

Beyond Parallel Sampling: Diverse Query Initialization for Agentic Search

Anonymous ACL submission

Abstract

Test-time scaling for agentic search typically increases depth (i.e., more turns and tokens per trajectory) or breadth (i.e., more parallel rollouts). Here we focus on breadth scaling, showing that standard parallel sampling yields diminishing returns, tracing this to query redundancy at the first turn. When models issue similar first queries across rollouts, the threads retrieve overlapping evidence, and subsequent turns are conditioned on this shared retrieval. We address this limitation with **DivInit**, a training-free intervention at the first turn. Rather than sampling k independent first queries, **DivInit** draws n candidates from a single call, picks $k < n$ diverse seeds, and runs them as parallel trajectories. Across five open-weight models and eight benchmarks, **DivInit** consistently improves over standard parallel sampling, with average gains of five to seven points on multi-hop QA at matched compute.¹

1 Introduction

Test-time scaling has become one of the most reliable ways to improve language model performance on complex tasks without training. On search agents, two practical levers dominate current work. The first, depth, extends a single trajectory through longer chains of thought, more reasoning turns, or more search-and-read iterations (Wei et al., 2022; Muennighoff et al., 2025; Yao et al., 2023; Li et al., 2025). The second, breadth, runs multiple trajectories in parallel and aggregates their outputs through voting, best-of- N selection, or related strategies (Wang et al., 2023; Cobbe et al., 2021; Brown et al., 2024; Snell et al., 2024).

In this work, we focus on parallel test-time scaling of search agents (i.e., breadth). We empirically identify an *anchor collapse* phenomenon in multi-turn agentic search, where the turn-1 query anchors the trajectory and the k parallel threads collapse

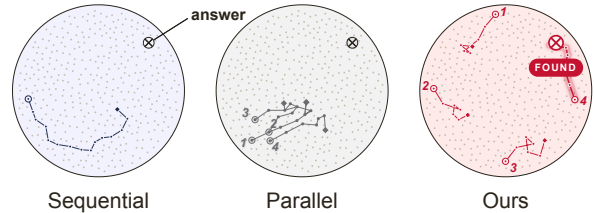


Figure 1: Conceptual illustration of retrieval-path exploration under sequential execution, standard parallel sampling, and our approach named **DivInit**.

onto a single retrieval path. When threads start with similar first queries, overlapping evidence is retrieved, and systems fail in correlated ways. Parallel sampling still outperforms a single sequential trajectory (Li et al., 2026), but the low diversity hints at a suboptimal usage of resources (Figure 1).

While standard parallel scaling launches k independent threads that often anchor on the same turn-1 query, one can instead choose a diverse starting set before any thread runs. The rest of the search agent would be unchanged, making this intervention compatible with any agentic search loop. We name this approach **DivInit**, sampling n candidate queries from a single shared call, picking $k < n$ through Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998), and running one trajectory per chosen query.

We evaluate **DivInit** on five open-weight models across eight benchmarks covering multi-hop QA and long-horizon reasoning. Results show that **DivInit** consistently improves over standard parallel sampling at matched compute, with average gains of five to seven points on multi-hop QA. The improvement grows with model size on most datasets, suggesting a capacity floor below which models cannot act productively on varied first queries. Moreover, diversity at turn 1 is sufficient on its own, and extending the pool selection to later turns yields little benefit, since the first turn separation carries through subsequent retrieval.

¹Code available at: [github](https://github.com)

2 Related Work

Test-time Scaling. Scaling inference-time compute without retraining has proven effective across reasoning and retrieval tasks, through approaches such as chain-of-thought prompting (Wei et al., 2022), budget forcing (Muennighoff et al., 2025), process-reward supervision (Lightman et al., 2023; Uesato et al., 2022), self-consistency (Wang et al., 2023), or best-of- N selection (Cobbe et al., 2021). For long-horizon agentic tasks, AggAgent (Lee et al., 2026) shows that treating parallel trajectories as an interactive environment outperforms voting and answer-concatenation baselines. **DivInit** addresses a different part of the same problem: rather than improving aggregation after trajectories complete, it diversifies where they start.

Agentic Search. ReAct (Yao et al., 2023) introduced the thought-action loop for tool-augmented agents, and IRCot (Trivedi et al., 2023) extended this to multi-hop retrieval. A separate line trains agents via outcome or process rewards (Jin et al., 2025a; Chen et al., 2025; Song et al., 2025; Sun et al., 2025; Wen et al., 2026; Anonymous, 2026). **DivInit** changes only the distribution from which the first query is drawn, before the loop runs, and is compatible with any of the above.

Diversity in Generation and Retrieval. Diverse beam search (Vijayakumar et al., 2016), nucleus sampling (Holtzman et al., 2020), maximal marginal relevance (Carbonell and Goldstein, 1998), and determinantal point processes (Kulesza and Taskar, 2012; Chen et al., 2018) promote diversity within a single generation or retrieval call. **DivInit** operates across parallel agent threads at the query level, directly spreading the retrieval neighborhoods each thread will explore.

3 Anchor Collapse in Agentic Search

Agent Setup. We build on the ReAct-style search agent of Li et al. (2026). Given a question q , the agent iteratively issues search queries, retrieves the top- N documents, and decides whether to continue searching or commit to an answer. Each trajectory runs for at most T turns. Standard parallel sampling draws k independent trajectories for the same question under temperature τ .

Anchor Collapse. To characterize interaction diversity across the k trajectories of a question, we measure two quantities. **Query Pairwise Dis-**

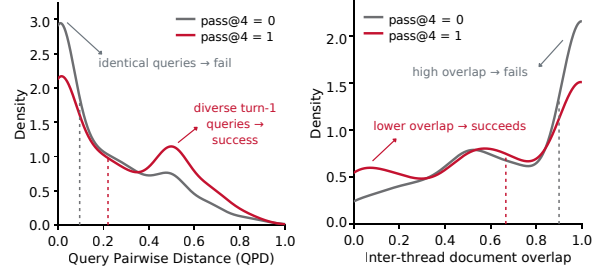


Figure 2: Turn-1 QPD and inter-thread document overlap under standard parallel sampling (Qwen3-8B, $k=4$, $\tau=1.0$), split by pass@4 outcome. Failed questions concentrate at low QPD and high overlap; successful questions show higher QPD and lower overlap.

tance (QPD) is the mean pairwise Jaccard distance among tokens from turn-1 queries, where higher QPD indicates that trajectories initialize from distinct retrieval directions. **Document overlap** is the mean pairwise Jaccard similarity between retrieved document sets, where higher overlap indicates that trajectories consume largely identical evidence despite surface-level query variation.

A preliminary study of standard parallel sampling on the GAIA benchmark (Mialon et al., 2024) shows that these quantities track pass@4 outcome (Figure 2). Questions where all trajectories fail (pass@4=0) cluster at low QPD and high document overlap, whereas questions with at least one successful trajectory (pass@4=1) show a secondary mode with higher QPD and lower overlap.

We hypothesize that this behavior arises from the sequential dependency structure of agentic retrieval. Because each turn conditions on previously retrieved evidence, trajectories with similar turn-1 retrieval actions tend to remain coupled throughout the search process. Early retrieval decisions therefore anchor subsequent reasoning and evidence acquisition, causing the k trajectories to collapse onto a narrow region of the search space. We refer to this phenomenon as **anchor collapse**.

As an example, consider a GAIA question asking for the NASA award number supporting a specific author’s work, linked from a Universe Today article. Under standard parallel sampling, all four trajectories issue identical queries packing every entity into one search (turn-1 QPD 0.098), retrieve the same wrong award number from a different paper, and exhaust their budget without finding the target record. A full trace is provided in Appendix C.

4 DivInit: Diverse Query Initialization

The diagnostic in Section 3 identified low turn-1 QPD as a predictor of failure, i.e., threads that issue near-identical first queries tend to retrieve overlapping evidence and collapse onto a single trajectory. Next we describe **DivInit**, which aims to control this behavior by oversampling queries in the first turn, and selecting a diverse subset.

Procedure. The intervention is restricted to the first turn. While standard parallel sampling has each of k threads generate its own turn-1 query independently, we draw a pool $\mathcal{C}$ of $n > k$ candidate queries in a single LLM call at temperature τ . We select k from $\mathcal{C}$ via MMR (Carbonell and Goldstein, 1998), initializing the selected set S with the pair of candidates at maximum pairwise distance and adding subsequent picks as follows:

$$c^* = \arg \max_{c \in \mathcal{C} \setminus S} \left[(1 - \lambda) \min_{s \in S} d_J(c, s) - \lambda d_J(c, q) \right], \quad (1)$$

where d_J is the token-level Jaccard distance, and q is the original question. We stop when $|S| = k$ and run one thread per selected candidate.

Compute Overhead. The shared pool replaces the k independent turn-1 LLM calls of standard sampling with a single call that emits n candidates in one decoding pass, consolidating k prefills into one at the cost of a longer output. From turn 2, the two methods are identical, so the total cost is $1 + k(T-1)$ calls, $k-1$ fewer than standard’s kT .

5 Experimental Setup

We evaluate on five open-weight models: Qwen3 (1.7B, 4B, 8B) (Yang et al., 2025) and Gemma3 (4B, 12B) (Kamath et al., 2025), all served locally via vLLM (Kwon et al., 2023). Benchmarks span two groups. First, multi-hop question answering (HotpotQA, MuSiQue, 2WikiMHQA, Bamboogle, FRAMES), for which the agent searches over a local Wiki18 BM25 index (Jin et al., 2025b). Second, larger models are also evaluated on open-web reasoning (GAIA, HLE, WebWalker), searching the live web via SERPER (Serper, 2025). We sample 500 questions per benchmark, except for GAIA which uses the full 103-question validation split.

The agent runs $T=8$ turns at $k=4$ threads and temperature $\tau=1.0$. Retrieval returns the textual content of the top 10 documents per query. For **DivInit** (Eq. 1), we draw $n=16$ candidates in the first turn, and set $\lambda = 0$ for MMR. Prompts are detailed in Appendix A.1.

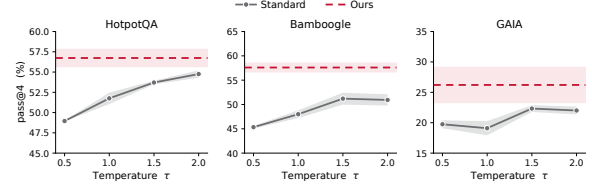


Figure 3: Pass@4 under standard parallel sampling at varying τ (Qwen3-8B, $k=4$), with **DivInit** at $\tau=1.0$ as reference. Higher temperature improves the baseline, but does not close the gap.

6 Experiments

In this section, we measure the empirical gains of **DivInit** over standard parallel sampling, and analyze the query-level patterns that drive them.

6.1 Empirical Results

Table 1 compares **DivInit** against standard parallel sampling under a fixed token budget. Results show that **DivInit** improves on nearly every cell. On the open-web group, WebWalker carries the largest improvements (six to seven points across all three models). While the table shows pass@4, the results also hold for $k = 8$ (Figures 5 and 6 of Appendix B.2) and carry through to pass@1 aggregation under AggAgent (Lee et al., 2026) (Appendix B.3). **DivInit** only differs from the standard parallel scaling strategy on the first turn. A wall-clock time comparison of both is provided in Appendix B.4.

Results also show that improvement scales with model size. Qwen3 averages move from $\Delta=2.8$ at 1.7B to $\Delta=7.4$ at 8B. Notably, the 1.7B numbers point to a capacity floor below which query diversification yields limited benefits.

Figure 3 sweeps $\tau \in \{0.5, 1.0, 1.5, 2.0\}$ for standard parallel sampling, with **DivInit** at $\tau=1.0$ as reference. Performance tends to improve with temperature, but saturates below **DivInit**, confirming that sampling noise is not a substitute for explicit diversity selection.

6.2 Qualitative Analysis

Figure 4 shows that standard parallel sampling concentrates turn-1 queries near low QPD values (≈ 0.2), while **DivInit** shifts them toward high diversity (≈ 0.85). To test if this separation persists beyond the first turn, we measure Across-Thread Distance (ATD), i.e. the mean pairwise Jaccard distance between tokens in queries from different threads, pooled across all turns.

Model	Multi-hop QA (Wiki18)												Open-Web Reasoning (Serper)							
	HpQA		MuSi		2Wiki		Bambo		FRAMES		Avg		GAIA		HLE		WebWalker		Avg	
	S	DI	S	DI	S	DI	S	DI	S	DI	S	DI	S	DI	S	DI	S	DI	S	DI
Qwen3-1.7B	42.9	43.8	14.5	15.6	37.6	41.5	16.8	24.3	13.1	13.6	25.0 $\pm$ 1.3	27.8 $\pm$ 1.1	-	-	-	-	-	-	-	-
Qwen3-4B	41.9	53.2	15.9	19.7	41.9	49.0	32.5	40.8	15.5	20.4	29.5 $\pm$ 1.0	36.6 $\pm$ 1.3	22.7	27.8	9.7	14.3	38.7	44.9	23.7 $\pm$ 1.6	29.0 $\pm$ 2.7
Qwen3-8B	50.4	57.0	23.9	29.7	46.3	55.1	47.7	57.6	24.8	30.8	38.6 $\pm$ 1.2	46.0 $\pm$ 1.3	26.0	30.2	10.0	14.1	41.6	46.8	25.2 $\pm$ 0.8	28.2 $\pm$ 2.6
Gemma3-4B	40.0	49.2	17.2	16.1	42.8	52.2	27.7	37.9	12.3	14.7	28.0 $\pm$ 1.0	34.0 $\pm$ 1.1	-	-	-	-	-	-	-	-
Gemma3-12B	54.9	59.1	31.6	36.1	52.0	53.9	55.7	64.3	31.0	37.5	45.0 $\pm$ 1.1	50.2 $\pm$ 0.9	34.0	35.0	12.7	14.8	38.0	45.2	28.2 $\pm$ 2.2	31.6 $\pm$ 1.4

Table 1: Pass@4 (%) results obtained with a mean over 3 seeds (with standard deviation reported on group averages only). Per-dataset columns show standard parallel sampling (S) vs **DivInit** (DI). Left: multi-hop QA over a local Wiki18 index. Right: open-web reasoning via SERPER.

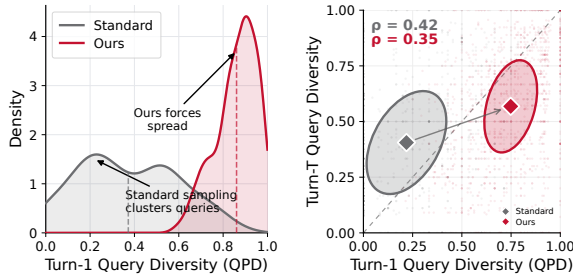


Figure 4: Turn-1 QPD distribution (left) and turn-1 QPD vs. ATD per question (right), Qwen3-8B, $k=4$. **DivInit** shifts queries toward high diversity, and this separation persists across all turns.

Questions with higher turn-1 QPD tend to exhibit higher ATD, with positive correlation in both conditions ($\rho_{\text{std}}=0.42$, $\rho_{\text{DivInit}}=0.35$). This further solidifies that turn-1 retrieval acts as a trajectory anchor, i.e., once threads retrieve different evidence early, subsequent reasoning and search remain separated with little additional intervention. Figures 7 and 8 of Appendix B.2 show this pattern across the full grid of benchmarks and models.

6.3 Ablations

Extent of diversification. **DivInit** applies pool selection only at turn 1. With Qwen3-8B, extending it to turns 1 through N for $N \in \{1, \dots, 8\}$ yields no gain on the open-web benchmarks (Table 2). This matches the observations from §6.2, where early trajectory separation persists throughout later retrieval turns without additional intervention.

Diversity strategy. **DivInit** selects k queries from the pool of n via MMR with $\lambda=0$, which maximizes the minimum pairwise distance in the selected set. On GAIA with Qwen3-8B, this scores 34.0 pass@4. Setting $\lambda \in \{0.5, 0.75\}$ adds weight to similarity between each candidate and the original question, and drops performance to 30–31, indicating that the LLM-generated pool is already

Model	Dataset	Turn N							
		1	2	3	4	5	6	7	8
Qwen3-8B	GAIA	30.7	25.9	27.8	30.7	28.2	28.2	28.2	27.8
	HLE	13.9	12.1	11.6	11.1	12.4	13.1	13.5	13.3
	WebWalker	47.1	45.3	47.1	46.3	45.1	44.3	44.8	44.8

Table 2: Pass@4 (%) results as **DivInit** pool selection extends to turns 1 through N .

on-topic. Uniform random selection from the same pool scores 27.2, below all MMR variants.

7 Conclusions and Future Work

We identified anchor collapse as a failure mode of standard parallel sampling within agentic search systems, where k threads converge onto near-identical first queries, retrieve overlapping evidence, and fail in correlated ways. We also proposed **DivInit** as a training-free fix that oversamples a shared candidate pool of queries and selects a maximally diverse subset before any thread runs. Across five open-weight models and eight benchmarks, the **DivInit** intervention improves over standard parallel sampling at matched compute, with gains of five to seven points on multi-hop QA.

The results suggest that standard breadth scaling underutilizes inference-time compute by allocating rollout budget to correlated trajectories. The same issue may arise in RL training: in group-based methods such as GRPO, near-identical search trajectories produce low-variance rollout groups and weak learning signals (Anonymous, 2026). Diversifying trajectory initialization before group formation is a natural next step. A complementary direction is aggregation. **DivInit** maximizes thread pool diversity but reports pass@ k , while exploiting the structural diversity of the pool to produce a single answer remains open.

Limitations

The $\text{pass}@k$ metric corresponds to the ceiling of what the thread pool can achieve, not the accuracy of any single answer the system would produce in deployment. We also report single-answer accuracy under AggAgent (Lee et al., 2026) in Appendix B.3 and find that diversification gains transfer to $\text{pass}@1$, but closing the remaining gap to $\text{pass}@k$ is an open problem.

Anchor collapse is characterized here for search agents, where the turn-1 decision is a query string and its downstream effect is observable through retrieved documents. Whether the same failure mode arises in other domains, such as tool-use and code, is not addressed in this work.

Ethical Considerations

All the evaluation datasets and pre-trained models used in our experiments are publicly available for research usage. We will provide the source code that allows for reproduction of the results.

By using large pre-trained language models, we acknowledge inherent biases embedded within the models, which may perpetuate or amplify societal biases present in the training data.

References

- Anonymous. 2026. Effective Reinforcement Learning for Agentic Search by Recycling Zero-Variance Queries During Later Training. *Under review*.
- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. *arXiv preprint arXiv:2407.21787*.
- Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *International Conference on Research and Development in Information Retrieval (SIGIR)*.
- Laming Chen, Guoxin Zhang, and Hanning Zhou. 2018. Fast Greedy MAP Inference for Determinantal Point Process to Improve Recommendation Diversity. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*.
- Mingyang Chen and 1 others. 2025. ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning. *arXiv preprint arXiv:2503.19470*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, and 1 others. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. *International Conference on Learning Representations (ICLR)*.
- Bowen Jin and 1 others. 2025a. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. *arXiv preprint arXiv:2503.09516*.
- Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025b. FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research. In *International Conference on World Wide Web (WWW)*.
- Aishwarya Kamath and 1 others. 2025. Gemma 3 Technical Report. *arXiv preprint arXiv:2503.19786*.
- Alex Kulesza and Ben Taskar. 2012. Determinantal Point Processes for Machine Learning. *Foundations and Trends in Machine Learning*, 5(2-3).
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Symposium on Operating Systems Principles*.
- Yoonsang Lee, Howard Yen, Xi Ye, and Danqi Chen. 2026. Agentic Aggregation for Parallel Scaling of Long-Horizon Agentic Tasks. *arXiv preprint arXiv:2604.11753*.
- Xiaochuan Li, Ryan Ming, Pranav Setlur, Abhijay Paladugu, Andy Tang, Hao Kang, Shuai Shao, Rong Jin, and Chenyan Xiong. 2026. Benchmark Test-time Scaling of General LLM Agents. *arXiv preprint arXiv:2602.18998*.
- Xiaoxi Li and 1 others. 2025. Search-o1: Agentic Search-Enhanced Large Reasoning Models. *arXiv preprint arXiv:2501.05366*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s Verify Step by Step. *arXiv preprint arXiv:2305.20050*.
- Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: a benchmark for General AI Assistants. In *International Conference on Learning Representations (ICLR)*.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, and 1 others. 2025. S1: Simple Test-Time Scaling. *arXiv preprint arXiv:2501.19393*.
- Serper. 2025. [Serper.dev Search API](https://serper.dev). Accessed May 19, 2026.

- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM Test-time Compute Optimally Can Be More Effective than Scaling Model Parameters. *arXiv preprint arXiv:2408.03314*.
- Huatong Song and 1 others. 2025. R1-Searcher: Incentivizing the Search Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2503.05592*.
- Hao Sun and 1 others. 2025. ZeroSearch: Incentivize the Search Capability of LLMs without Searching. *arXiv preprint arXiv:2505.04588*.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, and 1 others. 2022. Solving Math Word Problems With Process- and Outcome-Based Feedback. *arXiv preprint arXiv:2211.14275*.
- Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models. *arXiv preprint arXiv:1610.02424*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, and 1 others. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *International Conference on Learning Representations (ICLR)*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Tongyu Wen, Guanting Dong, and Zhicheng Dou. 2026. SmartSearch: Process Reward-Guided Query Refinement for Search Agents. *arXiv preprint arXiv:2601.04888*.
- An Yang and 1 others. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*.

A Reproducibility Details

A.1 Prompts

We use two agent prompts and one pool-generation prompt. All three are reproduced verbatim below (placeholders in curly braces are filled at run time). The multi-hop QA agent uses a ReAct loop with `<thought>`, `<search>`, and `<answer>` actions. The open-web agent adds a `<summary>` action to compress prior `<information>` blocks within longer trajectories. The pool-generation prompt is invoked once at turn 1 in every **DivInit** run to produce the candidate pool of $n=16$ queries.

Multi-hop QA agent prompt

You are a research assistant that answers questions by searching the web. You have {max_turns} turns total. You are on turn {turn}.

Question: {question}

History of searches and findings: {history}

Rules (follow strictly):

1. Every response must use exactly one of these two formats.
2. If you need more information: output `<thought>...</thought>` then `<search>your single search query here</search>`.
3. If you have the answer (or it is the last turn): output `<thought>...</thought>` then `<answer>your concise final answer here</answer>`.
4. Do not output an answer outside `<answer>...</answer>`. Do not output search and answer in the same turn.
5. On the last turn (turn {max_turns}) you MUST output `<answer>...</answer>` with your best guess if unsure.

Open-Web agent prompt

You are a precise research assistant answering web-reasoning questions by searching the internet. You have {max_turns} turns total. You are on turn {turn}.

Question: {question}

Search history: {history}

Available actions per turn (choose exactly one):

- `<search>query</search>` - issue a new search
- `<summary>...</summary>` - compress prior `<information>` blocks
- `<answer>...</answer>` - submit your final answer

Rules:

1. Use EXACTLY one action per response.
2. Answers must be exact: a name, number, date, or short phrase.

3. On turn {max_turns} you MUST give `<answer>...</answer>`.
4. Search for specific facts. Avoid generic queries.

Pool-generation prompt

Generate exactly {n} diverse search queries to investigate this question. Each query should approach the question from a different angle, specifically targeting different constraints or components of the question.

{history_block}

Question: {question}

Output exactly {n} queries, one per line, numbered 1-{n}. No other text.

A.2 Evaluation Protocol

All numbers reported in the main paper are evaluated by gpt-4o-mini (temperature 0, max 300 output tokens). For each question, $\text{pass}@k$ is 1 if at least one of the k candidates is correct, and 0 otherwise. We report the mean over questions. Because the same judge is applied to every condition, any bias affects all methods equally. The total judge cost is approximately \$175.

Judge prompt (multi-hop QA)

You are an expert evaluator. Determine if the generated answer correctly answers the question based on the ground truth answer.

Question: {question}

Ground Truth Answer: {ground_truth}

Generated Answer: {generated_answer}

Evaluation Rubric:

1. Factuality: the answer must contain the core correct information.
2. Semantic equivalence: mark CORRECT if the meaning is the same despite phrasing differences.
3. Completeness: for multi-part questions, all parts must be correctly answered.
4. Contradiction: mark INCORRECT only if the answer directly contradicts the ground truth.
5. Extra information: ignore extra details if the core answer is correct.

Briefly explain your reasoning, then output "CORRECT" or "INCORRECT" on the final line.

A.3 Infrastructure and Reproducibility

Open-weight models are served locally with vLLM in bf16 with prefix caching enabled on $4 \times$ NVIDIA L40S GPUs (48 GB VRAM). Per-rollout seeds are derived deterministically from a SHA-256 hash of the question identifier, the run-level seed, and the rollout index. Search backends are queried with

Dataset	Pass@4		AggAgent@1		Pass@4		AggAgent@1	
	S	DI	S	DI	S	DI	S	DI
<i>Qwen3-4B</i>					<i>Qwen3-8B</i>			
HotpotQA	41.9 $\pm$ 0.6	53.2 $\pm$ 0.5	38.5 $\pm$ 0.2	46.7 $\pm$ 0.7	49.3 $\pm$ 2.8	56.2 $\pm$ 2.0	44.6 $\pm$ 2.3	47.8 $\pm$ 2.1
MuSiQue	15.9 $\pm$ 0.8	19.7 $\pm$ 1.0	13.0 $\pm$ 0.8	13.6 $\pm$ 1.5	24.2 $\pm$ 0.9	31.3 $\pm$ 2.6	13.6 $\pm$ 0.4	17.4 $\pm$ 1.3
2WikiMultihopQA	41.9 $\pm$ 0.3	49.0 $\pm$ 1.0	29.0 $\pm$ 0.3	35.5 $\pm$ 1.3	46.8 $\pm$ 1.3	55.0 $\pm$ 1.2	35.2 $\pm$ 1.7	38.9 $\pm$ 0.9
Bamboogle	32.5 $\pm$ 1.5	40.8 $\pm$ 1.7	30.9 $\pm$ 3.4	37.1 $\pm$ 1.2	47.3 $\pm$ 2.3	57.8 $\pm$ 1.3	40.4 $\pm$ 0.9	46.1 $\pm$ 1.3
FRAMES	15.5 $\pm$ 0.3	20.3 $\pm$ 1.3	14.5 $\pm$ 0.3	19.1 $\pm$ 1.1	24.9 $\pm$ 0.4	30.5 $\pm$ 1.1	20.4 $\pm$ 0.9	22.5 $\pm$ 0.2
<i>Gemma3-4B</i>					<i>Gemma3-12B</i>			
HotpotQA	40.0 $\pm$ 0.8	49.2 $\pm$ 0.3	33.4 $\pm$ 0.2	39.2 $\pm$ 0.3	54.9 $\pm$ 1.0	59.0 $\pm$ 1.2	46.2 $\pm$ 0.3	50.4 $\pm$ 1.0
MuSiQue	17.2 $\pm$ 1.3	16.1 $\pm$ 1.1	17.0 $\pm$ 0.6	15.3 $\pm$ 0.6	31.6 $\pm$ 0.9	36.1 $\pm$ 0.2	22.8 $\pm$ 0.3	30.2 $\pm$ 0.3
2WikiMultihopQA	42.8 $\pm$ 0.7	52.1 $\pm$ 0.6	18.6 $\pm$ 2.2	34.3 $\pm$ 2.4	52.0 $\pm$ 1.0	53.9 $\pm$ 1.1	36.1 $\pm$ 0.2	36.1 $\pm$ 1.0
Bamboogle	27.7 $\pm$ 1.0	37.9 $\pm$ 1.9	20.8 $\pm$ 0.2	31.6 $\pm$ 2.6	55.7 $\pm$ 1.0	64.3 $\pm$ 0.4	46.1 $\pm$ 1.4	55.7 $\pm$ 3.1
FRAMES	12.3 $\pm$ 0.1	14.6 $\pm$ 0.5	12.1 $\pm$ 0.7	13.2 $\pm$ 1.0	30.9 $\pm$ 0.2	37.5 $\pm$ 0.6	25.8 $\pm$ 0.6	28.9 $\pm$ 0.7

Table 3: Pass@4 and AggAgent pass@1 aggregation, comparing standard parallel (S) against **DivInit** (DI) at $k=4$.

Model	Dataset	$n=4$	$n=8$	$n=16$	$n=32$
Qwen3-1.7B	HpQA	42.6	44.1	44.3	44.4
	GAIA	6.5	7.1	7.1	7.4
Qwen3-8B	HpQA	50.6	55.8	54.5	54.0
	GAIA	26.5	30.0	30.6	30.3

Table 4: Pass@4 (%) under **DivInit** as pool size varies.

up to five retries on transient failures, and retrieved text is truncated at 4,000 characters per turn. Total compute is approximately 590 GPU-hours plus \$185 in API costs. The Wiki18 backend is fully reproducible, although SERPER search results drift over time, producing 0.1 to 0.3 points of drift on pass@ k across re-runs.

B Additional Empirical Results

B.1 Pool Size

Table 4 shows results when varying the oversampled pool size n , reporting pass@4 values. The performance is essentially unchanged across $n \in \{8, 16, 32\}$, while $n=4$ shows a drop, suggesting the pool should be larger than k .

B.2 Performance Curves

Figures 5 to 8 report QPD and ATD alongside pass@ k curves over $k \in \{1, 4, 8\}$ for every (model, dataset) cell in both benchmark groups. The pattern from §3 is consistent across the grid: standard parallel sampling concentrates at low QPD, **DivInit** at high QPD, and ATD tracks QPD throughout.

B.3 Aggregation Accuracy

Aggregating k parallel rollouts into a single answer is not a trivial task, with current strategies falling short of the pass@ k ceiling (Li et al., 2026).

Method	Qwen3-1.7B		Qwen3-4B		Qwen3-8B	
	Wall	qps	Wall	qps	Wall	qps
Standard	96.6	2.65	204.6	1.25	285.3	0.90
DivInit	102.4	2.50	119.1	2.15	164.9	1.55

Table 5: Wall-clock time (seconds) and question throughput for first-turn search-query generation on 256 multi-hop questions, on a single L40S GPU.

Our work focuses on pass@ k , leaving aggregation to existing methods. To verify that **DivInit**’s gains transfer past the oracle regime, we run AggAgent (Lee et al., 2026), a state-of-the-art LLM-based aggregator, on both standard and **DivInit** trajectories. Table 3 reports the results. When compared to standard parallel sampling **DivInit** lifts both the pass@4, and AggAgent’s pass@1 metrics. The diversity gain carries through to single-answer accuracy, while a gap to the pass@4 ceiling remains under both conditions.

B.4 First Turn Wall Clock Time

We batch 256 questions in a single vLLM call and time the first turn under each method. Turns 2 through 8 issue one search per thread under both methods and have identical per-turn cost, so we benchmark turn 1 only.

Table 5 shows DivInit is slower at the 1.7B parameter scale, and faster at 4B and 8B. Standard runs $k=4$ reasoning traces per question, each followed by a single query. In contrast, **DivInit** runs a single reasoning trace for $n=16$ queries. At 4B and 8B, the k -fold multiplier on the long reasoning trace dominates, and wall time tracks total tokens. Turns 2 through 8 dilute this turn-1 gap across the full eight-turn rollout.

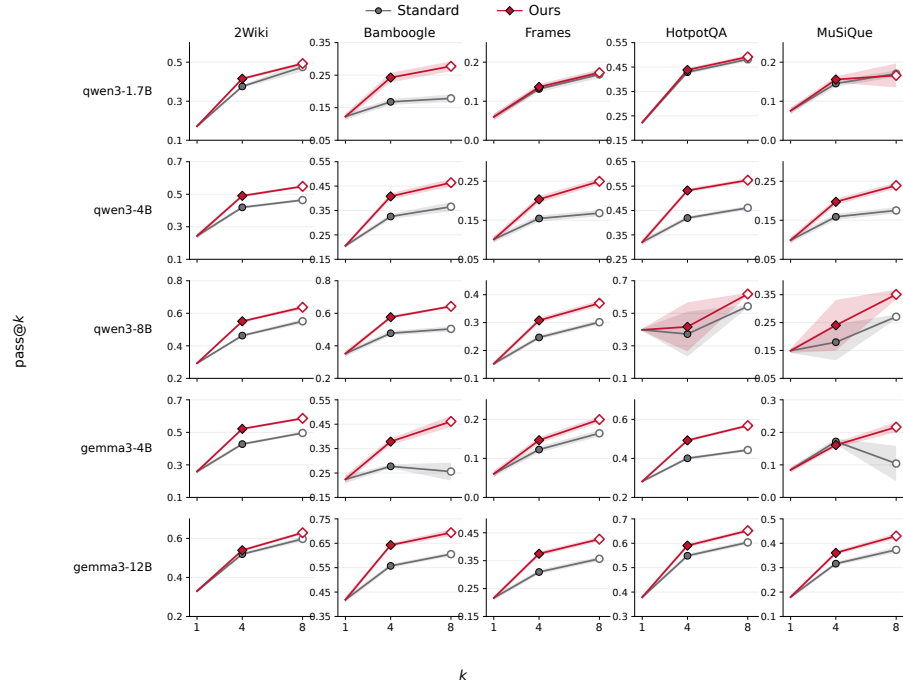


Figure 5: Multi-hop QA results, reporting pass@ k as k varies from 1 to 8, across five models and five datasets. **DivInit** retains its advantage at every k under matched compute.

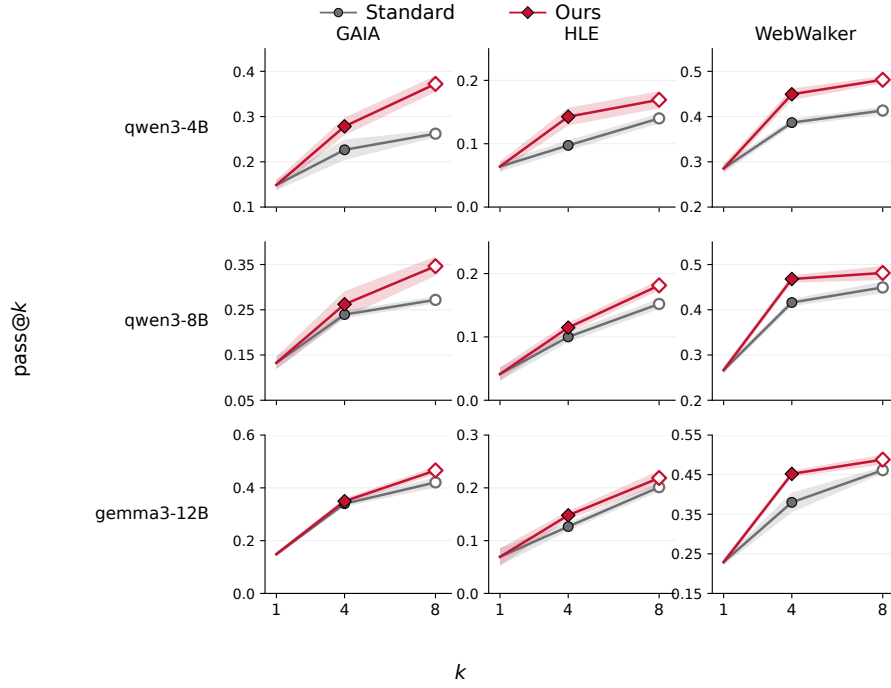


Figure 6: Open-web reasoning results, reporting pass@ k as k varies from 1 to 8. Layout matches Figure 5.

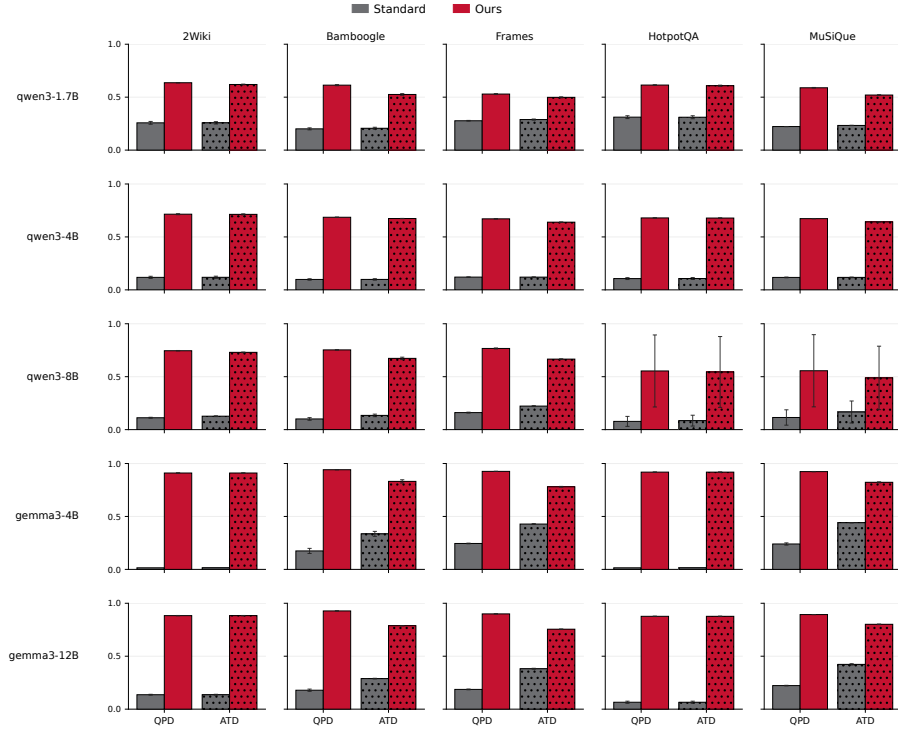


Figure 7: Multi-hop QA results, reporting per-cell QPD and ATD comparing standard (gray) and **DivInit** (red) at $k=4$, across five models and five datasets.

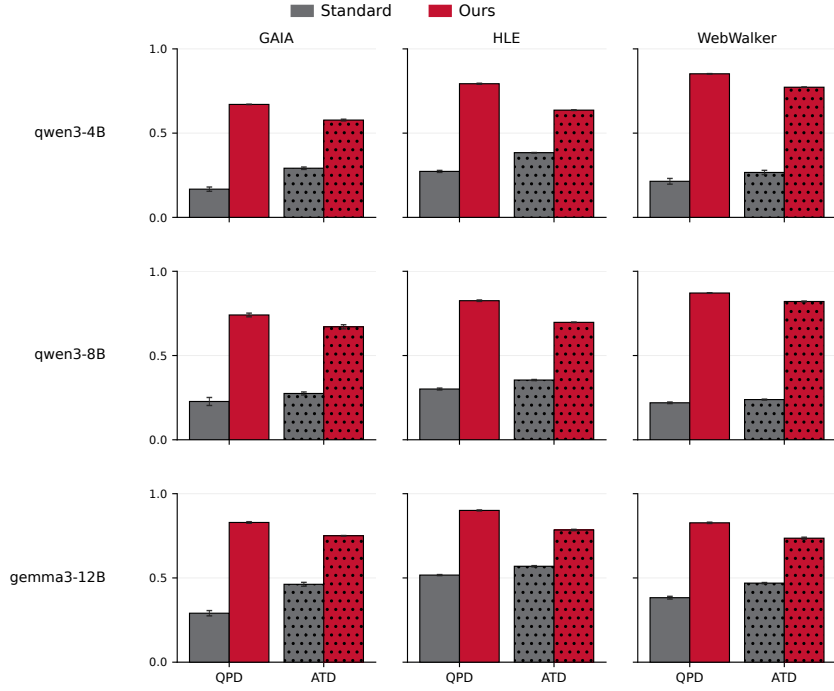


Figure 8: Open-web reasoning results, reporting per-cell QPD and ATD comparing standard and **DivInit** at $k=4$, across three models and three datasets.

C Case Studies

We present four trajectory-level case studies from GAIA experiments with Qwen3-8B at $k=4$ and $n=16$. Cases C.1 and C.2 show where **DivInit** succeeds while standard parallel sampling fails; Cases C.3 and C.4 show the reverse.

C.1 Statistical Significance in Nature 2020

Question. If we assume all articles published by Nature in 2020 relied on statistical significance to justify their findings and they on average came to a p -value of 0.04, how many papers would be incorrect as to their claims of statistical significance? Round up to the next integer.

Gold: 41 **Standard QPD:** 0.120 **DI QPD:** 0.901 **Standard:** ✗ **DI:** ✓

Diagnosis. A two-step problem: look up the Nature 2020 article count, then apply the false-positive rate. All four standard threads issue near-identical queries and either loop or mis-apply the calculation. DivInit seeds probe orthogonal sub-problems, letting one thread assemble both steps correctly.

Oversampled pool ($n=16$; **selected in bold**): **1.** Nature total articles 2020 excl. book reviews; **2.** common p -value thresholds; **3.** understanding p -values; **4.** Nature articles by year; **5.** percent findings incorrect at $p=0.04$; **6.** 2020 Nature by field; **7.** type I error; **8.** p -values in claims 2020; **9.** concerns about p -values; **10.** Nature 2020 by category; **11.** examples of incorrect significance; **12.** reliable significance methods; **13.** Nature on p -value interpretation; **14.** retracted articles for statistical misinterpretation; **15.** $p=0.04$ implications; **16.** Nature’s policy on significance 2020.

Standard Parallel (QPD = 0.120)

Thread 0

↔ search: total number of articles published by Nature in 2020
↔ result: *Volumes | Nature ... fewer than 8% of submitted papers ...*

T2–T5: Loops on same query; never extracts count.

Answer: [malformed]

Thread 1

↔ search: total number of articles published by Nature in 2020
↔ result: *Nature publishes about 15 articles per issue ...*

T2–T3: Finds “12,000” from aggregator.

Answer: 12000

✗

Thread 2

↔ search: number of articles published by Nature in 2020
↔ result: *Article Type Article (1002), Year 2020 (1002) ...*

Mis-applies false-positive rate as 100%.

Answer: 1002

✗

Thread 3

↔ search: total number of articles published by Nature in 2020
↔ result: *Article (1002), Year 2020 ...*

Applies 5% rate without re-reading question.

Answer: 50

✗

Oracle: ✗ All threads fail.

DivInit (QPD = 0.901)

Thread 0 – article-count seed

↔ search: Nature journal total articles 2020 excl. book reviews
↔ result: *Nature publishes peer-reviewed research ...*

T2–T5: Converges on 1002 but cannot finish calculation.

Answer: [malformed]

Thread 1 – false-positive-rate seed

↔ search: percentage of findings incorrect at p -value 0.04
↔ result: *p -value of 0.04 represents a 4% risk ...*

T2–T4: Retrieves rate but cannot find article count.

Answer: [malformed]

Thread 2 – policy seed

↔ search: Nature’s policy on statistical significance in 2020 articles
↔ result: *Scientists rise up against statistical significance ...*

T2–T5: Pivots to article count but loops.

Answer: [malformed]

Thread 3 – incorrect-significance seed

↔ search: examples of incorrect statistical significance in published research
↔ result: *Common statistical errors ...*

↔ search: number of articles published by Nature in 2020

↔ result: *Article (1002) ... $p=0.04 \Rightarrow 4\%$ false-positive rate ...*

Combines: $1002 \times 0.04 = 40.08 \Rightarrow \lceil 40.08 \rceil = 41$.

Answer: 41

✓

Oracle: ✓ Thread 3 assembles both steps correctly.

Mechanism. Standard seeds are lexical variants of the same article-count query; none frames the false-positive sub-problem. DivInit seeds target orthogonal aspects, giving Thread 3 the framing to combine both retrieved facts.

C.2 NASA Award Number Chain

Question. On June 6, 2023, an article by Carolyn Collins Petersen was published in Universe Today. This article mentions a team that produced a paper about their observations, linked at the bottom of the article. Find this paper. Under what NASA award number was the work performed by R. G. Arendt supported?

Gold: 80GSFC21M0002 **Standard QPD:** 0.098 **DI QPD:** 0.743 **Standard:** ✗ **DI:** ✓

Diagnosis. A multi-hop chain: locate the Universe Today article, follow the linked paper, then find the funding record for a specific author. All four standard threads pack every keyword into one query and retrieve the same wrong award number from a different paper. DivInit decompose the chain; three threads independently reach the correct record.

Selected seeds (from $n=16$): (1) Funding source for R. G. Arendt research Universe Today June 6, 2023; (2) Paper linked in Universe Today article Carolyn Collins Petersen June 6, 2023; (3) NASA grant number R. G. Arendt work cited Universe Today June 2023; (4) NASA award number research cited Universe Today article June 6, 2023; 5. Carolyn Collins Petersen Universe Today astrophysics June 2023; 6. R. G. Arendt NASA funding acknowledgment 2023; 7. Universe Today June 2023 linked paper authors; 8. galactic center observations paper June 2023 NASA grant.

Standard Parallel (QPD = 0.098)

Thread 0

↔ search: Carolyn Collins Petersen June 6 2023 Universe Today R. G. Arendt NASA award
↔ result: *NASA award NNX15AF61G ...*

Retrieves wrong award from a different paper.

Answer: NNX15AF61G

✗

Thread 1

↔ search: Carolyn Collins Petersen June 6 2023 Universe Today R. G. Arendt NASA award number
↔ result: *NASA award NNX15AF61G ...*

Same result; budget exhausted.

Answer: NNX15AF61G

✗

Thread 2

↔ search: Carolyn Collins Petersen June 6 2023 Universe Today R. G. Arendt NASA award number
↔ result: *NASA award NNX15AF61G ...*

Identical retrieval path.

Answer: NNX15AF61G

✗

Thread 3

↔ search: Carolyn Collins Petersen June 6 2023 Universe Today R. G. Arendt NASA award number
↔ result: *NASA award NNX15AF61G ...*

Identical retrieval path.

Answer: NNX15AF61G

✗

Oracle: ✗ All threads return the wrong award number.

DivInit (QPD = 0.743)

Thread 0 – funding seed

↔ search: Funding source for R. G. Arendt research in Universe Today June 6, 2023
↔ result: *NASA award 80GSFC21M0002 ...*

Answer: 80GSFC21M0002

✓

Thread 1 – article-first seed

↔ search: Paper linked in Universe Today article Carolyn Collins Petersen June 6, 2023

↔ result: *Paper on galactic center observations ...*

T2–T3: Locates paper but wrong funding entry.

Answer: 80NSSC22K0568

✗

Thread 2 – grant seed

↔ search: NASA grant number R. G. Arendt work cited Universe Today June 2023

↔ result: *NASA award 80GSFC21M0002 ...*

Answer: 80GSFC21M0002

✓

Thread 3 – award seed

↔ search: NASA award number research cited Universe Today article June 6, 2023

↔ result: *NASA award 80GSFC21M0002 ...*

Answer: 80GSFC21M0002

✓

Oracle: ✓ Three threads independently reach the correct award.

Mechanism. Standard threads overload the query with every entity and land on a prominent but wrong record. DivInit seeds approach the chain from different entry points; three of four phrase the funding lookup in a way that surfaces the correct paper and award.

C.3 Doctor Who Script Heading

Question. In Series 9, Episode 11 of Doctor Who, the Doctor is trapped inside an ever-shifting maze. What is this location called in the official script for the episode? Give the setting exactly as it appears in the first scene heading.

Gold: THE CASTLE **Standard QPD:** 0.183 **DI QPD:** 0.631 **Standard:** ✓ **DI:** ✗

Diagnosis. A direct script lookup. Standard Thread 1 happens to retrieve the correct answer through a broad query. DivInit seeds push toward conceptual sub-problems; all four threads land on plot-summary sources that return the episode title instead of the in-script heading.

Selected seeds (from $n=16$): (1) Doctor Who S9E11 official script first scene heading location; (2) exact setting name Doctor Who Series 9 Episode 11 maze script; (3) Doctor Who Heaven Sent script PDF scene heading; (4) Doctor Who S9E11 script download castle location name; 5. Doctor Who Series 9 Episode 11 maze setting name; 6. Heaven Sent Doctor Who location inside maze official name; 7. Doctor Who S9E11 castle setting script heading text; 8. BBC script Doctor Who Heaven Sent first scene.

Standard Parallel (QPD = 0.183)

Thread 0

↔ search: Doctor Who Series 9 Episode 11 location name

Retrieves episode title from plot summaries.

Answer: Heaven Sent

✗

Thread 1

↔ search: unknown language article DDC 633 Bielefeld University BASE 2020

↔ result: ...*exact setting name in script* ...*THE CASTLE* ...

Answer: THE CASTLE

✓

Thread 2

↔ search: Doctor Who Series 9 Episode 11 first scene heading

Finds fandom wikis; returns episode title.

Answer: Heaven Sent

✗

Thread 3

↔ search: Doctor Who Series 9 Episode 11 maze location

Identical retrieval path to Thread 0.

Answer: Heaven Sent

✗

Oracle: ✓ *Thread 1 incidentally retrieves the correct heading.*

DivInit (QPD = 0.631)

Thread 0 – scene-heading seed

↔ search: Doctor Who S9E11 official script first scene heading location

↔ result: *Script PDF ...INT. THE CASTLE ...*

Locates script but returns episode title on answer turn.

Answer: Heaven Sent

✗

Thread 1 – exact-setting seed

↔ search: exact setting name Doctor Who Series 9 Episode 11 maze script

↔ result: *Fandom wiki ...Heaven Sent episode ...*

Answer: Heaven Sent

✗

Thread 2 – script PDF seed

↔ search: Doctor Who Heaven Sent script PDF scene heading

↔ result: *Script ...first heading: THE CASTLE ...*

Retrieves script but mis-formats answer.

Answer: The Castle

✗

Thread 3 – download seed

↔ search: Doctor Who S9E11 script download castle location name

↔ result: *Heaven Sent script ...INT. THE CASTLE ...*

Finds script; answers with episode title instead.

Answer: Heaven Sent

✗

Oracle: ✗ *All DivInit threads miss the exact capitalised heading.*

Mechanism. Diverse seeds collectively land on sources that describe the episode rather than reproduce the script heading verbatim. Standard Thread 1's unstructured query happens to surface a page containing the exact text. This is a failure mode in which query diversity does not help when the answer requires an exact string match from a specific document.

C.4 Rooster and Hamster Song Composer

Question. Who composed the song that was performed by a rooster and a hamster in separate animated videos at separate tempos with different lyrics? Answer using the format “First Last”.

Gold: Roger Miller **Standard QPD:** 0.241 **DI QPD:** 0.784 **Standard:** ✗ **DI:** ✓

Diagnosis. Standard threads retrieve results about Hampton the Hamster, a viral internet character, and never connect the rooster video to identify the original song. Two DivInit threads approach the problem from different angles and independently identify Roger Miller as the composer.

Selected seeds (from $n=16$): (1) animated rooster video song original composer viral; (2) hamster rooster same song different tempo animated videos composer; (3) Roger Miller Robin Hood song rooster hamster animated cover; (4) original song covered by animated rooster and hamster internet videos; 5. Hampton the Hamster original song composer; 6. rooster animated music video viral internet song; 7. Robin Hood whistle-stop song animated animals; 8. viral hamster song same melody rooster video origin.

Standard Parallel (QPD = 0.241)

Thread 0

No query issued on turn 1; malformed output.

Answer: [malformed]

Thread 1

↔ search: song performed by rooster and hamster animated videos separate tempos
↔ result: *Hampton the Hamster ... viral character*
...

Answer: Hampton

✗

Thread 2

↔ search: song performed by rooster and hamster animated videos different tempos
↔ result: *Hampton The Hamster ...*

Answer: Hampton The Hamster

✗

Thread 3

↔ search: song performed by rooster and hamster animated videos
↔ result: *Hampton The Hamster ... Ray Stevens ...*

Answer: Ray Stevens

✗

Oracle: ✗ All threads fixate on the hamster character; none identifies the composer.

DivInit (QPD = 0.784)

Thread 0 – rooster-first seed

↔ search: animated rooster video song original composer viral
↔ result: *Robin Hood rooster animation ... Roger Miller ...*

Answer: Roger Miller

✓

Thread 1 – connection seed

↔ search: hamster rooster same song different tempo animated videos composer
↔ result: *Hampton Hamster ... Michael Stipe ...*

Answer: Michael Stipe

✗

Thread 2 – composer seed

↔ search: Roger Miller Robin Hood song rooster hamster animated cover
↔ result: *Roger Miller composer ... Robin Hood whistle-stop ...*

Answer: Roger Miller

✓

Thread 3 – original-song seed

↔ search: original song covered by animated rooster and hamster internet videos
↔ result: *Walter Scharf ... Robin Hood ...*

Answer: Walter Scharf

✗

Oracle: ✓ Threads 0 and 2 independently identify Roger Miller.

Mechanism. Standard threads all anchor on the hamster character and never reach the rooster side of the connection. DivInit seeds approach the question from different directions; two threads phrase queries in ways that surface the original song and its composer.