AIRE: CLOSED-LOOP AGENTIC RESEARCH FOR TEXT EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Developing effective text embedding models extends far beyond executing a training script. It requires researchers to navigate interdependent decisions across data curation, objective design, sampling strategies, and evaluation protocols. Because key design trade-offs often emerge only after end-to-end training and retrieval evaluation, embedding development is fundamentally iterative. While recent advances demonstrate that full research workflows can be automated, doing so for text embeddings requires orchestrating domain-specific decisions across data pipeline construction, model training, and retrieval benchmarking. In enterprise settings, this automation must additionally handle distributed user roles, shared accelerator clusters, and strict data governance.

To address these challenges, we introduce AIRE (*Autonomous Investigation and Research for Embeddings*), a multi-agent framework that structures these operations into an autonomous, persistent research loop. In AIRE, a central coordinator delegates literature review, data construction, implementation, training, and evaluation to specialized sub-agents, maintaining a shared cross-round state to guide iterative discovery. We also introduce *AIRE-Bench*, a benchmark for autonomous embedding research with three task environments built from public retrieval datasets. AIRE-Bench evaluates the resulting model and the research process, tracking model quality, improvement rate, exploration coverage, failure resilience, and compute cost. AIRE achieved an AIRE-Bench score of 89.61.

1 INTRODUCTION

An embedding run can fail for many reasons. The training data may contain weak negatives, the loss may not match the task, or the optimizer may be unstable. Because the source of the failure is unclear, diagnosis often sends the researcher back to the literature or the data. What changes in the next run depends on the evidence found there: the appropriate intervention may be a new sampler rather than a new model. That evidence is scattered across papers, repositories, datasets, logs, and people, creating substantial communication overhead at every iteration. Figure 1 contrasts this workflow with the research loop studied here.

These coordination costs grow in enterprise environments, where data, training, and evaluation are often handled by different teams that share limited accelerator resources. Repeated handoffs across these teams make each research iteration slower and more expensive.

Language-model agents offer a promising way to alleviate this coordination overhead. By interleaving reasoning with tool use (Yao et al., 2023), agents can execute non-trivial machine learning engineering tasks (Huang et al., 2024; Chan et al., 2025). Multi-agent frameworks further distribute complex workloads across specialized roles (Wu et al., 2024). This paradigm has been extended both to automated idea generation, experimentation, and report writing (Lu et al., 2024) and to human-seeded research workflows spanning literature review, experimentation, and reporting (Schmidgall et al., 2025). However, generic multi-agent workflows do not inherently capture the specialized structure of embedding research. Effective embedding development requires agents to reason over sample-level features and hard negatives, jointly adapt data curation and training strategies, evaluate runs under a unified retrieval protocol, and leverage past failures to avoid redundant exploration.

AIRE is designed around this tight coupling, formulating embedding research as a multi-round discovery process. Within each round, a Coordinator Agent ingests the cumulative experimental history

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

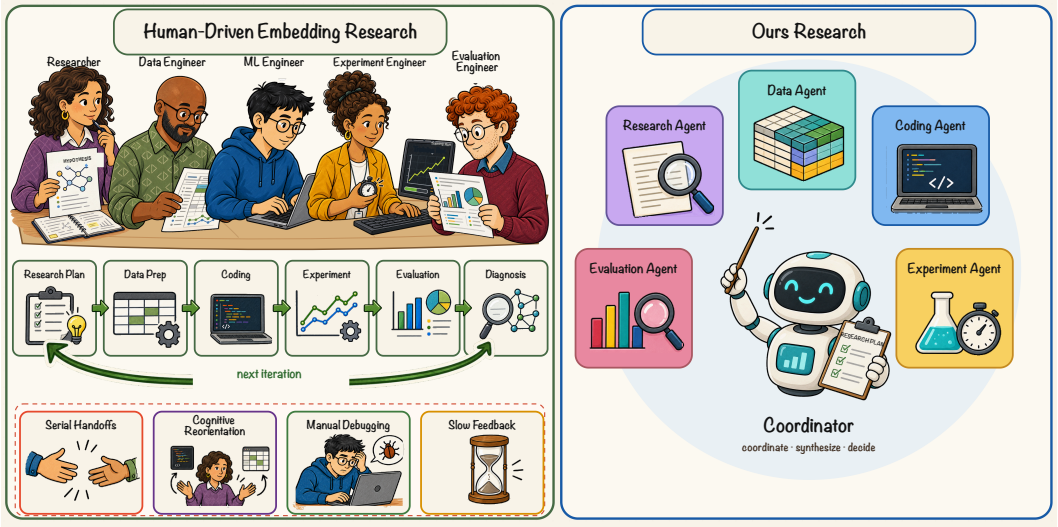

Figure 1: Human-driven embedding research in a multi-role organization (left) divides planning, data, coding, experimentation, evaluation, and diagnosis across serial handoffs. AIRE (right) places the same work in a persistent loop under a central Coordinator Agent. Specialist agents retain intermediate state across rounds, while people continue to provide scientific judgment.

and tasks specialized agents with surveying literature, constructing data pipelines, implementing changes, training models, and evaluating retrieval metrics. Based on the round's empirical findings, the Coordinator dynamically decides whether to iterate on the current idea, pivot direction, fork into parallel hypotheses, or halt exploration. By preserving historical context across these high-level decisions, AIRE automates a continuous, closed-loop research workflow rather than disjointed training scripts.

This paradigm motivates the central research question of this paper: *Can an agentic system sustain an auditable, multi-round research process that improves text embeddings?* To answer this question, we make three primary technical contributions:

1. **Multi-round embedding research system.** We formulate autonomous text embedding research as a sequential process and instantiate it in AIRE, where a Coordinator Agent and five specialists propose and execute data, model, and optimization interventions while preserving experimental state across rounds (Sections 3 and 4).
2. **Governed artifact-first harness.** Typed tasks, immutable treatment records, stage-completion contracts, and persistent artifacts make hypotheses, consumed treatments, outputs, failures, and lineage auditable and recoverable. Aggregated feedback guides later decisions, while evaluator-held test judgments remain isolated from model-facing workspaces (Sections 4.4 and 5.6).
3. **AIRE-Bench and empirical characterization.** We introduce AIRE-Bench, a standardized evaluation environment that scores the resulting model and the research process through model quality, improvement trajectory, exploration coverage, failures, and cost. Across three complete single-launch trajectories, AIRE obtains a mean open-ended AIRE-Bench score of 89.61; these are descriptive findings, not evidence of matched-seed stability or cross-system superiority (Section 5).

2 Related Work

Text representation and retrieval. Dual encoders map queries and documents into a shared space for efficient retrieval, as established by Sentence-BERT (Reimers & Gurevych, 2019) and DPR (Karpukhin et al., 2020). Later systems scale supervision and corpus scope (Wang et al., 2022; 2024; Zhang et al., 2024) while varying objectives, backbones, pooling, negative mining, and con-

textual conditioning (Chen et al., 2024; Liu et al., 2024b; Lee et al., 2025; Morris & Rush, 2025). These coupled choices motivate AIRE: rather than proposing another encoder, it evaluates how an autonomous system navigates data and training decisions.

Embedding evaluation. MTEB (Muennighoff et al., 2023) benchmarks text embeddings across tasks, while BEIR (Thakur et al., 2021) benchmarks retrieval systems across heterogeneous datasets. AIRE-Bench instead evaluates the end-to-end research system under a common execution protocol, reporting final model quality together with progress, coverage, failures, time, and compute.

Automated machine learning and ML agents. AutoML searches predefined model and hyperparameter spaces (Feurer et al., 2015); language-model agents can also edit repositories, run jobs, and interpret outputs. AgentBench, SWE-bench, and SWE-agent study interactive and repository-level agency (Liu et al., 2024a; Jimenez et al., 2024; Yang et al., 2024), while MLAgentBench and MLE-bench evaluate iterative ML engineering (Huang et al., 2024; Chan et al., 2025). AIRE-Bench retains this benchmark–task separation but targets retrieval-specific, multi-round research with state shared across related experiments.

Automated scientific research. AutoGen coordinates tool-augmented agents (Wu et al., 2024), and AFlow searches over agentic workflows using execution feedback (Zhang et al., 2025). The AI Scientist extends this pattern to automated idea generation, experimentation, and reporting (Lu et al., 2024); Agent Laboratory turns human-provided ideas into workflows spanning literature review, experimentation, and reporting (Schmidgall et al., 2025); AgentX applies an iterative loop to online recommender optimization (Lao et al., 2026). AIRE instead studies offline text embedding research with a dedicated data specialist, persistent cross-round state, and bounded, reproducible training trajectories.

3 AUTONOMOUS EMBEDDING RESEARCH

We model embedding research as a sequence of experiments beginning with a research objective O, a base encoder f_0, public training resources, and an evaluation policy. At round t, the state $s_t = (O, H_t, E_t, D_t)$ records accepted and failed hypotheses, permitted experimental observations, explored research directions, and references to persistent artifacts, without exposing test relevance labels.

Given this state, the Coordinator selects an action $a_t = (h_t, d_t, m_t, x_t)$ comprising a hypothesis, data operation, model or training treatment, and execution plan. Execution returns model metrics, failure records, and safe diagnostic summaries, producing a trajectory $\tau_\pi = (s_1, a_1, e_1, \ldots, s_T, a_T, e_T)$. We evaluate this trajectory by its peak admissible model quality, the responsiveness of later decisions to prior evidence, and its failure behavior. This requires expressive research actions, persistent state, auditable isolated execution, and controlled feedback—the four requirements implemented by AIRE's multi-agent harness.

4 AIRE: CLOSED-LOOP AGENTIC RESEARCH

Figure 2 shows the Coordinator Agent, five specialist agents, and their shared experimental environment. AIRE is a modular system in which capability plugins define the tools and typed actions available to each agent, while idea plugins package trainable research interventions. Its stages run in isolated worker processes. Agents exchange typed tasks and result artifacts through the coordinator; they do not call one another directly. A worker can therefore fail and resume without asking the coordinator to reconstruct the workflow state from a chat history.

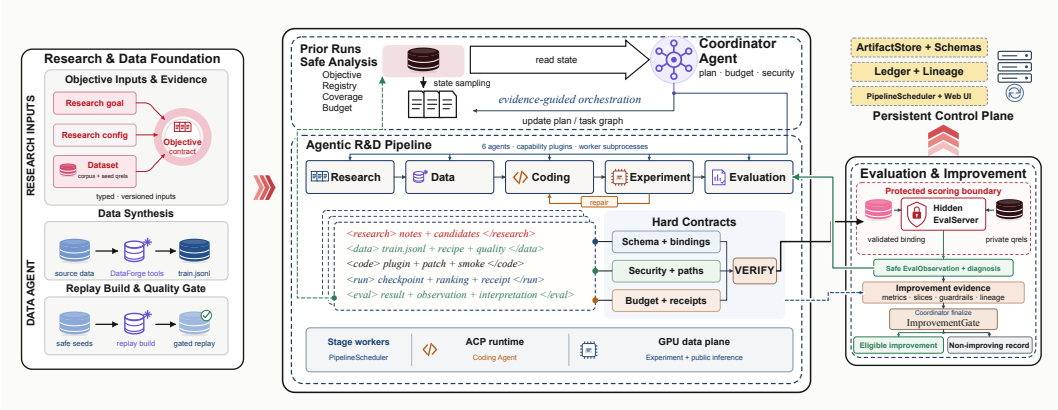

Figure 2: AIRE has one Coordinator Agent and five specialists: Research, Data, Coding, Experiment, and Evaluation. The lower path runs a proposed intervention. The right-hand path converts evaluation output into a safe observation for the following round. Artifacts, schemas, lineage records, and the scheduler keep state consistent across processes.

4.1 The Research Loop

A round has six logical stages: research, data construction, coding, experiment, evaluation, and finalization. Research turns the current state into literature-backed hypotheses. Data and coding produce a candidate dataset and implementation. The experiment stage trains the model, evaluation returns an observation, and finalization updates cross-round state. The coordinator then continues the idea, starts another one, forks a promising line, abandons it, or stops.

Failed experiments stay in the history. A blocked data build, training crash, or non-improving experiment can change the next decision even though it never produced the best checkpoint. A round may still remove a hypothesis or narrow its cause. One-shot code generation usually loses this information when execution fails.

4.2 Specialized Agents

Coordinator Agent. The Coordinator Agent ingests a compact world state representation, formulates subsequent actions, validates prerequisite dependencies, and dispatches tasks. To enforce governance, it is strictly barred from modifying training data directly. Deterministic rule-based validators intercept and reject any action that violates access permissions, stage prerequisites, or experimental validity constraints, allowing the underlying language model to navigate research directions strictly within these verified bounds.

Research Agent. The Research Agent conducts targeted literature surveys, synthesizes relevant papers, cross-checks for conceptual overlap with prior rounds, and produces ranked hypothesis candidates. Each candidate explicitly specifies the targeted behavioral change, the empirical failure mode it addresses, grounding citations, implementation risk, and expected compute overhead. The agent also tracks search-space exploration to detect idea saturation if recent rounds linger excessively within a single methodological family.

Data Agent. The Data Agent manages data curation pipelines via DataForge tooling—ingesting public corpora, mining hard negatives, constructing replay buffers from sanitized failure summaries, scoring sample utility, and materializing data variants. Before triggering model training, each materialized dataset must pass strict deterministic integrity checks for empty instances, exact duplicates, positive-negative label conflicts, missing negatives, false-negative hazards, and prompt template bias.

Coding Agent. Operating within an isolated workspace, the Coding Agent implements modifications to the model architecture, loss function, negative sampler, batch collator, or training routine.

Restricted to read and edit only whitelisted file paths, it emits a modular idea plugin, a file diff, and passing smoke-test telemetry. When a full training run encounters runtime exceptions, the coordinator dispatches a bounded repair task provisioned with sanitized error traces.

Experiment Agent. The Experiment Agent binds candidate code and datasets to concrete batching strategies, accelerator allocations, and training configurations. GPU experiments can run either on locally scheduled accelerators or on remote GPU workers submitted through a configured execution backend. The current implementation supports one internal cloud GPU platform as its remote backend. Both modes use the same frozen training specification and artifact contract. During execution, the agent periodically monitors training progress and health through public telemetry, allowing unhealthy runs to be identified and stopped cooperatively. The agent publishes resulting model checkpoints, intermediate rankings, execution logs, and hardware telemetry. Crucially, training failures are preserved as structured negative results rather than discarded, ensuring that failed trajectories remain informative within the cumulative research history.

Evaluation Agent. The Evaluation Agent verifies run outputs and ingests only aggregated diagnostic signals returned by an isolated external evaluator. It transforms overall retrieval metrics, fine-grained slice evaluations, and sanitized diagnostic summaries into structured observations for subsequent decision rounds. By construction, it is completely air-gapped from held-out test judgments, strictly precluding evaluation data leakage.

4.3 Cross-round research state

The Coordinator ingests a compact projection of the accumulated artifact history, capturing the current best-accepted experiment, recent hypotheses, categorized failure modes, and previously manipulated variables. This structured state enables principled, causal follow-up iterations; for example, a subsequent round can isolate and refine a data intervention that successfully addressed a specific failure slice while holding the remainder of the treatment configuration invariant.

Exploration coverage is computed by a direction classifier that categorizes accepted ideas into predefined research families and tracks untouched avenues and consecutive within-family streaks. These signals are advisory rather than prescriptive: the Coordinator may continue exploring an apparently saturated direction when it provides an explicit, testable hypothesis.

All intermediate artifacts—including research notes, data synthesis recipes, code patches, execution manifests, evaluation summaries, and complete lineage graphs—are persisted to disk. This persistence supports seamless recovery from system interruptions and facilitates post-hoc audits to reconstruct the empirical rationale behind each decision.

4.4 Trusted execution and evaluation

Sustaining an autonomous, long-running research loop demands rigorous execution semantics. Prior to job invocation, AIRE freezes each candidate intervention into an immutable experiment specification. Upon execution, the training runtime records the complete provenance of all consumed parameters—including dataset versions, loss formulations, negative samplers, optimizer configurations, batch schedules, and auxiliary runtime settings. Furthermore, a pipeline stage is marked as complete only when its worker process status, expected output artifacts, schema contracts, access permissions, and resource leases are mutually verified and satisfied. Runtime failures transition to explicit terminal states, ensuring that stale, partial, or empty artifacts can never inadvertently satisfy downstream stage dependencies.

Evaluation uses a separate trust boundary. Experiment workers may encode the public queries and corpus, but only the evaluation server reads test relevance judgments. A completed evaluation is bound to the checkpoint and ranking that produced it. A Promotion Gate admits an experiment as the current best model only after checking the primary metric, priority slices, guardrails, cost, and leakage flags. Agents cannot turn fine-grained hidden feedback into training data.

5 EXPERIMENTS

We evaluate AIRE with *AIRE-Bench*, one benchmark containing three end-to-end task environments. The benchmark defines a common execution contract; each task instantiates that contract with versioned public retrieval data and evaluator-held test judgments. The common contract fixes the starting model, metrics, and stopping conditions. A *benchmark run* is one complete execution of a comparison system on one task with one random seed; it returns the best admissible model and an auditable research trace. The system receives only the aggregate feedback allowed by the contract. The resulting records expose the outcome and cost of the complete research process.

5.1 AIRE-BENCH TASKS AND STARTING MODEL

AIRE-Bench contains task environments built from FiQA and NFCorpus as distributed through BEIR (Thakur et al., 2021), together with a retrieval view of QASPER (Dasigi et al., 2021). They cover financial, biomedical, and scientific-document retrieval and differ in corpus size and relevance structure. Public training judgments seed the initial training data. Each task uses nDCG@10 as its primary metric, with recall and mean reciprocal rank as secondary metrics.

Every task environment starts from `Alibaba-NLP/gte-multilingual-base`, a multilingual text representation model in the mGTE family (Zhang et al., 2024). Within a benchmark run, the system may change the training data, loss, sampler, collator, training strategy, or an approved idea plugin. It may not replace the starting-model family or evaluation protocol. The benchmark limits each run to at most 16 rounds.

5.2 MULTI-ROUND RESEARCH METRICS

Model quality and research progress. We assess research outcomes by the peak admissible nDCG@10 achieved during a trajectory along with its relative gain over the baseline model. Admissibility is governed by uniform evaluation and promotion criteria across all experimental conditions. To capture temporal dynamics and efficiency across the full trajectory, we further track the wall-clock time and GPU-hours to the first improvement, the area under the best-so-far curve, and the fraction of benchmark runs yielding at least one admissible gain. Together, these trajectory metrics distinguish sustained, systematic progress from isolated successful runs.

Exploration, reliability, and cost. Direction coverage is defined as the fraction of pre-configured research families explored by accepted ideas. We also track the number of distinct executable treatments and the maximum consecutive exploration streak within a single family. Coverage serves as a descriptive diagnostic rather than a strict optimization objective: concentrating multiple fruitful experiments within a single direction does not inherently constitute a search failure. Beyond coverage, the research log captures failure patterns and overhead—including invalid actions, blocked data pipelines, code repair iterations, failed training jobs, redundant evaluation attempts, wall-clock duration, GPU-hours, and LLM inference costs. To avoid survivorship bias, failed benchmark runs remain in the denominator when aggregating run-level success rates.

Open-ended AIRE-Bench score. We assign an open-ended score to each complete benchmark trajectory without cross-system normalization. Let $b > 0$ be the base-model nDCG@10, $m^*(t)$ the best admissible nDCG@10 available by wall-clock time t, and T the run duration. Let c be the completed-round fraction and p the pass fraction among worker attempts other than finalization. We compute

$$q(t) = \max\left\{0, \frac{\ln(m^*(t)/b)}{\ln(1.10)}\right\}, \qquad \bar{q} = \frac{1}{T}\int_0^T q(t)\,\mathrm{d}t, \tag{1}$$
$$\rho = 0.7c + 0.3p, \qquad S_{\mathrm{AIRE}} = 60q(T) + 25\bar{q} + 15\rho.$$

Here, $q(t)$ expresses improvement in multiplicative 10% units: a 10% gain gives $q = 1$, while a 21% gain gives $q = 2$ because $1.1^2 = 1.21$. The time average $\bar{q}$ rewards improvements that arrive early and persist, while ρ summarizes round completion and attempt reliability. An immediate, sustained 10% gain with perfect reliability yields a reference score of 100, but 100 is not an upper bound. Failed rounds consume time without increasing $q(t)$ and reduce reliability. We report the composite

score alongside its raw quality, trajectory, reliability, and cost components so that it does not obscure these trade-offs.

Beyond evaluating current research trajectories, we envision AIRE-Bench as the evidence source for a future outer loop that improves AIRE itself (Figure 3).

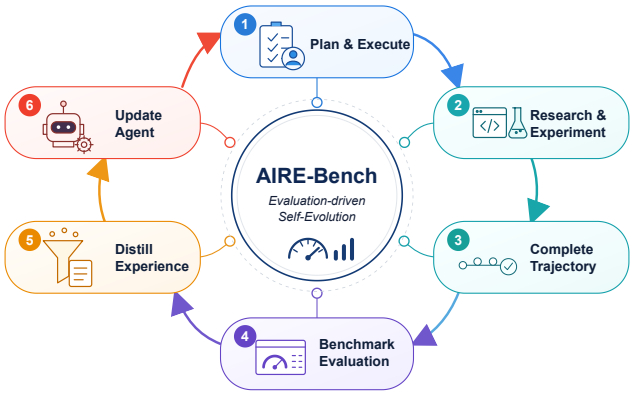

Figure 3: Envisioned use of AIRE-Bench in a future evaluation-driven self-evolution loop. Complete research trajectories would be evaluated and distilled into proposals for updating AIRE's policies, skills, prompts, and control logic. Agent updating is not part of the present implementation or reported experiments.

5.3 Embedding Quality Improvements

For each launch, we compare the measured zero-shot baseline with the best formally evaluated run marked valid for the leaderboard (Table 1).

Table 1: Artifact-backed single-launch results. Relative gain is computed from the measured zero-shot baseline. AIRE score is the open-ended index in Equation 1.

Task	Base nDCG@10	Best nDCG@10	Rel. gain	AIRE score
FiQA	0.4433	0.4979	12.32%	109.36
NFCorpus	0.3300	0.3622	9.75%	85.70
QASPER	0.0815	0.0879	7.91%	73.78

All three launches improve on their measured baselines. FiQA records the largest relative gain and the highest AIRE score. These results provide evidence of within-launch improvement across three task domains, but one launch per task cannot establish seed stability, statistical significance, or superiority over fixed search or another agent.

Internal proprietary-model comparison. We additionally evaluate AIRE in an internal e-commerce product-search setting built around our in-house proprietary untied dual-encoder embedding model. The *manual* curve in Figure 4 is the checkpoint produced by the existing manual training workflow; the *AIRE* curve is the best checkpoint produced by the automated research loop. AIRE improves recall at all six retrieval cutoffs, with relative gains ranging from 8.75% at Recall@1 to 0.97% at Recall@100. We subsequently adopted the AIRE checkpoint in the internal product-search system, replacing the checkpoint produced by the manual training workflow.

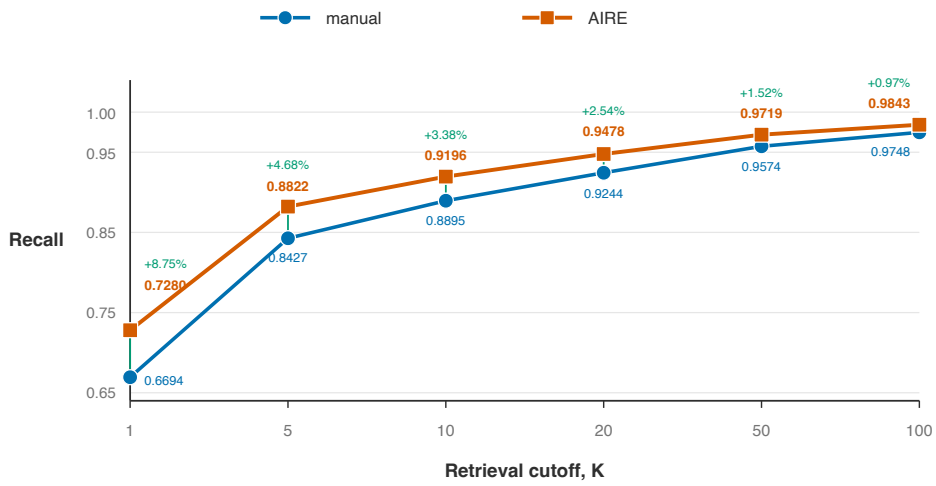

Figure 4: Recall@K on an internal e-commerce product-search evaluation using our in-house proprietary dual-encoder embedding model. *manual* denotes the checkpoint from the existing manual training workflow; *AIRE* denotes the best checkpoint produced by AIRE from the same starting model and evaluation protocol. Percentages show relative recall gains.

5.4 Controlled system ablations

We compare full AIRE with four removals: cross-round memory, direction guidance, data intervention, and role specialization. All conditions use the same starting checkpoint, task data, action space, evaluation feedback, stopping rule, and model and compute budgets. Safety checks and the protected evaluation boundary remain unchanged.

Table 2: AIRE-Bench scores for matched system ablations (higher is better). Mean averages the three task scores.

Condition	FiQA	NFCorpus	QASPER	Mean
AIRE (full)	109.36	85.70	73.78	89.61
w/o Cross-Round Memory	68.84	71.74	65.49	68.69
w/o Direction Guidance	72.96	71.55	69.12	71.21
w/o Data Intervention	79.62	72.32	72.59	74.84
w/o Role Specialization	74.59	61.19	63.43	66.40

5.5 Trajectory quality, reliability, and resource use

Table 3 reports the score components and resource use of the three launches. Figure 5 then traces their round-by-round relative nDCG@10 and best-so-far progress.

Table 3: Open-ended score components from the three complete trajectory exports. Token totals aggregate all recorded LLM actors.

Task	$q(T)$	$\bar{q}$	ρ	Wall-clock (h)	LLM tokens (M)
FiQA	1.2192	0.8484	1	5.08	29.40
NFCorpus	0.976	0.579	0.846	7.25	30.68
QASPER	0.799	0.523	0.853	5.49	20.08

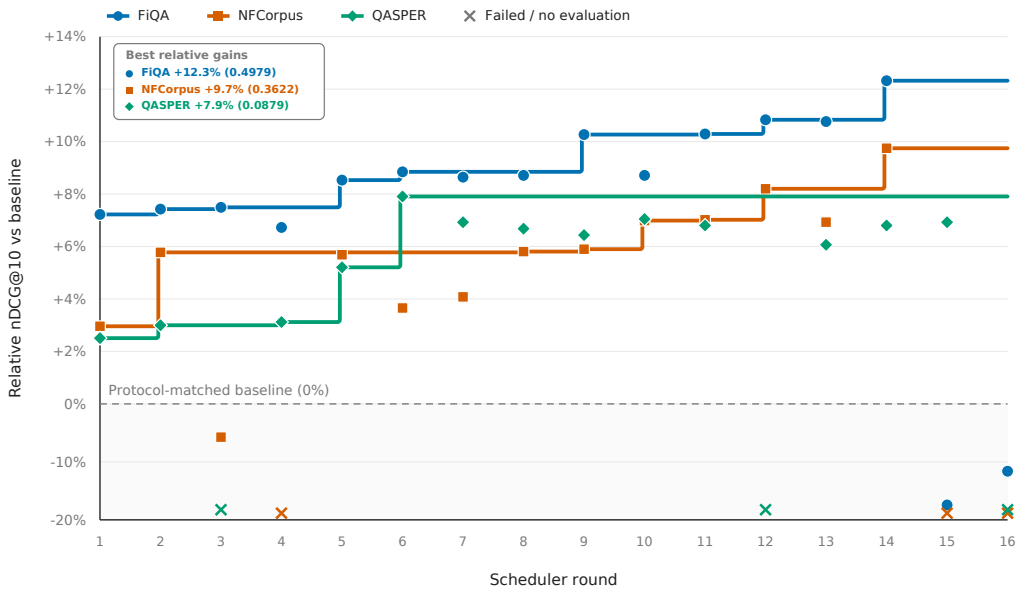

Figure 5: Round-by-round nDCG@10 relative to the protocol-matched baseline. Markers show evaluated rounds, step lines track the best valid gain reached so far, and crosses denote failed rounds or rounds without a formal evaluation. All three trajectories improve over their baselines, with FiQA reaching the largest final gain.

5.6 Analysis

Different paths to improvement. Figure 5 reveals three distinct search patterns. QASPER gains early and then plateaus, NFCorpus improves through round 14, and FiQA advances gradually before finishing with the largest relative gain. FiQA's higher time-averaged quality and reliability also yield the highest AIRE score, showing how the score distinguishes trajectories beyond their final metric.

Failed rounds remain part of the result. The crosses in Figure 5 preserve failed or unevaluated rounds rather than presenting monotone progress. They consume time, reduce reliability, and remain in the persisted sequence, making the path to an accepted result auditable and distinguishing clean from failure-heavy runs.

Research control and future self-evolution. Trajectory shape and direction coverage help the Coordinator balance continued exploitation against a change of research family. Figure 3 extends this idea to a future, protected outer-loop *self-evolution* process in which held-out-validated trajectories inform updates to AIRE's policies, skills, prompts, and control logic.

6 Conclusion

We presented AIRE, a system for carrying text embedding research across multiple rounds through explicit treatments, persistent artifacts, recoverable execution, and evaluator-isolated held-out judgments. In one complete launch on each AIRE-Bench task, AIRE improved all three domains under the same starting model and evaluation protocol, obtaining a mean open-ended score of 89.61.

On our internal product-search model, AIRE improved all six recall cutoffs by 0.97–8.75%; its checkpoint and research loop subsequently replaced the prior model and manual training workflow. Looking forward, AIRE-Bench can provide the evidence foundation for future *self-evolution*, with accumulated trajectories informing verifiable updates to AIRE's research policies, skills, and control mechanisms.

AI USE STATEMENT

The system studied in this paper uses generative AI as an experimental component. We also used generative AI for software development, literature discovery, and language editing. Human authors reviewed the research design, source code, citations, experimental artifacts, and AI-assisted text. We take responsibility for the paper's claims and contents.

ETHICS STATEMENT

AIRE trains on public data and keeps test relevance judgments outside agent workspaces. Language-model agents cannot read individual relevance labels or gold documents. The experiments do not involve human subjects. The main risks are wasted compute from invalid hypotheses, bias inherited from public corpora, and conclusions that exceed the evidence from automated experiments. Section 4.4 describes the corresponding controls.

REPRODUCIBILITY STATEMENT

The anonymous supplement will include the system source, configurations, prompt templates, artifact schemas, and scripts for each reported benchmark run. Section 5 specifies the AIRE-Bench tasks, metrics, and comparison conditions. Appendix A lists the artifacts needed to replay and audit a round.

REFERENCES

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*. OpenReview.net, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.365. URL https://aclanthology.org/2021.naacl-main.365/.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28*, pp. 2962–2970, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/11d0e6287202fced83f79975ec59a3a6-Abstract.html.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 20271–20309. PMLR, 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*. ICLR, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Changxin Lao, Fei Pan, Guozhuang Ma, Han Li, Huihuang Lin, Jijun Shi, Kangzhi Zhao, Kun Gai, Mo Zhou, Qinqin Zhou, Quan Chen, Ruochen Yang, Shifu Bie, Shijie Yi, Shuang Yang, Shuo Yang, Wenhao Li, Wentao Xie, Xiao Lv, Xuming Wang, Yijun Wang, Yiming Chen, Yusheng Huang, Zhongyuan Wang, Zibo Zhao, Zijie Zhuang, Baoning Xia, Chao Liu, Chaoyi Ma, Chubo He, Dawei Cong, Feng Jiang, Gang Wang, Guilin Xia, Hanwen Xu, Jiahong Xie, Jiahui Qiao, Jian Liang, Jiangfan Yue, Jing Wang, Jinghan Yang, Jinghui Jia, Kan Qin, Lei Wang, Ming Li, Peilin Song, Pengbo Xu, Qiang Luo, Ruiming Tang, Shiyang Liu, Shuxian Jin, Tao Wang, Tao Zhang, Xiang Gao, Xianghan Li, Yingsong Luo, Yiwen Ning, Yongcheng Liu, Yueyang Liu, Yuan Guo, Zhaojie Liu, and Zhenkai Cui. AgentX: Towards agent-driven self-iteration of industrial recommender systems, 2026. URL https://arxiv.org/abs/2606.26859.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. NV-Embed: Improved techniques for training LLMs as generalist embedding models. In *The Thirteenth International Conference on Learning Representations*. ICLR, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c4bf73386022473a652a18941e9ea6f8-Abstract-Conference.html.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*. ICLR, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Zheng Liu, Chaofan Li, Shitao Xiao, Yingxia Shao, and Defu Lian. Llama2Vec: Unsupervised adaptation of large language models for dense retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 3490–3500. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.acl-long.191. URL https://aclanthology.org/2024.acl-long.191/.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

John X. Morris and Alexander Rush. Contextual document embeddings. In *The Thirteenth International Conference on Learning Representations*. ICLR, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f79df6cbc6e5f708440004fad7ef64cc-Abstract-Conference.html.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 2014–2037. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.eacl-main.148. URL https://aclanthology.org/2023.eacl-main.148/.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.

findings-emnlp.320. URL https://aclanthology.org/2025.findings-emnlp.320/.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/65b9eea6e1cc6bb9f0cd2a47751a186f-Abstract-round2.html.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training, 2022. URL https://arxiv.org/abs/2212.03533.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 11897–11916. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.642. URL https://aclanthology.org/2024.acl-long.642/.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Proceedings of the First Conference on Language Modeling*. OpenReview.net, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent–computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems 37*. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*. ICLR, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. mGTE: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1393–1412. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-industry.103. URL https://aclanthology.org/2024.emnlp-industry.103/.

A Reproducibility and artifact contract

A.1 Round artifacts

Table 4 lists the persistent outputs needed to resume and audit a round. Paths are project-scoped. Each consumer validates the schema for its stage.

Table 4: Core artifacts retained for each AIRE research round.

Stage	Representative artifacts	Purpose
Research	research note, candidate ideas, search receipts	hypothesis evidence and ranking
Data	task, train data, recipe, quality report	data provenance and training readiness
Coding	task, changed files, smoke result	implementation and bounded repair
Experiment	experiment specification, checkpoint, consumption manifest	frozen treatment and execution
Evaluation	ranking, evaluation result, completion manifest	metric binding and safe feedback
Finalize	lineage, promotion decision, report, ledger record	cross-round state and reproducibility

A.2 Allowed research actions

The study will publish the action registry defined for each AIRE-Bench task and shared by all compared systems. Data actions include deterministic filtering, negative mining and backfill, replay construction, mixture planning, and teacher scoring. Training actions include the supported losses, sampler and collator changes, approved hyperparameters, and idea-plugin implementations. A hardware-aware preflight chooses the batch size. We count it as a scientific intervention only when the study explicitly designates it as one.

A.3 Multi-round execution

For each condition–task pair, every benchmark run starts from a clean project directory with a pinned configuration and source revision. The archive will contain the baseline receipt, round states, worker results, model checkpoints required by the reported lineage, public rankings, evaluation completions, and final research summary. The release script rejects a manuscript table when its source artifact is missing or a value cannot be traced to a completed real-mode benchmark run.

A.4 Resource-aware parallel execution

Research, data analysis, and coding mostly use CPUs. Training and corpus encoding use GPUs. AIRE can run these stages for different rounds at the same time once their local dependencies are met. GPU stages acquire explicit leases, and finalization stays local to the coordinator. If the requested hardware is busy, the stage waits; the scheduler does not alter the treatment to make it fit. Figure 6 contrasts sequential execution with this cross-round schedule. The figure explains the scheduling mechanism rather than claiming a measured speedup.

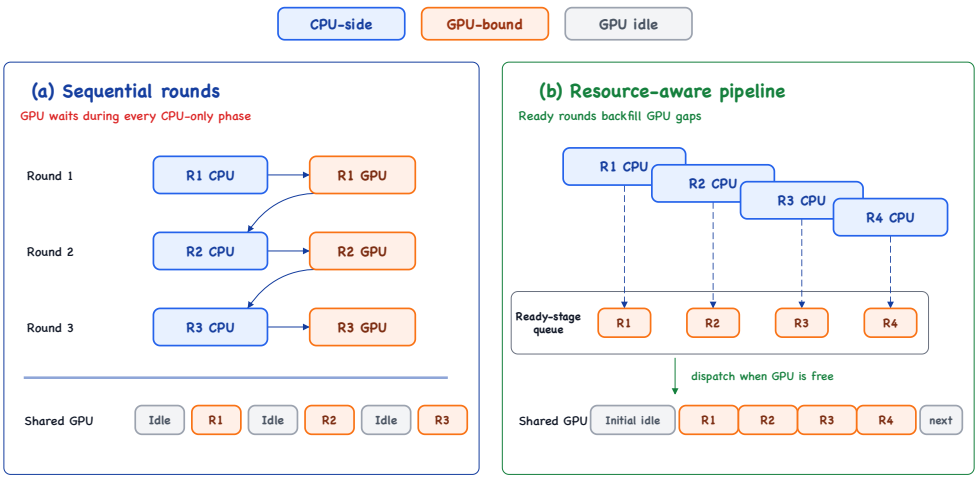

Figure 6: In sequential execution, the GPU waits while a round performs research, data analysis, and coding. AIRE can run a ready stage from another round during that interval without breaking dependencies inside either round. The diagram illustrates the scheduling mechanism rather than a measured utilization or wall-clock improvement.

A.5 Qualitative decision trajectory

Figure 7 instantiates the parallel scheduler with the complete NFCorpus launch. Each upper-panel edge links the round that produced a cross-round decision to the successor round created from that decision. The two branches therefore show concurrent frontier slots rather than independent repeated trials. The lower panel aligns those decisions with their evaluated outcomes under the same protocol-matched baseline.

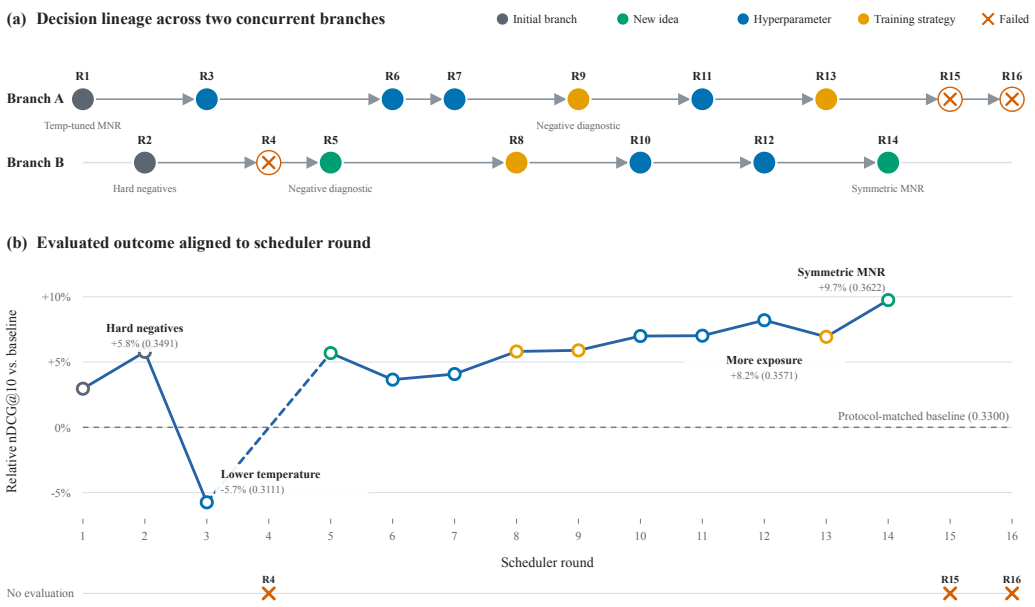

Figure 7: NFCorpus decision trajectory from a complete 16-round AIRE launch. (a) Two concurrent scheduler branches preserve the parent-to-successor lineage; node colors encode the primary intervention selected for each round. (b) Evaluated nDCG@10 is reported relative to the protocol-matched baseline of 0.3300. The dashed R3–R5 connector spans the intervening failed R4 and does not assign it a metric value. R4, R15, and R16 remain visible as execution failures without fabricated scores. R14 attains the best observed result, 0.3622, corresponding to a 9.75% relative gain.

The trace makes the search reversals explicit: hard-negative sampling improves the early result, aggressive temperature sharpening in R3 regresses, later exposure changes recover performance, and the symmetric-MNR branch reaches the best observed score. Retaining the three failed rounds separates execution failures from evaluated scientific outcomes instead of presenting the search as an uninterrupted monotone improvement.

A.6 Agent model configuration

The three reported AIRE-Bench launches use the same agent configuration, summarized in Table 5. These model names refer to the language models that reason and act inside each agent. They are distinct from the text embedding backbone: all three task environments start from `Alibaba-NLP/gte-multilingual-base`, as described in Section 5.

B Additional evaluation details

B.1 Failure injection

We terminate processes before and after result publication, inject invalid schemas and stale artifacts, trigger out-of-memory training, modify code outside the allowed path, and interrupt evaluation. A test passes only if AIRE records the failure, releases resources, and resumes without skipping required work or repeating a completed evaluation.

B.2 Reporting direction coverage

Configured research directions form the denominator of coverage. An accepted idea increments its matched direction. Generated candidates that were not selected do not count. An idea without a reliable match enters an unknown category and does not change the family streak. We report raw counts with the coverage ratio because the ratio can look large when few directions are configured.

Table 5: Language models used by the six AIRE agents in all reported launches.

Agent	Runtime	Language model	Use in AIRE
Coordinator Agent	LLM orchestrator	`qwen3.8-max`	Cross-round planning and next-action selection
Research Agent	Research skill	`deepseek-v4-pro`	Literature-grounded hypothesis generation and candidate ranking
Data Agent	LLM data tools	`deepseek-v4-pro`; `deepseek-v4-flash-0731` (generation)	Data planning and analysis; query and negative generation, reranking, and teacher scoring
Coding Agent	OpenCode	`glm-5.2`	Idea-plugin implementation, testing, and bounded repair
Experiment Agent	LLM tool loop	`glm-5.2`	Experiment execution and live training monitoring
Evaluation Agent	LLM tool loop	`glm-5.2`	Output verification and interpretation of aggregated evaluation results