Continuous Knowledge Metabolism: Generating Scientific Hypotheses from Evolving Literature

Anonymous ACL submission

Abstract

Scientific hypothesis generation is not a one-shot act of ideation but a continuous process of updating an evolving knowledge state against new evidence. Existing LLM-based hypothesis-generation systems, however, fall short in two ways: they operate on static literature snapshots, and they are graded by contemporary judges, making it hard to test whether their outputs anticipate where a field later moves. To address these limitations, we introduce *Continuous Knowledge Metabolism* (CKM), an AI workflow that slices the literature into sliding windows and maintains a structured knowledge state through contrastive updates, generating hypotheses from *what just changed* in that state rather than from a one-shot snapshot. To grade such workflows, we adopt a predictive-validation protocol: hypotheses generated from literature up to time t are scored only against papers published strictly after t. On a 50-topic ML benchmark, CKM-LITE (our lightweight CKM variant) matches future literature on 72% of topics (36/50, vs. 30% for a one-shot batch baseline), with a 404-day mean lead and $\sim 15\times$ lower per-hypothesis token cost. The protocol also surfaces diagnostic trade-offs across instrumentation level, topic category, and change-trigger type; for instance, adding explicit change-detection instrumentation lifts mean lead (from 404 to 537 days) and originality, yet cuts topic coverage from 72% to 22%. We release CKM, the benchmark, and the judging protocol at https://anonymous.4open.science/r/ckm-hypogen-DE84.

1 Introduction

Consider how a machine-learning researcher tracks a fast-moving subarea over a few months. They read each new paper not in isolation but *against* an evolving knowledge state, noting what extends prior findings, what strengthens them, what contradicts them outright, and what bridges previously separate threads. After enough readings they can sometimes anticipate the shape of the next paper before it appears: not its title or authors, but its central insight, at the level of specificity a research plan would require. This continuous, contrastive integration of new evidence against an existing knowledge state is the cognitive operation behind scientific hypothesis generation in practice, and it is a key capability for the wave of automated-research systems (Lu et al., 2026; Juraj Gottweis and Natarajan, 2026; Weng et al., 2025) to operationalize at the literature level. A concrete instance from our benchmark (a 2024 hypothesis about RLHF stability matched by a paper fifteen months later) appears in Figure 3 (§5).

Literature-driven hypothesis generation has been studied for decades (§2). Recent LLM-based systems fall into two lines of work: static-snapshot idea/hypothesis generation (Wang et al., 2024; Baek et al., 2025; Hu et al., 2025; Yang et al., 2026) and end-to-end automated research (Lu et al., 2026; Weng et al., 2025; Juraj Gottweis and Natarajan, 2026). Both underemphasize what the classical Swanson-style work (Swanson, 1986; Smalheiser and Swanson, 1998) made central: a contrastive step against an *evolving* state, and a falsifiable test of whether outputs anticipate where the field is heading rather than rationalize where it already is. Neither line of work models the evolution of the literature, and neither grades hypotheses against papers appearing strictly *after* generation; validation is drawn from the same present-day population the system is meant to extend.

To address both gaps, we introduce *Continuous Knowledge Metabolism* (CKM), an AI workflow that implements the contrastive-integration step at the literature level. CKM slices an evolving corpus into sliding windows and performs two operations at each window. *First*, it updates a structured knowledge state through a contrastive edit (Eq. 2) that tags each new finding as adding, confirming, contradicting, or bridging existing claims, the same

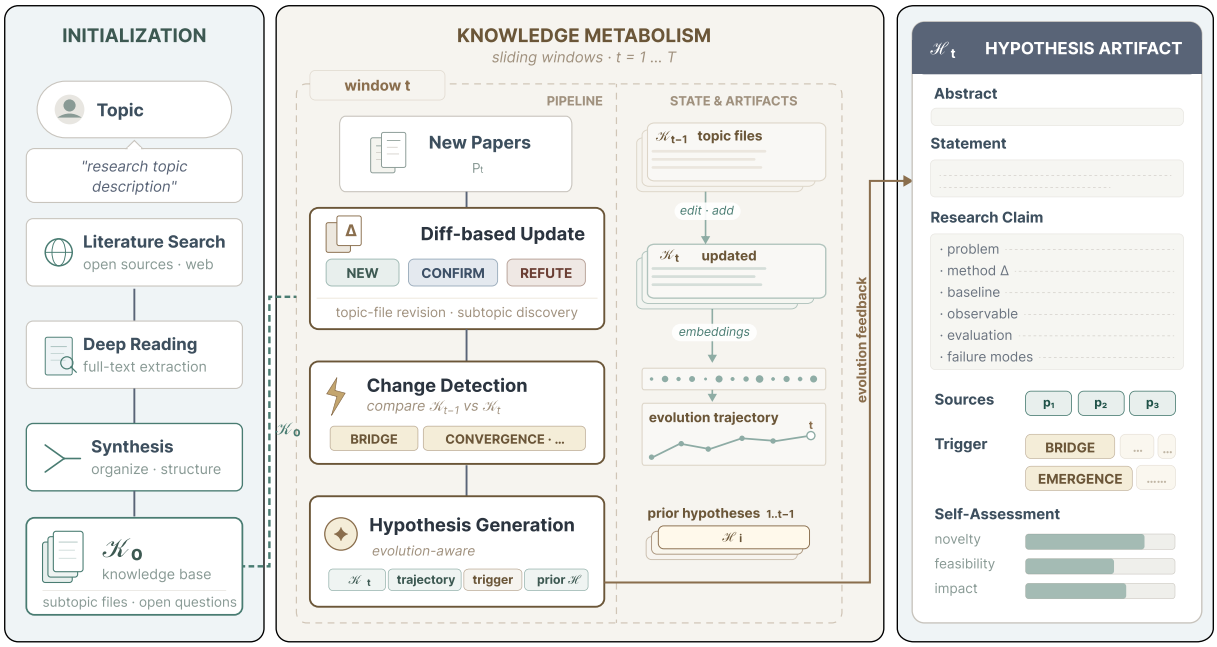

Figure 1: The CKM workflow. *Initialization* reads pre-window papers into a structured knowledge state $\mathcal{K}_0$. Each *evolution* window t updates $\mathcal{K}_{t-1} \to \mathcal{K}_t$ with new papers and emits structured hypotheses $\mathcal{H}_t$. CKM-LITE implements the core cycle; CKM-FULL adds an explicit change-detection probe (dashed) that we use as an analytical lens (Appendix L).

categories the researcher in our opening example sorts by hand. *Second*, it generates hypotheses conditioned on what just *changed* in the state rather than on a static snapshot. *To evaluate* whether the workflow anticipates research direction rather than only rationalizing it post hoc, we adopt a falsifiable protocol that grades each hypothesis against papers published strictly after the generation window. The two artifacts are co-developed: CKM is the cognitive workflow, and predictive validation against future literature is the falsifiability test that makes its success or failure measurable.

To summarize, we make the following contributions:

1. We propose **CKM**, a sliding-window workflow that updates a structured knowledge state through contrastive edits and generates hypotheses from what just changed (§3). On a 50-topic ML benchmark, CKM-LITE matches future literature on 72% of topics at $\sim 15\times$ lower per-hypothesis token cost than a one-shot batch baseline.

2. We propose a **predictive-validation protocol** that grades each hypothesis against papers published strictly after the generation window, with cross-provider judging to reduce same-model self-preference (§4). We release the 50-topic benchmark, prompts, frozen artifacts, and judging code so other workflows can be scored under the same falsifiability test.

3. We provide **robustness analyses** confirming the headline results hold under stricter judging thresholds, content-shuffled negative controls (0 HIT across 110 shuffled pairs), and per-hypothesis token-efficiency comparisons (§5; Appendices G, H, K).

2 Related Work

Literature-driven hypothesis generation has a long history. Swanson (1986) bridged disjoint biomedical literatures through manual co-citation analysis, and systems like ARROWSMITH (Smalheiser and Swanson, 1998) scaled the idea with information retrieval over decades of MEDLINE; these classical systems made the contrastive step explicit but were limited to surface co-occurrence over static corpora. Recent LLM-based work falls into two lines, discussed below.

Idea and hypothesis generation from static snapshots. A first line of work prompts LLMs for novel proposals against a fixed retrieval pool: SciMON (Wang et al., 2024), ResearchAgent (Baek et al., 2025), Nova (Hu et al., 2025), Scideator (Radensky et al., 2024), and Chain-of-

Ideas (Li et al., 2025) on the idea side; Zhou et al. (2024), Yang et al. (2024), MOOSE-Chem2 (Yang et al., 2026), and Xiong et al. (2024) on the hypothesis side. Validation is contemporary (automated scoring plus human review, agent peer review, Swiss-style tournaments, LLM critique, or known-target matching), and the corpus the system sees is treated as a snapshot rather than an evolving stream, so what just *changed* between time windows is never an explicit conditioning signal.

End-to-end automated research with contemporary signals. A second line closes the loop from idea to drafted paper: AI Scientist (Lu et al., 2026), Dolphin (Yuan et al., 2025), CycleResearcher (Weng et al., 2025), EvoScientist (Yougang Lyu, 2026), and the AI Co-Scientist (Juraj Gottweis and Natarajan, 2026). Validation is again contemporary: human reviewers (Si et al., 2025), agent peer-review loops (Weng et al., 2025), simulated tournaments (Hu et al., 2025), or self-evaluation (Lu et al., 2026). Wet-lab assays (Juraj Gottweis and Natarajan, 2026) provide the strongest correctness signal in this set but are limited to biomedicine. None of these regimes tests whether generated hypotheses align with literature appearing strictly after generation, so the verdict is drawn from the same present-day population the system is meant to extend.

Adjacent work. Adjacent work includes temporal knowledge graphs (Cai et al., 2024, 2022) at the fact-and-triple level and parameter-level knowledge editing (De Cao et al., 2021; Mitchell et al., 2022). IdeaBench (Guo et al., 2025) and LiveIdeaBench (Ruan et al., 2026) score generated ideas under minimal context, again with contemporary judges. For broader surveys of LLM-driven scientific discovery, see (Zhang et al., 2024; Reddy and Shojaee, 2025).

CKM differs from these systems along two axes. *On generation*, a sliding-window state $\mathcal{K}_t$ is updated incrementally and contrastively, so each window's hypotheses condition on what just *changed* in the literature rather than on a static snapshot. *On validation*, instead of asking a contemporary judge whether an output looks good, we ask whether subsequent literature actually pursued the predicted direction. A per-system comparison of validation regimes is given in Appendix A (Table 3).

3 The CKM Workflow

CKM converts a stream of dated papers into a sequence of structured hypotheses, each scored against papers published strictly after the generation window. The workflow itself has two phases, described in this section: (1) *initialization* builds a baseline knowledge state from pre-window literature; and (2) *metabolism* carries that state forward window by window while emitting hypotheses. A separate validation protocol (§4) then grades the hypotheses against future literature. CKM-LITE runs the lean metabolism cycle; CKM-FULL adds an explicit change-detection probe between state update and generator (Figure 1).

3.1 Initialization

Given a topic q and an initialization window $[t_{\text{init}}, t_0]$ that precedes the evolution period, initialization retrieves up to 48 papers (denoted $\mathcal{P}^{\text{init}}$), extracts structured per-paper notes (core methods, key findings, datasets, open questions) through a full-text reading step, and organizes them into Markdown *topic files* partitioned by sub-theme to produce the initial knowledge state $\mathcal{K}_0$:

$$\mathcal{K}_0 = \text{Init}\big(q,\, \mathcal{P}^{\text{init}}\big). \tag{1}$$

The state is stored as structured natural-language notes rather than a vector index because the downstream contrastive update (Eq. 2) requires editing existing claim text in place, which a chunk-level retrieval store does not naturally afford. The sub-theme partition is induced by the LLM at initialization rather than imposed from a fixed taxonomy, since no single taxonomy fits every topic. In practice the LLM produces 6–17 sub-themes per topic (median 9), each backed by its own Markdown topic file that serves as the per-call edit unit during the metabolism cycle (§3.2).

3.2 Metabolism: incremental, contrastive update

The evolution period $[t_0, t_T]$ is divided into T sliding windows of width Δt (we use $\Delta t = 2$ months, $T = 6$ in experiments). For window t, let $\mathcal{P}_t$ denote the new in-window papers. The CKM-LITE cycle for $t = 1, \ldots, T$ is

$$\mathcal{K}_t = \text{Update}(\mathcal{K}_{t-1},\, \mathcal{P}_t), \tag{2}$$

$$\mathcal{H}_t = \text{Gen}(\mathcal{K}_t,\, \mathcal{S}_{1:t-1},\, \mathcal{H}_{1:t-1}), \tag{3}$$

where $\mathcal{K}_t$ is the knowledge state after window t, $\mathcal{H}_t$ is the set of hypotheses generated in window t,

Operation	Trigger condition	Edit to $\mathcal{K}$
ADD	finding f is not covered by any existing claim	append new claim with f's citation, time-stamped to window t
CONFIRM	f supports existing claim c	attach f's citation to c, increment c's evidence count, append window-stamp
CONTRADICT	f challenges existing claim c	add a contradiction tag to c with f's citation and a scoping condition; do not delete c
BRIDGE	f connects two existing claims/sub-themes	create a cross-link with f's citation; optionally lift a method from one sub-theme into another

Table 1: The four contrastive update operations in CKM. Every edit is window-stamped, so $\mathcal{K}_t$ retains a typed record of what changed in window t rather than overwriting prior state. The Update prompt requires the LLM to classify each finding into exactly one operation before applying the edit.

$\mathcal{S}_{1:t-1}$ is a cross-window trajectory summary aggregating older trends, and $\mathcal{H}_{1:t-1} := \bigcup_{i=1}^{t-1} \mathcal{H}_i$ is the set of all hypotheses generated in prior windows.

Update is implemented as one LLM call per topic file (sub-theme): each call edits its topic file in place using only the new papers assigned to that sub-theme ($\leq$ 8 papers per call out of the $\leq$ 96 in-window pool), applying one of the four operations in Table 1 (ADD, CONFIRM, CONTRADICT, BRIDGE) to each new finding. Every edit is window-stamped and typed, so $\mathcal{K}_t$ itself exposes the current-window changes; the summary $\mathcal{S}_{1:t-1}$ is updated *after* Gen for use in the next cycle (Algorithm 1). Gen therefore reads what just changed (from window-stamped edits in $\mathcal{K}_t$) and what changed earlier (from $\mathcal{S}_{1:t-1}$), without redundancy.

The empirical cost discipline of this cycle is captured by the BATCH baseline (§4.3), which concatenates all in-window papers into one long prompt at each cycle and uses $\sim 15\times$ more tokens per generated hypothesis (§5). The typed operations in Table 1 make the contrastive separation explicit at edit time, the same categories the researcher in our opening example was sorting by hand.

Conditioning each hypothesis on prior hypotheses $\mathcal{H}_{1:t-1}$ is a smaller but important choice: without it, a direction that takes several windows to mature would be re-proposed as soon as it appears in $\mathcal{K}_t$, inflating yield without coverage. The generation prompt enforces non-repetition by requiring each new hypothesis to differ from earlier ones in its named method delta. The contrastive cycle is lossy by design (each window's $\mathcal{P}_t$ is compressed into edits on $\mathcal{K}$ rather than stored), so per-window cost grows with what *changed* rather than with the corpus; the predictive-validation protocol of §4 is the empirical check on whether the retained state preserves enough signal.

3.3 CKM-FULL: an instrumented analytical variant

CKM-FULL augments the cycle at two points. First, it adds a change-detection probe between Update and Gen:

$$\tau_t = \Theta\big(\Delta(\mathcal{K}_{t-1}, \mathcal{K}_t)\big), \tag{4}$$

where $\Delta(\mathcal{K}_{t-1}, \mathcal{K}_t)$ denotes the pre/post-update diff (the set of typed edits applied during window t), Θ classifies this diff into a categorical change-type signal τ_t, and τ_t is then appended to the generation context. Second, CKM-FULL's Update classifies each finding into one of the operations in Table 1 via a dedicated per-finding classification call, instead of CKM-LITE's single in-place edit-and-commit call that applies the operations implicitly. CKM-FULL is therefore an instrumented variant rather than a single-component ablation: it externalizes per-finding operation labels during Update *and* adds the Θ probe. We use it diagnostically (Appendix L); the name reflects "fully instrumented", not better performance. As the results in §5 show, the added instrumentation narrows the generator's distribution and trades coverage for per-hit alignment.

3.4 Hypothesis artifact and implementation

Each generated hypothesis $h \in \mathcal{H}_t$ is a JSON object with a one-sentence *statement*, a *research claim* (problem, method delta, baseline, expected observable, evaluation plan, failure mode), source-paper citations, and a trigger label; the full schema is in Appendix E.3. Structure matters for downstream judging because the protocol of §4 grades hypotheses field by field, whereas a paragraph-length brainstorm aligns inconsistently. Each Gen call emits at most three schema-valid hypotheses, so cross-system yield differences in Table 2 reflect schema-validity rates rather than tunable sampling. All Gemini-side calls use temperature 0.7 and all OpenAI-side judge calls use temperature 0.1; full

Algorithm 1 CKM cycle over one topic.

Require: topic q, init range $[t_{\text{init}}, t_0]$, window width Δt, horizon T

$\mathcal{K}_0 \leftarrow \text{Init}(q, \mathcal{P}^{\text{init}})$; $\mathcal{S}_0 \leftarrow \emptyset$; $\mathcal{H} \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  $\mathcal{P}_t \leftarrow$ fetch in-window papers
  $\mathcal{K}_t \leftarrow \text{Update}(\mathcal{K}_{t-1}, \mathcal{P}_t)$ {contrastive, in-place edit}
  if CKM-FULL: $\tau_t \leftarrow \Theta(\Delta(\mathcal{K}_{t-1}, \mathcal{K}_t))$
  $\mathcal{H}_t \leftarrow \text{Gen}(\mathcal{K}_t, \mathcal{S}_{1:t-1}, \mathcal{H}_{1:t-1}\,[, \tau_t])$
  $\mathcal{S}_t \leftarrow$ summary of $\mathcal{K}_{t-1} \rightarrow \mathcal{K}_t$; $\mathcal{H} \leftarrow \mathcal{H} \cup \mathcal{H}_t$
end for
return $\mathcal{H}, \mathcal{K}_T$

model, prompt, retrieval, and reproducibility details are in Appendices C–E and D.

3.5 Design rationale

Three design decisions distinguish this workflow from an end-to-end LLM call per topic. First, the phases are split externally rather than collapsed into one prompt, so the cross-window trajectory becomes a readable artifact rather than a hidden activation; this is what lets the generator condition on what *changed* between windows rather than on a snapshot of what is. Second, initialization is separated from metabolism so the first sliding window already conditions on a structured baseline of what the field knew *before* the evolution period, otherwise metabolism cannot distinguish what changed from what merely is. Third, the update operation is contrastive rather than additive: each new finding is tagged with one of four typed operations against the existing state, retaining the contrastive shape of the cognitive operation rather than letting the generator re-infer it from concatenated text.

4 Predictive Validation Protocol and Benchmark

4.1 Protocol

A generated hypothesis is a *hit* if at least one paper published strictly after the generation window scores ≥ 6.0 on a four-dimension judge rubric (research-direction alignment, technical-approach overlap, problem-formulation match, contribution anticipation; each scored 1–10, HIT iff average ≥ 6.0). We use *matched hypothesis* (not *validated*) to mark predictive alignment with future literature rather than formal scientific proof. The rubric requires a 6 to mean *"same problem with substantively overlapping method delta or evaluation target, not merely topic-level relatedness"*; full rubric and judge prompt in Appendix E.4. The protocol requires three pieces:

(i) Frozen temporal cutoffs. Each topic is split into three temporally disjoint phases. *Initialization* (≤ 48 papers from 2019–2023) builds $\mathcal{K}_0$ before the simulated deployment year. *Evolution* consists of six 2-month windows over 2024 (≤ 96 papers per window), which update the knowledge state and emit hypotheses; hypotheses are frozen at the window in which they are generated. *Validation* uses papers from the held-out post-evolution period, beginning Jan. 1, 2025 and continuing through the frozen benchmark crawl date in early 2026. Thus generation never observes validation papers.

(ii) Cross-provider judging. Generation uses Gemini-2.5-Flash (Gemini Team, 2025). For tractability, each hypothesis is first paired with the top-30 validation papers by embedding similarity within the same topic; only these candidate pairs proceed to LLM judging. Judging is two-stage and cross-provider: GPT-4o-mini (OpenAI, 2024) pre-filters at threshold 5.0, then GPT-4o re-judges surviving candidates with the 6.0 hit threshold. Both judges are from a different model family than the generator, reducing the risk that a hit merely reflects a model family judging its own outputs. We discuss the residual judge-side limitations (including possible LLM-family-wide stylistic preferences, the top-30 candidate window, and pre-filter false negatives) in §6.

(iii) Topic eligibility. A topic is admitted to the benchmark only if its arXiv corpus satisfies ≥ 30 papers in 2019–2023, ≥ 50 papers in the evolution period, and ≥ 30 papers in the validation period . The filter ensures every topic has enough validation material that "zero hits" is a property of the workflow, not of corpus availability. Topics were selected before any system was executed and held fixed across all experimental groups; no topic was added or removed based on preliminary results.

4.2 Benchmark

We selected 50 ML topics spanning eight categories (full list and selection in Appendix B): multilingual & speech (4), LLM core capabilities (10), LLM applications (8), safety & trust (4), multimodal (3), domain-specific AI (6), efficiency &

System	Hyp. /Topic ↑	Hit Rate ↑	Coverage ↑	Tokens /Topic ↓
CKM-LITE	17.3	**6.4%**	**72%**	**0.51 M**
CKM-ABSTRACT	17.2	3.8%	44%	1.52 M
BATCH	13.7	3.1%	30%	5.93 M
CKM-FULL	**17.8**	1.3%	22%	5.87 M

Table 2: Main results across 50 ML topics. **Hit Rate** is the pooled hit rate over all hypotheses (55/863, 33/858, 21/684, 12/892). **Coverage** is the fraction of topics with at least one matched hypothesis (36/50, 22/50, 15/50, 11/50); this is the primary deployment metric, since a portfolio user scoping 4–5 candidate directions needs at least one credible direction per topic. Arrows mark preferred direction; bold = best per column.

data (7), and other foundations (8). The mix is deliberate: a workflow that claims to track continuous knowledge evolution should behave differently in slow-moving subareas (machine translation, information retrieval) than in fast-moving ones (chain-of-thought, RLHF), and a benchmark dominated by either kind would mask the dependence.

4.3 Experimental groups

All groups use the same initialization and validation pipeline, differing only in state instrumentation and per-paper input content.

CKM-LITE is our primary deployment configuration: the lean incremental cycle of Eq. (2)–(3), using full source as input. It applies the contrastive update operations within a single edit-and-commit prompt, but does not run the explicit change-detection probe or emit separate per-finding classification records. **CKM-FULL** adds explicit instrumentation to CKM-LITE: per-finding change-typing during Update and the change-detection probe of Eq. (4). We use it as an analytical lens rather than a deployment configuration (§3). **CKM-ABSTRACT** is the abstract-only counterpart of CKM-FULL, with identical instrumentation but per-paper inputs restricted to abstracts. This tests full-text versus abstract-only inputs within the instrumented setting. **BATCH** is the non-incremental long-context control: at each cycle, all in-window papers are concatenated into one prompt and hypotheses are generated without state carryover.

5 Results

5.1 Main comparison

Table 2 summarizes performance across the four experimental groups on 50 topics.

Sliding-window state improves topic coverage. A researcher scoping a portfolio of candidate directions needs the system to surface at least one credible hypothesis per topic; coverage is the metric that maps to that need. CKM-LITE achieves $36/50 = 72\%$ coverage against $15/50 = 30\%$ for BATCH, more than doubling the baseline. On the primary binary coverage metric, the paired discordance is 21–0 in CKM-LITE's favor: CKM-LITE exclusively covers 21 topics where BATCH produces no hit, while BATCH covers no topic missed by CKM-LITE. Per-topic hit-rate differences are also significant under a Wilcoxon paired test ($n = 50$, $W = 182.5$, $|Z| = 2.73$, $p = 0.006$, $r = 0.44$ after dropping 12 ties). Thus the gain is not merely a higher pooled hit rate, but broader topic coverage.

Sliding-window state improves hits while reducing token cost. CKM-LITE produces more hits than BATCH (55 vs. 21 matched hypotheses) at $\sim 11\times$ lower per-topic token cost (0.51M vs. 5.93M). Per generated hypothesis (the natural output unit for a hypothesis-generation workflow), the ratio is $\sim 15\times$ ($\sim$ 30K tokens per hypothesis for CKM-LITE vs. $\sim$434K per hypothesis for BATCH). Whether a budget-matched BATCH (truncated to fit a smaller context window) would close the gap is an additional baseline we discuss in §6 and leave for follow-up work; the version reported here uses each workflow's natural per-window context budget, since context-budget choice is itself part of the structural design being compared.

CKM-FULL's cost comes from bundled instrumentation, not from a single probe. CKM-FULL's mean per-topic cost (5.87M) is more than $11\times$ CKM-LITE's (0.51M) and nearly equal to BATCH's (5.93M). The gap is not from Θ alone: CKM-FULL's extra cost comes from (a) the change-detection probe Θ reading the entire pre/post-update state pair per window, (b) explicit per-change-type categorization at Update time, and (c) the trajectory-conditioned Gen prompt that carries the typed signals forward. The cost-source breakdown is itemized in Appendix K. This is also why CKM-FULL is positioned as an analytical lens, not a deployment configuration (§3).

Threshold tightening preserves a nontrivial high-confidence hit set. CKM-LITE's 55 hits are not concentrated at the 6.0 threshold edge: tightening the threshold to 7.0 still leaves 14 hits

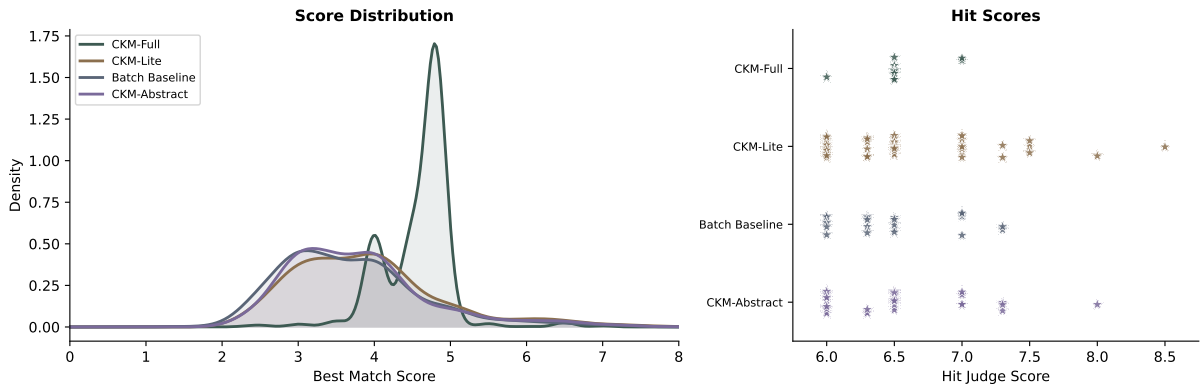

Figure 2: Best-match alignment score distribution across the four systems. CKM-FULL (green) concentrates in 4–5; CKM-LITE (orange) has a broader distribution with a thicker right tail crossing the hit threshold of 6.0; BATCH (gray) and CKM-ABSTRACT (purple) lie between the two extremes. The trade-off is between consistently-near hypotheses and a working population of threshold crossings (F1 in Appendix L).

across 12 topics (24% coverage), and to 7.5 leaves 5 hits across 4 topics. Mean temporal lead is also threshold-independent across 6.0–7.5 (404–446 days), so the predictive content of matched hypotheses is largely a property of the alignments rather than of the operating point (Appendix G).

Negative-control judge audit rules out the strongest forms of judge laxity. To test whether the 55 hits could reflect over-permissive judging, we re-judged every hit under three conditions with the same four-dimension prompt: **real** (the original verdict), **within_cat** (each hypothesis re-paired with the matched paper of another hit *from the same category*), and **cross_topic** (re-paired globally). Across the 110 shuffled pairs, zero re-judged as HIT, with mean per-pair scores around 2.0 versus ~ 6 on real pairs (a ~ 4-point gap). A stress test that allows up to 12 samples per pair leaves shuffled pairs at $0/110$ while real pairs recover to $48/55 = 87.3\%$, evidence that the misses on real pairs are threshold-edge variance rather than content disagreement. The audit rules out the strongest forms of category-level and content-agnostic judge laxity (Appendix H); subtler family-wide stylistic biases are discussed in §6.

Hits arrive well in advance of the paper they match. Across CKM-LITE's 55 hits, the mean generation-to-publication lead is 404 days (median 399, IQR 284–535, range 66–757). The median is essentially equal to the mean, indicating the average is not pulled by a few outliers but reflects the actual middle of the distribution. Per category, the lead is structurally consistent: 385 days for LLM core, 524 for LLM applications, 494 for domain-specific AI, 277 for safety & trust (Appendix I). Matched hypotheses therefore precede their paired papers by roughly a research cycle across diverse subareas. Lead times are measured from the hypothesis's generation cutoff to the matched paper's earliest arXiv submission date, not its later venue publication date, to avoid double-counting submission-to-camera-ready delays (Appendix K).

RLHF Stability Trigger: CONVERGENCE · Generated Nov 2024 · Hit ✓ 460-day lead

"A novel RLHF framework will integrate adaptive entropy regularization (H-DPO/SEE-DPO style) with theoretically grounded KL-regularization and dynamic uncertainty-aware policy optimization, demonstrating enhanced stability and reduced reward hacking compared to standard DPO or PPO-based RLHF."

Matched (Feb 2026) → *SAFE: Stable Alignment Finetuning with Entropy-Aware Predictive Control for RLHF*. The matched paper develops entropy-aware RLHF stabilization with KL-style regularization and predictive/uncertainty-aware policy control, matching the hypothesis at the level of method composition.

Figure 3: One CKM-LITE hit and the future paper it matched. Generation saw only pre-November-2024 literature; the matched paper appeared 15 months later. The hypothesis fused four pre-window papers (H-DPO, SEE-DPO, sharp KL analysis, and adversarial uncertainty estimation) along an axis that subsequent work converged on. Additional cases in Appendix N.

5.2 One hit, end to end

To make the predictive-validation protocol concrete, Figure 3 shows one of CKM-LITE's 55 hits in the benchmark, selected for illustration because

it pairs a structured hypothesis with a future paper whose title is unusually transparent about the alignment, not because it is the highest-scoring hit. The hypothesis was generated from a knowledge state restricted to pre-November-2024 literature and named three specific design components (adaptive entropy regularization, KL-regularization, and dynamic uncertainty-aware policy optimization) jointly. The matched paper appeared 460 days later with a title encoding the same three components; the hit verdict, however, is scored from the paper's body under the four-dimension H1–H4 rubric (Appendix E.4), not from title overlap alone. This is the operational meaning of "hit" under the protocol: the hypothesis is neither so general that any future paper would match it, nor so narrowly tied to a specific paper's wording that the match is rhetorical.

The RLHF case is typical of the hit set along its main compositional axes. Across CKM-LITE's 55 hits, the median hypothesis fuses three pre-window source papers into a single named direction (range 1–6), so the RLHF case (four sources) sits at the modal fusion depth rather than the tail. The matched paper population also spans all eight benchmark categories, with no category accounting for more than a quarter of the hits, so the 72% headline coverage is not concentrated in a small slice (Appendix I). Aggregate hit counts, lead-time distribution, and two further matched cases of different trigger types (BRIDGE, GAP_EXPLOITATION) are in §5 and Appendix N; frozen judging code is released so matching can be re-run.

5.3 Instrumentation–coverage trade-off between CKM-LITE and CKM-FULL

CKM-FULL is an instrumented variant of CKM-LITE we use to probe how explicit change-tagging shapes hypothesis character. Its hit rate and coverage drop sharply (1.3% vs. 6.4% pooled; 11/50 vs. 36/50), while mean temporal lead lengthens (537 vs. 404 days) and an independent novelty judge scores it higher on originality and cross-field synthesis (D1 5.64 vs. 4.20; D2 6.62 vs. 5.78; Appendix F). Figure 2 shows the resulting trade-off: CKM-FULL concentrates around 4–5, whereas CKM-LITE produces a broader right tail crosses the hit threshold. Under the same instrumentation, CKM-ABSTRACT improves coverage from 22% to 44%, showing that the bundled instrumentation does not interact monotonically with input length. We therefore use CKM-FULL as an analytical lens and CKM-LITE as the deployment configuration, leaving per-component factorization to §6.

What the evidence together makes more and less likely. The five results paragraphs of this section give a partial answer to the question §1 asked: does CKM anticipate where the field is heading rather than rationalize where it already is? On the 50-topic benchmark, CKM-LITE matches a future paper on 72% of topics with 404-day mean lead times and a 6.4% pooled hypothesis-level hit rate, the negative-control audit constrains the strongest forms of judge laxity, and the threshold sweep preserves a 24% coverage subset at ≥ 7.0. None of these checks is by itself decisive, but together they make the most natural alternative explanations less likely: judge over-fit (constrained by the 0/110 shuffle audit), threshold gaming (constrained by the sensitivity sweep), token-spend confound (constrained by the $\sim 15\times$ per-hypothesis efficiency gap against BATCH), and parametric leakage from the generator (the matched validation papers post-date Gemini-2.5-Flash's January 2025 training cut-off; Appendix D). What predictive validation cannot tell us is whether the matched directions were scientifically valuable, well-executed, or insightful; that is the boundary line between the empirical claim of this paper and the design-time bar future work would need to clear.

6 Conclusion

We introduced CKM, a sliding-window workflow that integrates new findings contrastively into an evolving knowledge state, paired with a predictive-validation protocol that grades hypotheses against future papers. On 50 ML topics, CKM-LITE produces at least one future-paper match on 72% of topics, with a 404-day mean lead and lower token cost than the batch baseline. The protocol surfaces diagnostic trade-offs between instrumentation, coverage, and alignment (Appendix L). Beyond this workflow, predictive validation against future literature is a tractable falsifiability test for LLM-driven scientific-discovery systems; whether matched directions were scientifically worth pursuing remains for domain-expert review. The protocol's role is to make the temporal question tractable so that expert audit can focus on a filtered candidate set rather than the full generated population. We release the benchmark, judging protocol, and reference implementation, hoping future-literature evaluation of LLM-driven research tools becomes one of NLP's new defaults rather than an exception.

Limitations

All quality assessments (hit verification, novelty, alignment) rely on LLM judges. Cross-provider judging and the negative-control audit (Appendix H) rule out the strongest forms of category-level laxity, but cannot exclude family-wide stylistic biases. The two-stage pre-filter at threshold 5.0 may also introduce false negatives, making reported hit counts a lower bound on what a single-stage judging pipeline might report. Predictive validation rewards trajectory-aligned predictions and structurally penalizes novel directions the field has not yet visited; we therefore treat hits as a falsifiable signal of anticipation, not as a measure of scientific value, novelty, or insight. The empirical evidence is ML/NLP-specific, although the protocol is domain-independent in principle. We do not run external hypothesis-generation systems end-to-end on the released benchmark at this scale, and a budget-matched BATCH ablation that isolates context-budget effects remains future work. Finally, the diagnostic trade-offs of Appendix L are descriptive rather than calibrated predictors, and CKM-FULL's bundled instrumentation is not factorized into single-component ablations.

Ethical considerations

The benchmark draws on dated metadata and full-text snapshots from arXiv; we do not redistribute paper content beyond the citation metadata and per-paper extraction notes required to reproduce the workflow, and generated hypotheses cite their source papers. The hypotheses scored in this paper were generated by an LLM-driven workflow that is itself the object of study (§3); AI coding assistants were used for routine implementation under author review, and no AI authored prose presented as our own. We see three risks specific to LLM-driven hypothesis generation: *spurious authority* (structured outputs inheriting the gravity of cited sources without their evidence), *idea-piling* (saturating subareas with low-cost candidates that crowd out human-led directions), and *attention concentration* (predictive validation amplifying already-popular subareas), discussed in Appendix M. The full 50-topic benchmark takes ∼44 hours and ∼$130 in API spend (Appendix D), keeping it accessible to researchers without industrial-scale compute. We did not collect data from human subjects or annotators; all hypothesis quality scoring is performed by LLM judges, whose limitations are discussed in §6.

References

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. 2025. Researchagent: Iterative research idea generation over scientific literature with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6709–6738.

Borui Cai, Yong Xiang, Longxiang Gao, He Zhang, Yunfeng Li, and Jianxin Li. 2022. Temporal knowledge graph completion: A survey. *arXiv preprint arXiv:2201.08236*.

Li Cai, Xin Mao, Yuhao Zhou, Zhaoguang Long, Changxu Wu, and Man Lan. 2024. A survey on temporal knowledge graph: Representation learning and applications. *arXiv preprint arXiv:2403.04782*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506.

Google Gemini Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2503.21218.

Sikun Guo, Amir Hassan Shariatmadari, Guangzhi Xiong, Albert Huang, Myles Kim, Corey M Williams, Stefan Bekiranov, and Aidong Zhang. 2025. Ideabench: Benchmarking large language models for research idea generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5888–5899.

Xiang Hu, Hongyu Fu, Jinge Wang, Yifeng Wang, Zhikun Li, Renjun Xu, Yu Lu, Yaochu Jin, Lili Pan, and Zhenzhong Lan. 2025. Nova: An iterative planning framework for enhancing scientific innovation with large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21330–21359.

Alexander Daryin Tao Tu Petar Sirkovic Artiom Myaskovsky Grzegorz Glowaty Felix Weissenberger Alessio Orlandi Dan Popovici Anil Palepu Keran Rong Ryutaro Tanno Khaled Saab Fan Zhang Jacob Blum Andrew Carroll Kavita Kulkarni Nenad Tomašev Dina Zverinski Ivor Rendulic Elahe Vedadi Florian Hasler Luka Rimanic Marina Boia Ivan Budiselic Ben Feinstein Mathias Bellaiche Tom Sheffer Jan Freyberg Jeremy Ratcliff Ottavia Bertolli Katherine Chou Avinatan Hassidim Burak Gokturk Amin Vahdat Yuan Guan Vikram Dhillon Eeshit Dhaval Vaishnav Byron Lee Tiago R. D. Costa José R. Penadés Gary Peltz Yossi Matias James Manyika Demis Hassabis Yunhan Xu Pushmeet Kohli Annalisa Pawlosky Alan Karthikesalingam Juraj Gottweis, Wei-Hung Weng and Vivek Natarajan. 2026. Accelerating scientific discovery with co-scientist. *Nature*, pages 1–3.

Long Li, Weiwen Xu, Jiayan Guo, Ruochen Zhao, Xingxuan Li, Yuqian Yuan, Boqiang Zhang, Yuming Jiang, Yifei Xin, Ronghao Dang, Yu Rong, Deli Zhao, Tian Feng, and Lidong Bing. 2025. Chain of ideas: Revolutionizing research via novel idea development with LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 8971–9004, Suzhou, China. Association for Computational Linguistics.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. 2026. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations*.

OpenAI. 2024. GPT-4o System Card. *Preprint*, arXiv:2410.21276.

Marissa Radensky, Simra Shahid, Raymond Fok, Pao Siangliulue, Tom Hope, and Daniel S Weld. 2024. Scideator: Human-llm scientific idea generation grounded in research-paper facet recombination. *arXiv preprint arXiv:2409.14634*.

Chandan K. Reddy and Parshin Shojaee. 2025. Towards scientific discovery with generative AI: Progress, opportunities, and challenges. In *Proceedings of the 39th Annual AAAI Conference on Artificial Intelligence*, volume 39, pages 28601–28609.

Kai Ruan, Xuan Wang, Jixiang Hong, Hao Sun, Peng Wang, and Yang Liu. 2026. Evaluating LLMs' divergent thinking capabilities for scientific idea generation with minimal context. *Nature Communications*.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2025. Can LLMs generate novel research ideas? a large-scale human study with 100+ NLP researchers. In *The Thirteenth International Conference on Learning Representations*.

Neil R Smalheiser and Don R Swanson. 1998. Using arrowsmith: a computer-assisted approach to formulating and assessing scientific hypotheses. *Computer methods and programs in biomedicine*, 57(3):149–153.

Don R. Swanson. 1986. Fish oil, raynaud's syndrome, and undiscovered public knowledge. *Perspectives in Biology and Medicine*, 30(1):7–18.

Qingyun Wang, Doug Downey, Heng Ji, and Tom Hope. 2024. SciMON: Scientific inspiration machines optimized for novelty. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–299, Bangkok, Thailand. Association for Computational Linguistics.

Yixuan Weng, Minjun Zhu, Guangsheng Bao, Hongbo Zhang, Jindong Wang, Yue Zhang, and Linyi Yang. 2025. Cycleresearcher: Improving automated research via automated review.

Guangzhi Xiong, Eric Xie, Amir Hassan Shariatmadari, Sikun Guo, Stefan Bekiranov, and Aidong Zhang. 2024. Improving scientific hypothesis generation with knowledge grounded large language models. *arXiv preprint arXiv:2411.02382*.

Zonglin Yang, Xinya Du, Junxian Li, Jie Zheng, Soujanya Poria, and Erik Cambria. 2024. Large language models for automated open-domain scientific hypotheses discovery. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13545–13565.

Zonglin Yang, Wanhao Liu, Ben Gao, Yujie Liu, Wei Li, Tong Xie, Lidong Bing, Wanli Ouyang, Erik Cambria, and Dongzhan Zhou. 2026. Moose-chem2: Exploring llm limits in fine-grained scientific hypothesis discovery via hierarchical search. *Advances in Neural Information Processing Systems*, 38:89045–89076.

Xinhao Yi Yuyue Zhao Shuyu Guo Wenxiang Hu Jan Piotrowski Jakub Kaliski Jacopo Urbani Zaiqiao Meng Lun Zhou Xiaohui Yan Yougang Lyu, Xi Zhang. 2026. Evoscientist: Towards multi-agent evolving ai scientists for end-to-end scientific discovery. *arXiv preprint arXiv:2603.08127*.

Jiakang Yuan, Xiangchao Yan, Bo Zhang, Tao Chen, Botian Shi, Wanli Ouyang, Yu Qiao, Lei Bai, and Bowen Zhou. 2025. Dolphin: moving towards closed-loop auto-research through thinking, practice, and feedback. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21768–21789.

Yu Zhang, Xiusi Chen, Bowen Jin, Sheng Wang, Shuiwang Ji, Wei Wang, and Jiawei Han. 2024. A comprehensive survey of scientific large language models and their applications in scientific discovery. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8783–8817, Miami, Florida, USA. Association for Computational Linguistics.

Yangqiaoyu Zhou, Haokun Liu, Tejes Srivastava, Hongyuan Mei, and Chenhao Tan. 2024. Hypothesis generation with large language models. In *Proceedings of the 1st Workshop on NLP for Science (NLP4Science)*, pages 117–139, Miami, FL, USA. Association for Computational Linguistics.

Appendix

The appendix consists of the following further discussion:

- **A** Extended related work and validation-regime comparison table.
- **B** 50-topic benchmark and dataset construction.
- **C** Models, infrastructure, frozen artifacts, compute cost.
- **D** Reproducibility paths and leakage controls.
- **E** Four core prompt templates.
- **F** Novelty-judge rubric (D1–D4) and per-system scores.
- **G** Hit-threshold sensitivity sweep.
- **H** Negative-control judge audit.
- **I** Per-category lead-time breakdown.
- **J** Pairwise Wilcoxon tests and Cohen's d.
- **K** Token-efficiency breakdown by phase.
- **L** Three diagnostic trade-offs (F1–F3).
- **M** Extended discussion and paradigm implications.
- **N** Two additional matched-hit case studies.

A Related Work (Extended)

Table 3 extends the prose discussion of §2 with a per-system comparison of validation regimes and the temporal frame on which truth signal is sourced.

B Benchmark Topics and Dataset Construction

Data source: arXiv. We build the benchmark from arXiv preprints because arXiv provides public v1 timestamps, broad ML coverage, and stable paper identifiers. We use each paper's earliest arXiv submission date as its canonical publication date, rather than later venue publication dates, to define temporal splits and measure lead times. Per-paper records were fetched via the arXiv API; full-text systems use LaTeX source when available and fall back to abstracts otherwise, while CKM-ABSTRACT always uses abstracts. We release paper identifiers, topic assignments, prompts, frozen outputs, and judging artifacts so the corpus can be re-crawled and the evaluation reproduced.

Temporal construction. Each topic is partitioned into three temporally disjoint phases matching the main protocol. *Initialization* uses papers dated from 2019–2023 to build $\mathcal{K}_0$. *Evolution* uses papers from calendar year 2024, organized into six 2-month sliding windows, and emits hypotheses that are frozen at the window in which they are generated. *Validation* uses papers from the held-out post-evolution period, beginning Jan. 1, 2025 and continuing through the frozen benchmark crawl date in early 2026. Thus no validation paper is available to the generation phase. For lead-time analysis, each match is measured from the hypothesis's own generation cutoff to the matched paper's v1 arXiv date.

Topic selection and eligibility. The 50 benchmark topics span eight ML categories selected for heterogeneous evolution rates, so the workflow is stress-tested under both stable subfields and fast-moving recent LLM topics. The candidate topic list was fixed before any system was run, drawn to cover the eight ML categories listed below, and then filtered only by corpus-size criteria: the underlying arXiv corpus must satisfy $\geq$ 30 papers in 2019–2023 for initialization, $\geq$ 50 papers in the 2024 evolution period, and $\geq$ 30 papers in the post-2024 validation period. Topics that fail any criterion are excluded. No topic was added or removed based on preliminary results. Topic queries, corpus counts, paper caps, and deterministic selection rules are included in the released benchmark artifacts.

Multilingual & speech (4): low-resource machine translation; cross-lingual transfer for low-resource languages; multilingual large language models; low-resource speech recognition.

LLM core capabilities (10): chain-of-thought reasoning in LLMs; long-context understanding in LLMs; factual consistency in LLMs; prompt engineering and in-context learning; instruction tuning for LLMs; complex reasoning for AI agents; tool-using agents with LLMs; accuracy of automated evaluation for LLMs; reinforcement learning from human feedback; code generation with LLMs.

LLM applications (8): AI for software engineering; security of code LLMs; document understanding and information extraction; aspect-based sentiment analysis; relation extraction from text; open-domain question answering with LLMs; diversity evaluation in text generation; conversational information retrieval.

Safety & trust (4): explainability and interpretability of neural networks; adversarial robustness of deep learning models; fairness and bias mitigation in machine learning; out-of-distribution detection for language models.

System	Output	Validation regime	Temporal frame
SciMON (Wang et al., 2024)	ideas	auto + human review	concurrent
ResearchAgent (Baek et al., 2025)	ideas+plans	agent peer review	concurrent
Nova (Hu et al., 2025)	ideas	Swiss tournament	concurrent
Scideator (Radensky et al., 2024)	ideas	human-in-the-loop	concurrent
Chain-of-Ideas (Li et al., 2025)	ideas	LLM critique	concurrent
AI Co-Scientist (Juraj Gottweis and Natarajan, 2026)	hypotheses	wet-lab (biomed)	in-domain assay
MOOSE-Chem2 (Yang et al., 2026)	hypotheses	known-target match	known answer
AI Scientist (Lu et al., 2026)	papers	self-evaluation	concurrent
Dolphin (Yuan et al., 2025)	papers	benchmark empirical	benchmark
CycleResearcher (Weng et al., 2025)	papers	simulated review	concurrent
EvoScientist (Yougang Lyu, 2026)	papers	peer review + experts	concurrent
CKM (ours)	**hypotheses**	**future-paper alignment**	**post-cutoff literature**

Table 3: Representative LLM-based scientific-discovery systems and the validation regimes they use, with the temporal frame on which truth signal is sourced. Prior systems span concurrent judging (human, agent, LLM, simulated review), in-domain experimental assays, known-target matches, and benchmark scoring; the wet-lab assay of AI Co-Scientist is the strongest correctness signal in this set when applicable, though limited to biomedicine. CKM contributes future-paper alignment as a complementary regime operating on a different temporal frame (post-cutoff literature), intended as an addition to rather than a replacement for the other regimes.

Multimodal (3): vision-language models and multimodal learning; visual question answering with multimodal models; text-to-image generation and diffusion models.

Domain-specific AI (6): clinical natural language processing; machine learning for drug discovery; deep learning for medical image analysis; protein structure prediction with deep learning; surrogate modeling in physics-informed machine learning; AI for hypothesis generation in science.

Efficiency & data (7): efficient fine-tuning of large language models; knowledge distillation for small language models; model compression and pruning for neural networks; mixture of experts routing for language models; data filtering for domain-specific models; synthetic data quality evaluation; data augmentation for natural language processing.

Other foundations (8): continual learning and catastrophic forgetting; domain adaptation and transfer learning; federated learning for language models; knowledge graph completion and reasoning; graph neural networks for natural language processing; active learning for natural language processing; deep learning for recommendation systems; user experience evaluation for large language models.

C Implementation Details

Table 4 lists all models used in the experiments, their roles, and configuration details.

Infrastructure. All API calls were routed through a single proxy endpoint. Papers were sourced from arXiv with full-text extraction via LaTeX-source parsing where available, falling back to abstract-only when full text was inaccessible. Embedding-based candidate pre-filtering used cosine similarity with top-30 candidates per hypothesis. Total compute cost for the full evaluation (3 systems $\times$ 50 topics) was approximately $130 USD.

Frozen artifacts. All hypotheses, knowledge states, judge verdicts, and per-topic logs are stored as frozen artifacts in the released reference repository (anonymous release link in the OpenReview supplementary submission). Every number in Table 2 of the main text is recomputable from these artifacts *without further API calls*, via the released `recompute.py` script.

D Reproducibility

Reproducibility comes in two flavours, which we distinguish:

Recomputing reported numbers. Every number in Table 2, the lead-time distribution, the Wilcoxon statistic in §5, the per-trigger and per-category breakdowns, and the audit numbers can be *recomputed without any further API calls* from the frozen artifacts in the released reference repository. Each per-topic record in the released JSON summaries, one for each experimental group, contains `duration`, `total_generation_tokens`,

Role	Model	Provider	Temp.
Initialization (Init)	Gemini-2.5-Flash (Gemini Team, 2025)	Google	0.7
Knowledge update (Update)	Gemini-2.5-Flash	Google	0.7
Change detection (Θ, CKM-FULL only)	Gemini-2.5-Flash	Google	0.7
Hypothesis generation (Gen)	Gemini-2.5-Flash	Google	0.7
Hit verify (pre-filter)	GPT-4o-mini (OpenAI, 2024)	OpenAI	0.1
Hit verify (re-judge)	GPT-4o (OpenAI, 2024)	OpenAI	0.1
Keyword extraction	GPT-4o	OpenAI	0.1
Validation profiles	GPT-4o	OpenAI	0.1
Novelty judge	GPT-4o (separate prompt, run blind to system identity)	OpenAI	0.1
Embedding (candidate filter)	text-embedding-3-small	OpenAI	—

Table 4: Models used across all experiments. Generation uses one provider (Google) and judging uses another (OpenAI) to reduce same-model self-preference. Hit verification uses a two-stage pipeline: GPT-4o-mini pre-filters candidates (threshold ≥ 5.0), and GPT-4o re-judges those that pass (threshold ≥ 6.0). The novelty judge uses the same model as hit re-judge but a separate prompt with no exposure to the hit verdict; we acknowledge in §6 that this leaves novelty and hit judging within a single provider family.

`yield`, `hit_rate`, and `best_match_score`; the released `recompute` script reads those records and prints every reported value, including the Wilcoxon statistic.

Full pipeline re-runs. A complete re-run of the workflow from raw topics requires API access. CKM ships as an installable plugin (anonymized release link in the OpenReview supplementary submission), and a single-topic run takes three commands: installing the plugin, launching its host runtime, and invoking the workflow's entry-point command with a topic name (e.g., "long-context modeling"). Defaults match the benchmark settings of §4: 2-month windows, ≤ 96 papers per window, Gemini-2.5-Flash for generation, and the GPT-4o-mini/GPT-4o two-stage pipeline for judging. Because metabolism-side LLM calls use a nonzero sampling temperature (Table 4), exact end-to-end re-runs do *not* reproduce the same hypothesis text bitwise; they should be read as stochastic replications, while the headline numbers in Table 2 remain recomputable from the frozen artifacts without re-running the pipeline.

Leakage controls and residual risks. We distinguish three channels: *in-context leakage*, *judge self-preference*, and *generator pretraining exposure*. The in-context channel is controlled by construction: generation prompts contain only initialization papers and evolution papers available up to the current 2024 window, while validation papers are drawn from the held-out post-evolution period beginning Jan. 1, 2025 (§4). Thus validation papers are never included verbatim in the generation context.

Cross-provider judging mitigates a different risk: same-model or same-family self-preference in evaluation. Generation uses Gemini-2.5-Flash, while hit verification uses GPT-4o-mini and GPT-4o, reducing the chance that a hit is merely a model family judging its own outputs. This does not by itself eliminate generator pretraining exposure.

Generator training cutoff. Gemini-2.5-Flash, used for all metabolism-side calls, has a publicly reported training-data cutoff of January 2025 (Gemini Team, 2025), approximately coinciding with the end of our 2024 evolution period. Validation papers appearing after the reported cutoff fall outside the generator's reported training data, while near-boundary cases are retained but treated as residual retrospective-benchmarking risk in §6. This caveat is specific to retrospective benchmarking with contemporary LLMs. In prospective deployment, the generator would be queried as a fixed model snapshot before later papers exist, so those papers cannot enter its training data. Our protocol therefore targets the same deployment-time question: whether hypotheses generated from literature available during the evolution period align with research that appears later.

E Prompt Templates

We provide the four core prompts used in CKM's knowledge metabolism cycle. All metabolism-side prompts use Gemini-2.5-Flash at the temperatures listed in Table 4. The released reference implementation contains the prompts verbatim; this section reproduces them in an abbreviated form for the printed paper.

E.1 Diff-based knowledge update

The update prompt processes each topic file against the new papers of one window and applies one of the four operations from Table 1 (ADD / CONFIRM / CONTRADICT / BRIDGE). The key design choice is the contrastive instruction that forces explicit categorization of each finding.

System Prompt

```
You are a research knowledge base updater. Update a single topic file based on new papers. Work strictly from the provided content; no hallucination. Cite arxiv_id for every claim.
```

User Prompt (abbreviated)

```
Topic file: {topic_name}
Period: {period}

Current content: {topic_content}
New papers this period: {papers_text}

For EACH new paper, apply exactly one operation:
- Add: new finding/method absent from current content → append claim with citation
- Confirm: supports an existing claim → attach evidence and note increased confidence
- Contradict: refines/rebuts an existing claim → rewrite preserving both positions; mark [Revised]
- Bridge: cross-domain connection → note the link

Each edit is window-stamped ([t={period}]). Size control: keep the file under 200 lines; when approaching the limit, compress older entries.
```

E.2 Change detection (CKM-Full only)

This prompt compares knowledge states before and after a window's update to produce the trigger signal τ_t of Eq. (4).

System Prompt

```
You are a research intelligence analyst monitoring a knowledge base for changes. Characterize what changed in the latest update: what type of change occurred and what is notable. Be factual and precise. Return valid JSON only.
```

User Prompt (abbreviated)

```
Knowledge State BEFORE: {knowledge_before}
Knowledge State AFTER: {knowledge_after}
New Papers ({period}): {papers_text}

Classify the change type:
1. Incremental: routine extension of existing themes
2. Contradiction: revises established finding
3. Convergence: independent papers point to same conclusion
4. Bridge: connection between previously unrelated topics
5. Trend_Confirmed: earlier pattern is validated by new evidence

Return JSON: {change_type, reason, key_changes}. The LLM may occasionally emit additional labels (e.g., Gap_Exploitation, Non_Obvious_Bridge); we retain these as-is and report all types with n ≥ 5 in Appendix L.
```

E.3 Hypothesis generation

The hypothesis prompt receives the full evolution context and produces structured, schema-valid JSON hypotheses. Yield is bounded to at most three hypotheses per call by the schema instruction (§3.4).

User Prompt (abbreviated)

```
You have been tracking "{topic}" for {n_windows} periods.

Knowledge Evolution Trajectory: {evolution_trajectory}
Current Knowledge State (with window-stamped tags): {knowledge}
New Papers ({period}): {papers_text}
What Changed (CKM-Full only): {trigger_type}, {trigger_reason}
Previously Generated Hypotheses: {existing_hypotheses}

Generate at most three hypotheses that are conceptually novel and distinct from any previously generated hypothesis. Consider:
(1) contradictions & tensions in the state;
(2) non-obvious bridges between sub-themes;
(3) trend extrapolation;
(4) gap exploitation;
(5) cross-paper synthesis (must yield insight beyond the sum of parts).

Avoid hypotheses that merely integrate existing methods. Each hypothesis MUST include the schema fields: statement (1 sentence), research_claim ({problem, method_delta, baseline, expected_observable, evaluation_plan, failure_mode}), source_papers (arxiv_id list), trigger (one of the change types above), and self_assessment ({novelty, feasibility, impact}, 1-10 each).
```

E.4 Hit-verification judge prompt

The hit-verification judge grades each (hypothesis, candidate-paper) pair on a four-dimension rubric (D1–D4, each 1–10); the per-pair score is the per-dimension average, with HIT iff average ≥ 6.0. The pre-filter (GPT-4o-mini, threshold ≥ 5.0) and re-judge (GPT-4o, threshold ≥ 6.0) share the same prompt skeleton. The input includes the paper's full text where available (truncated to a fixed character budget) rather than the abstract alone.

System Prompt

```
You are a rigorous scientific reviewer judging whether a published paper substantively matches a previously generated research hypothesis. Score each dimension strictly according to the rubric. Superficial topic overlap does NOT constitute a match; you must find specific, concrete alignment in the paper's actual content.
```

User Prompt (abbreviated)

```
## Hypothesis: {hypothesis_content}
## Candidate Paper: title / arxiv_id / published / full paper text (or abstract fallback)

Score each dimension 1-10 with a one-line
```

```
evidence quote:
H1 - Research Direction Alignment: same gap/problem motivation.
5-6: same general direction, divergent framing.
7-8: same problem and motivation; hypothesis anticipated this gap.
H2 - Technical Approach Overlap: similar method/architecture.
5-6: similar design principles, different execution.
7-8: similar architecture; key design choice anticipated.
H3 - Problem Formulation Match: same task / I/O / dataset / constraints.
5-6: same broad task; different dataset or protocol.
7-8: same task with minor setup differences.
H4 - Contribution Anticipation: paper's reported result matches hypothesis prediction.
5-6: right direction, some detail mismatch.
7-8: specific type and scale of contribution anticipated.
Output: Avg Score: (H1+H2+H3+H4)/4 / Verdict: HIT if avg ≥ 6.0 else MISS / Reasoning: 2-3 sentences citing evidence.
```

F Novelty Judge Scoring Dimensions (D-Score)

The novelty judge is independent of the generation model and run blind: it receives only the hypothesis text and not the identity of the generating system. It scores each hypothesis on four 1–10 dimensions, labelled **D1–D4** below (distinct from the hit-verification dimensions H1–H4 in Appendix E.4).

- **D1 – Conceptual Originality.** Does the hypothesis propose a genuinely new idea, or is it a predictable combination of existing methods? 1–3: incremental extension of a single paper. 4–6: novel combination or non-obvious application. 7–10: fundamentally new concept or surprising cross-domain connection.
- **D2 – Cross-field Synthesis.** Does the hypothesis meaningfully bridge distinct research areas? 1–3: single-subfield work. 4–6: connecting related but distinct areas (e.g., NLP + IR). 7–10: bridging fundamentally different domains (e.g., language-evolution theory + LLM agent design).
- **D3 – Gap Identification Precision.** Is the research gap clearly articulated and supported by specific evidence? High score requires naming specific prior works, identifying what they cannot do, and explaining why the gap matters. Vague claims ("improve efficiency") score low.
- **D4 – Specificity & Falsifiability.** Does the hypothesis specify a concrete task, baseline, expected outcome, and evaluation plan? A score of 8+ requires named models (e.g., "LLaMA-2-7B"), datasets (e.g., "Davis, KIBA"), metrics (e.g., "RMSE, Hits@1"), expected effect sizes, and explicit failure conditions.

	D1 Orig.	D2 Cr.-f.	D3 Gap	D4 Fal.
CKM-LITE	4.20	5.78	6.21	**7.86**
CKM-ABSTRACT	4.76	6.22	6.19	7.75
CKM-FULL	**5.64**	**6.62**	6.59	7.84
BATCH	4.77	6.22	**6.75**	7.71

Table 5: Per-hypothesis novelty by dimension (1–10), independent novelty judge. Bold = best per column.

The system-level means used in §5 are reproduced in Table 5: the largest gap between CKM-FULL and CKM-LITE is on D1 (+1.44), followed by D2 (+0.84); D3 is closer (+0.38); D4 is essentially identical (∼7.8 across all systems), confirming that specificity is driven by the shared output schema rather than the metabolism pipeline.

G Hit-Threshold Sensitivity

The protocol's 6.0 hit threshold (§4) sits at the midpoint of the four-dimension judge rubric (Appendix E.4) and was fixed before any system was run. A common reviewer concern is that the headline coverage of 72% is driven by hypotheses sitting right at the 6.0 edge and would collapse under a stricter threshold. Table 6 recomputes coverage, hit counts, and mean lead time for CKM-LITE at progressively tighter thresholds. Coverage decays gradually rather than collapsing, lead time stays in the 404–446 day band, and a nontrivial high-confidence subset survives at $\geq$ 7.0 (24% topic coverage, 14 hits).

Threshold	Hits	Coverage	Cov. %	Mean lead (d)
≥ 6.0	55	36/50	72%	404
≥ 6.5	28	21/50	42%	446
≥ 7.0	14	12/50	24%	429
≥ 7.5	5	4/50	8%	440
≥ 8.0	2	2/50	4%	350

Table 6: CKM-LITE sensitivity to the hit threshold. The protocol's 6.0 threshold is the operating point chosen to balance volume against confidence; tightening to 7.0 retains 24% topic coverage, demonstrating the hit set is not a thin film at the threshold edge. Lead time is essentially threshold-independent (404–446 days), indicating temporal predictive power is a property of the matched alignments, not of the threshold setting.

H Negative-Control Judge Audit

To test whether the 55 hits in Table 2 could be artefacts of an over-permissive LLM judge, we re-judged every hit under three conditions using the same hit-verification prompt from Appendix E.4 (four-dimension scoring on full paper text or abstract fallback, GPT-4o, threshold ≥ 6.0 on the per-dimension average). The same hypothesis is held fixed; only the paper it is paired with changes:

- **real**: the original hit pair (independent re-judge of the original verdict).
- **within_cat**: each hypothesis is re-paired with the matched paper of a randomly chosen other hit *from the same topic category* (derangement with category fixed, seed 42). Tests whether the judge accepts within-subarea but content-mismatched pairs.
- **cross_topic**: each hypothesis is re-paired with the matched paper of a random other hit *globally* (derangement without category constraint). Tests whether the judge accepts content-mismatched pairs at all.

Protocol. The audit uses GPT-4o at temperature 0.1 on every (hypothesis, paper) pair from the three conditions; the rubric is the four-dimension hit-verification prompt of Appendix E.4, applied to the same full-text-or-abstract source the original lite judge saw. We report two complementary settings: a *single-pass* re-judge (one sample per pair, the primary audit), and a *stress test* that re-judges each pair up to 12 times at the same temperature and accepts HIT if any sample crosses the 6.0 threshold. The stress test is *not* a re-statement of the production protocol (which uses a single pass); it is a permissive upper-bound test of whether repeated sampling could ever push shuffled controls into HIT.

Real-pair concordance reflects threshold-edge variance. A direct re-judge concordance of 63.6% on real pairs reflects threshold-edge sampling variance in the LLM judge: of the 20 real pairs that did not re-cross the 6.0 threshold on a single pass, 13 scored 5.0–5.99 – within a single point of the threshold and well above any shuffled pair (which never exceeds 4.0 on either control). The judge is stochastic at temperature 0.1, so per-dimension sample noise can move the per-pair average by $\sim$0.3–0.5 points across runs, which is the regime that produces these flips at the threshold edge. The stress-test row in Table 7 confirms this interpretation: when each pair is given up to 12 chances to cross the threshold, real pairs recover from 63.6% to 87.3% while shuffled controls stay at $0/110$. We treat the single-pass concordance as a property of LLM-as-judge stochasticity at the threshold edge and report it as a limitation (§6); the audit re-judge is not a re-definition of ground truth, and the original hit set is not relabelled by it.

What the audit rules out. What this audit helps rule out is the strongest version of the "LLM judge over-credits superficially related pairs" objection. On a single pass, shuffled pairs do not become hits at all ($0/110$ across both conditions), and the mean-score gap of $\sim$4 points between real (5.95) and shuffled ($\sim$2.0) exceeds anything that would be expected if the judge were ignoring content. The stress test strengthens this: even when each shuffled pair is given up to 12 opportunities at the same temperature, none of the 110 shuffled pairs crosses the threshold on any sample. The headline coverage and lead-time numbers in Table 2 therefore provide evidence that the reported matches are not merely an artifact of category-level or content-agnostic judge laxity. The audit cannot exclude subtler family-wide LLM biases (e.g. stylistic preferences shared across models), which we flag in Limitations (§6).

I Per-Category Lead-Time Breakdown

The overall 404-day mean lead reported in §5 aggregates over all 55 matched hits. Table 8 decomposes the lead by topic category. The per-category mean spans 277 days (safety & trust) to 524 days (LLM applications), and the median is within $\sim$40 days of the mean in every category, so the lead-time distribution is not driven by outliers. The overall IQR $= [284, 535]$ days is approximately symmetric about the median (399), consistent with an approximately normal distribution over a research-cycle scale.

J Full Statistical Significance Tests

Table 9 reports pairwise Wilcoxon signed-rank tests and Cohen's d effect sizes for all system pairs across four metrics, computed on the 50 common topics. We use the non-parametric Wilcoxon signed-rank test rather than paired t-test because hypothesis-level metrics are not normally distributed across topics. Cohen's d is computed as

Condition	n	HIT %	Mean avg. score	Median
Single-pass re-judge (primary)				
real	55	**63.6%**	**5.95**	6.0
within_cat	55	0.0%	2.10	1.5
cross_topic	55	0.0%	1.81	1.5
Stress test (permissive, up to 12 samples per pair)				
real	55	87.3%	—	—
within_cat	55	0.0%	—	—
cross_topic	55	0.0%	—	—

Table 7: Negative-control judge audit on all 55 hits, under two settings. Under the primary single-pass re-judge, real pairs re-judge as HIT $35/55 = 63.6\%$ of the time (mean score 5.95, median 6.0), while shuffled pairs (both within-category and cross-topic) produce *zero* HIT verdicts across 110 shuffled pairs combined; the ≈ 4-point gap between real (5.95) and shuffled ($\sim$2.0) mean scores provides evidence that the LLM judge is not simply category-gullible or content-gullible. Under a permissive stress test that allows up to 12 samples per pair and accepts HIT if any sample crosses the threshold, real pairs recover to 87.3% while shuffled pairs *remain at* $0/110$, confirming that even repeated permissive judging does not produce false positives when the content is mismatched. The stress-test rows report HIT recovery only (mean/median not shown), since the statistic of interest is whether any of the 12 repeated samples crosses the threshold.

Category	n	Mean (d)	Median (d)	Range (d)
LLM core	14	385	398	[93, 682]
Other foundations	9	278	260	[66, 635]
LLM apps	7	524	491	[390, 757]
Efficiency & data	7	421	402	[330, 547]
Domain-specific AI	6	494	510	[269, 692]
Multimodal	6	440	360	[217, 735]
Multilingual & speech	3	436	440	[226, 641]
Safety & trust	3	277	280	[130, 421]
Overall	55	404	399	[66, 757]

Table 8: Lead time of CKM-LITE's future-paper-matched hypotheses by topic category. IQR for the overall distribution is $[284, 535]$ days; distribution is approximately symmetric about the median.

the mean difference divided by the pooled standard deviation.

Key patterns. CKM-LITE's hit-rate advantage over BATCH is significant ($p = .006$) with a medium effect ($d = 0.55$). CKM-FULL's novelty advantage over all other systems is large and highly significant ($d > 2.4$, $p < .001$), but it comes at a significant cost in hit rate ($p < .001$ vs. CKM-LITE, $p = .034$ vs. BATCH). Yield is significantly higher for all incremental systems vs. BATCH ($d > 1.4$, $p < .001$), confirming that temporal decomposition produces more hypotheses regardless of other design choices.

K Efficiency Analysis

CKM-LITE's efficiency advantage has two structural sources: (1) Update is dispatched as one LLM call per topic file, so each call sees only the new papers assigned to that sub-theme (≤ 8 per call out of ≤ 96 per window), bounding per-call prompt length; (2) no per-change-type categorization or trajectory instrumentation overhead. The per-window context budget forces BATCH to concatenate all in-window papers into one prompt, so structural choice and budget are coupled by design (§3).

Cost composition of CKM-FULL. CKM-FULL consumes $\sim 11\times$ more tokens per topic than CKM-LITE (5.87M vs. 0.51M) because it bundles multiple instrumentation components (§3). The cost decomposes into three sources:

- *Per-change-type categorization at* Update *time.* CKM-FULL classifies every finding into a typed operation (ADD/CONFIRM/CONTRADICT/BRIDGE) before committing the edit; CKM-LITE applies the same operations inline within one edit-and-commit prompt. This approximately doubles the in-Update token budget.
- *The Θ probe reads the full pre/post-update state pair per window.* For each of the $T = 6$ windows, Θ sees $\mathcal{K}_{t-1}$, $\mathcal{K}_t$, and the in-window

Comparison	Metric	Mean A	Mean B	Δ	d	p
CKM-LITE vs CKM-ABSTRACT	Hit Rate	5.79	4.00	+1.80	+0.34	$.042^{*}$
	Novelty	6.03	6.25	−0.22	−0.91	$<.001^{***}$
	Best Match	3.90	3.48	+0.42	+0.51	$.009^{**}$
	Yield	17.26	17.16	+0.10	+0.08	.333
CKM-LITE vs BATCH	Hit Rate	5.79	3.03	+2.76	+0.55	$.006^{**}$
	Novelty	6.03	6.39	−0.36	−1.42	$<.001^{***}$
	Best Match	3.90	3.46	+0.43	+0.58	$<.001^{***}$
	Yield	17.26	13.68	+3.58	+1.78	$<.001^{***}$
CKM-LITE vs CKM-FULL	Hit Rate	5.79	1.39	+4.40	+1.14	$<.001^{***}$
	Novelty	6.03	6.82	−0.79	−3.46	$<.001^{***}$
	Best Match	3.90	4.58	−0.68	−2.25	$<.001^{***}$
	Yield	17.26	17.84	−0.58	−0.24	.818
CKM-ABSTRACT vs BATCH	Hit Rate	4.00	3.03	+0.97	+0.18	.516
	Novelty	6.25	6.39	−0.14	−0.71	$<.001^{***}$
	Best Match	3.48	3.46	+0.01	+0.01	.173
	Yield	17.16	13.68	+3.48	+1.72	$<.001^{***}$
CKM-ABSTRACT vs CKM-FULL	Hit Rate	4.00	1.39	+2.61	+0.58	$.003^{**}$
	Novelty	6.25	6.82	−0.57	−3.58	$<.001^{***}$
	Best Match	3.48	4.58	−1.11	−1.40	$<.001^{***}$
	Yield	17.16	17.84	−0.68	−0.28	.840
CKM-FULL vs BATCH	Hit Rate	1.39	3.03	−1.64	−0.39	$.034^{*}$
	Novelty	6.82	6.39	+0.43	+2.44	$<.001^{***}$
	Best Match	4.58	3.46	+1.12	+1.57	$<.001^{***}$
	Yield	17.84	13.68	+4.16	+1.43	$<.001^{***}$

Table 9: Full pairwise statistical comparisons across system pairs and metrics (Wilcoxon signed-rank test, $n = 50$ topics). $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$. "Mean A/B" shown here are *per-topic* means of each metric (the inputs Wilcoxon takes as paired observations), which is why the hit-rate values (5.79, 3.03, 1.39) differ slightly from the pooled rates (6.4%, 3.1%, 1.3%) reported in Table 2; the test statistic, effect size, and p-value are unaffected by which summary is shown.

System	Tokens	Tok/Hyp	Tok/Hit	Hyps
CKM-LITE	26M	**30K**	**0.47M**	863
CKM-ABSTRACT	76M	89K	2.31M	858
CKM-FULL	293M	329K	22.6M	892
BATCH	297M	434K	14.1M	684

Table 10: Token efficiency. CKM-LITE achieves the lowest cost per hypothesis (30K) and per hit (0.47M), at 11× lower per-topic cost than BATCH; the per-hypothesis ratio is ∼ 15× (§5). CKM-FULL and BATCH consume similar total tokens (∼295M) for different reasons, decomposed below.

paper set $\mathcal{P}_t$ to emit the typed change signal τ_t, with input typically 20–40K tokens per window.

- *Trajectory-conditioned generation prompt.* CKM-FULL's Gen receives $\mathcal{K}_t$, $\mathcal{S}_{1:t-1}$, $\mathcal{H}_{1:t-1}$, *and* the typed signal τ_t; CKM-LITE's Gen receives only the first three.

Across $T = 6$ windows, these three sources account for the ~ 5.4M extra tokens per topic. The similar cost of CKM-FULL and BATCH is therefore coincidental: CKM-FULL runs many small instrumented calls, BATCH runs few large unstructured calls; only CKM-LITE avoids both overheads.

Hit-rate definition. The Hit% column of Table 2 reports the pooled hit rate (total hits / total hypotheses across the 50 topics): $55/863$ for CKM-LITE, $21/684$ for BATCH, $12/892$ for CKM-FULL. Coverage, the fraction of topics with ≥ 1 hit, captures per-topic deployment behavior; the two metrics are not redundant.

Lead-time provenance. Lead times (mean 404, median 399) are measured from the hypothesis's generation cutoff to the matched paper's *arXiv first-submission date*, not its conference publication date, to avoid double-counting submission-to-camera-ready delays.

Cost breakdown by phase. Most tokens are consumed during metabolism (Phase 2). CKM-LITE's Phase 2 uses Gemini-2.5-Flash for generation (∼\$0.50 per topic) and GPT-4o-mini + GPT-4o for two-stage judging in Phase 3 (∼\$2–3 per topic). Total cost for the full evaluation (3 systems × 50 topics) was ∼\$130, with CKM-LITE under 10% of this.

Scalability. CKM-LITE's low cost makes it practical for continuous deployment: monitoring 100 topics with monthly updates would require ∼5M tokens per cycle (∼\$0.75 at current API prices for generation). The bottleneck for sustained deploy-

ment is judging (Phase 3), which scales with both hypothesis count and candidate-pool size.

L Diagnostic Trade-offs Surfaced by the Protocol

Running predictive validation at scale surfaces three workflow-level diagnostics that are difficult to observe from contemporaneous judge scores alone. We report them as descriptive properties of this benchmark run, not as calibrated predictors for future deployments.

F1: Instrumentation changes the coverage–novelty profile. The CKM-LITE/CKM-FULL comparison bundles several instrumentation choices rather than isolating a single component (§6). In the released run, CKM-FULL reduces topic coverage from $36/50$ to $11/50$ and pooled hit rate from 6.4% to 1.3%, while increasing mean temporal lead from 404 to 537 days and improving originality under an independent novelty judge (D1 $4.20 \rightarrow 5.64$; Appendix F). Figure 2 shows the resulting distributional shift: CKM-FULL concentrates more mass near the 4–5 range, whereas CKM-LITE produces a broader right tail that crosses the 6.0 hit threshold more often. We therefore treat CKM-FULL as useful instrumentation for analyzing hypothesis character, but not as the deployment configuration.

F2: Coverage is category-dependent. CKM-LITE coverage varies substantially across categories, from $10/10$ for LLM core and $3/3$ for multimodal to $2/6$ for domain-specific AI (Appendix I). This variation does not reduce cleanly to a simple fast-moving-vs.-stable distinction: high-coverage and low-coverage categories both contain mixtures of mature and rapidly shifting subareas. We also tested a simple per-topic drift proxy, the first-to-last centroid shift in hypothesis embedding space, and found no relationship with CKM-LITE coverage (Spearman $r = 0.03$, $p = 0.83$, $n = 50$). Category mix should therefore be treated as part of the benchmark design rather than as a nuisance averaged away.

F3: Trigger labels vary in hit rate but are not yet stable predictors. In CKM-FULL, the change-detection step assigns trigger types to window-level updates. Hit rates differ across trigger types: an omnibus χ^2 test over the 9 trigger types with $n \geq 5$ rejects uniformity ($\chi^2 = 20.80$, dof $= 8$, $p = 0.008$). However, the ordering is workflow-specific and underpowered for rare triggers. For example, GAP_EXPLOITATION has the highest CKM-FULL hit rate ($3/28 = 10.7\%$), whereas CKM-LITE's independent trigger taxonomy gives CONVERGENCE the largest hit count and an 8.1% hit rate ($30/369$). We therefore use trigger labels as diagnostic handles for human inspection, not as deployment-ready filters.

Together, F1–F3 show that the protocol does more than assign a single score: it exposes how workflow design, topic regime, and trigger type shape the distribution of future-paper matches. These diagnostics are useful for comparing runs and guiding follow-up analysis, but they should be recalibrated before being used as selection rules.

M Extended Discussion

What predictive validation tells us that contemporary judges cannot. The hit metric is not a substitute for novelty (the two are essentially uncorrelated in our data, $r \approx 0$), and we do not claim it captures everything one wants hypothesis-generation workflows to do. What it does provide is a falsifiable, decision-independent signal: "hypothesis h generated at time t aligned with paper p published at $t' > t$" is a fact about the literature, not an opinion drawn from the same population of expertise the workflow is meant to extend. The three trade-offs of Appendix L (in particular the instrumentation–coverage axis) are the kind of structural property a contemporary-judge evaluation would be blind to: a contemporary judge cannot tell whether a per-hit-quality gain comes at the cost of coverage that future literature would have revealed.

Evaluation paradox: coverage vs. concentration. The hit metric (strictest among our predictive measures) still systematically favors broad, partially-aligned hypotheses over narrowly-specified ones. CKM-LITE's higher hit rate comes from a wide score distribution with a thick right tail; CKM-FULL's concentrated distribution reflects a different strategy whose precise predictions either align fully or not at all. Any hit-based evaluation that uses partial-alignment scoring will exhibit this bias. We therefore recommend that future evaluations of hypothesis-generation workflows report *both* coverage and concentration of alignment rather than collapse them into a single number, and that protocol authors design judges that can be tightened

or loosened without changing the validation pool. The threshold-sensitivity sweep in Appendix G demonstrates this style of joint reporting on our own benchmark.

Implications for evaluating LLM-as-research-tool systems. Predictive validation as designed here is one falsifiability test; it is not the only one a community building this class of systems will need. What we hope the protocol contributes beyond CKM is a tractable template: any workflow whose output is a structured proposal, run against any corpus with dated publication records, can be scored under a comparable protocol with judge prompts adapted to its output format. The artifacts released with this paper (topic list, validation pool, judging code, and reference implementation) are intended to make that adaptation cheap, so the next workflows of this kind can be evaluated against the same falsifiability bar from the start rather than retrofitted into it later.

N Additional Hit Cases

Figure 3 in the main text shows a single CKM-LITE hit and the matched future paper, selected for transparency of the title-level alignment. The released repository contains the full set of 55 CKM-LITE hits with hypothesis text, matched paper IDs, lead times, and judge scores. We describe two additional matched cases below to illustrate the diversity of the hit set.

N.1 Case A: methodological transfer across domains

Topic: AI for hypothesis generation in science. **Generated**: 2024-01. **Trigger**: BRIDGE. **Lead**: 639 days.

> *"Implementing a Test-Driven Development (TDD)-inspired framework for scientific hypothesis generation, where 'scientific tests' (e.g., formal logical consistency checks, computable simulations, or statistical tests on existing datasets) are defined prior to or concurrently with LLM-based generation, will reduce the number of hypotheses requiring costly experimental validation by at least 30%."*
>
> Matched → **HARPA: A Testability-Driven, Literature-Grounded Framework for Research Ideation** [2510.00620], judge score 6.5/10.

The BRIDGE trigger connected two previously disjoint literatures, software engineering methodology (TDD) and scientific hypothesis generation, to predict that a testability-first framework would emerge. HARPA was published 639 days later with exactly this approach, providing concrete substantiation for the cross-domain transfer the workflow proposed.

N.2 Case B: precise gap exploitation

Topic: protein structure prediction. **Generated**: 2024-01. **Trigger**: GAP_EXPLOITATION. **Lead**: 462 days.

> *"Extending the PSC-CPI multi-scale contrasting framework to explicitly incorporate atomic-scale representations and contrasting objectives will achieve at least a 5% improvement in RMSE for CPI binding-affinity prediction and 10% improvement in atomic contact prediction accuracy on the 'Unseen-Both' setting of the Davis, KIBA, and Mert datasets."*
>
> Matched → **EquiCPI: SE(3)-Equivariant Geometric Deep Learning for Structure-Aware CPI Prediction** [2504.04654], judge score 6.5/10.

The GAP_EXPLOITATION trigger named a specific deficiency (PSC-CPI is residue-level only, no atomic-scale conditioning) and predicted that a method adding atomic-scale representations would yield measurable improvements on named benchmarks. EquiCPI was published 462 days later with SE(3)-equivariant geometric deep learning addressing the same gap with a different technical approach, a closer match in problem than in method, but well above the 6.0 hit threshold.

Patterns across the hit set. A substantial fraction of CKM-LITE's hits fuse signals from multiple pre-window papers into a single named direction (median 3 source citations per hit). Long-lead hits ($\geq$ 600 days) concentrate in multimodal, domain-specific AI, and LLM applications; short-lead hits ($\leq$ 200 days) cluster in safety & trust and other foundations. The full per-category lead-time breakdown is in Appendix I; the JSON of all 55 hits, with arXiv IDs and judge verdicts, is in the released repository.