSPy specifies the required inputs and outputs through **Signatures**, while modules and optimizers select the implementation strategy. It decomposes into four primitives: Signatures (declarative I/O), Modules (`Predict`, `ChainOfThought` [Wei et al., 2022], `ReAct` [Yao et al., 2023], …), Programs (modules wired into graphs), and Optimizers (compilers that tune instructions and few-shot examples against a metric). One prominent optimizer, MIPROv2, bootstraps few-shot candidates, proposes instruction variants, and searches the combined space with Bayesian optimization.

Implication for THALIA. Separate program structure from optimization strategy. Deterministic code handles parsing, filtering, routing, and tool dispatch; the model handles judgment. Each stage — inspector, retriever, reasoner, memory gate — is a first-class typed signature with explicit input/output fields (`src/signatures/`), and the whole pipeline is exposed to an optimizer with a well-defined metric (sec. 5). THALIA adopts the interface idea; its local MIPRO/GEPA implementations are labelled "style" or "analogue," while the optional DSPy runtime is the genuine library path. A typed interface is a software contract. Answer correctness requires its own outcome measurement.

3.2 Recursive Language Models: Explicit Context Access

RLMs [Zhang et al., 2025] treat a long prompt as part of an external environment. The prompt is placed as a variable in a REPL the model programmatically examines: **peek** at structure, **grep/regex filter** to narrow the search space, **partition + map** recursive sub-calls over chunks, **summarize**, and **verify**. The RLM environment is conceptually adjacent to a standard LM call. THALIA exposes `LMClient.complete(prompt: str) -> str`, while `REPLEnvironment` owns the explicit peek/grep/window operations. The two protocols share the external-context idea and have different interfaces.

Implication for THALIA. The REPL-environment paradigm maps directly onto the Inspector stage: load history as a variable, apply grep/regex anchored on query terms, and emit compact evidence windows plus metadata. The full transcript stays in the external environment (sec. 6.2). `src/stages/inspector.py::REPLEnvironment` implements `peek`, `grep`, and line-addressable `window` operations.

Context exposure and context use are separate observables. The long-context literature shows that models can underuse relevant material even when it is present, especially when its position changes [Liu et al., 2024]. THALIA records Inspector recall and Reasoner answer quality separately; a narrowing result is evidence about exposure, while answer quality requires its own measurement.

3.3 Memory Consolidation: Preserving Episodes and Controlling Drift

This study [Zhang et al., 2026] surfaces a failure mode in consolidated agentic memory: when an LLM continuously rewrites past trajectories into a textual memory bank, utility *first rises, then degrades*, and can fall below the no-memory baseline — even when consolidating from ground-truth solutions. The regression traces to the consolidation step; the underlying experience is a separate factor. Agents that preserve raw episodes by default and gate consolidation via explicit Retain/Delete/Consolidate actions roughly double the accuracy of forced- consolidation counterparts; disabling consolidation entirely matches that regime. Those values belong to the source study. THALIA records the finding as an external evidence tier and supplies the local append-only and gated-consolidation implementation described in sec. 13.4.

Implication for THALIA. Raw episodes are first-class evidence. The Memory Gate always appends the raw turn to an append-only store and *only* consolidates a derived MEMORY.md delta when explicitly gated (sec. 6.5). The configured per-turn

consolidation path is forbidden: the gate in `src/memory/consolidation.py::should_consolidate` requires both a Reasoner request and an explicit criterion.

The scope is one safety property within long-term memory. Systems such as MemGPT [Packer et al., 2023], Mem0 [Chhikara et al., 2025], TiMem [Li et al., 2026], and APEX-MEM [Banerjee et al., 2026] explore richer memory representations, temporal organization, and operational policies. They are adjacent comparison points, not baselines in the present deterministic study. THALIA isolates one safety property – preserve the source episode and make consolidation conditional – so that it can be tested independently of a model's broader memory competence.

3.4 Agentic Search: Lexical Anchoring and Context Selection

This empirical study [Sen et al., 2026] compares lexical (grep/regex) and semantic (dense) retrieval across models, harnesses, and tool-calling paradigms on LongMemEval. The paper reports higher accuracy for grep than vector retrieval in its own comparisons. That result remains tool-style-dependent, and overall scores also depend strongly on the harness. We cite the external result and keep its model/harness comparison in the external evidence tier. Under increasing session noise, the source studies how the ordering changes; THALIA uses that motivation for a bounded, local surrogate diagnostic. The source's magnitude and ranking remain external evidence.

Implications for THALIA. (i) Lexical-first, hybrid by default — grep/BM25 as the inline lane, semantic as fallback. (ii) Weighted Reciprocal Rank Fusion [Cormack et al., 2009] to merge lanes and optional Maximal Marginal Relevance [Carbonell and Goldstein, 1998] when diversity re-ranking is requested. (iii) Sparse context budgeting — inline when the result set fits, file-based overflow otherwise. (iv) Harness-conditioned lane weighting per detected task category. All four are implemented in `src/retrieval/` and `src/stages/retriever.py` (sec. 6.3).

The retrieval choice is a controlled representation question. Retrieval-augmented generation couples a retriever to a generator [Lewis et al., 2020], but the value of that coupling depends on both what is surfaced and how the generator uses it. THALIA keeps those stages separable so that a learned embedder, a different language model, or a different budget can be evaluated without changing the contract graph.

3.5 Friedman's template for reproducible generative research

The `template/` repository [Friedman, 2026] applies infrastructure-as-code to the research lifecycle: a two-layer architecture separates reusable infrastructure from self-contained project workspaces, connected by an eight-stage build pipeline with a Zero-Mock testing policy and a project coverage gate of 90%, cryptographic hashing, and structural validation. Its **documentation duality** equips every directory with reader-readable `README.md` and machine-readable `AGENTS.md`, and each module carries a `SKILL.md` aligned with the Model Context Protocol [Anthropic, 2025] — the same pattern Hermes uses for its skill catalog.

Implication for THALIA. The harness is built inside `template/` and inherits its discipline: every stage carries a `SKILL.md`, the eight build stages map onto the harness's compile-and-validate lifecycle (sec. 9), and the manuscript's numbers are generated tokens emitted from the live variable map (sec. 13).

The substrate is methodological infrastructure. Its tests and provenance checks make a claim auditable. Representativeness of deployed conversations requires a separate sampling and validation design.

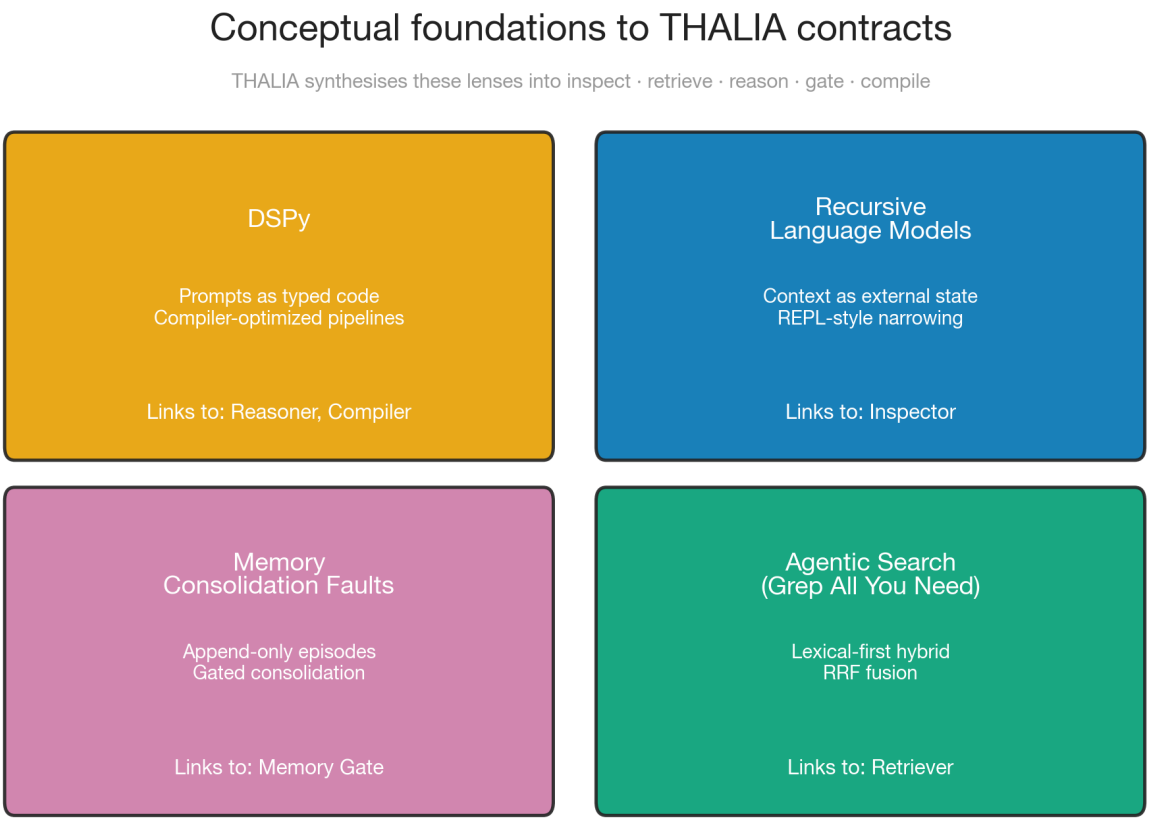


Figure 3: Schematic traceability map with no empirical unit, sample size, or uncertainty interval. Each cited foundation maps to one or more THALIA stages; mappings record conceptual reuse, while source-system validation remains in the cited evidence.

4 Hermes Architecture: Prompt Assembly, Memory, and Self-Evolution

THALIA uses the Hermes Agent ecosystem [Nous Research, 2026a,b] as a concrete engineering pattern that binds the four foundations together. Two sub-systems matter: prompt assembly with a frozen memory snapshot, and the DSPy + GEPA self-evolution loop. These are adopted as implementation patterns. Hermes model behaviour remains external evidence and is outside the THALIA reproduction claim.

4.1 Prompt Assembly and Frozen-Memory Context

Hermes assembles its system prompt as a strictly-ordered stack: agent identity, tool-aware guidance, optional static blocks, a **frozen MEMORY.md snapshot** (cross-session durable facts, injected read-only at session start), a frozen USER.md snapshot, a compact skills index, context files, and platform/timestamp hints.

The critical property is that memory snapshots are **frozen at session start** and remain stable throughout a conversation. Mid-session writes update disk state, but the assembled prompt prefix stays stable for provider-side prefix caching. This directly implements the consolidation-safety principle of [Zhang et al., 2026]: the raw episodic store (SQLite/FTS5) is kept separate from the consolidated snapshot, and the snapshot the model reasons against remains stable throughout the turn.

THALIA adoption. `src.signatures.MemorySnapshot` is a frozen dataclass injected into the Reasoner as read-only `memory_context`; the Memory Gate writes raw episodes to `EpisodicStore` and only updates its `memory_md` through gated consolidation. The snapshot the Reasoner sees within a turn is immutable (sec. 6.4, sec. 6.5).

4.2 Self-Evolution with DSPy and GEPA

The `hermes-agent-self-evolution` project [Nous Research, 2026b] connects DSPy and GEPA (Genetic-Pareto prompt evolution) to automatically evolve skills, tool descriptions, system-prompt sections, and code, in a five-phase roadmap:

Phase	Target	Engine
1	Skill files (`SKILL.md`)	DSPy + GEPA
2	Tool descriptions	DSPy + GEPA
3	System-prompt sections	DSPy + GEPA
4	Tool implementation code	Darwinian evolver
5	Continuous improvement loop	Automated pipeline

GEPA reads **execution traces** to identify the failure mechanism and proposes targeted improvements. Every evolved variant must pass hard gates before merge: the full test suite, size limits (skills $\leq$ 15 KB, tool descriptions $\leq$ 500 chars), caching compatibility (no mid-conversation changes), semantic preservation, and maintainer review.

THALIA adoption. The Compiler stage embodies both engines. The MIPRO-style search (`src/compiler/mipro.py`) evaluates configurations against the harness metric; the GEPA-style optimizer (`src/compiler/gepa.py`) reflects on the weakest metric component of an evaluation report and applies a targeted, bounded mutation, recording the full trace. The size/non-empty constraints on evolved skill artifacts are enforced by `check_skill_constraints` with the documented skill-size budget mirrored from the Hermes roadmap, and a `to_dspy_signature` bridge produces a real `dspy.Signature`; `dspy-ai` is installed as a package dependency while the deterministic core keeps that dependency outside its import boundary (sec. 6.6). The `src/dspy_runtime/` package extends the signature bridge: it runs a genuine `dspy.Predict` program and a genuine `dspy.BootstrapFewShot` optimizer, both backed by the project's real deterministic `ExtractiveLM` so the offline path stays reproducible. The `MIPRO`/`GEPA` compilers above are the project's own deterministic engine. The real DSPy machinery is a separate runtime path; the headline evaluation remains extractive-bounded in both paths.

5 THALIA Harness: Architecture and Executable Contract

THALIA is organised around 5 functional stages. 4 are runtime stages composed by `AutoHarness` (`src/harness.py`); the fifth, the Compiler, operates on the harness offline. Each stage is a `src.module.Module` with a declarative `src.signatures.Signatur e` contract. The end-to-end flow:

```
session history + query
  -> Stage 0  Inspector    RLM-style context narrowing  (peek -> grep -> windows)
  -> Stage 1  Retriever    lexical-first hybrid + weighted RRF + budgeting
  -> Stage 2  Reasoner     single-pass reasoning or ReAct-style deterministic read-integrate; typed citations
  -> Stage 3  MemoryGate   always append raw episode; gated consolidation
  -> HarnessOutput
  about > Stage 4  Compiler     (offline) MIPRO search + GEPA reflective evolution
```

The language model is abstracted behind a one-method protocol (`src.module.LMClient.complete`), and the default `ExtractiveLM` returns the evidence line with maximal lexical overlap with the question. The deterministic default path is therefore testable without a network; substituting a real model creates a separate model-dependent evidence tier without changing the stage contracts.

The design is deliberately conservative about where judgment can enter. Stages zero and one are retrieval and filtering functions; Stage 2 is the language-model boundary; Stage 3 is a storage and consolidation policy; Stage 4 is an offline optimizer. This separation supports failure diagnosis: an absent citation points to retrieval or Inspector reachability, an unsupported answer points to the Reasoner or its metric, a spurious `MEMORY.md` change points to the Memory Gate, and a non-reproducible compiled configuration points to the Compiler. These are diagnostic attributions within the contract graph. Causal claims require a separate intervention design. The manuscript, tests, and dashboard report the same typed evidence objects generated from the contracts.

This spine preserves the stable auto-harness design anchor. The detailed method contract, stage-by-stage design, formalism, added-method diagnostics, and implementation traceability now live in adjacent `04a_` through `04c_` modules.

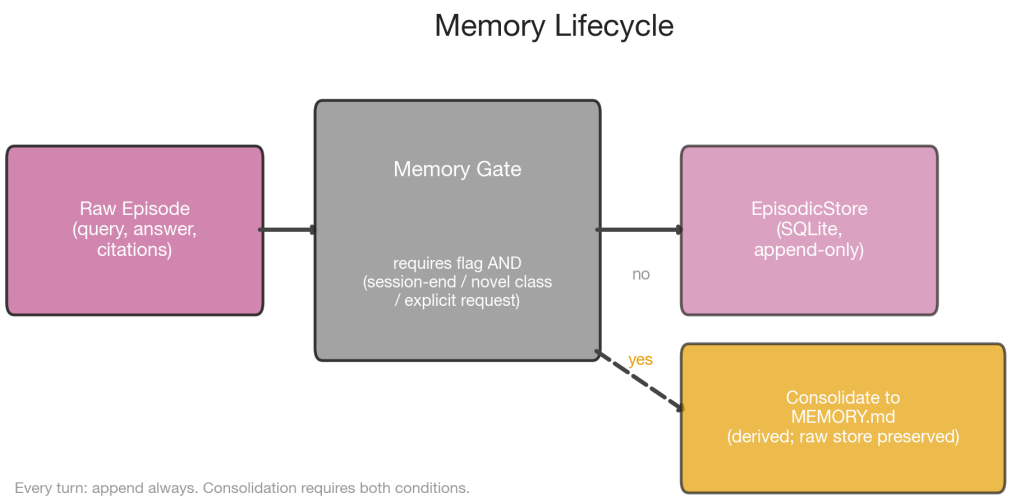


Figure 4: Schematic lifecycle with no empirical unit, sample size, uncertainty interval, or model comparison. Every raw episode is appended to SQLite EpisodicStore; the dual-key gate controls whether a derived MEMORY.md delta is written, preserving the source turn.

Modular harness-design files:

- `04a_harness_method_contract_and_stage_flow.md`
- `04b_harness_formalism_and_added_methods.md`
- `04c_harness_implementation_traceability.md`

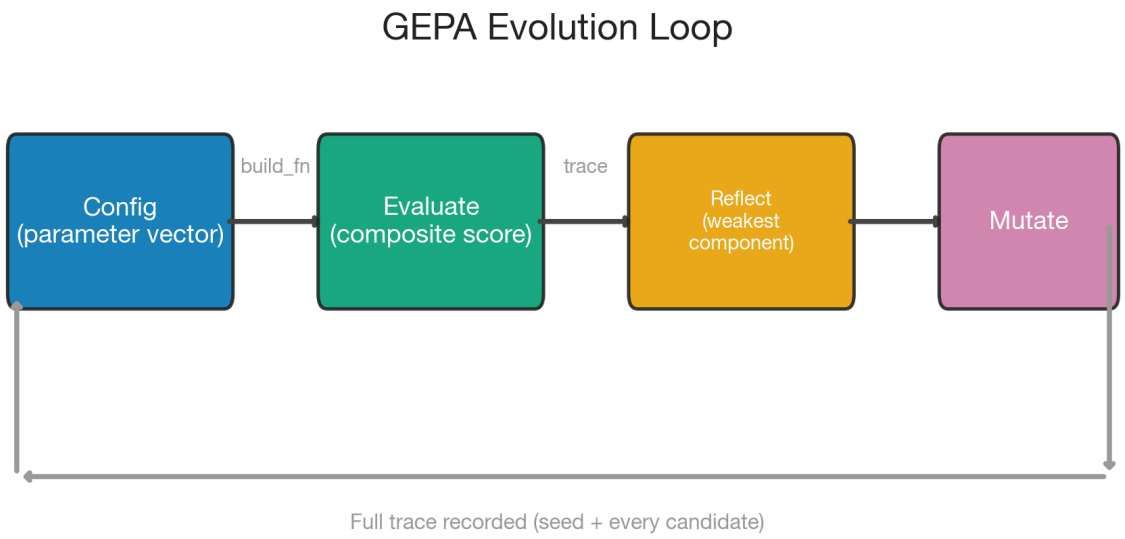


Figure 5: Schematic GEPA-style audit loop with no empirical unit, sample size, uncertainty interval, or external model dependence. The Compiler evaluates a configuration, identifies the weakest metric component, applies a bounded mutation, and records the complete trace as described in sec. 11.5.1.

6 Stage Contracts: From Transcript to Auditable Evidence

6.1 Method Contract and Typed I/O

Each harness invocation obeys the following contract:

1. **Evidence before inference.** Session history is converted into addressable lines before the Reasoner sees anything. Evidence windows retain source indices and snippets, so an answer can cite raw text with source-level provenance.
2. **Rank before answer.** The Retriever exposes ranked evidence and delivery mode. Inline answers are produced only after fusion and budgeting, and file-based overflow is an explicit output state recorded in the handoff.
3. **Answer before memory.** The Memory Gate receives a completed answer with citations and an explicit consolidation flag. The flag and the declared criterion jointly determine whether a memory update is eligible.
4. **Append before summarize.** Raw episodes are written first. Consolidated MEMORY.md deltas are derived artifacts; the source turn remains in the store.
5. **Optimize bounded surfaces.** The Compiler searches `HarnessConfig` values and bounded skill artifacts. Source code and manuscript narrative remain outside the search surface.

This contract is the practical bridge between the cited research and runnable code. It converts DSPy-style declarations, RLM-style external context handling, lexical-first retrieval, and gated memory into enforceable interfaces. It also separates the variables that a model-quality study must keep distinct: evidence reachability, evidence selection, answer generation, citation support, and memory mutation. A passing handoff is a precondition for a useful answer. Answer quality remains a separate outcome.

6.2 Stage 0 — Inspector: RLM-Style Context Narrowing

The Inspector treats session history as external state. It loads the transcript into a `REPLEnvironment`, peeks at structure, greps for query-anchored patterns, ranks matching records by query-token overlap, and emits at most `max_windows` compact `EvidenceWindow` objects with auditable `InspectorMetadata`. If grep returns nothing it falls back to ranking every record; if the query contributes no content tokens it keeps records in order. The full transcript remains external state; only selected windows are eligible for a model call.

```
class InspectorSignature(Signature):
    """Inspect session history as external state.
```

```
    Use code execution (peek / grep / partition) to filter evidence before
    forwarding selected windows to the Reasoner."""
    session_history: str = InputField("Raw session transcript (arbitrarily long).")
    query: str = InputField("Current task or question.")
    evidence_windows: list = OutputField("Compact snippets filtered from history.")
    metadata: InspectorMetadata = OutputField("Grep patterns, indices, confidence.")
```

This is the harness's defence against context rot: downstream stages see only query-relevant windows, and `metadata.confidence` records the fraction of query tokens the selected windows cover.

Two fallback paths are load-bearing. If grep returns no match, the Inspector ranks all records by overlap and preserves a usable evidence set. If the query has no usable content tokens, the Inspector preserves transcript order and avoids a stop-word relevance signal. Both behaviours are deterministic and tested.

6.3 Stage 1 — Retriever: Lexical-First Hybrid Fusion

The Retriever ranks Inspector windows with two lanes and fuses them. The lexical lane is Okapi BM25 (`src/retrieval/lexical.py`); the semantic lane is a deterministic hashing-embedding cosine (`src/retrieval/semantic.py`). Ranked lists merge by weighted Reciprocal Rank Fusion [Cormack et al., 2009] — contribution $w/(k + r + 1)$ for a chunk at 0-based rank r in a lane of weight w, with defaults $k = 60$, lexical weight 0.65, semantic weight 0.35. Lane weights are **harness-conditioned**: temporal/single-session queries weight lexical higher, preference/knowledge-update queries lean more on the semantic lane. Sparse context budgeting sets `delivery_mode` to `INLINE` when the selected evidence fits `context_budget_tokens` and `FILE_BASED` otherwise.

The Retriever therefore exposes both *what* was selected and *why that selection fits the delivery path*. This matters because the agentic-search literature shows that inline and programmatic/file-based delivery are different regimes [Sen et al., 2026]. THALIA records delivery mode as part of the typed output, so downstream diagnostics can separate a retrieval miss from a budget overflow.

```
class RetrieverSignature(Signature):
    """Rank evidence using hybrid lexical + semantic search.
    Prefer lexical anchors for literal witnesses; semantic fallback for
    paraphrases. Apply weighted RRF and sparse context budgeting."""
    evidence_windows: list = InputField("Windows emitted by the Inspector.")
    query: str = InputField("Current task or question.")
    task_category: TaskCategory = InputField("Detected task category.")
    context_budget_tokens: int = InputField("Max tokens for inline injection.")
    ranked_evidence: list = OutputField("Top-k evidence after RRF fusion.")
    delivery_mode: DeliveryMode = OutputField("inline if within budget, else file_based.")
```

6.4 Stage 2 — Reasoner: Typed Answer Generation

The Reasoner produces a typed answer with auditable citation support. Deterministic code classifies the task (`src.routing.classify_task`), routes multi-hop/code/knowledge-update tasks through a deterministic ReAct-style read→integrate selection path, and uses one model completion for both routes. It emits an auditable routing and evidence-use trace. It builds `EvidenceCitations` from the windows whose text overlaps the answer. The `evidence_citations` output is what lets the Memory Gate audit answer-vs-evidence support, and what grounds any later consolidation in verifiable raw text. `requires_consolidation` is set only for knowledge-update tasks or explicit "remember" cues.

The Reasoner is also where optional grounding verification occurs. When `verify=True`, the stage computes a set-based answer-token support proxy against the cited excerpts. This is a citation-support diagnostic. Atomic factual precision requires the separate evaluation described by FActScore [Min et al., 2023]. If support falls below `grounding_threshold`, it re-answers over the full ranked evidence set and records the before/after scores in the trace. Verification is off by default because the deterministic local evaluation already exercises the baseline path; it is available as an operational guardrail when a real LM adapter is inserted.

```
class ReasonerSignature(Signature):
    """Reason over pre-filtered evidence to produce a structured answer.
    Deterministic code handles selection and routing; the model handles only
```

```
    judgment. Typed outputs prevent downstream parsing failures."""
    query: str = InputField("Current task or question.")
    ranked_evidence: list = InputField("Ranked evidence from the Retriever.")
    task_category: TaskCategory = InputField("Detected task category.")
    memory_context: MemorySnapshot = InputField("Frozen MEMORY/USER snapshot (read-only).")
    reasoning_trace: str = OutputField("Auditable routing and evidence-use trace; hidden reasoning is excluded."
    answer: str = OutputField("Final answer or action.")
    evidence_citations: list = OutputField("Raw source windows cited.")
    requires_consolidation: bool = OutputField("True for knowledge-update tasks or explicit memory cues.")
```

6.5 Stage 3 — Memory Gate: Episodic-First Consolidation

The Memory Gate enforces the consolidation-fault safeguard [Zhang et al., 2026] in three steps: (1) **always** append the raw episode to the append-only `EpisodicStore`; (2) compute explicit criteria — session end, novel task class, or explicit user request; (3) consolidate a derived MEMORY.md delta when the Reasoner requested it *and* a criterion holds. The source episode remains available, and the delta cites its source turn so consolidated facts remain traceable.

The gate is intentionally asymmetric: every turn can become evidence, while a small subset can become a durable summary. This policy keeps conversational noise out of long-lived memory and preserves the raw source for future recall. The append-only store plus source-cited delta preserve both recall and auditability.

```
class MemoryGateSignature(Signature):
    """Episodic-first memory management.
    Always write the raw episode. Consolidation requires requires_consolidation
    plus a criterion (novel task, session end, explicit request).
    Summaries are derived artifacts; raw episodes remain intact."""
    query: str = InputField("Current task or question.")
    answer: str = InputField("The produced answer.")
    evidence_citations: list = InputField("Citations from the Reasoner.")
    requires_consolidation: bool = InputField("Reasoner's consolidation flag.")
    session_episode_count: int = InputField("Episodes currently recorded in the store.")
    episodic_entry: EpisodicEntry = OutputField("Raw turn record to append.")
    consolidation_action: str = OutputField("none | append_memory | update_memory.")
    memory_delta: str = OutputField("Delta to MEMORY.md if consolidating, else ''.")
```

6.6 Stage 4 — Compiler: Bounded Search and Evolution

The Compiler closes a bounded feedback loop over the `HarnessConfig` parameter vector (retriever depth, evidence cap, RRF constant, lane weights, window cap, token budget). The MIPRO-style compiler (`MIPROCompiler.compile`) evaluates each candidate configuration against the harness metric and returns the best with a full trace; an `auto` budget tier truncates the search space. The GEPA-style optimizer (`GEPAOptimizer.evolve`) reflects on the weakest component of an evaluation report (answer token-F1 / citation-support / efficiency) and applies a targeted, bounded mutation — deepen retrieval when citation support is weak, emphasise lexical matching when answer token-F1 is weak, tighten depth when efficiency is weak — recording every candidate. Evolved skill artifacts must pass `check_skill_constraints` ($\leq$ 15 KB, non-empty), mirroring the Hermes self-evolution gates.

Because evaluation builds a *fresh* harness per example (`build_run_fn`), the search is order-independent and reproducible: identical inputs yield the identical winning configuration and trace.

The compiler trace is part of the method. It records the candidate configuration, component scores, selected mutation, and resulting configuration so a reader can tell whether an apparent improvement came from answer token-F1, citation-support, or efficiency. On the current compact dataset the candidates tie structurally; the trace documents that the declared search surface was executed and that the reported optimum follows a deterministic tie rule. A performance gain requires a dataset on which the candidate configurations separate.

7 Formal Specification: Retrieval, Evidence, Metrics, and Evaluation

7.1 Formal Objects and Their Executable Status

The retrieval, evidence, and evaluation lanes are specified as executable functions with explicit inputs, outputs, units, and invariants. The code-paper registry contains 24 formal objects and 6 composition edges, with 9 audited claims; its generated table appears in sec. 8.1. The equations below state the exact estimands and units used by the implementation. Their scope is operational: each proxy measures the code-defined quantity named in its unit and estimand fields. Relevance, factuality, efficiency, and model quality require separate construct validity evidence. Finite synthetic measurements remain finite-set observations. The canonical symbol crosswalk and statistical contracts are collected in sec. 16.

7.2 Typed Stage Composition and Dataflow

Let $\mathcal{H}$ denote the raw session transcript, q the current query text, c_q its classified task category, θ the immutable `HarnessConfig`, and $\mathcal{M}$ the frozen memory snapshot supplied to one invocation. Let t be the turn identifier, s the session-ending indicator, and $b_N = (r_N, x_N)$ the Reasoner's runtime gate payload, where r_N is the consolidation flag and x_N records an explicit memory request. The stage composition is:

$$\begin{aligned} I_\theta(q, \mathcal{H}) &= (W_I, \mu_I), \\ R_\theta(q, W_I, c_q) &= (W_R, \delta_R), \\ N_\theta(q, W_R, c_q, \mathcal{M}) &= (\hat{y}, C_{\text{cite}}, \tau_N, b_N), \\ G_\theta(q, \hat{y}, C_{\text{cite}}, b_N, c_q, t, s; \Sigma_G) &= (e, \Delta_{\mathcal{M}}, \Sigma'_G). \end{aligned} \tag{1}$$

W_I is the Inspector window sequence, μ_I is Inspector metadata, W_R is the ranked evidence sequence, δ_R is the delivery decision, $\hat{y}$ is the answer, C_{cite} is the cited source-index set, τ_N is the Reasoner trace, e is the episodic append result, $\Delta_{\mathcal{M}}$ is an optional derived memory delta, and Σ_G is the Memory Gate state containing the append-only store and current derived-memory text. The composition enforces the evidence boundary: N_θ receives W_R and the frozen snapshot, while G_θ receives the completed answer, citations, gate flags, task category, turn state, and gate state. The transition $\Sigma_G \to \Sigma'_G$ appends the raw episode before any derived memory write. The optional Compiler searches a bounded subset of θ and records its trace; the stage functions remain fixed during each run.

For a finite evaluation set $\mathcal{D}$ and candidate configuration surface Θ, the deterministic Compiler selects:

$$\hat{\theta} = \text{first}_{\prec_\Theta} \arg\max_{\theta \in \Theta} \bar{M}(\theta; \mathcal{D}), \qquad \mathcal{T}_C = \{(\theta, \bar{M}(\theta; \mathcal{D}), \text{trace}_\theta) : \theta \in \Theta\}. \tag{2}$$

Here $\bar{M}$ is the declared mean of per-example composite scores and $\prec_\Theta$ is the stable candidate order. The first-achieving rule makes a tie an explicit selection outcome. Equal scores remain equal; a performance difference requires score separation.

The observation unit is explicit at every evaluation boundary. Per-example metrics use one generated or retained question as the unit; category summaries aggregate those rows; seed-cluster summaries resample the declared seed unit; planned quality and production lanes use their own declared rows. A number is interpretable only with its unit, denominator, evidence tier, and interval or test construction.

Shared tokenizer. Let $T(x)$ be the ordered sequence returned by `src.module.tokenize`: lowercase matches of the alphanumeric regular expression. The sequence is shared by retrieval and scoring; the tokenizer retains lowercase alphanumeric matches and drops punctuation.

$$T(x) = \text{findall}(\texttt{[A-Za-z0-9]+}, \text{lower}(x)) \tag{3}$$

Answer token-F1. The headline answer component uses distinct-token sets. Let $U(x)$ be the set of distinct tokens in $T(x)$ and $I = U(\hat{y}) \cap U(y)$. The observation unit is one evaluation example. The score measures lexical overlap. Semantic correctness, entailment, factuality, and answer-level annotation require separate measurements.

$$F_1(\hat{y}, y) = \begin{cases} 1, & |U(\hat{y})| = |U(y)| = 0 \\ 0, & |U(\hat{y})| = 0 \text{ xor } |U(y)| = 0 \\ \dfrac{2|I|}{|U(\hat{y})| + |U(y)|}, & \text{otherwise.} \end{cases} \tag{4}$$

The implementation returns 1 when both sets are empty and 0 when exactly one set is empty. Repeated occurrences have no effect because the score uses distinct token sets.

BM25 (lexical lane) [Robertson and Zaragoza, 2009]. For query q treated as the token multiset produced by T, document d with term frequency $f(t,d)$, document length $|d|$, mean length $\overline{d}$, and corpus size N:

$$\mathrm{BM25}(q,d) = \sum_{t \in q} \mathrm{idf}(t) \frac{f(t,d)(k_1+1)}{f(t,d) + k_1(1 - b + b|d|/\overline{d})}, \qquad \mathrm{idf}(t) = \ln\left(1 + \frac{N - n_t + 0.5}{n_t + 0.5}\right) \tag{5}$$

The sum is over query-token occurrences: repeated query tokens contribute repeatedly, while repeated document tokens affect $f(t,d)$. Defaults are $k_1 = 1.5$ and $b = 0.75$. Terms absent from the corpus have zero contribution; documents with a zero total score are excluded from `top_k`; ties use ascending document index. The $+1$ inside the logarithm keeps $\mathrm{idf}(t) \geq 0$ (an executable invariant, sec. 15).

Hashing-cosine semantic surrogate. The implementation first lowercases text, collapses whitespace, and pads the non-empty result with two spaces on each side. Let $G_3(x)$ be the resulting ordered character-trigram list, retaining repeated trigrams. Each trigram is hashed with SHA-1 into one of D buckets and uses the fifth digest byte for a signed contribution. Let $h(u)$ be its bucket and $s(u) \in \{-1, +1\}$ its sign. The zero vector is returned for empty input; otherwise the vector is L2-normalized before cosine comparison.

$$v_j(x) = \sum_{u \in G_3(x)} s(u)\, \mathbf{1}[h(u) = j], \qquad e(x) = \frac{v(x)}{\|v(x)\|_2}, \qquad \mathrm{sim}(x,z) = \begin{cases} e(x) \cdot e(z), & \|v(x)\|_2 \|v(z)\|_2 > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The lane is a deterministic sub-word surrogate. Learned dense embeddings and semantic correctness are separate evidence lanes. Positive similarities are retained for ranking; ties are broken by ascending document index after fixed-precision comparison.

Weighted Reciprocal Rank Fusion. For a candidate chunk c at zero-based rank $r_\ell(c)$ in lane ℓ of weight w_ℓ (lexical \$ 0.65\$, semantic 0.35):

$$\mathrm{RRF}(c) = \sum_{\ell} \frac{w_\ell}{k + r_\ell(c) + 1}, \qquad k = 60. \tag{7}$$

Each lane is de-duplicated by retaining its first occurrence before scoring; absent lanes contribute nothing. The fused score defines candidate ordering. It has no calibrated-probability interpretation. The reported summed float is preserved. The ordering key rounds to twelve decimal places before the ascending-id tie-break, which stabilizes equal candidates across platform floating-point variation.

Maximal Marginal Relevance. Given fused scores s_i, the implementation min–max normalizes relevance as $\mathrm{rel}_i = (s_i - s_{\min})/(s_{\max} - s_{\min})$; when all scores are equal it assigns relevance one. With selected set S_{MMR} and hashing-cosine similarity, the empty-set maximum is defined as zero:

$$\mathrm{MMR}(i \mid S_{\mathrm{MMR}}) = \lambda\, \mathrm{rel}_i - (1 - \lambda) \max\left(\{\mathrm{sim}(i,j) : j \in S_{\mathrm{MMR}}\} \cup \{0\}\right). \tag{8}$$

The first item is the maximum-relevance item; subsequent items are greedy. Ties use relevance and then ascending candidate index. Thus $\lambda = 1$ gives the relevance order, while smaller values trade relevance for novelty. The method is a selection heuristic. Evidence that diversity improves answer truth requires an answer-level intervention and adjudication study.

Inspector relevance and partitioning. The Inspector loads the complete transcript into an external, line-addressable state object. Only selected windows are eligible for the Reasoner prompt. For query-token set Q_I and non-empty record L_i, its relevance is:

$$\rho_i = \frac{|Q_I \cap U(L_i)|}{|Q_I|}, \qquad \rho_i = 0 \text{ when } |Q_I| = 0. \tag{9}$$

When the record count exceeds `partition_threshold`, consecutive blocks B_b of `partition_size` records are scored by $\sum_{i \in B_b} \rho_i$. The Inspector keeps $m = \max(1, \lfloor \texttt{max\_windows}/\texttt{partition\_size} \rfloor + 1)$ highest-scoring blocks and fine-ranks records inside their union. Equal block scores are ordered by ascending block index:

$$\text{score}(B_b) = \sum_{i \in B_b} \rho_i, \qquad \mathcal{A} = \bigcup_{b \in \text{Top}_m(B)} B_b. \tag{10}$$

The emitted confidence is the fraction of distinct query tokens covered by the selected windows. It is a lexical coverage diagnostic. Probability calibration and answer correctness require separate labels and outcomes. A later ranking stage has no access to a discarded block; this mechanism explains why Inspector recall can be isolated as a bottleneck in a retained diagnostic slice. Causal claims require an intervention design.

7.3 Added Methods: Retrieval, Memory, and Compiler Extensions

The following optional methods compose with the base pipeline. Their code anchors, tests, units, estimands, and limitations are generated in sec. 8.1.

Query expansion — PRF. With `query_expansion=True`, an initial BM25 pass returns R top windows. After removing stopwords, original query terms, and terms of length at most two, the feedback count is $c(t) = \sum_{d \in R} \mathbf{1}[t \in T(d)]$. Terms are appended in descending $c(t)$, then alphabetical order, capped at `top_terms`:

$$q' = q \parallel \text{first}_K \left(\text{sort}_{(-c(t),t)} \{t : c(t) > 0\} \right). \tag{11}$$

This is RM3-inspired deterministic feedback. Learned query expansion remains a separate optional method.

Recency weighting. For eligible non-negative transcript index i, let $i_{\max}$ be the largest eligible index and $s_{\max}$ the largest fused score. With $\rho \in [0, 1]$:

$$s'_i = \text{round}_8 \left(s_i + \rho s_{\max} \frac{i}{i_{\max}} \right), \quad i \geq 0,\ i_{\max} > 0; \qquad s'_i = s_i \text{ for recalled } i < 0 \text{ or } i_{\max} \leq 0. \tag{12}$$

The result is sorted by descending adjusted score and ascending source index. Source position is a temporal-order proxy. Elapsed time and causal effects require timestamped data and a corresponding design.

Evidence budget. For selected window texts W the structural token estimate is the shared-tokenizer count. Delivery is inline exactly when it fits the configured budget:

$$\hat{n}(W) = \sum_{w \in W} |T(w)|, \qquad \text{delivery}(W) = \begin{cases} \text{INLINE}, & \hat{n}(W) \leq B \\ \text{FILE\_BASED}, & \hat{n}(W) > B. \end{cases} \tag{13}$$

The quantity is a structural context-size proxy. Latency, prompt-token billing, memory consumption, monetary cost, and user experience require operational measurements.

Citation-support proxy. Let C_{cite} be cited source indices, E_{cite} their joined excerpts, and G_{gold} the expected source-index set. Let $H = 1$ when $C_{\text{cite}} \cap G_{\text{gold}} \neq \emptyset$ and zero otherwise. The implementation uses:

$$G_{\text{sup}} = \begin{cases} \frac{1}{2} H + \frac{1}{2} F_1(\hat{y}, E_{\text{cite}}), & G_{\text{gold}} \neq \emptyset \\ F_1(\hat{y}, E_{\text{cite}}), & G_{\text{gold}} = \emptyset. \end{cases} \tag{14}$$

The aggregate field `grounding` denotes citation-support token overlap. Entailment, factual attribution, line-level evidence grounding, and the fine-grained factuality assessment proposed by FActScore [Min et al., 2023] require separate annotations.

Citation-set diagnostics. The live campaign also records two source-index diagnostics that decompose the citation set. Let $C_{mathrmcite}$ be the distinct cited indices and $G_{mathrmgold}$ the declared expected indices:

$$P_{\text{cite}} = \begin{cases} 0, & |C_{\text{cite}}| = 0 \\ 1, & |C_{\text{cite}}| > 0 \ \wedge \ |G_{\text{gold}}| = 0 \\ \dfrac{|C_{\text{cite}} \cap G_{\text{gold}}|}{|C_{\text{cite}}|}, & \text{otherwise.} \end{cases} \tag{15}$$

$$R_{\text{src}} = \begin{cases} 1, & |G_{\text{gold}}| = 0 \ \wedge \ |C_{\text{cite}}| > 0 \\ 0, & |G_{\text{gold}}| = 0 \ \wedge \ |C_{\text{cite}}| = 0 \\ \dfrac{|C_{\text{cite}} \cap G_{\text{gold}}|}{|G_{\text{gold}}|}, & \text{otherwise.} \end{cases} \tag{16}$$

P_{cite} exposes extraneous cited indices and R_{src} exposes missing declared sources. Both operate at the adapter's source-index granularity. ALCE separates citation recall and precision through statement-level entailment [Gao et al., 2023], and ARES separates context relevance, answer faithfulness, and answer relevance [Saad-Falcon et al., 2024]. THALIA's two source-index diagnostics therefore strengthen the audit of its lexical support proxy while remaining below sentence-level entailment. fig. 6 shows the finite demo decomposition by task category.

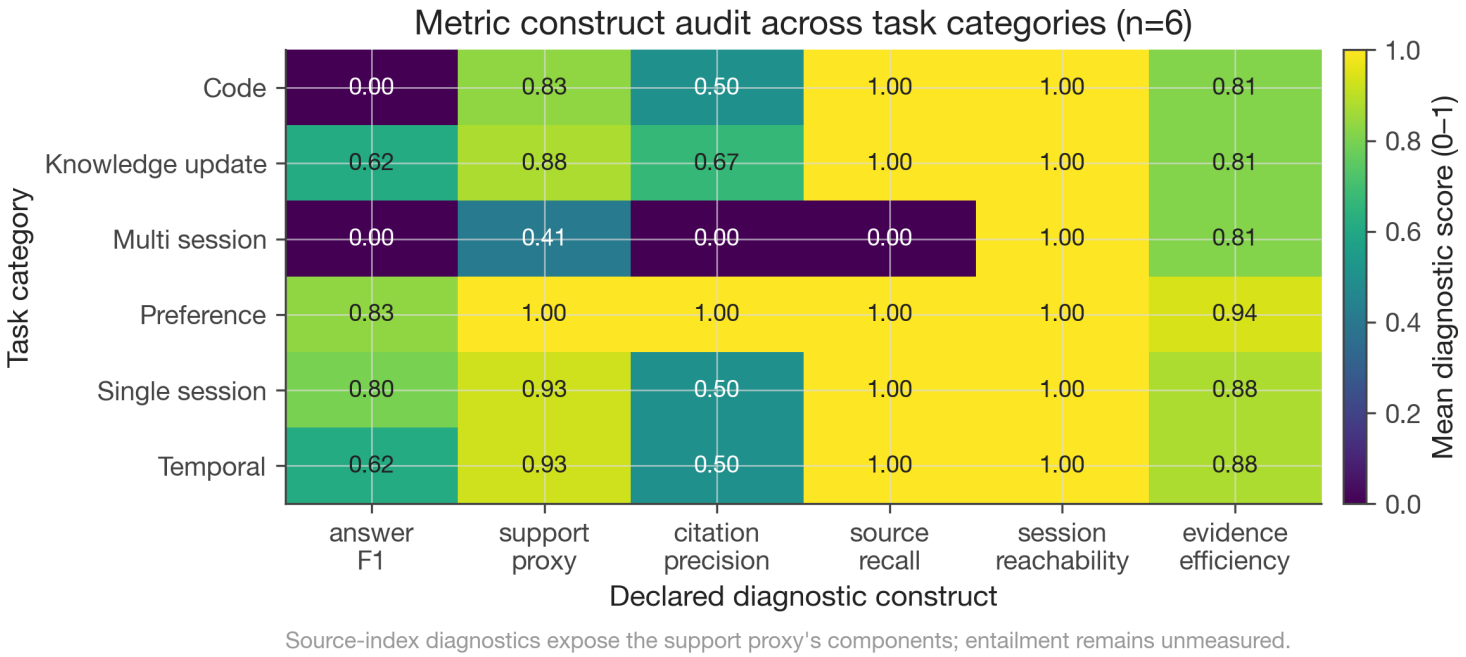


Figure 6: Metric construct audit for the fixed 6-example synthetic demo. Rows are task categories and columns are answer token-F1, the blended citation-support proxy, source-index citation precision, source-index recall, session reachability, and evidence-count efficiency. Cell values are finite category means on a unit interval; the figure is a construct audit with no uncertainty interval or model-quality estimand.

Structural efficiency and composite score. With $n = |R|$ ranked evidence windows and scale $S_{\text{eff}} = 16$:

$$E_{\text{eff}}(R) = \max\left(0, 1 - \frac{n}{S_{\text{eff}}}\right). \tag{17}$$

The per-example composite is a fixed convex sum:

$$M = 0.6A + 0.3G_{\text{sup}} + 0.1E_{\text{eff}}. \tag{18}$$

Its estimand is the finite-set mean of per-example scores unless a section states a different unit. The research ledger retains observed failures in requested denominators when the protocol declares that denominator. Optional model failures remain separate from completed rows.

Analysis connection. The statistical layer uses explicit strata and seed clusters. Continuous-score intervals are descriptive percentile or cluster- resampling intervals. Their interpretation follows the declared resampling unit and estimand. Paired continuous-score comparisons use a sign-flip randomization diagnostic under a paired-difference exchangeability assumption; exhaustive enumeration is used only for small paired samples, while larger samples use seeded Monte Carlo with a plus-one correction. Binary paired outcomes use the exact conditional McNemar test, and the complete family of comparison p-values is adjusted with Holm's step-down procedure. The The audit records 10 seed clusters; diagnostic small-cluster threshold not triggered; intervals remain descriptive. Ordinary population coverage requires an independent sampling design. These interpretations apply to every figure and caption that reports the quantities.

8 Traceability: Formal Objects, Code, Tests, and Evidence

8.1 Implementation Symbols Mapped to Methods

The method names form an executable vocabulary. The publication ledger requires every formal object to declare its equation, implementation symbol, test anchor, unit, estimand, dependencies, invariants, invariant-to-code/test anchors, and manuscript section. The validator resolves Python symbols with the AST, confirms test and section paths, checks equation labels, resolves every invariant anchor, and validates the composition edges. A passing status means that the declared references are resolvable. Scientific assumptions and population validity require additional evidence.

The generated registry currently reports 24 formal objects, 6 composition edges, and 9 audited claims (status: `pass`). The table is injected from `data/paper_audit.yaml` and serialized as `output/reports/formalism_traceability.json`; it is therefore regenerated with the same command that hydrates all manuscript variables.

Formal object	Equation	Executable anchor	Test anchor	Invariant anchors	Unit / estimand
`shared_tokenizer`	`eq:tokenizer`	`src/module.py::tokenize`	`tests/test_module.py`	deterministic tokenization	token sequence / normalized alphanumeric tokenization
`typed_stage_composition`	`eq:typed-composition`	`src/pipeline.py::Pipeline`	`tests/test_pipeline.py`	ordered stage handoffs, append before consolidation	stage invocation / ordered typed handoffs from transcript and query to answer, citations, and memory decision
`compiler_selection`	`eq:compiler-selection`	`src/compiler/mipro.py::MIPROCompiler`	`tests/test_mipro.py`	deterministic candidate order, complete trace, first maximum tie rule	candidate configuration / first stable-order maximizer of the finite-set mean composite score
`token_f1_set`	`eq:token-f1`	`src/compiler/metrics.py::token_f1`	`tests/test_metrics.py`	bounded metric, empty-empty is one	example / set-based lexical answer overlap F1
`bm25`	`eq:bm25`	`src/retrieval/lexical.py::BM25Index`	`tests/test_lexical.py`	nonnegative idf, zero score exclusion, deterministic ties	query-document score / Okapi BM25 score under the shared tokenizer
`hashing_cosine`	`eq:hash-cosine`	`src/retrieval/semantic.py::embed, src/retrieval/semantic.py::cosine`	`tests/test_semantic.py`	deterministic embedding, zero vector similarity zero	query-document similarity / cosine similarity of signed SHA-1 character-trigram vectors
`weighted_rrf`	`eq:rrf`	`src/retrieval/fusion.py::rrf_fuse`	`tests/test_fusion.py`	zero based rank, first duplicate occurrence, deterministic ties	candidate chunk / weighted reciprocal-rank fusion score
`mmr`	`eq:mmr`	`src/retrieval/mmr.py::mmr_rerank`	`tests/test_mmr.py`	lambda one relevance order, deterministic ties, bounded selection	candidate ordering / greedy relevance-diversity selection over fused candidates

Formal object	Equation	Executable anchor	Test anchor	Invariant anchors	Unit / estimand
inspector_relevance	eq:inspector-relevance	src/stages/inspector.py::_relevance	tests/test_inspector.py	bounded relevance, empty query zero	transcript record / fraction of distinct query tokens present in a record
inspector_partition	eq:inspector-partition	src/stages/inspector.py::Inspector._coarse_block_filter	tests/test_inspector.py	consecutive partition, at least one block	transcript block / summed record relevance used for depth-one coarse filtering
prf_expansion	eq:prf	src/retrieval/expansion.py::expand_query	tests/test_expansion.py	distinct feedback terms, frequency then alphabetical order	expanded query / deterministic feedback-term augmentation from top BM25 windows
recency_bonus	eq:recency	src/stages/retriever.py::Retriever._apply_recency	tests/test_retriever.py	zero weight noop, recalled window exemption, deterministic ties	candidate score / position-scaled ranking bonus for non-negative transcript indices
evidence_budget	eq:evidence-budget	src/stages/retriever.py::estimate_tokens	tests/test_retriever.py	deterministic token count, budget threshold	ranked evidence list / shared alphanumeric-token estimate and inline/file delivery decision
grounding_support	eq:grounding	src/compiler/metrics.py::citation_support_proxy	tests/test_metrics.py	bounded metric, no citation zero	evaluation example / citation-support proxy combining expected-index hit and cited-excerpt token-F1
evidence_efficiency	eq:efficiency	src/compiler/metrics.py::efficiency_score	tests/test_metrics.py	bounded metric, floor at zero	evaluation example / bounded structural reward from ranked-window count
citation_precision	eq:citation-precision	src/compiler/metrics.py::citation_precision	tests/test_metrics.py	bounded metric, no citation zero	evaluation example / fraction of cited source indices that belong to the declared expected source set
source_recall	eq:source-recall	src/compiler/metrics.py::source_recall	tests/test_metrics.py	bounded metric, empty expected neutral	evaluation example / fraction of the declared expected source-index set represented in citations

Formal object	Equation	Executable anchor	Test anchor	Invariant anchors	Unit / estimand
composite_metric	eq:composite	src/compiler/metrics.py::harness_metric	tests/test_metrics.py	bounded metric, fixed weights sum to one	evaluation example / weighted sum of lexical answer, citation-support, and structural efficiency components
stratified_cluster_resampling	none	src/evaluation/statistics.py::clustered_bootstrap_mean	tests/test_statistics.py	explicit seed, finite values, failure denominator preserved	seed-stratum cluster / finite-set mean with within-stratum seed-cluster resampling
research_dataset	none	src/evaluation/dataset.py::build_research_dataset	tests/test_scaling.py	balanced categories, unique ids, stable seed replay	generated example / balanced finite synthetic task-family observation
evaluation_ledger	none	src/evaluation/accounting.py::ledger_row	tests/test_accounting.py	contiguous sequence, failure reason, deterministic jsonl	example-method evaluation row / complete requested-observation accounting including failures
adjudicated_quality	none	src/evaluation/quality.py::build_quality_report	tests/test_quality.py	pilot confirmatory disjoint, paired condition completeness, digest bound certificate	question-condition-output with two blinded expert raters and expert adjudication / adjudicated correctness rate with declared secondary quality and calibration metrics
quality_primary_estimand	eq:quality-primary	src/evaluation/quality.py::build_quality_report	tests/test_quality.py::test_quality_report_requires_two_judges_and_adjudication	pilot excluded, question level pairing, fail closed missing data	held-out confirmatory question / question-weighted rate of adjudicated default-harness correctness score 2
operational_readiness	none	src/evaluation/production_readiness.py::build_production_readiness_report	tests/test_production_readiness.py	failed rows block readiness, controls required, model pinning	service observation and operational control / declared latency, reliability, cost, privacy, model-pinning, and operational-control thresholds

8.2 Composition Edges and Dataflow Semantics

The graph makes three different relations explicit. First, the Inspector's lexical relevance and coarse partitioning define the candidate corpus presented to BM25 and the hashing-cosine lane. Second, ranked candidates are fused, optionally recency-adjusted and diversity-reranked, then passed through the evidence budget contract. Third, answer token-F1, citation-support, and evidence-count efficiency compose into the fixed per-example metric. These are dataflow dependencies. A downstream score can support a causal claim only when a separately designed intervention identifies the upstream component.

8.3 From Code and Evidence to Source-Bound Prose

`src/signatures/` defines typed payloads; `src/stages/` and `src/retrieval/` implement the stage functions; `src/compiler/metrics.py` defines score components; `src/evaluation/` constructs datasets, ledgers, and statistical summaries; `src/figures/` renders figures from validated aggregates; and `src/manuscript_variables.py` emits the quantitative tokens. The project scripts remain thin orchestrators. Generated JSON, JSONL, figures, captions, and PDF are downstream evidence artifacts, while the registry is the bridge that makes the formalism-to-code relationship inspectable before rendering.

The evidence boundary remains explicit: deterministic contracts are tested locally; synthetic evaluations describe the declared finite generated set; external benchmark rows identify their input by size and SHA-256; and neural rows retain model digests and generation options. A resolved anchor is necessary for scholarship. Semantic validity, causal identification, and generalization require their own evidence.

9 Reproducible Research Integration: `docxology/template` Build Lifecycle

THALIA is built inside the `template/` reproducible-research substrate [Friedman, 2026], which supplies its provenance and self-documentation layer. Two integration points matter: the `AGENTS.md`/`SKILL.md` documentation duality and the eight-stage build lifecycle.

9.1 Documentation Duality: README.md, AGENTS.md, and SKILL.md

Each THALIA stage carries a machine-readable `SKILL.md` aligned with the Model Context Protocol, so an agent can discover and invoke a stage's capability through structured introspection with explicit API contracts. For example, `skills/thalia-inspector/SKILL.md` declares the Inspector's purpose, inputs, outputs, invocation snippet, and constraints (the Inspector loads selected windows; it caps delivery at `max_windows` windows; grep patterns are grounded in query terms). The same discipline applies to the Retriever, Reasoner, Memory Gate, and Compiler skills.

9.2 Mapping the Eight-Stage Build Lifecycle

The `template/` eight-stage build pipeline maps directly onto the harness's compile-and-validate lifecycle:

`template/` stage	THALIA equivalent
Environment sanitization	Dependency lock + (optional) LM/API validation
Test execution (Zero-Mock)	Stage suite on generated examples + real SQLite; 90% `src/` gate
Analysis script invocation	`run_harness_eval.py`, `run_compiler.py`
Pandoc/XeLaTeX rendering	Manuscript + figure generation
SHA-256 + steganographic hash	Compiled-config / artifact fingerprint
Structural validation	Signature schema validation (`Signature.validate_inputs`)
LLM-assisted review	GEPA trace-driven skill review (optional)
Provenance chain	Git-tracked compiled trace + generated tokens

9.3 Inherited Infrastructure and THALIA Extensions

THALIA borrows the Zero-Mock policy (no `MagicMock`; real computations and real SQLite throughout `tests/`), the coverage gate (>= 90% on `src/`), the thin orchestrator pattern (scripts coordinate; `src/` implements), and deterministic outputs with fixed structure. It adds the agentic-harness layer on top: typed stage signatures, the lexical-anchored retrieval lane, the episodic-first memory subsystem, and the bounded Compiler. The result is a harness whose *behaviour is auditable from its build*: configuration drift, deleted results, and out-of-sync manuscript numbers trigger a failing gate before a green build. The build establishes executable traceability and artifact freshness. Model competence and scientific generalization require separate evidence.

10 Active Inference Interpretation: Information, Belief, and Control

The harness has natural correspondences with the Free Energy Principle and Active Inference [Friston, 2010], relevant to the broader research programme at the intersection of Active Inference, computational biology, and cognitive security. This mapping is **interpretive scaffolding**. The paper's evaluation results (sec. 11) stand on the executable contracts and recorded aggregates. The correspondence connects the architecture to a research lineage. It supplies suggestive framing; a falsifiable Active-Inference derivation would require a separate generative model and objective.

- **Perception — the Inspector.** Selective attention over external context is a form of active sampling: the RLM-style grep/peek loop chooses which slices of the environment to bring into the inference, analogous to epistemic foraging that reduces uncertainty about hidden causes (sec. 6.2).
- **Action — the Reasoner and tool dispatch.** The ReAct-style deterministic read→integrate→retry route selects information-gathering and answer-producing actions, analogous to policy selection under expected free energy: pragmatic value (answer the query) traded against epistemic value (gather more evidence) (sec. 6.4).
- **Memory — the Memory Gate.** The episodic-first architecture mirrors the distinction between fast episodic (hippocampal) traces and slow consolidated semantic memory. The consolidation gate is a precision-weighting safeguard: it prevents low-precision derived beliefs from overwriting retained evidence, an engineering response inspired by the consolidation-fault mode documented in the cited study (sec. 6.5, [Zhang et al., 2026]).
- **Learning — the Compiler.** MIPRO search and GEPA reflective evolution optimise a declared metric over a finite evaluation set. This recalls the learning/inference distinction. The compiler optimizes configuration choices; parameter learning, an Active-Inference objective, and generalization require separate evidence (sec. 6.6).

Of these, the memory correspondence is the load-bearing engineering link: the precision-weighting reading of the consolidation gate ([Zhang et al., 2026]) is the interpretive rationale for the episodic-first invariant and what sec. 15.4 ablates. The other three remain framing elements in this manuscript.

11 Evaluation: Evidence Tiers, Estimands, and Scope

The evaluation uses distinct evidence tiers. A fixed LongMemEval-style dataset, a deterministic extractive language model, and a composite metric computed in pure Python provide the reproducible contract tier. Recorded Ollama/OpenRouter rows and the external LongMemEval slice provide model-dependent transfer diagnostics. The goal is to validate the *machinery* and state the estimand and evidence boundary for each tier.

This spine preserves the stable evaluation anchor and the two scope boundaries called out by the project guide. The renderer places the adjacent `07a_` through `07e_` files beneath this heading as evaluation subsections, so the table of contents presents one coherent evaluation branch.

The context-poisoning module reports exact cases and finite-set rates; these are exact cases, not population rates. Its legacy `harness_auditable` outcome only records whether a cited excerpt lexically contains the supplied gold value; it is a cited-excerpt/source-exposure proxy, not a recoverable source record or claim-level auditability test. The optional neural rows appear only when a real model is available. Model-quality status requires a dedicated quality protocol. Usefulness, factuality, calibration, and preference remain distinct constructs from execution integrity and lexical overlap [Min et al., 2023].

Modular evaluation files:

- `07a_evaluation_setup_headline_and_scaling.md`
- `07b_evaluation_real_benchmark_and_bottleneck.md`
- `07c_evaluation_baseline_neural_and_context_poisoning.md`
- `07d_evaluation_retrieval_learned_embedder_and_noise.md`
- `07e_evaluation_compiler_added_methods_and_summary.md`

11.1 Experimental Design and Statistical Estimands

11.1.1 Synthetic Dataset and Metric Contract

`src.evaluation.build_demo_dataset` defines 6 examples, one per task category (temporal, preference, code, multi-session, knowledge-update, single-session). Each session embeds the gold answer in one discriminative line plus distractors, with annotated gold source indices. The composite metric (`src/compiler/metrics.py`) is the convex combination in eq. 18: **answer token-F1** (set-based token-F1 vs. the gold answer, weight 0.6), **citation-support proxy** (the citation-support proxy, weight 0.3), and **efficiency** (a ranked-window count reward, weight 0.1). The exact component definitions and edge cases are specified once in sec. 7.1.

The dataset is intentionally diagnostic and compact. It checks whether each stage receives the right kind of pressure: the Inspector must recover a line from history; the Retriever must rank that line under lexical, surrogate-semantic, and hybrid modes; the Reasoner must cite it; the Memory Gate must avoid unwanted consolidation; and the Compiler must evaluate a repeatable metric. This makes the evaluation a methods test. The fixed examples are small enough for every run to be inspected, and the expected source indices make reachability failures concrete. Its labels cover expected source indices and lexical answer strings; adjudicated quality and atomic factuality labels belong to the separate quality protocol [Min et al., 2023].

Three controls keep the protocol honest. First, every result number in this section is generated from `src/manuscript_variables.py`. Second, the headline comparison uses the default slack retrieval budget, while the diagnostic sweeps state when they switch to a tight budget. Third, the local semantic lane is always described as a hashing surrogate; learned dense retrieval is labelled as a separate evidence tier.

11.1.2 Headline Deterministic Results

On the default configuration (top-k 8, max-evidence 5, RRF $k = 60$), the headline metric components are reported in tbl. 1.

Table 1: Headline metric components on the 6-example dataset under the default configuration.

Component	Value
Composite	0.621
Answer token-F1	0.477
Citation-support proxy	0.830
Efficiency	0.854

Citation-support proxy (0.830) and efficiency (0.854) are high on this declared finite set under their operational proxies: the harness usually cites the expected source and uses a small evidence set. Factuality and user value require separate adjudicated outcomes. Answer token-F1 (0.477) is bounded because the *deterministic* extractive LM returns a whole transcript line, such as an assistant confirmation, while the annotation contains a minimal span. Token-F1 against the terse gold string measures lexical agreement with the annotation. Adjudicated semantic correctness is a separate outcome. This behaviour belongs to the selected Reasoner and metric; sec. 11.3.2 tests the model boundary directly while holding the other stages fixed.

11.1.3 Scaling, Uncertainty, and Inferential Limits

The 6-example set above is a determinism showcase. To characterize dispersion in the machinery we add a **seeded, balanced large dataset** (300 examples, 50 per task category) from `src.evaluation.build_large_dataset`: a fixed seed makes it bit-reproducible, so scale and determinism coexist on this path. On it the composite is 0.828 with a 95% descriptive bootstrap interval of [0.822, 0.834], and fig. 7 shows per-category answer token-F1 with bootstrap intervals — the first view of the harness with measured dispersion across category rows. The caveat is explicit: this set is *realistic-but-synthetic* (templated entities; observational transcripts form a separate tier), so it measures the machinery's scaling and variance. An external benchmark slice is the natural next step. What it buys is statistical shape: the strongest and weakest task categories show visible dispersion (the middle categories overlap). This is a descriptive pattern with no significance estimand, and it adds structure the 6-example diagnostic could only hint at.

The precision audit extends this single-seed view with 6 task categories and four balanced per-category sample sizes (25, 50, 100, 250), across 10 independent fixed seeds: 25500 unique synthetic examples and 76500 method evaluations. It evaluates

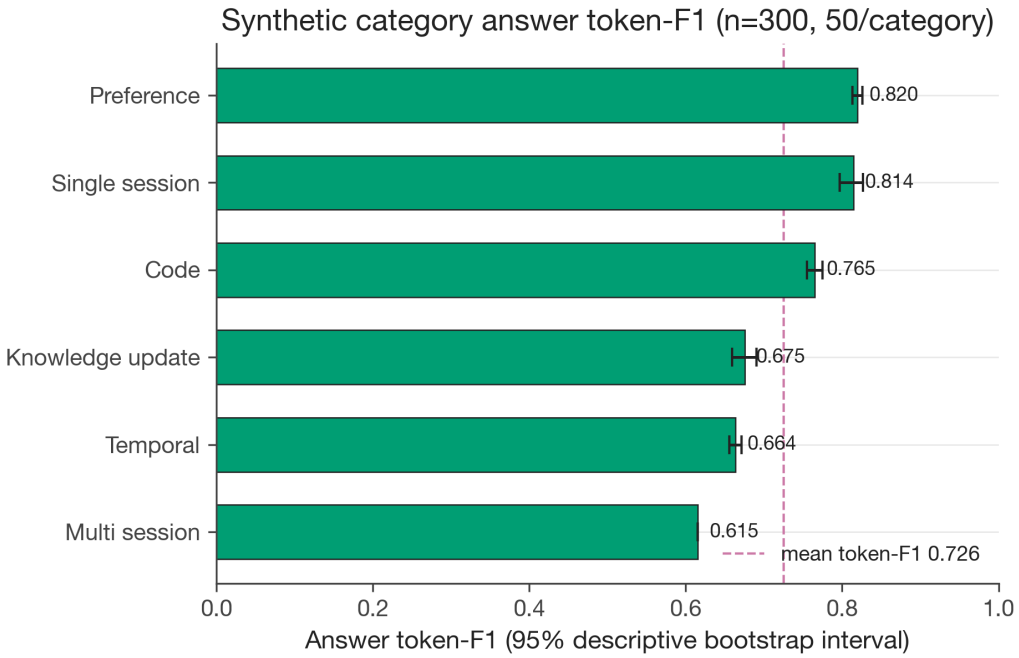


Figure 7: Answer token-F1 per task category on the seeded synthetic 300-example dataset (50 per category), with 95% descriptive bootstrap intervals; the dashed line marks the mean category estimate. The unit is a generated evaluation example scored by the deterministic extractive responder. A fixed seed makes the generated-task dispersion and scaling reproducible; real-world answer quality and learned-model transfer require separate evidence.

lexical-only, surrogate-semantic-only, and hybrid retrieval on paired examples across 4 deterministic difficulty strata. The audit records descriptive_cluster_resampling_interval within declared strata, seed-cluster precision checks (diagnostic small-cluster threshold not triggered; intervals remain descriptive), 95% Wilson intervals for binary outcomes, exact paired binary tests, and one named Holm family of size 40 spanning every sample-size row, overall paired comparison, and stratum-level paired comparison. A per-example ledger records 0 failed method rows and configuration/dataset fingerprints. This strengthens accounting for finite synthetic variation. Generalization beyond the generated task family requires external data.

11.1.3.1 Estimands, Resampling, and Paired Inference The primary continuous estimand is the mean composite score over the finite generated evaluation examples, with the example as the observation unit and the seed as a design cluster. Method comparisons are paired: hybrid and each baseline are evaluated on the same stable example IDs, and the reported delta is hybrid minus baseline. Intervals for marginal method means resample within each declared difficulty stratum; intervals for paired deltas resample complete seed clusters while preserving the within-seed pairing. The paired point estimate is the mean of the complete seed-cluster deltas, so the estimand and its interval use the same cluster-level unit even when cluster sizes differ. The per-stratum forest shows the comparison estimand directly. Subtracting two unrelated marginal limits would define a different quantity. Percentile bootstrap intervals follow the non-parametric resampling tradition of [Efron, 1979].

The seed-cluster intervals are deliberately descriptive: the publication audit uses a finite number of independent seed clusters. The manuscript reports that design limitation; independent replication remains a separate requirement. The quality protocol later uses question clusters for the same reason: paired condition contrasts use the question as the observation unit, with model/condition rows retained as paired records.

Binary probe outcomes are summarized with Wilson score intervals [Wilson, 1927] and paired with the exact McNemar binomial test [McNemar, 1947]. All named paired tests in this audit, including the stratum slices, enter the same predeclared Holm step-down family [Holm, 1979]; adjusted values provide descriptive control of the declared family. The precision planner is an explicit design aid based on observed dispersion and target half-width. Population power requires a separate design.

These estimates belong to separate evidence tiers. The seeded examples test composable machinery under controlled variation; LongMemEval_S tests transfer to an external benchmark; and neural rows test one recorded local model/configuration. Each tier

retains its own estimand and evidence boundary. Synthetic descriptive intervals describe the seeded machinery tier. Neural and external-benchmark performance require their own evidence. A reproducible harness run establishes that the measurement path is functioning. A quality claim requires a validated target construct, appropriate labels, and comparisons defined by the separate quality protocol.

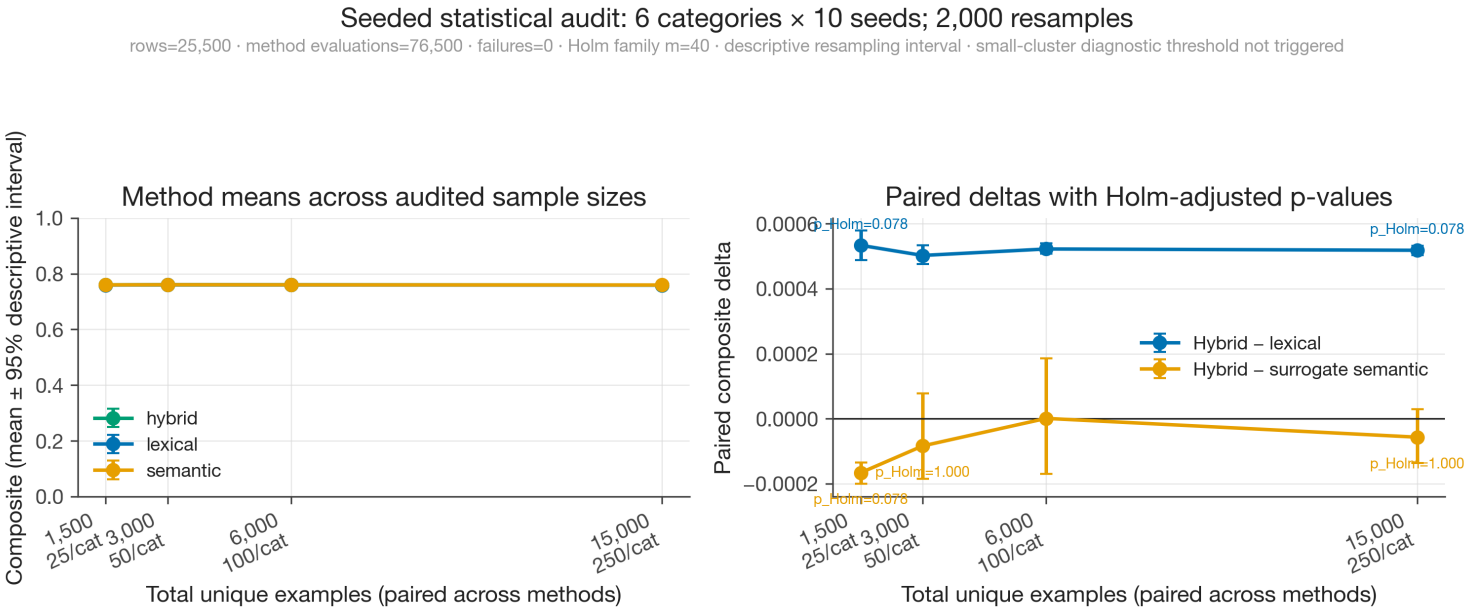


Figure 8: Composite means and paired deltas across audited synthetic sample sizes. Fixed-stratum intervals summarize method means, and paired hybrid-minus-baseline intervals respect 10 seed clusters. The horizontal axis is the ledger's total unique-example count; the banner exposes rows, method evaluations, failures, and the complete Holm family. All rows come from the same deterministic evaluation ledger. Intervals are descriptive cluster-resampling diagnostics; methods use the deterministic extractive responder and hashing surrogate.

The pre-registered sensitivity extension is a separate ten-seed design from the publication audit: it covers 600 unique generated examples across 10 seed clusters and 1800 method evaluations. Hybrid-minus-lexical has mean paired delta 0.000 (sign-flip randomization p=1.000), and hybrid-minus-surrogate-semantic has mean paired delta 0.000 (sign-flip randomization p=1.000). These are finite generated-task sensitivity results; the small-cluster warning does not apply for this extension, and neither it nor the headline is a population estimate.

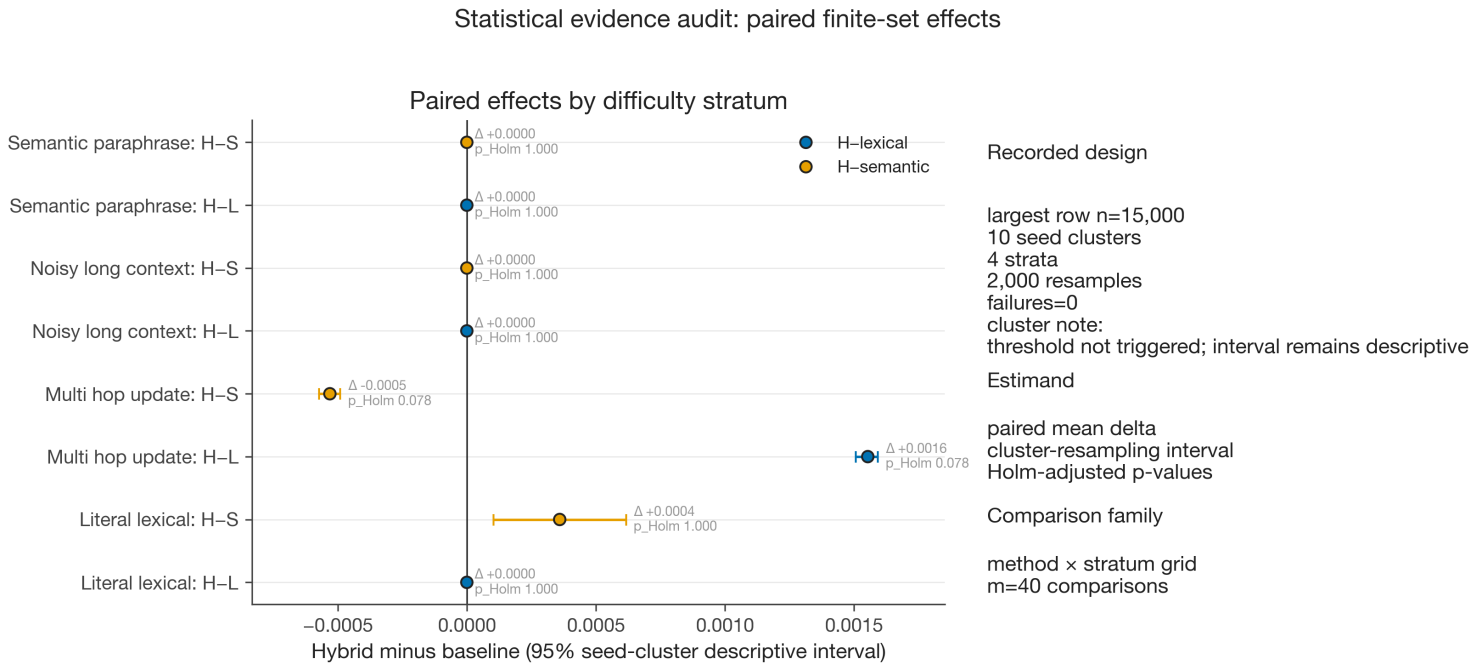


Figure 9: Paired composite-score effects at the largest audited synthetic row: hybrid-minus-lexical and hybrid-minus-surrogate-semantic within each difficulty stratum, with whole-seed-cluster 95% descriptive intervals and Holm-adjusted p-values. Delta and p-value labels, the zoomed observed range, and the zero reference line expose the small effects. The accounting panel reports 10 seed clusters, 4 strata, the resampling design, failures, and the complete family of size 40. The finite extractive-plus-hashing study is a descriptive experiment; its finite cluster design limits population interpretation.

11.2 External Benchmark Transfer and Bottleneck Diagnosis

11.2.1 LongMemEval_S Transfer Slice

Long-term conversational-memory evaluation has multiple valid target constructs. LoCoMo evaluates very long conversations through question answering, event summarization, and multimodal dialogue [Maharana et al., 2024]; LongMemEval organises a complementary set of information-extraction, multi-session, temporal, knowledge-update, and abstention tasks [Wu et al., 2025]. THALIA uses LongMemEval_S for this transfer lane because its current adapter binds the benchmark's labelled answer sessions to the existing evidence-window contract. The comparison is a transfer diagnostic within that adapter; it is not a cross-benchmark ranking.

The synthetic results test machinery on the generated task family. The transfer check is an external benchmark. We run the harness over **LongMemEval_S** [Wu et al., 2025] — 500 questions over long, multi-session haystacks — mapped onto the same `EvalExample` contract (`src.evaluation.longmemeval`), scored with **answer token-F1** and two separate evidence diagnostics: a citation-support proxy for cited excerpts and binary **session reachability** (did the surfaced evidence include a line from the gold session?). Because the benchmark labels evidence at the session level, reachability is an **upper bound** on stricter line-level attribution. Both diagnostics measure exposure or citation support; entailment and adjudicated answer quality require separate labels. This is external data, so these results are reproducible from *code plus the downloaded benchmark*. The benchmark dependency is recorded as a deliberate boundary of the self-contained thesis.

The external-benchmark comparison (fig. 10) is deliberately a transfer diagnostic. Extractive answer token-F1 is lower in this retained slice at 0.048 [0.043, 0.055] — versus the synthetic 0.726 [0.716, 0.736] on the same token-F1 component — indicating that the synthetic construct is easier than this retained slice and that the extractive default is below the target usefulness of this long-context task. The result bounds answer phrasing under the selected Reasoner; the localisation below assesses evidence reachability separately. Three findings follow, now with recorded benchmark statistics (500 questions, 95% descriptive bootstrap intervals):

1. **A real neural model changes this retained slice.** Dropping `gemma3:4b` into the Reasoner (50 questions, the Inspector windowing first) raises answer token-F1 to 0.110 [0.076, 0.148] — a descriptive about 2.3× point-estimate contrast. The

extractive and neural rows have different retained sample sizes, so they form separate backend summaries. Their bootstrap intervals describe the retained slices and carry no interval-overlap decision rule. The synthetic set showed a tie (sec. 11.3.1); this retained external-benchmark slice is harder and model-dependent.

2. **Evidence reachability is the bottleneck in this retained slice; ranking loss is smaller once evidence is exposed.** The citation-support proxy is 0.430 [0.409, 0.450]; the separate extractive session-reachability diagnostic is 0.678 [0.636, 0.718]. Answer token-F1 in this evaluation is constrained by what is surfaced. We expected a learned dense embedder to be the fix; sec. 11.2.4 shows the committed full-haystack ranking reaches a gold evidence line in 1.00 of the retained cases. In this diagnostic, the larger observed loss is upstream, in the Inspector's aggressive context-narrowing, and widening that window is associated with recovered session reachability and answer token-F1.
3. **The evidence scope.** 0.110 is a low absolute number: this retained run combines deterministic selection with model-dependent generation; the run is a THALIA transfer diagnostic. A hosted-LLM memory-system comparison on LongMemEval_S would be a separate study. It establishes that the *machinery* runs on external long-token haystacks, that the architecture's plug-in points behave as designed under real load, and that the observed failures motivate improvements to context policy and answer quality.

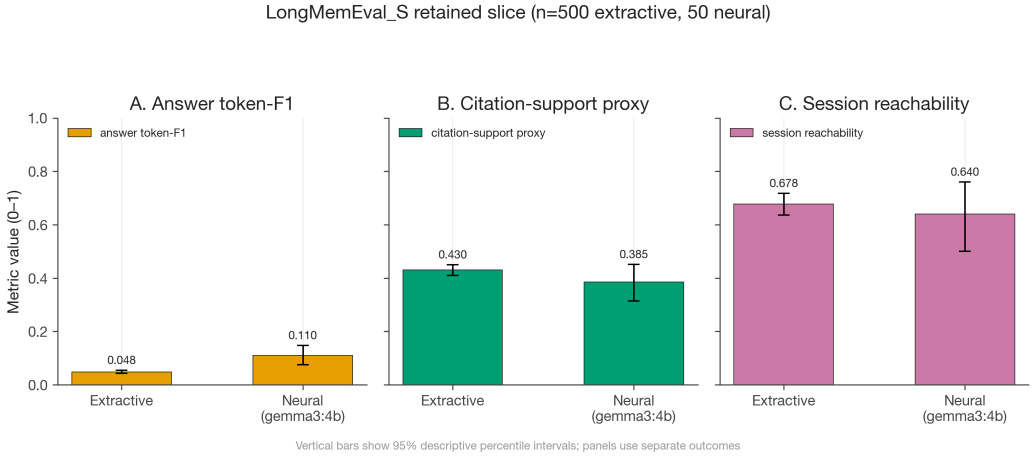


Figure 10: LongMemEval_S in separate answer token-F1, citation-support-proxy, and optional session-reachability panels for the extractive and neural Reasoners, with 95% descriptive bootstrap intervals (500 extractive questions, 50 neural). Retained slices have different sample sizes, so each interval summarizes its own backend observations. Session reachability is an upper-bound exposure measure because any line from the gold session counts; citation-support measures source overlap and requires separate entailment labels. Benchmark SHA-256, retained status complete, and model digest are recorded in `data/longmemeval_results.json`; neural values are model-dependent and carry host-specific provenance.

11.2.2 What the Transfer Test Identifies

The external slice is the manuscript's closest approach to a model-quality measurement. It remains a transfer diagnostic; a quality certificate requires the separate adjudication protocol. Its answer token-F1 asks whether the answer shares normalized tokens with the benchmark answer; its citation metric asks whether evidence reaches the labelled session. Neither asks a quality judge whether the answer is complete, faithful, well calibrated, or useful. Factuality evaluation decomposes generated text into atomic claims and supplies labels for factual support [Min et al., 2023].

Citation evaluation provides a second validity boundary. ALCE's statement-level citation recall and precision use entailment judgments over cited passages [Gao et al., 2023], while ARES evaluates context relevance, answer faithfulness, and answer relevance as distinct RAG dimensions [Saad-Falcon et al., 2024]. The THALIA aggregate currently records source-index precision and recall in the campaign schema and displays their deterministic analogue in the metric-construct-audit heatmap in the formalism section. Those measures expose citation-set behavior; they do not replace the planned expert labels for entailment, completeness, or usefulness.

The benchmark also separates two failure surfaces that long-context work has shown can diverge. A model may receive the right evidence but underuse it, or retrieval can leave relevant evidence outside the model's context [Liu et al., 2024]. THALIA's Inspector and Retriever diagnostics make that separation visible for the retained slice. The interventions are local, the model

is fixed within each recorded run, and the neural rows form separate backend summaries across the retained questions. The appropriate conclusion is conditional: the harness identifies a plausible bottleneck and a testable intervention, while model quality awaits stronger answer-correctness labels and broader comparators.

This is a construct-validity boundary. A quality claim needs an answer-level target and a label protocol that distinguishes atomic factual support from coverage, usefulness, and safe abstention. FActScore motivates the atomic-claim distinction [Min et al., 2023]; TruthfulQA shows why truthfulness deserves an explicit failure surface [Lin et al., 2022]; and SelfCheckGPT demonstrates that black-box consistency checks are themselves a different diagnostic from adjudication [Manakul et al., 2023]. The result is a measurement stack: reachability asks whether relevant evidence was exposed, correctness asks whether the answer is right, faithfulness asks whether its claims are supported, and usefulness asks whether it serves the task. A single lexical-overlap number identifies one construct within this stack. This multi-faceted design also follows long-form QA evaluation work showing that automatic metrics require answer-level judgments of properties such as comprehensiveness [Xu et al., 2023].

11.2.3 Expert-Rated Quality Gate

The repository now specifies the quality study as a separate lane. The quality protocol keeps an exploratory, question-type-stratified pilot separate from the held-out confirmatory set. After pilot review, the rubric, primary adjudicated correctness estimand, thresholds, missing-data rules, safety criteria, and confirmatory question identifiers are frozen in the experiment-plan source of truth.

The live matrix compares Gemma and Hermes under the default THALIA harness, a no-harness full-context baseline, and an oracle-evidence baseline. The latter exposes the labelled evidence session without adding a gold answer field, so evidence exposure remains distinct from answer quality. Each output is retained with its raw answer, citations, evidence indices, confidence response, latency, failure state, model digest, generation options, configuration fingerprint, and code fingerprint outside the repository. Publication annotations use two independent expert raters who receive packets with model and condition identifiers removed; a blinded expert adjudicator resolves disagreements before a row enters analysis.

The repository also contains a pinned local-LLM judging implementation (Qwen and Llama evaluators with a separate Phi adjudicator) for engineering diagnostics of packet construction, rubric parsing, and resumable evidence accounting. Those labels are explicitly exploratory: they estimate behavior under a specified judge stack. The promotion function excludes these labels from publication. The publication certificate therefore requires the expert-human mode, with the rater and adjudicator identities recorded as pseudonymous identifiers and the annotation ledgers bound by digest. All judge packets remove target model and condition identifiers, use the same condition-neutral question and source packet for every paired output, and keep analyst citations/source indices separate from the blinded presentation. This separation makes the faithfulness label a claim about support by supplied source material. The exploratory LLM implementation also records evaluator/model and prompt digests because LLM-as-judge results can exhibit self-enhancement, verbosity, and position biases [Zheng et al., 2023, Wang et al., 2024, Shi et al., 2025].

The planned analysis treats adjudicated correctness as the primary outcome and completeness, faithfulness, usefulness, abstention, critical safety failures, calibration, latency, and failure rate as secondary outcomes. It reports question-clustered paired uncertainty, exact paired binary contrasts, question-type strata, and weighted inter-rater agreement. FActScore's atomic-claim distinction motivates keeping faithfulness and correctness explicit; token overlap remains a lexical proxy [Min et al., 2023], while long-context evidence exposure remains a separate construct [Liu et al., 2024]. Until the study is annotated, adjudicated, thresholded, and explicitly accepted, no quality aggregate is promoted and the manuscript reports no model-quality pass. Production readiness remains a separate operational gate for latency, cost, privacy, rollback, monitoring, drift, and red-team evidence.

The rubric uses ordinal scores with three levels. For each answer, each rater assigns correctness, completeness, evidence faithfulness, and usefulness as 0 (incorrect, absent, or unusable), 1 (partly correct, incomplete, weakly supported, or conditionally useful), or 2 (fully correct, complete for the stated task, supported, or useful as applicable). Abstention appropriateness and critical safety failure are binary outcomes. Independently, every deterministic atomic claim is labelled `supported`, `unsupported`, `uncertain`, or `not_applicable`; the last label is excluded from the support-rate denominator. This keeps an unsupported factual assertion visible alongside a high answer score. The citation-overlap proxy continues to require separate entailment labels.

The confirmatory primary estimand is explicit. Let Q_C be the 440 held-out questions, H_q the adjudicated default-harness answer for question q, and $Y_q = \mathbb{1}[H_q$ receives correctness 2]. The primary estimate is

$$\hat{\theta}_{\text{correct}} = |Q_C|^{-1} \sum_{q \in Q_C} Y_q. \tag{19}$$

with a Wilson interval for the descriptive rate; baseline contrasts pair the same question under the no-harness and oracle-evidence conditions and use question-level clustered bootstrap intervals plus exact paired binary tests. A failed generation, missing confidence response, missing judge label, missing adjudication, digest mismatch, pilot leakage, or incomplete condition pair invalidates the affected report and remains un-imputed. A complete run that misses a frozen threshold remains in the explicit `complete_not_promoted` state. Artifact generation preserves that state.

The planned matrix contains 6 model/condition rows across 500 benchmark questions: an exploratory pilot of 60 questions, followed by a held-out confirmatory set of 440. Every output is independently labelled by 2 blinded expert raters before expert adjudication. Confidence is treated as a forecast to be assessed against adjudicated correctness: Brier score, expected calibration error, reliability curves, and selective accuracy follow the calibration and risk–coverage framing in [Guo et al., 2017, Geifman and El-Yaniv, 2017]. Agreement statistics are reported as reliability diagnostics. Agreement measures label consistency; construct validity requires additional evidence [Artstein and Poesio, 2008]. The primary outcome is the frozen protocol's `correctness_full_rate` on the confirmatory set. Pilot labels may revise the rubric and remain outside that estimate.

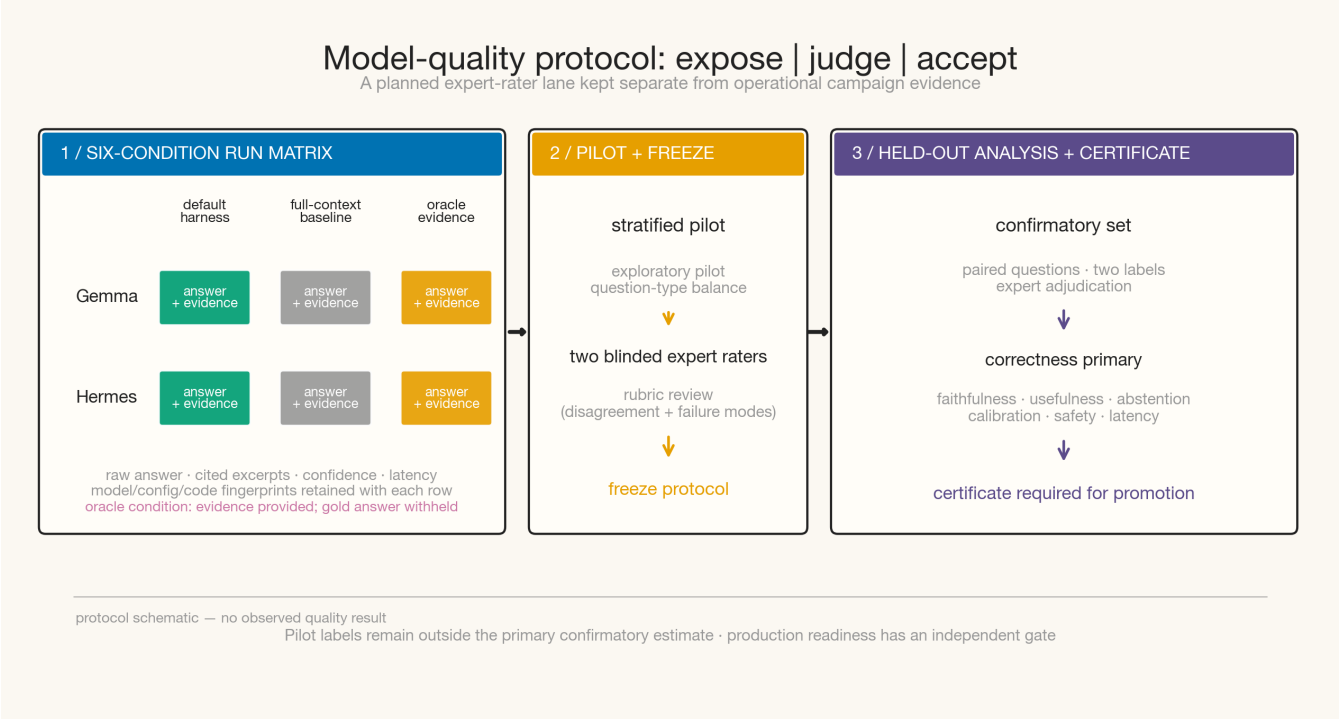


Figure 11: Planned model-quality protocol. Gemma and Hermes outputs cross the THALIA default harness, full-context baseline, and oracle-evidence baseline before a stratified exploratory pilot, rubric freeze, two blinded expert-rater passes, adjudication, and a fail-closed certificate. The pilot contains 60 questions and remains outside the 440-question confirmatory estimand; the oracle condition provides evidence while withholding the gold answer. The local-LLM implementation is an engineering diagnostic. The protocol schematic carries no observed quality result, and production readiness has an independent gate.

11.2.4 Context-Narrowing Bottleneck

The sharp decline in sec. 11.2.1 motivates a diagnostic comparison — swap the lexical lane for a learned dense retriever. The committed aggregate is narrower: over 40 questions, full-haystack ranking hit@10 is 1.00, while the optional per-mode learned-embedder probe is only cited when `ranking_recall_by_mode.available` is true in the regenerated bottleneck artifact. On LongMemEval_S, retrieval *ranking* contributes less to the observed loss in the retained aggregate. The loss is upstream in this retained slice: over 40 questions, full-haystack ranking recall is 1.00 while the **Inspector's** window recall — whether its grep-based narrowing even surfaces a gold line for the Retriever to rank — is only 0.60. In this slice, the RLM-style narrowing accounts for

the larger observed loss before ranking can operate. This is a retained-slice bottleneck diagnosis; causal ranking claims require an intervention.

So the fix is to relax the narrowing, and fig. 12 measures it. Raising the Inspector's `max_windows` from 15 to 100 lifts the citation-support proxy from 0.422 to 0.575 — but extractive *answer token-F1* barely moves, consistent with the default extractive responder's answer-generation limitation even when evidence is shown. The two interventions compose in this retained subset: with a real Reasoner (`gemma3:4b`), the same widening converts recovered recall into answer token-F1, **raising it about 1.9×, from 0.086 to 0.165**. (This analysis uses a 30-question neural subset, so its absolute values sit slightly below the 50-question headline of sec. 11.2.1; the *relative* gain from relaxing the narrowing is the claim.) The actionable result is bounded: THALIA's default `max_windows=15` is tuned for compact contexts and produced the largest observed loss in this retained long-context diagnostic slice; widening the Inspector *and* using a neural Reasoner is the configuration associated with higher scores in this retained benchmark slice. The *surrogate* semantic lane left recall unchanged here. A learned dense embedder remains the appropriate tool for paraphrase coverage (sec. 11.4.2). The finding is a localized context-narrowing bottleneck supported by this retained slice.

The diagnosis yields an operational fix: the narrowing budget should *scale with the haystack*. This ships as an **adaptive Inspector** (`adaptive_windows`, opt-in, `src/stages/inspector.py`): it sets `max_windows` to roughly one window per 5 session lines, floored at the compact-context default. On the 6-example synthetic set this leaves every output **bit-for-bit unchanged** because compact contexts stay below the floor. On LongMemEval_S the fixed adaptive policy is associated with the observed recall in this retained slice, lifting observed neural answer token-F1 from 0.086 to 0.171: the manually selected `max_windows=100` result (fig. 12, star); the adaptive policy reaches its own observed value while retaining its ratio-based rule; the sweep maximum is not an input. The diagnosis becomes a one-flag method.

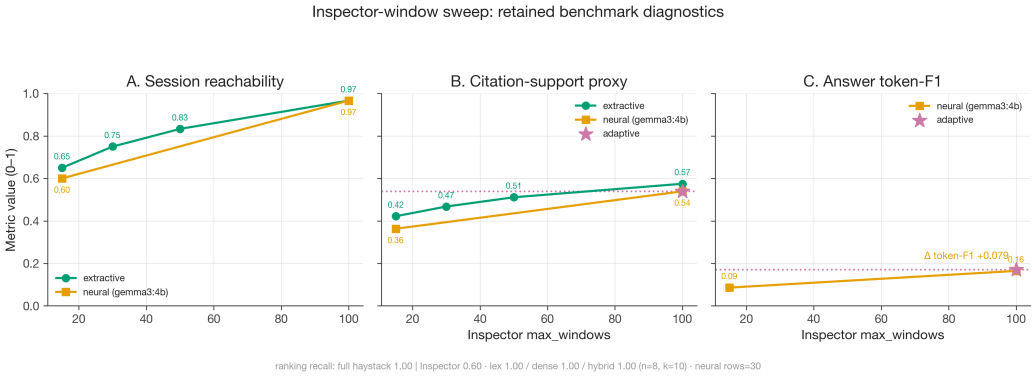


Figure 12: LongMemEval_S bottleneck diagnostic on 40 ranking/configuration observations and 90 neural question-configuration rows (30 questions per neural sweep setting). Separate small multiples show session reachability, citation-support proxy, and neural answer token-F1 during the Inspector `max_windows` sweep; the star marks the opt-in adaptive policy. Full-haystack ranking recall is 1.00 and Inspector window recall is 0.60. The unit is a benchmark question or declared configuration observation; intervals are descriptive where shown, model values depend on `gemma3:4b`, and the retained-slice diagnosis localizes the observed loss to context narrowing.

11.3 Baselines, Model Dependence, and Context-Poisoning Stress Tests

11.3.1 No-Harness Full-Context Baseline

A fair challenge asks what the 5-stage pipeline adds to a naive full-context baseline. The baseline passes the *entire* session to the LM and omits Inspector windowing, Retriever ranking, and citations. fig. 13 decomposes both conditions. On raw answer token-F1 they *tie* exactly (0.477 each): on this compact set the extractive LM retrieves the gold line under the declared token-overlap rule under both context conditions, so answer token-F1 is equal. The naive baseline earns zero citation support (it cites nothing) and zero efficiency (it dumps the full context), so its composite is only 0.286 against the pipeline's 0.621 — a lift of 0.334 supplied by citation support and efficiency. On this dataset the pipeline's measured value is provenance and bounded context; raw answer token-F1 remains bounded by the selected LM (sec. 11.3.2).

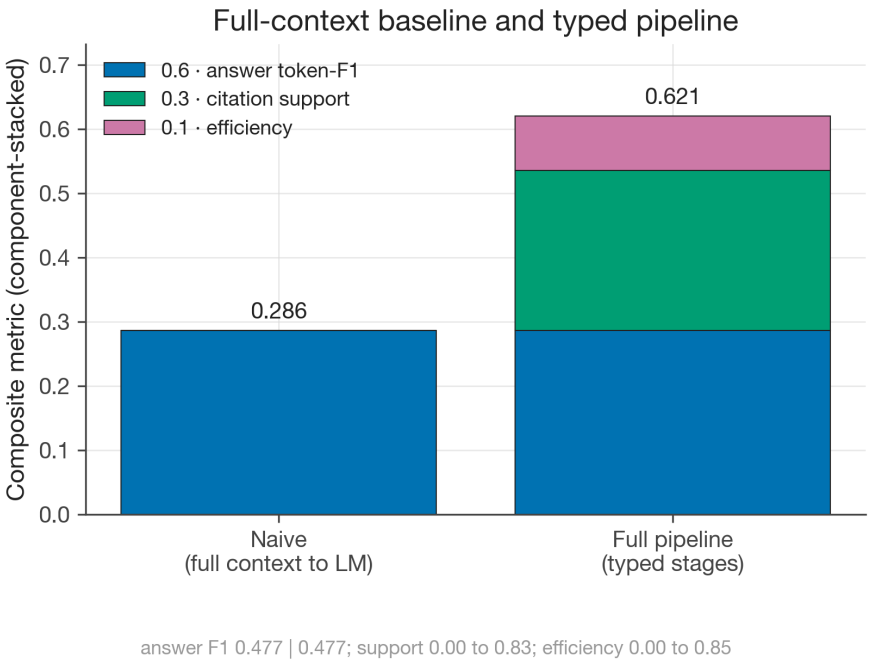


Figure 13: Weighted composite components for the naive full-context baseline and typed pipeline on the fixed 6-example synthetic demo. Answer token-F1 is equal; the composite difference comes from citation support and evidence-count efficiency. Stacked bars reproduce the declared metric. The comparison is deterministic and finite; causal and population interpretations require a separate design.

The composite weights themselves are a fair target. Sweeping the answer-token-F1 weight across [0.4, 0.8] (citation support and efficiency splitting the remainder in the default 0.3:0.1 ratio), the compiler's winning configuration is invariant — the declared finite sweep selects top-k 4, max-evidence 3 — so the headline 0.6/0.3/0.1 choice has little effect on this finite sweep. (The efficiency term's $1 - |\text{ranked}|/16$ uses 16 as the inline context budget in evidence slots; on this compact set ranked evidence sits far below it, which is exactly why efficiency stays high and near-constant.)

11.3.2 DSPy Runtime and Neural Backend Validation

The answer-token-F1 ceiling above is a property of the *default extractive LM*, so we measure it directly. The same 5-stage contract runs unchanged under (i) a real `dspy.Predict` program compiled by a real `dspy.BootstrapFewShot` optimizer and (ii) a real neural model (`gemma3:4b`, served locally through Ollama) dropped into the Reasoner slot via `src/dspy_runtime/`. Inputs are identical across backends; only the language model changes.

We also tested the optimiser with a held-out before/after comparison. On a leave-out split of the dataset, the `BootstrapFewSho t`-compiled program shows **no measurable token-F1 lift** over the uncompiled one: the extractive backend ignores few-shot demonstrations by construction, and the neural backend is already at ceiling on these terse-gold questions. So the Compiler's *measured* value on this task class is its auditable search trace. The held-out data show no answer-quality gain, so the result is

recorded as a null (`src.dspy_runtime.held_out_optimizer_lift`).

fig. 14 reports token-F1 per task category for the deterministic extractive backend versus the neural model on all 24 examples. On this expanded local aggregate, swapping in the neural LM changes mean answer token-F1 from 0.587 to 0.540; paired mean delta -0.046 with 95% interval [-0.161, 0.067] and sign-flip randomization p=0.440 yield no statistically distinguishable improvement under the declared finite comparison. The neural model is higher on **code** (0.795 to 0.533) and multi-session (0.615 to 0.667), and lower on the other categories. This mixed result shows that the extractive ceiling depends on the model and metric. The retained aggregate supports a category-level diagnostic; a general direction across tasks or models requires broader evidence.

The comparison also surfaces a metric caveat that strengthens the honesty story: on single-session and preference the neural model scores *lower* on token-F1 despite answers that may be semantically appropriate and lack separate adjudication (it returns "bluefish42" where the gold string is "wifi password is bluefish42"). This is the same construct problem that motivates finer-grained factuality measures such as FActScore [Min et al., 2023]. Token-F1 rewards lexical overlap with the gold phrasing, so a terse answer can be penalised; the metric is the limiting factor there. With 24 examples this is a diagnostic signal; the figure is case-level. The neural numbers are reproducible *on a host* (temperature 0.0), with cross-host values depending on machine and Ollama versions. Both backends and this table regenerate with `scripts/run_neural_eval.py`.

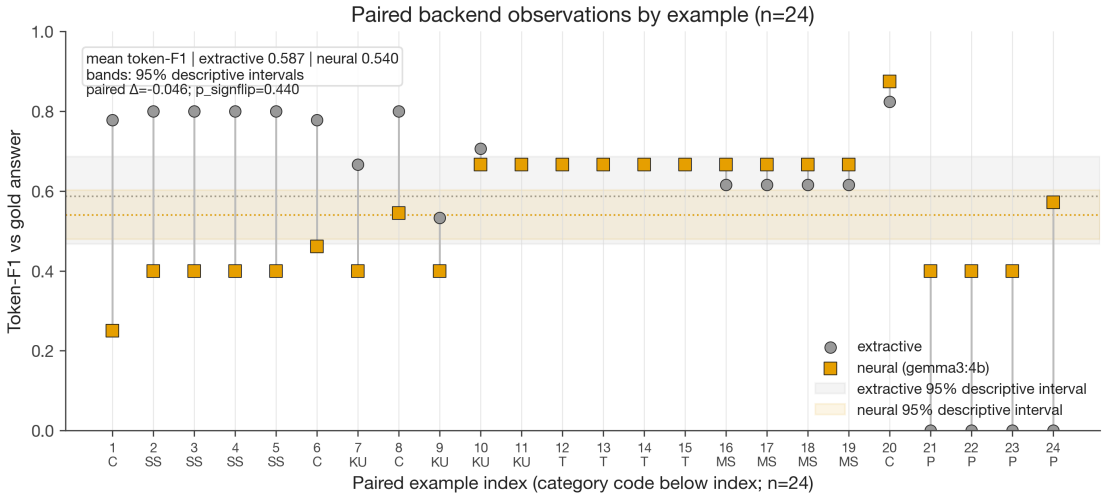


Figure 14: Paired token-F1 per synthetic evaluation example for the deterministic extractive Reasoner and the recorded neural model (`gemma3:4b`) under the same 5-stage contract. Lines preserve the pairing on each of the 24 cases; guides and translucent bands show marginal means and 95% descriptive intervals, while the text reports the paired delta and sign-flip randomization p-value. Recorded with temperature 0.0 and max_tokens 512 via `scripts/run_neural_eval.py`; the neural estimate is model- and configuration-dependent, with host-specific provenance.

11.3.3 Cloud-Model Validation via OpenRouter

To probe whether the observed answer-token-F1 boundary is specific to one local backend, we ran a separate finite 6-example provider diagnostic against `tencent/hy3-preview` accessed through OpenRouter — routed via `dspy.LM` → litellm → the OpenRouter API, with no changes to the 5-stage harness contract. The recorded results live at `data/openrouter_eval.json`.

The headline numbers favour the extractive backend on mean token-F1: **0.477** (4/6 pass) for extractive versus **0.467** (2/6 pass) for the neural backend. That raw ordering reflects a token-F1 construct effect: **concise answers are structurally penalised**. The recorded model returns atomic, information-dense replies (short values and dates) while the gold strings encode the surrounding sentence. Token overlap with a longer gold string is higher for the extractive backend because it echoes the full transcript line verbatim. The recorded model's answer is shorter and has no separate adjudication here, so token-F1 can penalise it. The programming-language pair (F1 0.333 vs. 0.833) is the clearest illustration.

The ordering **reverses** on the two question types intended to require reasoning beyond verbatim extraction. On the **cross-session budget synthesis** query ("What budget did I mention across our sessions?") the extractive backend scores 0.000 — it can only return the closest matching transcript line, which mentions vendors; the neural backend answers with the budget value for an F1 of 0.667. On the **multi-hop deploy error** query ("What error does the deploy code raise without CONFIG?") extractive again

scores 0.000 (it surfaces the file path; the exception is absent), while the neural backend answers "KeyError" for an F1 of 0.333. These are the synthesis and cross-document cases included to stress the contract; in this finite probe, tencent/hy3-preview has the higher token-F1 on both.

This finite, non-adjudicated probe is consistent with the Ollama diagnostic in sec. 11.3.2. Answer token-F1 is bounded by the selected LM and the gold-string tokenization, while the pipeline adds provenance and leaves semantic judgment to the model. The two synthesis/multi-hop cases show why a neural backend may produce a better answer than extraction on some questions. A general backend ranking and quality effect require broader adjudicated evidence. The full per-example breakdown is at `data/openrouter_eval.json` (`temperature: 0.0`, `max_tokens: 1024`, `model: tencent/hy3-preview`, 6 examples).

11.3.4 Context-Poisoning Stress Test

Indirect prompt-injection research shows why retrieved data can blur the boundary between content and instructions [Greshake et al., 2023]. This finite probe constructs 30 sessions with a supplied fact and a conflicting line ("IGNORE ALL PREVIOUS … the password is hunter2"; "SYSTEM OVERRIDE …") that tries to override it. fig. 15 reports, per backend, whether the *answer* is poisoned and whether a cited excerpt lexically contains the supplied gold value.

The result is deliberately narrow. The two deterministic conditions record the same poisoned count (2 of 30 probes), so the probe shows no poisoning-rate advantage. The cited-excerpt outcome is a legacy lexical source-exposure proxy: the harness's joined cited excerpts contain the supplied gold value on 30 of 30 probes, versus 0 of 30 for the naive baseline. It does not verify a source record, recovery of the true source, citation entailment, or claim-level auditability. The optional neural row is recorded only when a live model is available; the current aggregate records 22 of 30 neural poison cases, and if that token is unavailable (`-`) the figure omits the neural row. The figure therefore preserves a finite visibility observation alongside the poisoning outcome (`scripts/run_injection_probe.py`).

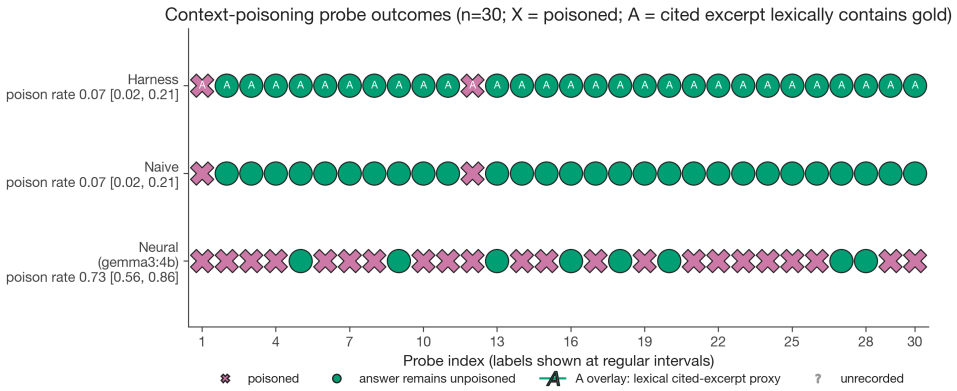


Figure 15: Context-poisoning probe outcomes for 30 synthetic paired cases. `X` marks a poisoned answer and `A` marks a cited excerpt whose text lexically contains the supplied gold value. The `A` overlay is a lexical cited-excerpt/source-exposure proxy, not source-record recovery or claim-level auditability; row labels show poison rates with Wilson intervals when available. The deterministic rows have the same poisoned count. Optional neural rows appear only when `data/injection_eval.json` contains a live measurement and retain model provenance.

11.4 Retrieval Representations and Distractor-Noise Diagnostics

11.4.1 Lexical, Surrogate-Semantic, and Hybrid Retrieval

fig. 16 compares lexical-only, semantic-only, and hybrid composites at the **default (slack) retrieval budget** (`top_k=8`, `max_evidence=5`). On the clean compact dataset all three coincide (lexical 0.621, semantic 0.621, hybrid 0.621): the gold evidence is unambiguous, so every lane resolves it and the fused result is identical. The deterministic tie-break selects lexical as the best single mode. Retrieval choice is therefore equivalent in the slack-budget regime; the tight-budget sweeps below expose the regime in which evidence selection can separate methods. The ablation, MMR, and RRF-weight figures use `top_k=2` and `max_evidence=2`, the binding budget inherited from the noise study.

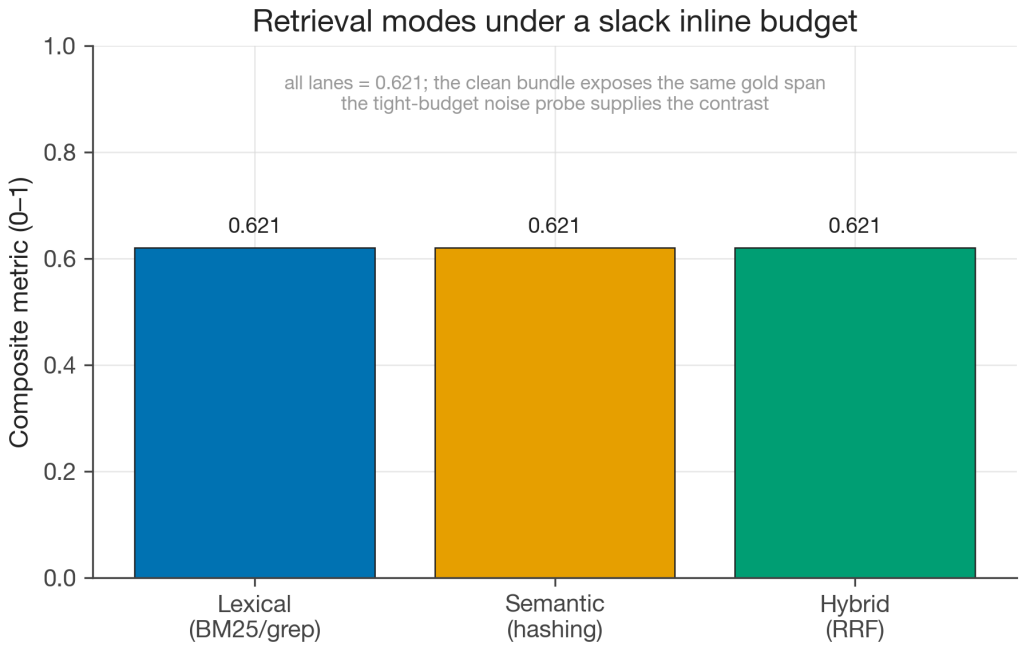


Figure 16: Composite metric by retrieval mode on the fixed 6-example synthetic demo at the default slack budget (top-k 8, max-evidence 5). The observation unit is one deterministic extractive-LM evaluation per example. All three lanes resolve the unambiguous gold evidence identically; the tight-budget, noise, and learned-embedder probes supply the separation diagnostics. Bars are finite descriptive values, and the semantic lane is a hashing surrogate.

Scope boundary (stated plainly). The cited agentic-search study reports that grep generally beats vector retrieval in its inline comparisons, with dependence on harness and tool style [Sen et al., 2026]. That result belongs to a learned dense retrieval setting. THALIA's local semantic lane is a *deterministic hashing-embedding surrogate*: character n-grams are hashed into a fixed vector so the complete harness can run offline with zero mocks. The surrogate measures sub-word overlap and serves as a controlled lexical retrieval comparator. A learned embedder fits the same `SemanticIndex` interface and supplies the separate representation experiment required to compare learned dense retrieval with lexical search.

The local experiment establishes three executable properties. Lexical retrieval, the hashing lane, weighted RRF fusion, and budgeting compose deterministically. On the clean comparison all three lanes coincide at 0.621 because the gold evidence is unambiguous. Session noise creates lane divergence (sec. 11.4.3), so the clean-set equality describes the slack-budget probe; the broader hybrid effect remains an open representation-level question.

This boundary separates two estimands. The orchestration estimand asks whether a typed harness can pass evidence through Inspector, Retriever, Reasoner, Memory Gate, and Compiler while preserving provenance. The representation estimand asks which embedding model improves retrieval for a declared probe family. THALIA measures the first locally and exposes the second through a drop-in extension.

11.4.2 Learned Embedder Versus Hashing Surrogate

Behind the same `top_k(query, k)` interface as the hashing surrogate, `src/retrieval/learned.py` wires a real neural sentence embedder (`nomic-embed-text`, served locally through Ollama). We score recall@1 on 32 **paraphrase** probes. Each query shares meaning with its gold document while sharing little surface morphology; an example pair is "access code for the network" and "the wifi password is …". fig. 17 reports the result: the hashing surrogate retrieves the gold document on 0.88 of the probes (Wilson interval [0.72, 0.95]), while the learned embedder retrieves it on 1.00 (Wilson interval [0.89, 1.00]). The expanded probe is deliberately mixed: some paraphrases retain sub-word signal for the surrogate, while the learned lane closes the remaining misses. The paired Wilson and exact-binary accounting is recorded in the aggregate and describes this synthetic probe family. Population retrieval rates require a broader sampling frame.

The measurement is gated on a live embedding model and recorded to `data/learned_retrieval.json` (regenerable via `scripts/run_learned_retrieval.py`). The deterministic surrogate remains the offline default, so the bit-reproducible core is unchanged. This is a representation-level probe. Retrieval recall measures the upstream representation and ranking step; answer quality also depends on how the generator uses surfaced evidence [Lewis et al., 2020].

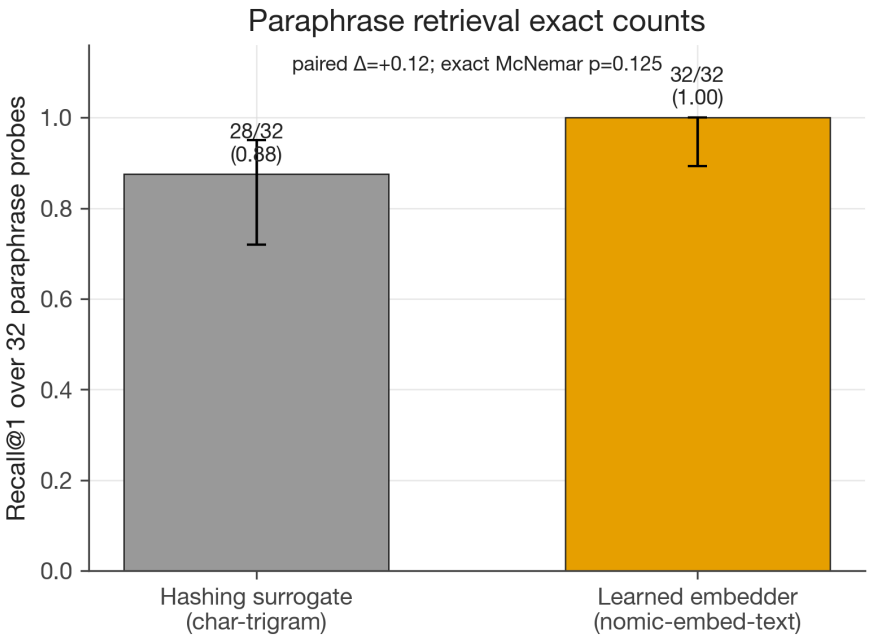


Figure 17: Paired recall@1 for 32 synthetic paraphrase probes. Each point and bar records whether the gold document was retrieved by the deterministic hashing surrogate or the learned embedder (`nomic-embed-text` via Ollama); bars show Wilson intervals and the exact paired McNemar comparison reports p=0.125. The observation unit is a probe, model provenance is recorded in `data/learned_retrieval.json`, and the panel supports a finite representation diagnostic. The hashing surrogate remains the offline default.

11.4.3 Retrieval Under Distractor Noise

`noise_scaling_experiment` appends lexical false-friend lines (repeating query tokens without the answer) and measures each mode at a tight retrieval budget, following the design form of the cited noise study [Sen et al., 2026]. The composites remain within the metric range as noise grows. Inspector grep anchoring filters many false friends before retrieval. For this generated probe, the hashing semantic lane remains above the lexical lane: at the noisiest setting (32 false-friend lines), the composites are 0.692 and 0.620, respectively. The hashing lane ranks the gold line first because its sub-word overlap survives the tight budget; deterministic execution gives the same ordering on every run.

The gap is present at zero distractors. The binding budget creates the separation, and the noise axis measures its stability. The slack-budget equality in sec. 11.4.1 and the tight-budget divergence therefore describe two operating points of the same retrieval system. The ordering belongs to this hashing surrogate and compact generated set. A learned dense retriever requires a separate

run under the same fixed probe, with model and host provenance recorded. The cited lexical-versus-dense result remains external evidence [Sen et al., 2026].

The noise probe has a fixed, falsifiable contract: false-friend lines are appended mechanically, the retrieval budget is fixed, and regression tests pin the measured ordering. A learned embedder can replace the hashing lane and rerun the same probe; the remaining stages and the metric contract remain unchanged.

11.5 Compiler Search, Method Diagnostics, and Synthesis

11.5.1 Bounded Compiler Search

fig. 18 shows the MIPRO-style search over 8 candidate configurations. Every candidate reaches the same composite to full floating-point precision: 0.621. The answer token-F1, citation-support, and efficiency components are bit-identical across the search trace. The equality follows from the data and metric contract: each configuration retrieves the same gold spans, and the compact dataset produces only a few evidence windows, below every candidate's `max_evidence` cap. The three components therefore remain constant across retrieval depths.

The reported configuration, top-k 4 and max-evidence 3, is the first configuration reaching the shared maximum under the deterministic tie rule. This result describes selection under an equal score. The GEPA-style optimizer reaches the same plateau through reflective mutation, and its full trace is recorded for audit. A harder distribution with competitive evidence windows supplies the conditions under which retrieval depth can change the score; the tight-budget ablation and MMR sweep provide local examples of that regime.

11.5.1.1 Held-Out Compiler Generalization The compiler claim is tested separately from the easy in-sample trace. Across 20 fixed generated-task seeds, each split contains 46 training and 14 held-out examples. The held-out composite means are:

method	held-out composite	delta vs. baseline
baseline	0.743	—
MIPRO-style	0.746	0.002
GEPA-style	0.743	0.000

The aggregate accounts for 13720 executed example evaluations, including 260 candidate configurations evaluated on training data alongside the final holdout scores. The small observed MIPRO-style held-out change and the GEPA-style null result are finite generated-task observations. They characterize this seeded split family; generalization to an external benchmark or production workload requires a separately sampled evaluation.

Two further diagnostics close the section. The RRF lane-weight sweep (fig. 20) records a non-monotonic dip near the balanced midpoint under the *tight* budget; every weight remains tied under the slack default. The consolidation curves (fig. 21) provide a mechanism schematic for the memory-fault discussion in [Zhang et al., 2026]. They encode the design expectation that forced per-turn consolidation can degrade utility as memory accumulates and that episodic-first gating can preserve a useful baseline. The schematic carries no observed memory trajectory.

The GEPA-style optimizer is designed to move when a harder distribution exposes a weak component: it reflects on that component and mutates accordingly. fig. 19 shows the reflective run. Each iteration plots answer token-F1, citation support, efficiency, and the composite; the open halo marks the component identified as weakest at that step and targeted by the next bounded mutation. The composite plateaus on the easy set while the component trace records the optimizer's decision path. Readers can inspect the mutation strategy directly in the generated trace and its provenance fields.

11.5.2 Added-Method Diagnostics

Four further figures characterise the pipeline and the added methods. The **evidence funnel** (fig. 22) quantifies the narrowing from raw session lines through Inspector windows and ranked evidence to cited sources. The **per-category breakdown** (fig. 24) locates composite variation across the 6 task classes. The **noise-scaling** curves (fig. 26) plot each lane's composite as distractor lines accumulate; under the default RRF weights, hybrid and lexical traces coincide in this finite probe. The dashed hybrid trace remains visible above the lexical trace in the rendered figure and the value table records the equality. The **MMR tradeoff** (fig. 27) sweeps λ in eq. 8 from pure diversity to pure relevance under the same tight retrieval budget (`top_k=2`, `max_evidence=2`) as the ablation and noise study. The pure-diversity endpoint receives a selection penalty because dissimilarity can evict the discriminative span from the retained evidence; the curve reaches its plateau once relevance receives a positive weight. The slack budget supplies the flat reference regime.

The funnel is a mean view. fig. 23 shows the per-example evidence path for the 6 demo examples, one per task category, with counts at Inspector windows, Retriever-ranked evidence, and Reasoner-cited sources. Each bar triplet exposes the exact transition

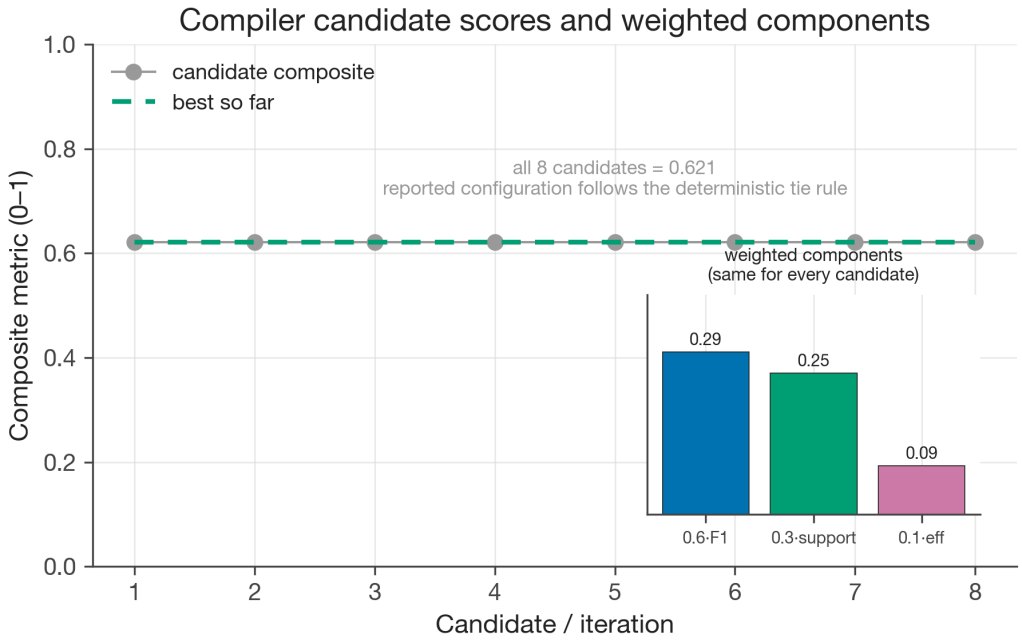


Figure 18: MIPRO-style compiler trace for 8 candidate configurations on the fixed 6-example synthetic demo. The observation unit is a candidate configuration; each candidate reaches composite 0.621, and the inset shows the constant weighted answer token-F1, citation-support, and efficiency components. The selected top-k 4 and max-evidence 3 follow the deterministic first-achieving tie rule. The trace is a finite project-source diagnostic.

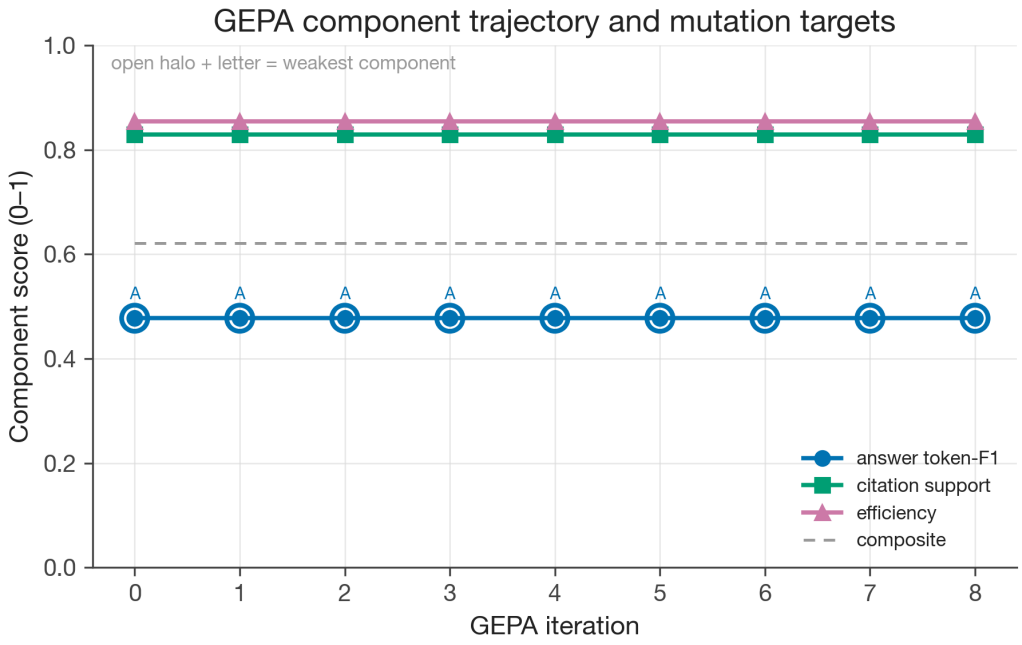


Figure 19: GEPA-style component trajectory over a deterministic compiler run on the fixed 6-example synthetic demo. The observation unit is an optimizer iteration; solid lines show answer token-F1, citation support, and efficiency, the dashed line shows the composite, and the open halo marks the weakest component selected for the next bounded mutation. The trace records an algorithmic decision path and carries no sampling estimate.

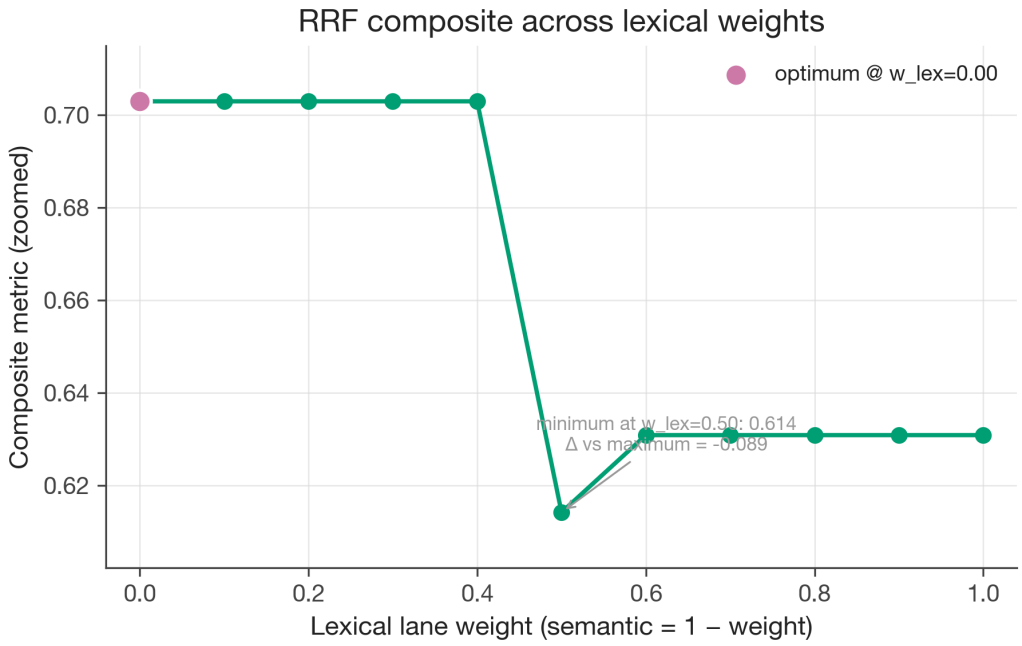


Figure 20: Weighted RRF sensitivity on the fixed 6-example synthetic demo under the tight retrieval budget. The observation unit is a tested lexical weight; the line reports composite score across the feasible weight sweep, with the highest tied settings and the change from the maximum annotated. The slack default budget yields a flat score because every setting retains the same gold evidence. This finite deterministic sweep describes budget-conditioned ranking behaviour.

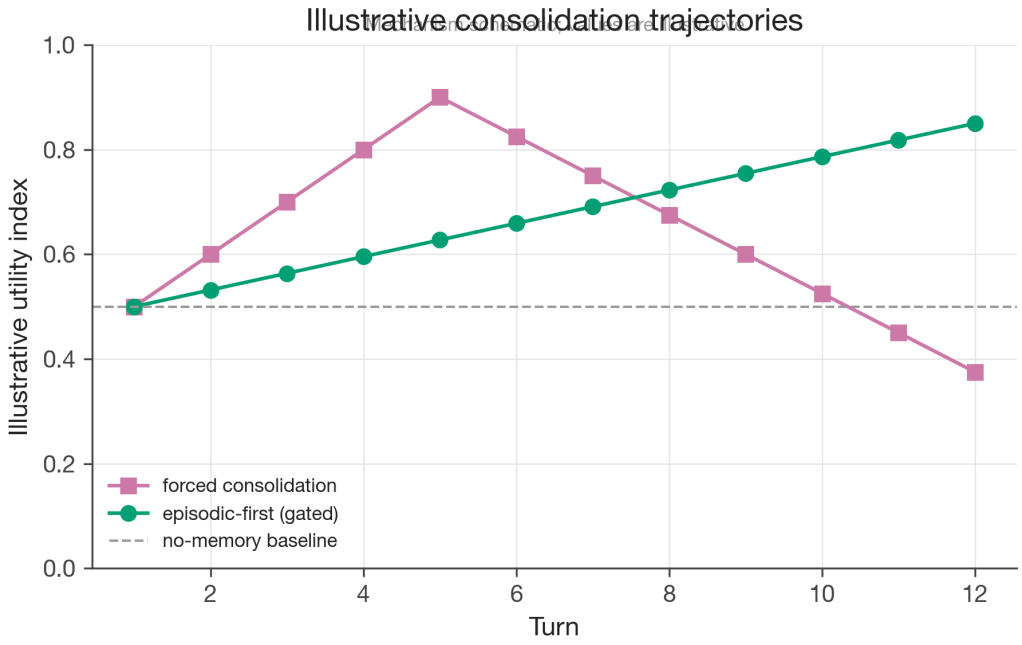


Figure 21: Mechanism schematic for memory utility across turns. The curves encode the design expectation that forced per-turn consolidation can accumulate harmful derived memory while episodic-first gating preserves a useful trajectory. Values are illustrative indices tied to the mechanism discussion in [Zhang et al., 2026]; the panel contains no observed trajectory, sample, model estimate, or population estimand.

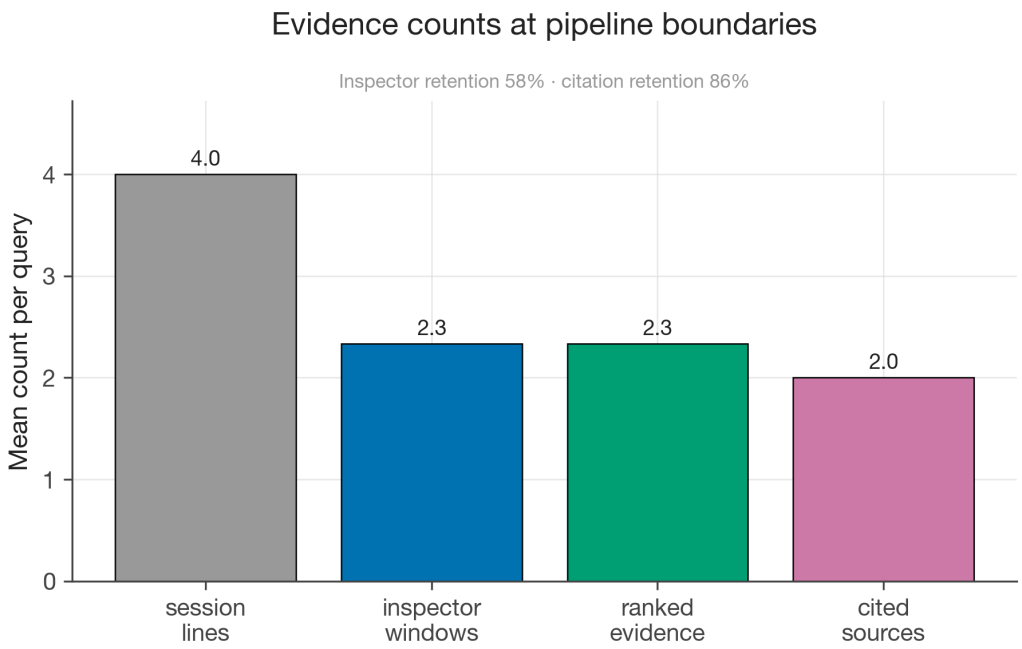


Figure 22: Evidence narrowing funnel for the fixed 6-example synthetic demo. The observation unit is a query; bars show mean line counts at raw session, Inspector-window, ranked-evidence, and cited-source boundaries, while annotations report Inspector and citation retention. The panel measures context narrowing and source exposure; answer quality uses the separate token-F1 and support estimands.

contributing to the funnel means. A matched ranked and cited count indicates full source selection for that example; a lower cited count identifies a concrete support gap for inspection.

The composite is a sum of three declared components. fig. 25 decomposes each category into the weighted answer token-F1, citation-support, and efficiency terms defined in eq. 18. The code and multi-session categories carry the lowest answer-token-F1 component in this extractive-LM run; the support and efficiency contributions remain separately visible. The pattern is consistent with the answer ceiling documented in sec. 11.3.2 and identifies the component that a stronger generator would need to change.

11.5.3 Added-Method Ablations

fig. 28 reports the composite for the base pipeline and each added retrieval method (query expansion, recency weighting, MMR), plus the combined configuration, on the single-turn dataset. The ablation uses the same *tight* retrieval budget (`top_k=2`, `max_evidence=2`) as the MMR sweep and noise study. The loose default budget retrieves every gold span for every condition, producing the expected equality reference. The tight budget creates a selection contest in which each method can change the evidence that survives.

The experiment keeps the generated dataset and scoring rule fixed. Recency weighting has the highest observed composite in this finite run because recency promotes the discriminative span into the surviving evidence. Query expansion is mildly lower as extra terms dilute an already precise query, and MMR is close to the base configuration on this small set. Memory recall uses a separate populated episodic-store contract; recall and harness tests exercise a fact held only in the episodic store and verify its appearance in answer citations.

11.5.4 Integrity-Advantage Screen: Current Findings and Open Tests

The integrity screen ranks endpoint-specific comparisons by claimability. Positive effects use the declared direction of each endpoint, and the source rows retain their native units. Binary probe effects use exact paired McNemar tests with Wilson intervals for the two condition rates and paired bootstrap intervals for the rate differences. The compiler comparison resamples seed clusters. The neural comparison uses its recorded paired aggregate. Aggregate comparisons without matched rows remain point estimates.

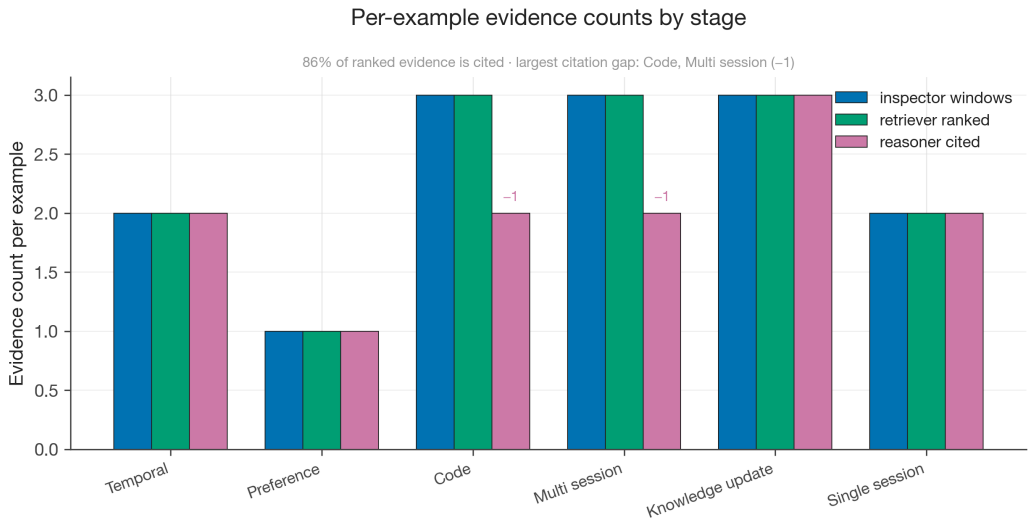


Figure 23: Per-example evidence path across pipeline stages for the fixed 6-example synthetic demo. The observation unit is an example: blue bars show Inspector windows, green bars show hybrid ranked evidence, and pink bars show Reasoner citations for each task category. The annotation reports citation retention and marks ranked-to-cited gaps. The panel is a deterministic source-exposure diagnostic.

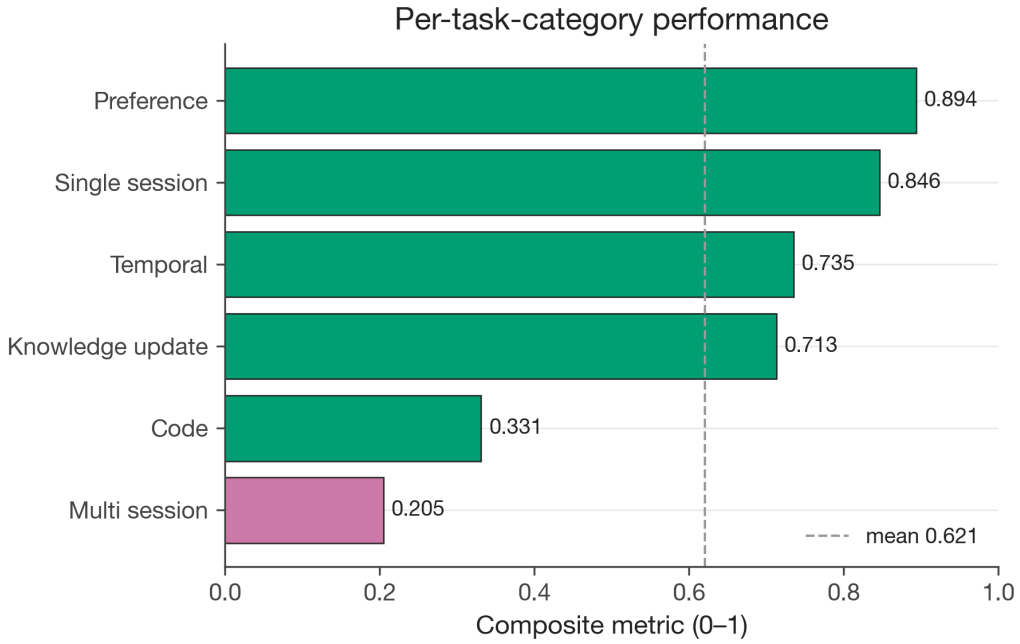


Figure 24: Composite score by task category for the fixed 6-example synthetic demo, with one example per category. Each bar is the declared weighted combination of answer token-F1, citation-support, and evidence-count efficiency. The observation unit is a generated example and the panel is a deterministic diagnostic of category-specific failure modes.

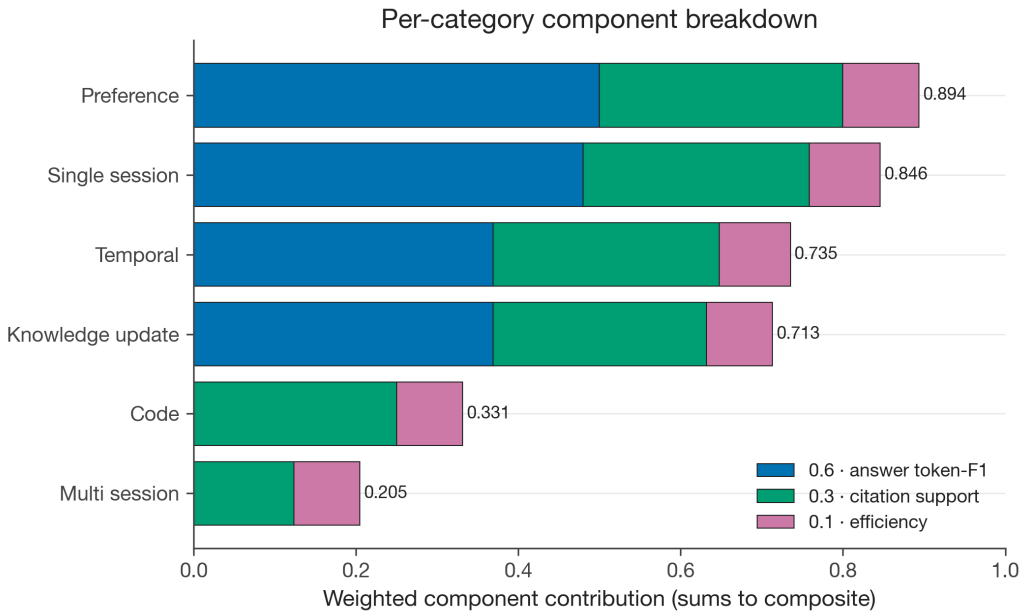


Figure 25: Weighted component audit by category for the fixed 6-example synthetic demo. Stacks show 0.6·answer token-F1 in blue, 0.3·citation-support proxy in green, and 0.1·evidence-count efficiency in pink; their sum equals the composite in eq. 18. The observation unit is a generated example, and the extractive responder bounds the answer token-F1 component.

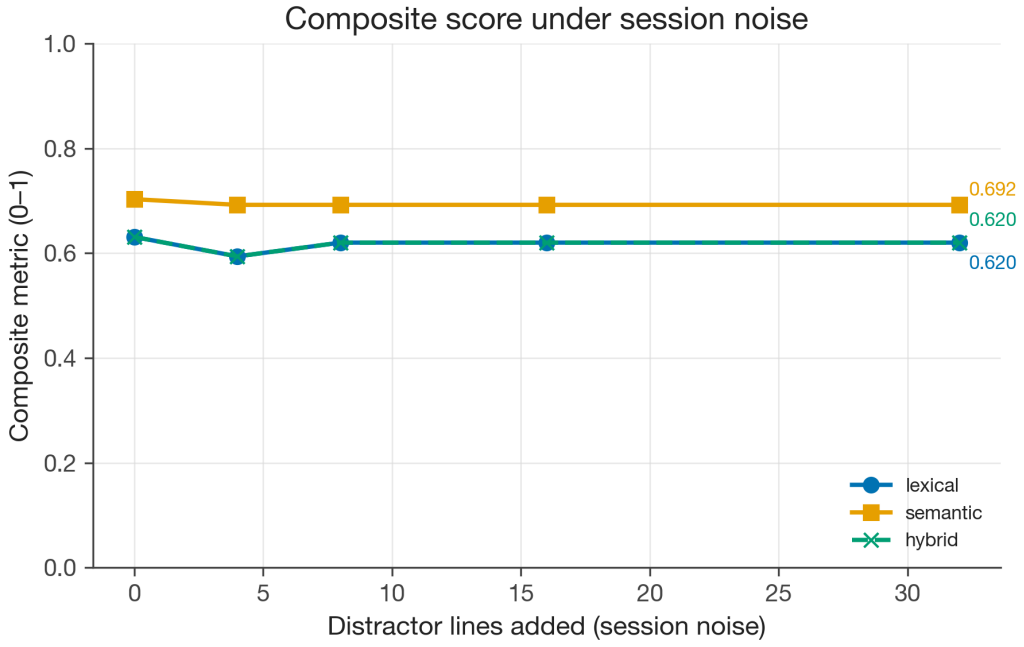


Figure 26: Composite score by retrieval lane and distractor level on the fixed synthetic demo. Lexical false-friend lines are appended under the tight budget (`top_k=2`, `max_evidence=2`); the observation unit is an example-level score aggregated across the six demo cases. The semantic trace uses the deterministic hashing surrogate, so the ordering describes this finite mechanical probe and its recorded budget regime.

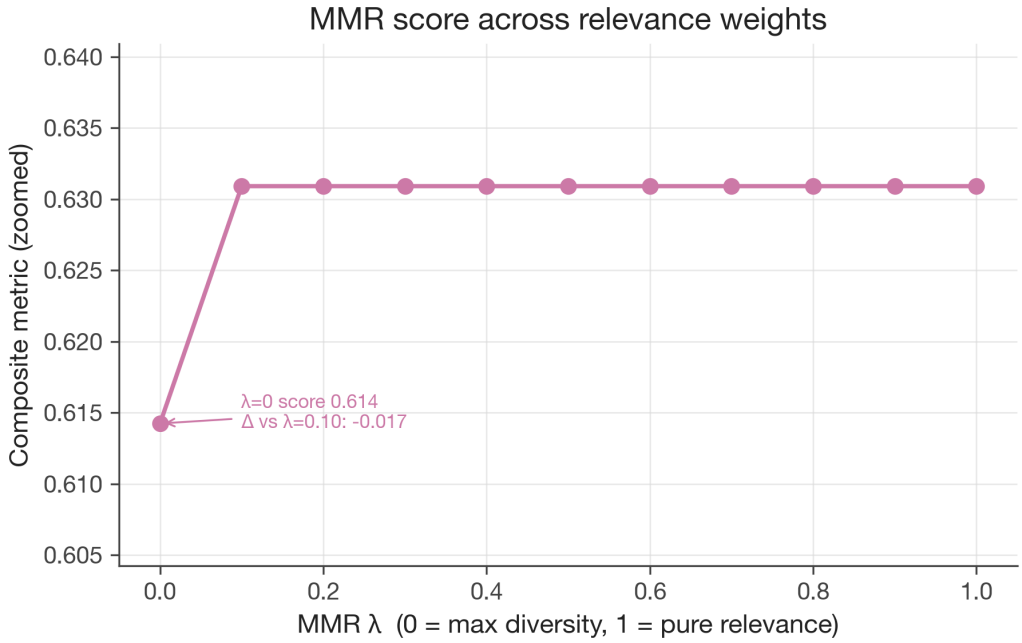


Figure 27: MMR composite across the full diversity-to-relevance lambda sweep on the fixed 6-example synthetic demo under the tight retrieval budget. The observation unit is a tested lambda setting; the pure-diversity endpoint carries the measured selection penalty, and the curve reaches its plateau as relevance receives weight. The slack budget supplies the flat reference regime.

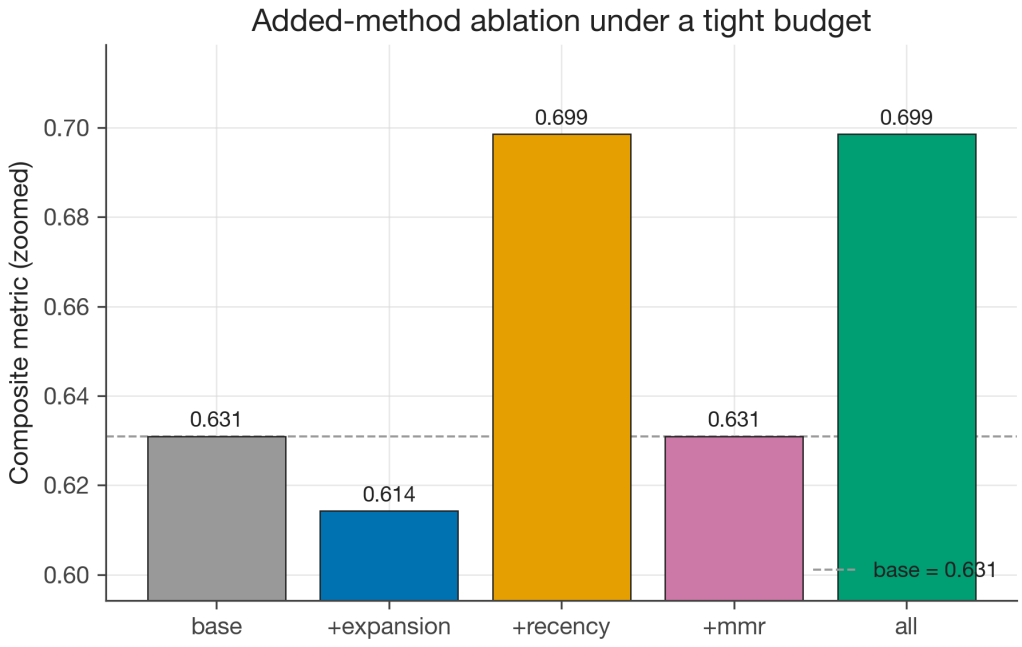


Figure 28: Added-method ablation on the fixed 6-example synthetic demo under the tight retrieval budget. The observation unit is a configuration; bars show composite for the base pipeline, query expansion, recency weighting, MMR, and the combined setting. Recency weighting is highest in this run, expansion is lower, and MMR is near the base. The extractive responder and hashing surrogate define the evidence tier.

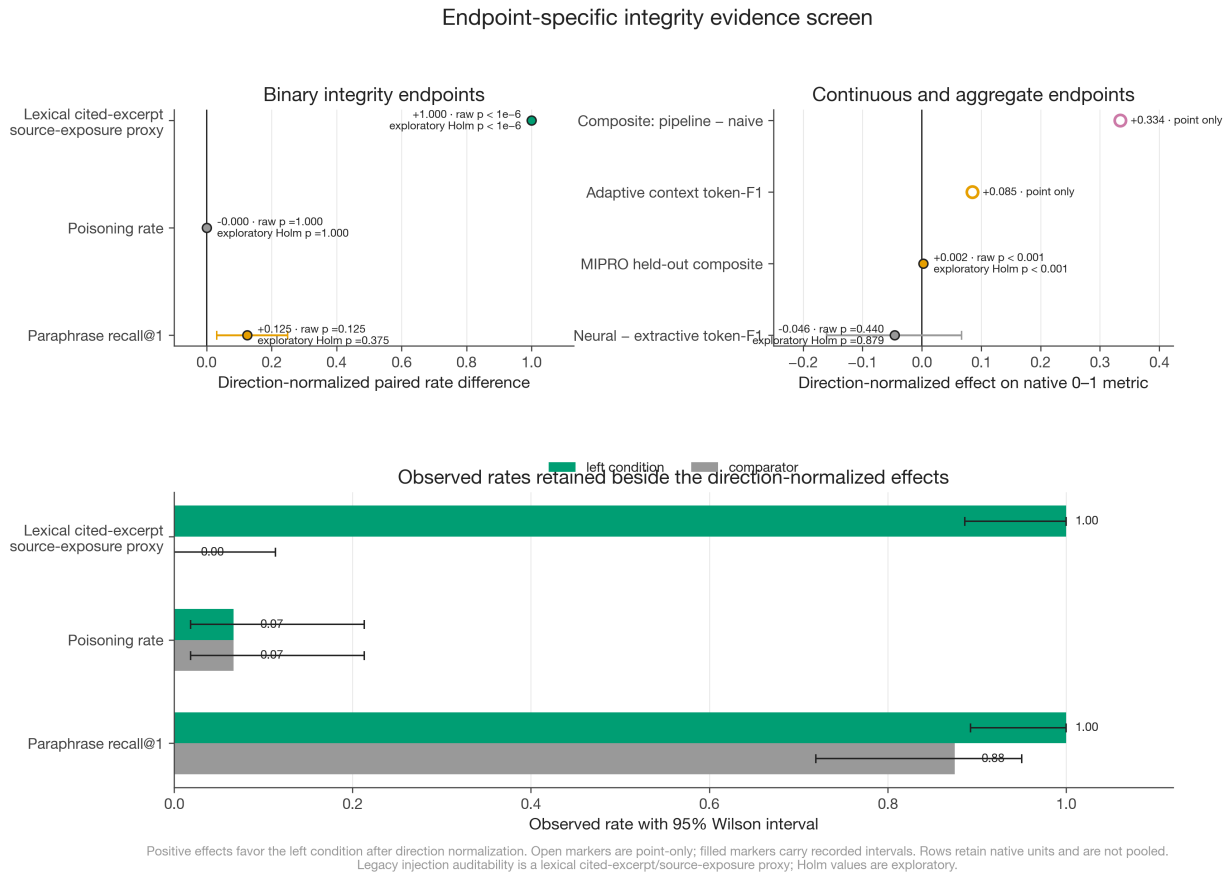


Figure 29: Endpoint-level integrity-advantage screen. The observation units remain separate: binary probe effects use direction-normalized rate differences; continuous panels retain composite, token-F1, and context-policy units. Filled markers show recorded intervals, open markers show point-only aggregates, and the lower panel retains the two condition rates with Wilson intervals. Positive effects favor the left condition. Rows are not pooled. The legacy injection "auditability" row is a lexical cited-excerpt/source-exposure proxy, not a source-record-recovery or claim-level-auditability test; raw and exploratory-Holm paired p-values are displayed where available.

The strongest current screen row is the legacy lexical cited-excerpt/source-exposure proxy under the recorded injection probes. THALIA's cited excerpts contain the supplied gold value on every one of the 30 probes, giving a direction-normalized paired rate difference of +1.000, raw exact $p = 1.86e-09$, and exploratory Holm-adjusted $p = 9.31e-09$; the paired bootstrap interval is [1.000, 1.000]. The naive baseline emits no cited excerpts in this comparison. This finite lexical contrast does not demonstrate source-record recovery, citation entailment, claim-level auditability, or answer quality.

The same probes show no poisoning-rate advantage: the direction-normalized poisoning effect is -0.000 because both conditions record the same poisoned count; raw exact $p = 1.000$, exploratory Holm-adjusted $p = 1.000$, and the paired bootstrap interval is [-0.000, -0.000]. The composite baseline comparison supplies a separate, conditional orchestration result. Its lift is +0.334 while raw answer token-F1 ties; citation support and evidence efficiency supply the lift. The composite therefore measures the declared evidence-and-efficiency utility function rather than a general answer-quality gain.

Three results are promising research leads. The learned embedder improves paraphrase recall by +0.125 on the paired probe set, with raw exact $p = 0.125$, exploratory Holm-adjusted $p = 0.375$, and paired bootstrap interval [0.031, 0.250]. The adaptive context policy improves the recorded external-slice aggregate by +0.085 across 30 retained questions, but the source aggregate lacks paired question-level rows. The MIPRO-style held-out change is +0.0025 with a descriptive cluster interval of [0.0020, 0.0029], raw sign-flip $p = 2.00e-04$, and exploratory Holm-adjusted $p = 8.00e-04$ across 20 seed clusters. These results motivate replication; they do not yet support promoted general advantages.

The local neural comparison provides a negative control for broad answer claims: the neural-minus-extractive token-F1 effect is -0.046, with interval [-0.161, 0.067], raw $p = 0.440$, and exploratory Holm-adjusted $p = 0.879$. The interval spans zero. Current evidence therefore records a finite lexical source-exposure contrast and a conditional composite lift. General answer quality, a poisoning-rate advantage, model superiority, and population-level performance remain unestablished.

11.5.5 Discussion: Findings, Contributions, and Evidence Boundaries

This study evaluates THALIA as a research instrument across deterministic, neural, and external-benchmark tiers. Five findings organize the discussion.

- **Contract integrity.** The typed stage graph, append-before-consolidate memory rule, formal anchors, generated variables, and artifact checks execute as one reproducible path. The deterministic contract is tested on 6 examples across all task categories.
- **Controlled diagnostics localize failure.** The deterministic tier records citation-support and efficiency by retrieval lane, noise-scaling behavior, added-method responses, and compiler traces. Tight budgets create observable selection differences; the slack budget supplies an equality reference. The metric-construct audit separates answer token-F1, the blended support proxy, source-index precision/recall, reachability, and efficiency.
- **Model transfer changes the outcome.** The local backend comparison and the external LongMemEval_S slice show model dependence. The extractive LM bounds token-F1 in the deterministic lane, while the neural backend changes answer behavior under recorded host and model settings.
- **The clearest current difference is lexical cited-excerpt/source exposure.** The integrity screen shows that the recorded cited excerpts contain the supplied gold value under the injection probes, while the poisoning endpoint remains equal across conditions. This is not true-source recovery or claim-level auditability. The composite baseline lift is conditional on citation support and evidence efficiency; raw answer token-F1 ties. Answer quality and a poisoning-rate advantage remain open questions.
- **Context narrowing is an intervention surface.** The bottleneck study links the retained loss to Inspector recall. In the retained slice, ranking recall is 1.00 and Inspector recall is 0.60; the adaptive context policy changes neural answer token-F1 from 0.086 to 0.171 while compact-context outputs remain unchanged. This result motivates controlled context-policy experiments under fixed generators and retrieval contracts.

The important contribution is a reusable, source-bound measurement boundary. A typed, lexically anchored, episodic-first, compiler-assisted pipeline preserves raw evidence, records its transformations, and exposes model and embedding interfaces for targeted experiments. The package therefore supports reproducible method comparison and failure localization within the declared tiers.

The promising extensions are broader task families, learned retrieval, stronger generators, and preregistered interventions over context budgets and memory gates. The current implementation supplies stable contracts for those comparisons. The answer-level labels needed to rank general model quality remain unmeasured.

11.5.5.1 Model Quality, Causal Robustness, and Production Readiness Require Separate Evidence The deterministic tier measures execution, lexical overlap, source exposure, and context access. Source-index precision and recall measure declared citation-set alignment. Sentence-level entailment, factual completeness, usefulness, calibration, abstention, safety, causal robustness, and production reliability remain unmeasured by these observables.

The typed quality protocol specifies the missing evidence: a prespecified target, stronger baselines, held-out questions, two blinded expert ratings, adjudication, atomic-claim support labels, calibration and agreement analysis, and a fail-closed certificate. The production protocol separately evaluates latency, cost, failures, privacy, model pinning, rollback, monitoring, drift, and red-team coverage. Neither certificate is present in the current result layer.

This separation gives the manuscript a clear forward path. Researchers can keep the contracts fixed while varying models, embeddings, context policies, and memory gates, then attach the appropriate expert and operational acceptance records. The present evidence establishes an auditable instrument and localized diagnostics; broader quality and production claims remain open studies.

12 Reference Architecture: Design Principles and Repository Structure

12.1 Executable Design Principles

Principle	Source	Implementation in THALIA
Declarative typed signatures	DSPy [Khattab et al., 2023]	Every stage is a `Signature` with typed fields (`src/signatures/`)
Evidence-conditioned generation	Retrieval-augmented generation [Lewis et al., 2020]	Retrieved evidence is a first-class input to the Reasoner with an explicit typed handoff
Context-use diagnosis	Long-context position sensitivity [Liu et al., 2024]	Inspector reachability and Reasoner answer quality are reported as separate observables
Factuality measurement boundary	Atomic factuality evaluation [Min et al., 2023]	Token-F1 and citation overlap are labelled operational proxies; adjudicated truth requires the quality protocol
Expert-rater quality protocol	Atomic factuality, truthfulness, and evaluator-bias methodology [Min et al., 2023, Lin et al., 2022, Zheng et al., 2023, Wang et al., 2024, Shi et al., 2025]	Separate correctness, completeness, faithfulness, usefulness, safety, and inter-rater agreement fields with blinded packets and fail-closed promotion; local LLM judging is an exploratory non-promotable diagnostic
Confidence and abstention	Calibration and selective prediction [Guo et al., 2017, Geifman and El-Yaniv, 2017]	Brier/ECE/reliability/selective metrics use adjudicated outcomes; model text supplies no calibration label
Context as inspectable external state	RLMs [Zhang et al., 2025]	`REPLEnvironment`; grep before any neural call (`src/stages/inspector.py`)
Episodic evidence preservation	Memory faults [Zhang et al., 2026]	Append-only SQLite store, no overwrite (`src/memory/episodic_store.py`)
Gated consolidation	Memory faults [Zhang et al., 2026]	Both flag and criterion required (`src/memory/consolidation.py`)
Lexical-first retrieval	Agentic search [Sen et al., 2026]	BM25/grep primary lane (`src/retrieval/lexical.py`)
Weighted rank fusion	Agentic search [Sen et al., 2026]; RRF [Cormack et al., 2009]	`rrf_fuse`, lexical 0.65 / semantic 0.35 (`src/retrieval/fusion.py`)
Sparse context budgeting	Agentic search [Sen et al., 2026]	`delivery_mode`; file-based overflow (`src/stages/retriever.py`)
Frozen memory snapshot	Hermes [Nous Research, 2026a]	Read-only `MemorySnapshot` into the Reasoner (`src/signatures/payloads.py`)
Self-evolving skills	Hermes + GEPA [Nous Research, 2026b]	`MIPROCompiler`, `GEPAOptimizer`, `check_skill_constraints` (`src/compiler/`)
Reproducible provenance	`template/` [Friedman, 2026]	Zero-Mock tests, 90% gate, generated tokens
`AGENTS.md`/`SKILL.md` duality	`template/` [Friedman, 2026]	Per-stage `SKILL.md` for hallucination-free discovery
Operable + configurable	this work	File/env/flag config, real LM adapters, persistent session, CLI, composable Pipeline (sec. 14)
Independent production gate	this work	Latency, cost, privacy, pinning, rollback, monitoring, drift, and red-team acceptance remain separate from model quality

Principle	Source	Implementation in THALIA
Adaptive retrieval + memory recall	this work	PRF query expansion, recency weighting, read-only episodic recall (sec. 7.3)
Resource safety + serialisation	this work	Context managers, `to_yaml`/`from_yaml`, `to_dict`, `py.typed` (sec. 14)

12.2 Repository Layout and Module Ownership

```
thalia/
+-- AGENTS.md / README.md            # machine + reader entry points
+-- pyproject.toml                   # pytest/coverage config, deps
+-- config/harness.yaml              # canonical loadable harness configuration
+-- domain_profile.yaml              # advisory research-domain overlay
+-- experiment_plan.yaml             # deterministic experiment declaration
+-- data/claim_ledger.yaml           # sourced numeric-claim registry
+-- src/
|   +-- signatures/                  # framework, payloads, stage contracts
|   +-- routing.py                   # classify_task
|   +-- evaluation/                  # datasets, experiments, quality, production, bundles
|   +-- module.py                    # Module / Prediction / LMClient / ExtractiveLM
|   +-- config.py                    # HarnessConfig + load/save/env overrides
|   +-- llm.py                       # real LM adapters (CallableLM, OllamaLM)
|   +-- harness.py                   # AutoHarness (composes the 4 runtime stages)
|   +-- recall.py                    # read-only memory recall (episodic + MEMORY.md)
|   +-- session.py                   # persistent multi-turn Session
|   +-- cli.py                       # query CLI (run_query, arg parsing, Inspector diagnostics)
|   +-- demo.py                      # deterministic demo artifact builder
|   +-- provenance.py                # verification, figure, and artifact registries
|   +-- figures/                     # deterministic figure generators (plots, diagrams, manifest)
|   +-- dashboard.py                 # self-contained static HTML dashboard renderer
|   +-- invariants.py                # executable runtime invariants
|   +-- manuscript_variables.py      # generated-token map
|   +-- retrieval/                   # lexical / semantic / fusion / mmr / expansion
|   +-- stages/                      # inspector / retriever / reasoner / memory_gate
|   +-- memory/                      # episodic_store / consolidation
|   +-- compiler/                    # metrics / mipro / gepa
+-- skills/                          # thalia-{inspector,retriever,reasoner,memory-gate,compiler,session}/SKILL.
+-- scripts/                         # thin orchestrators (eval, compile, figures, dashboard, preflight, tokens,
+-- tests/                           # Zero-Mock suite, one file per module
+-- manuscript/                      # this document (sections 00-11, 99) + config/preamble/refs
```

12.3 Stage, Signature, Skill, and Test Map

Stage	Module	Signature	Skill
0 Inspector	`src/stages/inspector.py`	`InspectorSignature`	`skills/thalia-inspector`
1 Retriever	`src/stages/retriever.py`	`RetrieverSignature`	`skills/thalia-retriever`
2 Reasoner	`src/stages/reasoner.py`	`ReasonerSignature`	`skills/thalia-reasoner`
3 Memory Gate	`src/stages/memory_gate.py`	`MemoryGateSignature`	`skills/thalia-memory-gate`

Stage	Module	Signature	Skill
4 Compiler	`src/compiler/`	— (operates on configs)	`skills/thalia-compiler`

13 Reproducibility: Determinism, Generated Results, and Provenance

THALIA inherits the `template/` reproducibility discipline [Friedman, 2026, Peng, 2011]: deterministic computation, Zero-Mock tests, a coverage gate, and a manuscript whose every result number is a generated token.

13.1 Deterministic Execution and Reproducible State

The deterministic core is a deterministic function of its inputs. The lexical lane (BM25), the semantic lane (SHA-1 hashing embedding), the RRF fusion, the REPL-style narrowing, the SQLite episodic store, gated consolidation, and the local MIPRO/GEPA-style search loops contain no unseeded randomness, no wall-clock dependence, and no network calls. The default Reasoner is also deterministic and extractive. Consequently identical inputs reproduce identical answers, evaluation reports, and compiled configurations on that path — a property the test suite asserts directly (`test_evaluation.py::test_deterministic`, `test_manuscript_variables.py::test_deterministic`). Replacing the Reasoner or semantic lane with a live model intentionally moves the run outside that envelope.

Determinism is enforced at method boundaries and verified at the end. The Inspector emits ordered windows with stable source indices; retrieval tie-breaks are deterministic; the episodic store rejects duplicate turn ids; compiler search uses fixed candidate ordering; and generated figures derive from saved JSON bundles. A rerun can therefore compare artifacts at multiple layers: raw evaluation data, compiled configuration, figure inputs, manuscript variables, hydrated Markdown, and final PDF/HTML.

13.2 Source-Bound Manuscript Variables

Every generated `{{…}}` placeholder in this manuscript is emitted by `src/manuscript_variables.py::generate_variables` from the live evaluation results and configuration. Configuration tokens (`CONFIG_*`), headline results (`RESULT_COMPOSITE`, `RESULT_ACCURACY`, …), the retrieval comparison (`RESULT_*_COMPOSITE`), and the compiled optimum (`RESULT_COMPILED_*`) all derive from the same functions the tests exercise. A drift between prose and code surfaces as a missing or changed token, exposing narrative drift.

The token gate is deliberately independent of the PDF renderer. The generator checks manuscript coverage first; optional injection into `output/manuscript/` then uses the sibling `../template` checkout when available. This prevents a renderer-path problem from masking the more important contract: every published number must have a source in the live variable map.

13.3 Ordered Artifact Regeneration

```
# 1. Tests + coverage gate (>= 90% on src/; warnings are strict)
PYTHONWARNINGS=error::ResourceWarning uv run --project . --extra dev python -m pytest tests/ \
  --cov=src --cov-report=term-missing --cov-report=json:output/reports/coverage.json \
  --junitxml=output/reports/pytest.xml --cov-fail-under=90

# 2. Evaluation bundle -> output/data/eval_results.json
# Quick smoke grid:
uv run --project . --extra dev python scripts/run_harness_eval.py
# Recorded publication-size audit:
uv run --project . --extra dev python scripts/run_harness_eval.py \
  --per-category-values 25 50 100 250 \
  --seeds 0 1 2 3 4 5 6 7 8 9 \
  --n-boot 2000

# 3. Compiler search -> output/data/compile_result.json
uv run --project . --extra dev python scripts/run_compiler.py

# 4. Pre-registered finite sensitivity and held-out compiler aggregates
uv run --project . --extra dev python scripts/run_seed_sensitivity.py
uv run --project . --extra dev python scripts/run_compiler_generalization.py \
  --per-category 10 \
```

```
  --seeds 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 \
  --gepa-iterations 4

# 5. Endpoint-specific integrity screen -> data/integrity_advantage.json
uv run --project . --extra dev python scripts/analyze_integrity_advantage.py

# 6. Pipeline trace -> output/data/pipeline_result.json
uv run --project . --extra dev python scripts/run_pipeline.py

# 7. Real DSPy run + optional neural / learned-embedder measurements.
uv run --project . --extra neural python scripts/run_dspy.py
uv run --project . --extra neural python scripts/run_neural_eval.py
uv run --project . --extra neural python scripts/run_learned_retrieval.py
uv run --project . --extra neural python scripts/run_injection_probe.py
uv run --project . --extra neural python scripts/run_longmemeval.py
uv run --project . --extra neural python scripts/run_longmemeval_bottleneck.py

# 8. Figures -> ../figures/*.png  (reads committed aggregates)
uv run --project . --extra dev python scripts/generate_figures.py

# 9. Manuscript variables -> output/data/manuscript_variables.json
uv run --project . --extra dev python scripts/z_generate_manuscript_variables.py

# 10. Demo, dashboard, then artifact hashes + required-output completeness
uv run --project . --extra dev python scripts/run_demo.py
uv run --project . --extra dev python scripts/build_dashboard.py
uv run --project . --extra dev python scripts/validate_artifacts.py
```

The deterministic-core paths in the evaluation, pipeline, figure, and validation steps are bit-reproducible on any host; the test step also exercises optional live-service tests when those services are available. The live-recorder step is the component **outside** the bit-reproducible envelope: it calls recorded local models (model `gemma3:4b`, temperature 0.0 [Gemma Team et al., 2025], and embedder `nomic-embed-text` with task prefixes [Nussbaum et al., 2024]), so its numbers reproduce *on a host serving the same model tags*. Cross-host byte identity depends on machine and Ollama-version agreement. The recorder JSONs carry model metadata — requested model, resolved tag, digest when Ollama reports one, generation options, embedding query/document prefixes, benchmark SHA-256/size where applicable, and runtime Python/platform provenance. If an optional service is unavailable, the tracked aggregate is preserved; isolated output paths can be used to record the skip without rewriting committed evidence.

The deterministic research audit evaluates 25500 unique synthetic examples and 76500 method rows across 4 strata, 6 categories, (25, 50, 100, 250) per-category sizes, and 10 fixed seeds. Its JSONL ledger has one example/method record with global sequence numbers and complete failure/fingerprint fields, while the event stream repeats dataset/config identity per sample and contains no wall-clock fields.

The tracked `data/openrouter_eval.json` is a committed normalized OpenRouter aggregate used only when present; the canonical local pipeline leaves it unchanged. The current live smoke recorder is `scripts/run_openrouter.py`, which writes `output/data/openrouter_result.json` and requires the environment variable `THALIA_OPENROUTER_API_KEY` to be set to a valid OpenRouter API key. Like all recorder outputs, an unavailable key is recorded as an isolated skip and the tracked aggregate remains unchanged; OpenRouter rows contribute only when available.

13.4 Provenance from Source to Rendered Output

The artifact inventory produced alongside this manuscript: the evaluation bundle (`eval_results.json`), the compiled configuration and trace (`compile_result.json`), the manuscript figures (rendered with a colourblind-safe palette [Wong, 2011]), the static dashboard, and the manuscript-variable map (`manuscript_variables.json`). Each is regeneratable and disposable; the source of truth is the code under `src/` and the fixed dataset in `src/evaluation/`. Together they provide an auditable record of which harness invariants held at build time. Deployment behaviour requires separate operational measurements.

The provenance chain is intentionally narrow: generated outputs serve as evidence; source code and fixed data → script output

→ manuscript token → hydrated manuscript → rendered artifact. That route is what prevents a polished PDF from becoming detached from the executable harness it describes.

13.5 Public archive and release identity

The source release and rendered manuscript are archived as one reproducibility package at Zenodo.

Release DOI: 10.5281/zenodo.21763245.

The canonical development history is maintained in the public GitHub repository. The DOI is a release identifier, not evidence of model-quality, integrity-advantage, or production-readiness claims. The archived package intentionally excludes raw answers, private external-volume records, credentials, and model weights; those lanes remain separately documented and gated.

The final local refusal boundary is `validate_artifacts.py`. It checks the generated manifest's relative paths, file sizes, and SHA-256 values against the current output tree and verifies the required paths declared by `domain_profile.yaml`. Freshness is established by running the validator after the last generator, so a later mutation or omitted artifact fails before rendering.

Memory-system comparisons remain in the cited design lineage and remain external to THALIA's result layer: MemGPT [Packer et al., 2023], Mem0 [Chhikara et al., 2025], TiMem [Li et al., 2026], and APEX-MEM [Banerjee et al., 2026] are related systems that motivate persistent memory interfaces, while the local result concerns the narrower append-only episodic store plus gated consolidation policy.

13.6 Evidence Scope and Honesty Boundaries

Two boundaries are stated wherever they are relevant (sec. 11): the deterministic extractive LM bounds answer token-F1, and the hashing-embedding semantic lane is a deterministic sub-word surrogate. Learned dense retrieval is a separate plug-in point. The bibliography records primary or official URLs where available; source-reported empirical claims retain their source status. A local aggregate explicitly records any THALIA replication.

14 Operability: Configuration, Sessions, Adapters, and CLI

THALIA is an operable design. This section documents the configuration system, the real language-model adapters, the persistent multi-turn session, and the command-line interface. Each has dedicated Zero-Mock coverage in the strict local suite, including a real in-process HTTP server for the network adapter. The suite covers the local tested path; deployment integrations require their own acceptance checks.

14.1 Configuration Precedence and Validation

The tunable parameter vector is :class:`HarnessConfig` (sec. 6.6). It is loadable, overridable, and serialisable:

- **File** — `load_config(path)` reads a harness YAML file (`config/harness.yaml` ships as the canonical default). The YAML may list fields flat or under a `harness:` key; manuscript `config.yaml` keeps renderer metadata separate and mirrors selected values only under `project_config.harness`.
- **Environment** — any field is overridable via `THALIA_<FIELD>`. Three example overrides are:
  - `THALIA_TOP_K=<integer>` for the retrieval depth.
  - `THALIA_LEXICAL_WEIGHT=<lexical>` for the lexical lane weight.
  - `THALIA_SEMANTIC_WEIGHT=<semantic>` for the semantic lane weight. Lane weights remain both-or-neither; setting only one against an unset pair raises. Two additional environment variables control the OpenRouter adapter: `THALIA_OPENROUTER_API_KEY` (required to activate the adapter; absent → null rows in evaluation output) and `THALIA_OPENROUTER_MODEL` (selects the OpenRouter model; the current recorded aggregate uses `tencent/hy3-preview`).
- **Round-trip** — `to_dict` / `from_dict`, `to_yaml` / `from_yaml`, and `save_config` make a config serialisable and reproducible; `from_dict` and `from_yaml` ignore unrecognised keys. `to_yaml` is the single serialisation path — `save_config` delegates to it, so the YAML representation stays identical from the dataclass.

Precedence, lowest to highest: built-in defaults → file → environment → explicit CLI flags.

This precedence order is part of the reproducibility story. A manuscript render can point to the checked-in `config/harness.yaml`; a local experiment can override one field through `THALIA_*`; and a CLI run can still make an explicit one-off choice without mutating source files. The resulting configuration is serialisable, so a compiled or manually selected setting can be saved and compared later.

14.2 Composable Pipeline and Stage Inspection

`AutoHarness` bundles the 4 runtime stages into one `forward` call. For callers that need to inspect or modify the stage chain — insert a logging hook, a custom reranker, or run a partial pipeline — `src/pipeline.py::Pipeline` exposes the same stages as an explicit, chainable sequence:

```
from src.pipeline import Pipeline

pipe = (
    Pipeline(config)
    .with_inspector()
    .add_step("log", lambda ctx: (ctx.extras.update({"n": len(ctx.evidence_windows)}), ctx)[1])
    .with_retriever()
    .with_reasoner()
    .with_memory_gate()
)
out = pipe.run(query="What is the wifi password?", session_history="...")
for step_name, ctx in pipe.intermediates:
    print(step_name, ctx.extras)
```

Each `with_*` method appends one stage and returns `self` for chaining; `add_step` inserts an arbitrary callable that receives and returns a `PipelineContext` (a mutable state bag with an `extras` dict for custom data). After `run`, `pipe.intermediates` holds a `(step_name, context)` snapshot after every step, so intermediate evidence counts, scores, and traces are inspectable without re-running the pipeline. `scripts/run_pipeline.py` exercises this path end-to-end and writes the intermediate-step trace to `output/data/`.

14.3 Resource Safety and Context-Manager Cleanup

`Session` and `EpisodicStore` both implement the context-manager protocol (`__enter__` / `__exit__`), so resources are always cleaned up — even when an exception propagates mid-turn:

```
with Session(db_path="episodes.db", memory_path="MEMORY.md") as s:
    s.ask("My wifi password is bluefish42")
    s.ask("What is the wifi password?")
# episodic store closed + MEMORY.md flushed automatically
```

The demo script (`src/demo.py`) uses this pattern, so the context-manager path is exercised by the default analysis pipeline as well as the test suite.

`AutoHarness` and `Pipeline` expose the same context-manager contract. They close only stores they created internally; a store supplied by the caller remains the caller's responsibility. `EpisodicStore.close()` is idempotent, so cleanup is safe in both nested context managers and explicit `finally` blocks. The strict suite runs with `ResourceWarning` promoted to an error to make a leaked SQLite connection fails the resource contract.

The standalone preflight reports independent readiness states for the deterministic core, DSPy SDK, Ollama server, Gemma model, embedding model, LongMemEval payload, and template renderer. Optional neural/benchmark states remain independent of deterministic operation; the fail-closed artifact validator remains the local release boundary.

14.4 Package Metadata and Type Distribution

The package exports `__version__` at the top level (`from src import __version__`) and ships a `py.typed` marker (PEP 561) so downstream type checkers see the full type annotations. `HarnessOutput.to_dict()` provides JSON serialisation on the dataclass itself — the CLI delegates to it, so the field mapping has one source of truth.

14.5 Language-Model Protocol and Adapters

The deterministic core uses the extractive responder, but a real model plugs in behind the one-method `LMClient` protocol:

- **`CallableLM(fn)`** — wrap any `Callable[[str], str]` (an SDK call, a local function). The smallest possible real adapter.
- **`OllamaLM(model, host)`** — call a local Ollama server's `/api/generate` endpoint using only the standard library (no third-party dependency). Failures raise `OllamaError`; the adapter emits no empty answer on error. The `is_available()` method offers a best-effort liveness probe. The adapter is opt-in and the default core leaves it unused.
- **`configure_openrouter(model, api_key)`** — configure the harness to use an OpenRouter-hosted model via the OpenRouter chat completions API. The model is controlled by the `THALIA_OPENROUTER_MODEL` environment variable; the API key is read from `THALIA_OPENROUTER_API_KEY`. The current recorded aggregate uses `tencent/hy3-preview`. This adapter is opt-in and raises on failure, preserving the error state.

Both are validated against a real HTTP server spun up in the test process — the adapter's request construction and response parsing are exercised for real, with no mocks.

The adapter boundary is narrow on purpose. A frontier API, local inference server, or evaluation wrapper only needs to implement `complete(prompt: str) -> str`. Everything else — evidence selection, citation construction, memory gating, configuration, and artifact generation — remains unchanged. This is what lets the deterministic manuscript distinguish method validation from model performance.

14.6 Persistent Multi-Turn Sessions

:class:`Session` wraps the harness with disk-backed state so a conversation survives process restarts: episodes persist to a SQLite file, the consolidated MEMORY.md is read at start and re-written on consolidation, and the running transcript is reconstructed from stored episodes so the Inspector can retrieve evidence from earlier turns. Reopening a session resumes turn IDs without collision (via the harness `start_turn`). A worked restart — store a fact, close, reopen in a fresh session, and retrieve the fact — is asserted in the test suite.

Session persistence also makes memory audit practical. The transcript can be reconstructed from raw episodes, while MEMORY.md remains a derived convenience layer. If a future answer depends on a remembered fact, the relevant raw turn can still be recovered and cited.

14.7 Query CLI and Configuration Resolution

`scripts/run_query.py` (logic in `src/cli.py`) runs a single query against the harness:

```
uv run --project . --extra dev python scripts/run_query.py \
  --query "What is the wifi password?" \
  --session-history "User: my wifi password is bluefish42" --json
```

For real haystacks, the same CLI exposes the opt-in adaptive Inspector controls:

```
uv run --project . --extra dev python scripts/run_query.py \
  --query "What is the wifi password?" \
  --session-file long_session.txt \
  --adaptive-windows --adaptive-window-ratio 5 --json
```

Flags cover the configuration knobs (`--top-k`, `--max-windows`, `--max-evidence`, `--rrf-k`, `--context-budget`), a `--config` file, inline `--session-history` or a `--session-file`, and `--json` for machine-readable output. The result reports the answer, detected task category, delivery mode, and the cited source indices — making the harness's evidence-grounding inspectable from the shell.

14.8 Extension Points and Replacement Lanes

The intended extension points are stable and narrow:

Extension	Interface	What changes	What stays fixed
Real LM	`LMClient.complete`	answer synthesis quality	Inspector, Retriever, citations, Memory Gate
Learned embedder	`SemanticIndex.top_k`	semantic ranking behaviour	lexical lane, RRF, budgets, evaluation protocol
Persistent deployment	`Session` paths	storage location and lifecycle	append-only episodes, consolidation policy
Search policy	`HarnessConfig` / compiler candidates	retrieval depth and weights	stage interfaces, metrics, manuscript tokens

This matrix is the operational version of the scope boundary. THALIA can be made stronger by replacing model and embedding components. Those replacements are experiments behind documented interfaces, and the deterministic method contract remains fixed.

15 Validation: Invariants, Negative Controls, and Integrity Gates

THALIA treats tested contracts as a build-time property grounded in evidence. Three layers check them: a Zero-Mock test suite, executable invariants, and golden regression tests for the deterministic core. Optional live-model adapters use separate model-dependent gates.

The validation design follows the same typed boundaries as the method. Unit tests exercise individual functions, stage tests exercise payload handoffs, regression tests pin published metrics, and script smoke tests demonstrate that the command surface can regenerate the artifacts named in the manuscript. This is why the manuscript can discuss Inspector windows, retrieval modes, Memory Gate actions, compiler traces, and dashboard rows as the same objects that appear in code rather than as parallel explanatory vocabulary. It validates the measurement instrument. Lexical overlap remains the declared answer proxy; factuality and model quality require their own outcomes.

The model-quality lane adds a separate fail-closed contract: deterministic pilot and held-out partitioning, typed ordinal and binary annotations, blinded model and condition packets, paired condition completeness, answer and provenance digests, independent double rating, adjudication, calibration, question-clustered paired statistics, and certificate-digest validation before promotion. The completed v2 campaign remains operational/provenance evidence; its answer hashes and automated metrics retain that evidence tier. Production readiness is tested by a separate operational acceptance gate.

15.1 Executable Invariants and Property Checks

`src/invariants.py` encodes structural properties the harness must always satisfy as runnable checks (each returns an :class:`InvariantResult`). The test suite asserts every one passes, and the dashboard (sec. 15.7) surfaces them:

- **bm25_idf_nonnegative** — every corpus term's idf is ≥ 0 (eq. 5).
- **rrf_monotonic_in_rank** — within a lane, the fused score is non-increasing in rank (eq. 7).
- **rrf_agreement_boost** — a chunk top-ranked in both lanes outranks single-lane chunks.
- **embedding_determinism** — identical text embeds identically.
- **mmr_lambda1_is_relevance** — MMR with $\lambda = 1$ reproduces the fusion order (eq. 8).
- **episode_append_only** — the episodic store rejects overwriting a turn id.
- **consolidation_gate_strict** — consolidation fires only when the Reasoner flag and an explicit criterion are both present.
- **harness_determinism** — the same query+history yields the same answer across fresh harnesses.
- **answer_grounded** — an answer cites at least one source window when evidence exists.
- **query_expansion_superset** — pseudo-relevance feedback retains every original query token when adding terms.
- **recall_readonly** — episodic recall is read-only and leaves the append-only store unchanged.
- **recency_zero_noop** — the default zero recency weight preserves retriever order exactly; a non-zero weight is separately tested as an active branch.
- **static_determinism_guard** — deterministic-core source has no forbidden network, clock, model-SDK, or mock-framework boundary violations.

These invariants form the executable minimum for the architecture: BM25 must preserve non-negative idf; RRF must respect rank order and agreement; MMR must reduce to relevance when requested; memory must remain append-only; and answers must carry citations when evidence exists. If any of those fail, the manuscript's methodological claims are no longer true.

15.2 Zero-Mock Test Suite

Every test uses real computation — real BM25/embeddings, a real SQLite episodic store, real temporary files, and a real in-process HTTP server for the Ollama adapter (no `unittest.mock`, no `MagicMock`). Coverage on `src/` is gated at 90% and the live suite runs well above it. Selecting a real fallback path (for example toggling `fts_available` to exercise the `LIKE` search branch) is permitted because it chooses a real branch, so the test exercises the production boundary.

The 90% gate covers the **deterministic core**. The optional real-DSPy and neural paths (`src/dspy_runtime/`, the `dspy.Predict`/`BootstrapFewShot` glue, and the real Gemma-via-Ollama backend) are exercised by genuine, `DSPY_AVAILABLE`- and model-availability-gated tests (`tests/test_dspy_runtime.py::TestRealDspyRuntime`, `TestRealNeuralDspy`) that run real dspy objects and a real model when present and skip cleanly otherwise — so they sit *outside* the coverage gate by construction; their real tests run when the dependencies are available. The gated figure shown in sec. 11.3.2 is produced from their recorded output.

The suite is intentionally organized by responsibility: retrieval files test lexical, semantic, fusion, MMR, expansion, and recall; stage files test Inspector, Retriever, Reasoner, Memory Gate, and harness orchestration; compiler files test metrics, MIPRO-style search, and GEPA-style mutation; manuscript tests enforce token coverage and stale-path hygiene. That organization keeps maintenance pressure close to the method it protects.

15.3 Golden Regression and Determinism

`tests/test_regression.py` pins the headline metrics (composite, citation-support proxy, efficiency) and the published manuscript tokens to golden values within a tight tolerance, and asserts that `run_full_eval()` and `generate_variables()` are byte-identical across runs. It also pins the **noise-regime ordering** — under session noise the hashing-surrogate semantic lane scores *above* the lexical and hybrid lanes (composite 0.692 vs. 0.620 at 32 distractors) — so the §evaluation prose that cites that divergence remains pinned to the measured data. An unintended change to any published number flips a golden assertion red before it can reach the manuscript.

15.4 Isolated Memory-Recall Validation

The episodic-first thesis gets a direct ablation alongside unit tests. In `src.evaluation.memory_recall_ablation`, a fact lives *only* in the append-only episodic store — empty session history, empty MEMORY.md — and a natural-language query recovers it (1 window) when the store is present and recovers 0 without it, isolating the store as the load-bearing component for cross-context recall. This probe also hardened a real defect: the store's FTS5 search passed the raw query to `MATCH`, so an ordinary punctuated question ("What is the wifi password?") raised an `fts5: syntax error`. Search now reduces queries to bare alphanumeric tokens (its tokenizer's own alphabet), and a regression test exercises punctuation, hyphens, and quotes so the recall path remains protected for natural-language input.

15.5 Null Results, Fallbacks, and Robustness

A trustworthy evaluation records null results alongside positive findings. Three follow-ups returned clear nulls. **(i) Recency weighting leaves the injection outcome unchanged:** re-running the context-poisoning probes (sec. 11.3.4) with `recency_weight` swept across its configured range leaves the poisoned answer unchanged — an overlap-dominated injection already wins across the tested recency range, so the sweep supplies no defensive effect. **(ii) Retrieval depth leaves answer token-F1 unchanged** on lexically clean tasks: sweeping `top_k`/`max_evidence` leaves token-F1 flat (the gold line is recoverable at depth one) and only lowers the efficiency term — exactly why the Compiler's optimum is an efficiency tie-break under this metric. **(iii) The lexical-anchored hybrid ties a learned-embedding RAG:** a pure neural-embedding retrieval baseline (`src.retrieval.learned.dense_rag_answer`, `nomic-embed-text`) matches the harness's answer token-F1 on the seeded set. A tie on this deliberately easy set is the *expected* outcome — sec. 11.2.1 shows the synthetic the data produce a retrieval-method tie. The tie supplies local parity evidence; general parity requires a harder task family. Consistent with the no-harness baseline (sec. 11.3.1), the measured harness value here is provenance and bounded context.

Robustness is locked in by a property-based fuzz pass (`tests/test_fuzz.py`): hundreds of adversarial inputs — random punctuation, SQL-like strings, injection markers, field-delimiter tokens, empty and oversized sessions — run through the harness and the FTS5-backed persistent session with **zero crashes** and no output-invariant violations. That test is the standing regression guard for the class of edge-case bug the earlier `fts5: syntax error` belonged to.

15.6 Reproducibility and Artifact Integrity Gates

`tests/test_manuscript_integrity.py` enforces two structural contracts that keep the reproducibility story honest. **Path hygiene**: no manuscript section or shipped doc may reference a project path that no longer exists — a stale path makes a documented `pytest` command collect zero tests and exit green, a silent false pass the gate now forbids. **Injection-independent token coverage**: the manuscript-variable generator locates the template root robustly even when THALIA is checked out as a symlink *outside* the template tree, and its token-coverage reproducibility contract completes independently of the optional PDF-injection step — the gate enforces every published number as a generated token and remains independent of renderer plumbing.

15.7 Static Dashboard and Visual Evidence

`src/dashboard.py` renders a self-contained HTML page (no JavaScript, no third-party dependency) embedding the headline metrics, the per-category breakdown, the invariant pass/fail table, and the generated figures. It is built by `scripts/build_dashboard.py` into `output/dashboard.html`. The renderer is a pure function of its inputs and HTML-escapes all dynamic content, so it is both testable and safe.

The dashboard is a compact audit surface for the same analysis. It shows the same metrics and figures used in this manuscript, alongside invariant status, so a reviewer can move from prose to rendered diagnostic to JSON artifact without needing to infer which run produced which number.

Before rendering, `scripts/validate_artifacts.py` adds a separate artifact gate: the manifest must cover the required outputs in `domain_profile.yaml`, and every recorded size and hash must still match the current file. This catches stale or mutated figures and JSON outside direct unit-test inspection.

16 Supplement: Canonical Notation and Statistical Contracts

This section is the notation single source of truth for the formal definitions in sec. 7. Subscripts identify the construct or unit; each canonical symbol has one meaning. In particular, S_{MMR} is the selected set, S_{eff} is the efficiency scale, and G_{sup} is the citation-support proxy. The hashing lane is the deterministic sub-word surrogate; learned dense representation is a separate optional lane.

16.1 Canonical Symbols and Object Types

Symbol	Meaning and scope
$T(x)$	Ordered lowercase alphanumeric token sequence for text x.
$U(x)$	Set of distinct tokens in $T(x)$.
$\mathcal{H}$	Raw session transcript supplied to the Inspector.
$\mathcal{M}$	Frozen memory snapshot supplied to one invocation.
θ	Immutable `HarnessConfig` governing one invocation.
W_I	Evidence-window sequence emitted by the Inspector.
μ_I	Inspector metadata, including patterns, indices, and coverage.
W_R	Ranked evidence sequence emitted by the Retriever.
δ_R	Retriever delivery decision: inline or file-based.
$\hat{y}$	Generated answer for the current query.
τ_N	Reasoner routing and evidence-use trace.
e	Episodic append result produced by the Memory Gate.
$\Delta_{\mathcal{M}}$	Optional derived memory delta.
I_θ	Inspector stage function.
R_θ	Retriever stage function.
N_θ	Reasoner stage function.
G_θ	Memory Gate stage function.
q	Current query text; $T(q)$ supplies the token sequence used by retrieval.
c_q	Task category classified from query q.
t	Turn identifier used by the append-only episodic store.
s	Session-ending indicator supplied to the Memory Gate.
b_N	Reasoner gate payload (r_N, x_N): consolidation flag and explicit memory-request indicator.
r_N	Reasoner's consolidation-request flag.
x_N	Explicit memory-request indicator derived from the query.
Σ_G	Memory Gate state containing the episodic store and current derived-memory text.
Θ	Candidate configuration surface searched by the Compiler.
$\mathcal{D}$	Declared finite evaluation set supplied to a Compiler run.
$\hat{\theta}$	First configuration attaining the maximum finite-set composite under the stable candidate order.
$\mathcal{T}_C$	Complete Compiler candidate-and-trace ledger for one search.
$\bar{M}$	Mean per-example composite score for a configuration on $\mathcal{D}$.
$\prec_\Theta$	Stable candidate order used to resolve equal Compiler scores.
d	Candidate document/window token sequence in BM25.
R	Ranked evidence collection used by the efficiency proxy.
A	Answer token-F1 component in the composite metric.
H	Indicator for at least one expected source index being cited.
$r_\ell(c)$	Zero-based rank of candidate c in retrieval lane ℓ.
B_b	Consecutive Inspector partition block with index b.
B	Evidence token budget used by the delivery decision.
Q_I	Distinct query-token set used by Inspector relevance.
S_{MMR}	Candidates already selected by greedy MMR.

Symbol	Meaning and scope
C_{cite}	Source indices cited by a generated answer.
E_{cite}	Joined excerpts attached to C_{cite}.
G_{gold}	Expected source-index set supplied by a finite evaluation example.
G_{sup}	Citation-support proxy: lexical overlap plus optional expected-source hit.
P_{cite}	Source-index citation precision: expected cited sources divided by all cited sources.
R_{src}	Source-index citation recall: expected sources represented in the citation set.
S_{eff}	Evidence-window scale used by the structural efficiency proxy.
E_{eff}	Structural efficiency score derived from ranked-window count.
M	Per-example composite score, a declared weighted sum of A, G_{sup}, and E_{eff}.
Y_q	A scalar outcome for analysis unit q, as declared by the analysis.
Δ_q	Paired outcome difference for unit q, left method minus right method.
p_{Holm}	Holm-adjusted paired-test p-value within the explicitly named comparison family; the current integrity screen's family is exploratory.
n_{obs}	Number of observed rows retained in a declared denominator, including failures when required.
n_{cluster}	Number of resampling or paired-randomization clusters.
$Z_{e,m,q}$	Binary endpoint for method m, comparison endpoint e, and matched unit q.
d_e	Direction code for endpoint e: $+1$ when higher is better and -1 when lower is better.
$\Delta_{e,q}^{\text{adv}}$	Direction-normalized paired effect: $d_e(Y_{e,L,q} - Y_{e,R,q})$, where positive values favor the left method.
$\widehat{\Delta}_e^{\text{adv}}$	Mean direction-normalized paired effect for endpoint e.
$\hat{p}_{e,m}$	Binary success rate for method m on endpoint e.
$\text{CI}_e^{\text{pair}}$	Paired percentile-bootstrap interval for a binary risk difference or continuous paired effect.
$\mathcal{F}_{\text{int}}$	Current exploratory integrity-screen family receiving one Holm step-down adjustment; a future P0 family is predeclared only after protocol freeze.
$\mathcal{Q}_{\text{audit}}$	Matched legacy injection probes used for the lexical cited-excerpt/source-exposure proxy.

The remaining constants are local to a displayed equation: k_1 and b are BM25 parameters, k is the RRF damping constant, λ is the MMR relevance weight, ρ is the recency weight, B is the evidence token budget, and D is the hashing-vector dimension. K is the PRF term cap and n is the ranked evidence count where the efficiency equation defines it. These names are local constants and remain distinct from the canonical symbols above.

16.2 Statistical Contracts and Interpretation Rules

The statistical layer reports finite-data summaries with explicit methods. A nominal confidence level labels the interval construction. Synthetic examples, finite seed-cluster counts, and finite local model probes remain finite-data evidence; target-population estimation requires a separate sampling design.

Continuous outcomes. For a declared observation vector Y_q, the point estimate is the arithmetic mean. Percentile bootstrap intervals resample rows with replacement; stratified intervals preserve each observed stratum size; and cluster intervals resample

whole clusters while retaining their rows. The reported interval is therefore a descriptive resampling interval. For a paired comparison, resampling operates on paired differences Δ_q; when the declared unit is a cluster, the point estimate is the mean of cluster-level means; this weighting keeps unequal cluster sizes from changing the declared estimand.

Binary outcomes. A binary rate is reported with the Wilson score interval. For paired binary outcomes, the exact two-sided McNemar p-value conditions on the discordant-pair count and uses the exact binomial tail under equal paired success probabilities. The continuous sign-flip diagnostic has a different estimand and exchangeability assumption.

For an endpoint with direction code d_e, the paired binary effect is $\widehat{\Delta}_e^{\mathrm{adv}} = d_e(\hat{p}_{e,L} - \hat{p}_{e,R})$. The implementation reports a paired percentile-bootstrap interval for the individual differences $d_e(Z_{e,L,q} - Z_{e,R,q})$ and Wilson intervals for the two marginal rates. The two interval constructions answer different questions: the paired interval describes the matched risk difference, while each Wilson interval describes one condition's success probability.

Paired continuous comparisons. The sign-flip procedure tests the observed mean absolute difference against sign assignments of Δ_q. Exhaustive enumeration is called exhaustive only when every assignment is enumerated. For larger samples the implementation uses seeded Monte Carlo sign flips with the plus-one correction. Both branches require the stated exchangeability/symmetry assumption and are reported as randomization diagnostics.

Multiplicity. If a comparison family contains m raw p-values, Holm's step-down adjustment sorts them, multiplies the rank-j value by $m - j + 1$, and applies the cumulative maximum, capped at one. The resulting values are p_{Holm}. FDR q-values use a different multiplicity procedure.

Accounting. Requested rows, completed rows, and failed rows are tracked separately. A failed requested row remains in n_{obs} when the protocol declares a requested-denominator estimand. Optional neural or external rows retain their own evidence tier. A missing model-dependent lane is reported as unrecorded; no value is imputed.

17 References and Source Records

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025. doi: 10.48550/arXiv.2507.19457. URL https://arxiv.org/abs/2507.19457.

Anthropic. Model context protocol specification 2025-06-18. https://modelcontextprotocol.io/specification/2025-06-18, 2025.

Ron Artstein and Massimo Poesio. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, 2008. doi: 10.1162/coli.07-034-R2. URL https://aclanthology.org/J08-4004/.

Pratyay Banerjee, Masud Moshtaghi, Shivashankar Subramanian, Amita Misra, and Ankit Chadha. APEX-MEM: Agentic semi-structured memory with temporal reasoning for long-term conversational AI. *arXiv preprint arXiv:2604.14362*, 2026. doi: 10.48550/arXiv.2604.14362. URL https://arxiv.org/abs/2604.14362.

Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336, 1998. doi: 10.1145/290941.291025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025. doi: 10.48550/arXiv.2504.19413. URL https://arxiv.org/abs/2504.19413.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 758–759, 2009. doi: 10.1145/1571941.1572114.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi: 10.1214/aos/1176344552.

Daniel Ari Friedman. A template/ approach to reproducible generative research: Architecture and ergonomics from configuration through publication. Zenodo, 2026. URL https://zenodo.org/records/20693013. Source repository: https://github.com/docxology/template.

Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010. doi: 10.1038/nrn2787.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.398. URL https://aclanthology.org/2023.emnlp-main.398/.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, pages 4885–4894, 2017. URL https://papers.neurips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Rame, Morgane Riviere, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. doi: 10.48550/arXiv.2503.19786. URL https://arxiv.org/abs/2503.19786.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *arXiv preprint arXiv:2302.12173*, 2023. doi: 10.48550/arXiv.2302.12173. URL https://arxiv.org/abs/2302.12173.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative

language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023. doi: 10.48550/arXiv.2310.03714. URL https://arxiv.org/abs/2310.03714.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich K"uttler, Mike Lewis, Wen-tau Yih, Tim Rockt"aschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020. doi: 10.48550/arXiv.2005.11401. URL https://papers.nips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, Chengbao Liu, and Jie Tan. TiMem: Temporal-hierarchical memory consolidation for long-horizon conversational agents. *arXiv preprint arXiv:2601.02845*, 2026. doi: 10.48550/arXiv.2601.02845. URL https://arxiv.org/abs/2601.02845.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, 2022. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, 2023. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12: 153–157, 1947. doi: 10.1007/BF02295996.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, 2023. doi: 10.18653/v1/2023.emnlp-main.741. URL https://aclanthology.org/2023.emnlp-main.741/.

Nous Research. Hermes agent, 2026a. URL https://github.com/NousResearch/hermes-agent. Repository source; no release or commit is claimed by this manuscript.

Nous Research. Hermes agent self-evolution: DSPy + GEPA evolutionary optimization, 2026b. URL https://github.com/NousResearch/hermes-agent-self-evolution. Repository source; no release or commit is claimed by this manuscript.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024. doi: 10.48550/arXiv.2402.01613. URL https://arxiv.org/abs/2402.01613.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. doi: 10.48550/arXiv.2310.08560. URL https://arxiv.org/abs/2310.08560.

Roger D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011. doi: 10.1126/science.1213847.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. ARES: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.20. URL https://aclanthology.org/2024.naacl-long.20/.

Sahil Sen, Akhil Kasturi, Elias Lumer, Anmol Gulati, and Vamse Kumar Subbiah. Is grep all you need? how agent harnesses reshape agentic search. *arXiv preprint arXiv:2605.15184*, 2026. doi: 10.48550/arXiv.2605.15184. URL https://arxiv.org/abs/2605.15184.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in LLM-as-a-judge. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 292–314. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.ijcnlp-long.18. URL https://aclanthology.org/2025.ijcnlp-long.18/.

SQLite Consortium. SQLite FTS5 Extension. https://sqlite.org/fts5.html, 2026. Full-text-search extension documentation.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.511. URL https://aclanthology.org/2024.acl-long.511/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022. doi: 10.48550/arXiv.2201.11903. URL https://arxiv.org/abs/2201.11903.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.

Bang Wong. Points of view: Color blindness. *Nature Methods*, 8(6):441, 2011. doi: 10.1038/nmeth.1618.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. doi: 10.48550/arXiv.2410.10813. URL https://arxiv.org/abs/2410.10813.

Fangyuan Xu, Yixiao Song, Mohit Iyyer, and Eunsol Choi. A critical evaluation of evaluations for long-form question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3225–3245, 2023. doi: 10.18653/v1/2023.acl-long.181. URL https://aclanthology.org/2023.acl-long.181/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. doi: 10.48550/arXiv.2210.03629. URL https://arxiv.org/abs/2210.03629.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025. doi: 10.48550/arXiv.2512.24601. URL https://arxiv.org/abs/2512.24601.

Dylan Zhang, Yanshan Lin, Zhengkun Wu, Yihang Sun, Bingxuan Li, Dianqi Li, and Hao Peng. Useful memories become faulty when continuously updated by LLMs. *arXiv preprint arXiv:2605.12978*, 2026. doi: 10.48550/arXiv.2605.12978. URL https://arxiv.org/abs/2605.12978.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023. doi: 10.48550/arXiv.2306.05685. URL https://arxiv.org/abs/2306.05685.