An Automated Pipeline for Provably Retrieval-Dependent Benchmark Construction over Dynamic Knowledge

Anonymous Authors[1]

Abstract

Static benchmarks often conflate memorization with reasoning, failing to capture the dynamic nature of world knowledge. We present LIVESEARCHBENCH, a scalable pipeline that automatically constructs provably retrieval-dependent benchmarks from knowledge graph differentials. Unlike prior template-based approaches, our method leverages topology-aware synthesis to generate multi-constraint compositional questions with unique answers guaranteed via SPARQL validation. Evaluating 1,000 questions across three difficulty levels on both open-weight and frontier models, we expose a pronounced "Recency Gap" on novel facts, particularly for compositional queries. Through Oracle and Wikidata-as-corpus analyses, we disentangle dual failure modes: retrieval coverage limitations for single-hop questions and reasoning failures for compositional queries even with ground-truth evidence. LIVESEARCHBENCH provides both a rigorous construction methodology and a continuously updatable testbed for evaluating evidence integration under evolving knowledge.

1. Introduction

Large language models (LLMs) have demonstrated remarkable progress across diverse natural language processing tasks, with solid performance on prominent search question answering (QA) benchmarks such as Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and HotpotQA (Yang et al., 2018). Recent reinforcement learning (RL) methods have further improved headline performance, strengthening the perception that LLMs possess sophisticated reasoning and knowledge-intensive inference capabilities (Jin et al., 2025; Fan et al., 2025). However, a fundamental limitation persists: most search-oriented benchmarks are static snapshots. Many were collected years ago, raising the risk that answers are encoded in models' parametric memory due to pre-training contamination rather than discovered via retrieval (Wu et al., 2025a).

World knowledge is inherently dynamic, news breaks, software versions change, and policies evolve. Yet, prevailing benchmarks often lack mechanisms to incorporate real-time updates. Even recent efforts to introduce dynamic evaluations often focus on checking simple, single-hop fact updates, such as "Who won the 2024 Super Bowl?", which tests a model's *knowledge updating* rather than its *retrieval-based reasoning* process. Because of this, evaluating retrieval systems remains unreliable: models can often answer questions without invoking search, or fail to disambiguate complex queries when new entities emerge. As emphasized by the notion of a *Knowledge Boundary* (Wang et al., 2025; Chen et al., 2025), there is a critical distinction between what a model remembers and what it must acquire externally. Our preliminary experiments in Section 3 corroborate this: models often achieve strong scores even when retrieval is disabled, suggesting that memorized knowledge dominates and obscures the true capacity for reasoning over up-to-date information.

To contextualize the evolution of QA evaluation, Figure 2 highlights key datasets and model milestones. As the timeline shows, community efforts have largely prioritized model development over evaluation under dynamic, complex conditions. Motivated by these gaps, and inspired by the contamination-aware practices of bench (Jain et al., 2024; White et al., 2025), we introduce **LiveSearchBench**, a continually updated benchmark built via a scalable pipeline that synthesizes complex reasoning questions from *Wikidata snapshot differentials*. Unlike prior work, LiveSearchBench goes beyond simple fact-checking. It constructs multi-hop and multi-constraint queries at L2 and L3 levels where the answer is guaranteed to be unique and verifiable through rigorous SPARQL validation against the specific knowledge snapshot. By design, success hinges on up-to-date retrieval and compositional reasoning rather than parametric recall.

Our evaluation yields three key insights. First, by systematically testing LLMs and retrieval-augmented generation

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

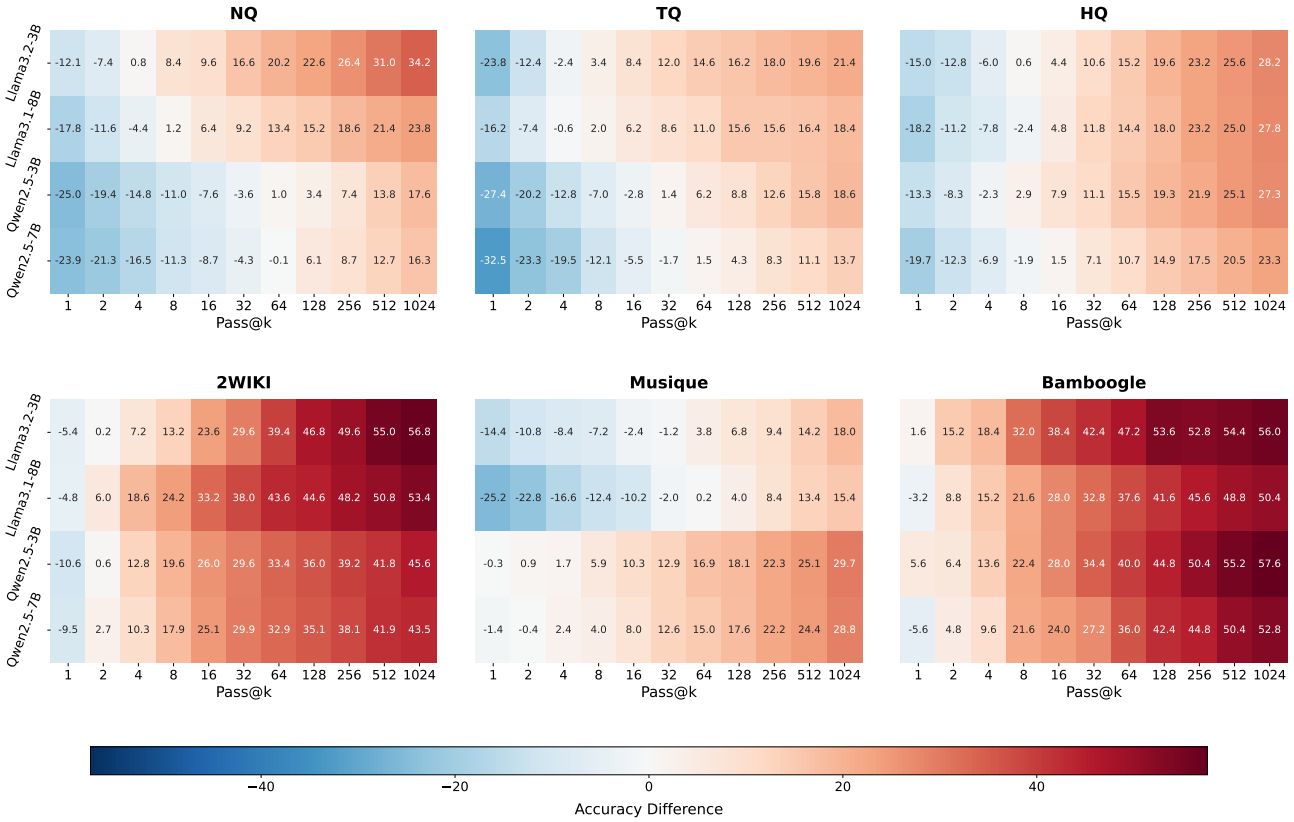

Figure 1. Accuracy difference $\Delta_k = \text{Pass@}k_{\text{no-search}} - \text{Pass@}1_{\text{search}}$ across six QA benchmarks. **Red regions** indicate that scaling parametric sampling (without search) outperforms standard retrieval-augmented generation. **Blue regions** indicate where retrieval is essential. The dominance of red areas in static benchmarks suggests that correct answers are often accessed via memorization rather than necessary retrieval.

(RAG) systems, we expose marked differences in their ability to handle dynamic knowledge compared to static facts. Second, we observe a persistent gap between memorization-driven responses and genuine retrieval-based inference, particularly in complex reasoning tasks. These findings underscore the need for benchmarks that reflect realistic, time-sensitive conditions. This paper makes the following contributions:

◇**Topology-Aware Question Synthesis:** We introduce a data-driven pipeline that mines complex logical structures such as intersection and fuzzy constraints directly from knowledge graph differentials. Unlike template-based approaches limited to predefined schemas, our method adaptively constructs multi-hop reasoning tasks over evolving data topologies.

◇ **Deterministic Logical Grounding:** We address the ambiguity challenge in automated benchmarking by enforcing formal solvability constraints. By anchoring every natural language question to a unique, mathematically verifiable subgraph, we ensure that the evaluation measures genuine reasoning capability rather than guessing games.

◇ **Disentangling Retrieval and Reasoning:** Through extensive evaluations and Oracle-based analysis, we quantify the "Recency Gap" and disentangle the distinct failure modes of LLMs. We show that while retrieval aids in accessing novel entities, current models fundamentally struggle to perform deductive reasoning over retrieved evidence in dynamic contexts.

2. Related Work

Retrieval-Enhanced Inference. Approaches to ground LLMs in external knowledge span three paradigms (Zhang et al., 2025): (i) *Standard RAG*, utilizing retriever-reader architectures (Gao et al., 2024; Fan et al., 2024); (ii) *Agentic Search*, where systems explicitly plan and browse in multi-step loops (Zhou et al., 2025; Li et al., 2025a;b); and (iii) *RL for Search*, which optimizes the search-reasoning coordination via end-to-end reinforcement (Jin et al., 2025; Fan et al., 2025; Sun et al., 2025). These lines differ primarily in policy autonomy and optimization objectives.

Static QA Benchmarks. Static datasets remain the standard for single-hop (Kwiatkowski et al., 2019; Joshi et al., 2017; Wei et al., 2024) and multi-hop reasoning (Yang et al., 2018; Ho et al., 2020; Wei et al., 2025). However, fixed

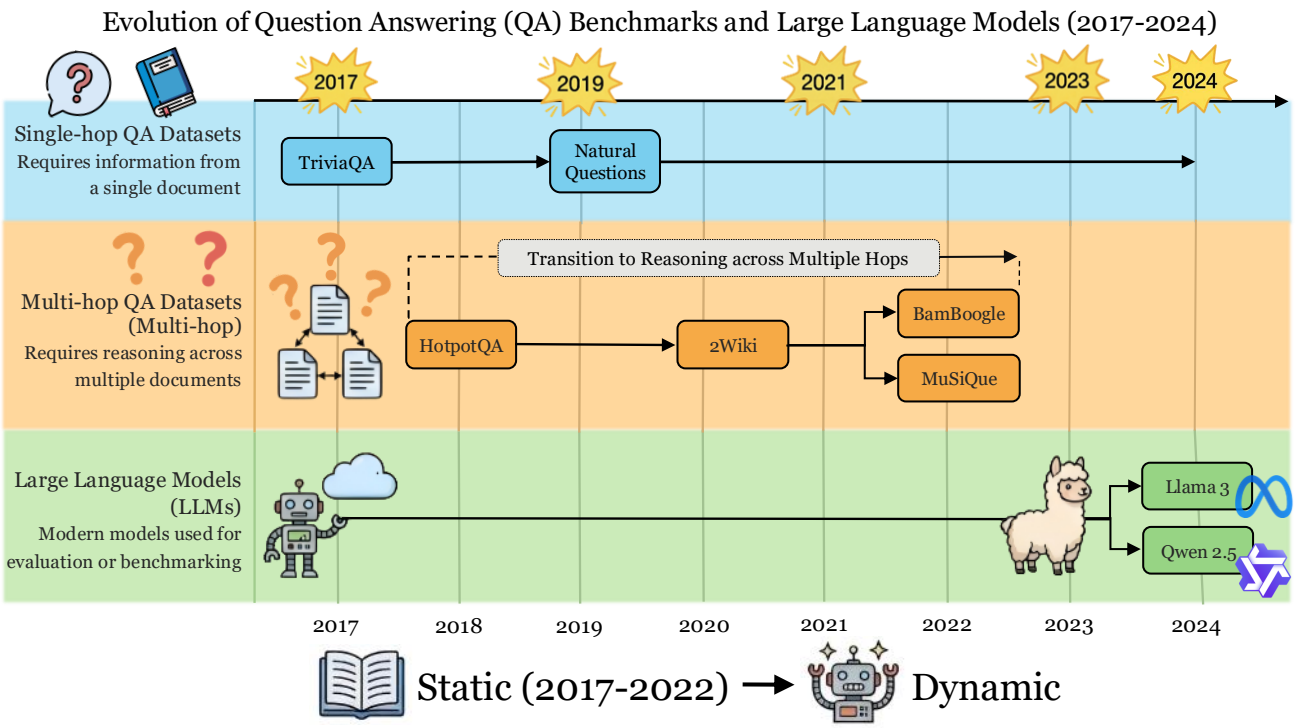

Figure 2. A timeline of major QA benchmarks and model releases. The figure illustrates the historical reliance on static benchmarks, motivating the need for dynamic evaluation resources.

snapshots fail to capture evolving knowledge.

Dynamic and Continual Benchmarks. Recent work addresses this via dynamic evaluation on news streams (Liska et al., 2022; Kasai et al., 2023) or changing facts (Vu et al., 2023). Most relevant is AntiLeakBench (Wu et al., 2025b), which also utilizes Wikipedia edits for contamination-free splits. A critical distinction lies in the generation methodology: while AntiLeakBench supports both single-hop and chain-based multi-hop questions, its reliance on fixed relation templates limits diversity to anticipated schemas and a curated subset of relations. Our method employs data-driven topology synthesis, automatically discovering any changed relation from the knowledge graph delta and identifying compositional structures such as intersections and attribute fuzzing directly from the data. This enables open-ended reasoning beyond rigid templates. Furthermore, we enforce strict answer uniqueness via SPARQL `COUNT=1` validation, providing a mathematical guarantee absent in heuristic template filling and human spot-check verification.

3. Preliminary Analysis: Internal Memory vs. Tool-Augmented Retrieval

Metric Definitions. To rigorously quantify the necessity of retrieval, we compare two inference modes. First, Parametric-Only (Pass@$k_{\text{no-search}}$) measures the probability that a model generates at least one correct answer within k independent samples using only its internal weights via temperature sampling. Second, Retrieval-Augmented (Pass@1_{search}) measures the accuracy of standard RAG, where the model generates a single answer based on retrieved documents via greedy decoding. We define the performance gap as $\Delta_k = \text{Pass@}k_{\text{no-search}} - \text{Pass@}1_{\text{search}}$. A positive Δ_k, shown as red in Figure 1, implies that internal memory scaling is superior to retrieval, while a negative Δ_k, shown as blue, indicates that retrieval provides essential information absent from parameters.

Benchmark-Level Patterns. Figure 1 reveals a systematic trend across widely used datasets. On single-hop benchmarks such as NQ and TQ, Pass@1_{search} offers limited gains over parametric baselines, and as k increases, Pass@$k_{\text{no-search}}$ rapidly overtakes retrieval, turning the heatmap red. This confirms that for static facts, scaling inference simply extracts memorized training data (Wu et al., 2025a). Crucially, this pattern persists even on multi-hop benchmarks such as HotpotQA, 2Wiki, and MuSiQue. At larger k, the red regions dominate, suggesting that what appears to be "reasoning over documents" in these benchmarks can often be solved by sampling from memory. In these cases, retrieval may even degrade performance by introducing noise or distractors, a phenomenon known as the distractor effect, while parametric inference remains robust.

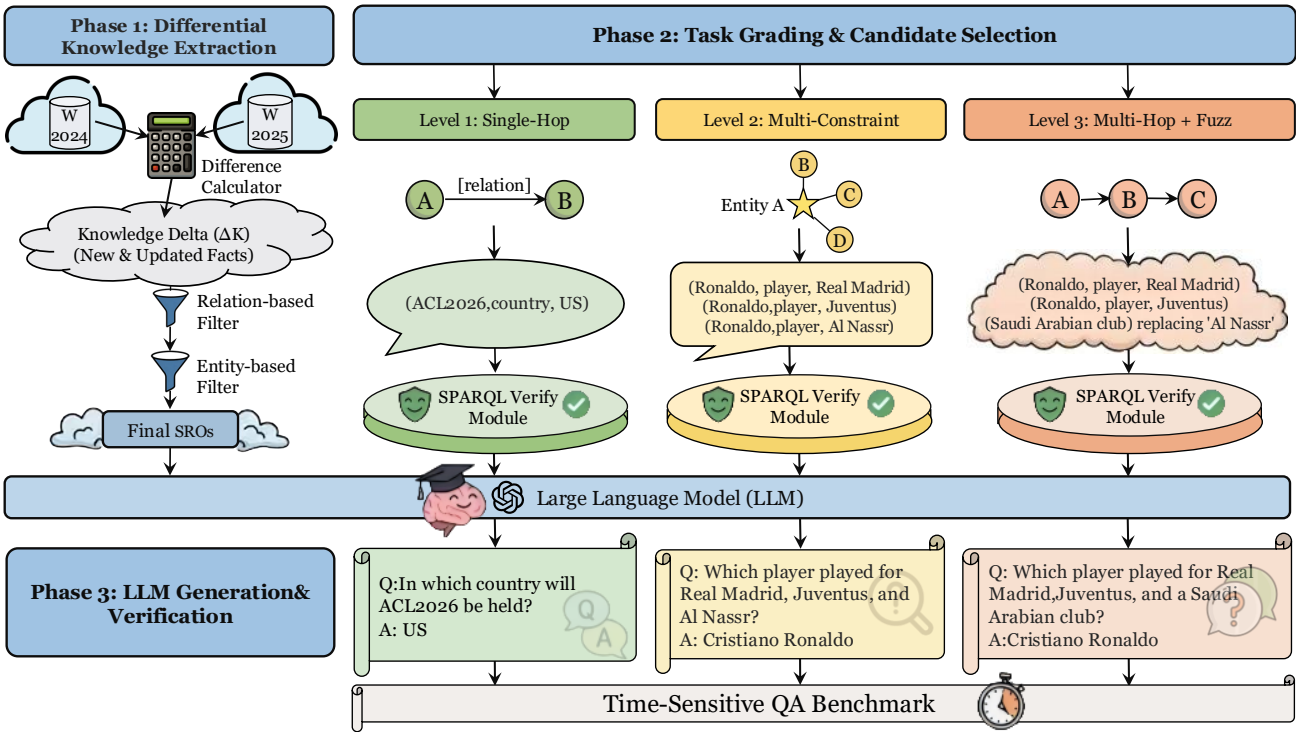

Figure 3. Overview of the generation pipeline. We compute a *knowledge delta* between two Wikidata snapshots to obtain new or updated subject–relation–object (SRO) triples. After relation and entity based filtering, candidate triples are used to synthesize questions at three difficulty tiers: (L1) single-hop, (L2) multi-constraint multi-hop, and (L3) multi-hop with attribute fuzzing. All questions are verified against the current snapshot via SPARQL.

The Illusion of Retrieval Necessity. These observations validate a core hypothesis: static benchmarks overestimate the utility of retrieval tools. They often reward distributional familiarity rather than the ability to acquire and process new information. The fact that Pass@$k_{\text{no-search}}$ can match or exceed RAG suggests that future RL-based reasoning models could "solve" these benchmarks solely by optimizing internal thought chains (Fan et al., 2025; Guo et al., 2025), without ever needing to search. This motivates **LiveSearchBench**, which is designed to ensure a persistent "Blue Region", where Δ_k remains negative regardless of sampling scale, forcing models to rely on external verification rather than memory.

4. LiveSearchBench

4.1. Problem Formulation

To address the evolving nature of world knowledge, we propose leveraging the dynamic updates of the Wikidata knowledge graph to construct question-answering (QA) problems. As Wikidata continually incorporates new information, it provides a rich source of facts that can be used to generate up-to-date QA instances. Building on this idea, we formalize QA in the context of dynamic knowledge graphs. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the Wikidata knowledge graph, where $\mathcal{V}$ is the set of entities and literals, and $\mathcal{E}$ is the set of directed triples (h, r, t) with head $h \in \mathcal{V}$, relation r, and tail $t \in \mathcal{V}$. A question q is formalized as a constrained path query over $\mathcal{G}$, and the gold answer $a^\star \in \mathcal{V}$ (or a literal) must be *unique* under these constraints.

$$\text{Answer}(q, \mathcal{G}_{T_1}) \;=\; a^\star \tag{1}$$

This uniqueness requirement, validated against the snapshot $\mathcal{G}_{T_1}$ via SPARQL queries, ensures that every benchmark instance admits a single, verifiable solution. Consequently, once the *new or updated* triples between two snapshots are extracted, the benchmark can be constructed automatically through a unified pipeline, without the need for manual annotation or domain-specific heuristics.

4.2. Benchmark Design and Generation Pipeline

Design Goals. Our aim is to build a continually updating benchmark that faithfully reflects the evolving nature of world knowledge. The design is guided by four principles: ① questions should target *recent* facts unlikely to reside in an LLM's parametric memory; ② each instance must admit a *unique*, verifiable answer grounded in a public knowledge base; ③ the benchmark should offer controllable difficulty through structured hop levels; and ④ the pipeline should be *fully automated*, ensuring scalability and sustainability with minimal human intervention. We instantiate these goals on WIKIDATA, leveraging its continually evolving knowledge

graph and SPARQL endpoint. This setup guarantees freshness and verifiability while enabling systematic control over reasoning complexity without costly manual curation. Figure 3 presents an overview of our pipeline, which transforms evolving knowledge in WIKIDATA into retrieval-dependent QA instances. The process is fully automated and proceeds in four main stages. Pseudocode for the full pipeline is provided in Appendix §D.2.

Step 1: Differential Knowledge Extraction. We take two Wikidata snapshots at times T_0 and T_1 ($T_1 > T_0$) and normalize each into a set of SRO triples, $\mathcal{G}_{T_0}$ and $\mathcal{G}_{T_1}$. We then construct the *knowledge delta* as the union of insertions and updates:

$$\Delta^{+} = \{t \in \mathcal{G}_{T_1} \setminus \mathcal{G}_{T_0}\}, \tag{2}$$

$$\Delta^{\circ} = \{(s, r, o_1) \in \mathcal{G}_{T_0}, \; (s, r, o_2) \in \mathcal{G}_{T_1} : o_1 \neq o_2\}, \tag{3}$$

$$\Delta = \Delta^{+} \cup \Delta^{\circ}. \tag{4}$$

Here, Δ^{+} captures newly added facts, and Δ° captures *updated* statements where the object set for a given (s, r) changed between snapshots. Every instance therefore anchors to information that post-dates typical pretraining corpora, discouraging memorization and encouraging retrieval.

Step 2: Candidate Filtering. The raw delta may contain noisy or underspecified triples. We apply three filters: (i) Relation allow-list. We exclude non-informative predicates using a curated allow-list. (ii) Entity quality and disambiguation. We require language coverage for labels/aliases, prune entities with incomplete metadata, and remove items whose surface forms are highly ambiguous without additional qualifiers. (iii) Statement validity. We drop deprecated or contradictory statements and deduplicate near-duplicates using normalized keys. The result is a pool of recent, interpretable triples suitable for question synthesis.

Step 3: Hierarchical Question Synthesis. From the curated triples, we synthesize questions at three levels, enforcing a single correct answer via SPARQL `COUNT=1`. L1 (single-hop): directly materialize a triple (a, r, b) and keep it only if b is uniquely identifiable in $\mathcal{G}_{T_1}$. L2 (multi-constraint): start from a target entity and iteratively add attribute constraints such as occupation, country, and affiliation, checking after each addition whether uniqueness is achieved; we stop when `COUNT=1`. L3 (multi-hop with fuzz): extend L2 by first relaxing an attribute to a broader type or hypernym, a process we call fuzzing, and then appending one relational hop; we verify that, despite fuzzing and the extra hop, the query still resolves to a single answer.

Step 4: Natural Language Synthesis and Contextual Verification. We convert the logical structures into natural language and rigorously verify their solvability using a generation-validation loop: (1) Triple-to-Question Generation: We feed the filtered knowledge triples to `gpt-5`. The model is prompted to synthesize a natural language question that strictly requires reasoning over these provided facts to answer. (2) Contextual Solvability Verification: To ensure the generated text semantically aligns with the underlying logic, we provide the generated question along with the original triples, which serve as a "gold context," to Qwen3-235B-A22B-Instruct-2507 without external retrieval tools. The benchmark instance is retained *only if* the model can correctly derive the gold answer solely from the provided triple context. This guarantees that the question is strictly answerable given the evidence and free from ambiguity or hallucinations that might arise during generation.

4.3. Question Complexity Levels

As illustrated in Figure 3, we define three levels of difficulty. The L1–L3 hierarchy defines a controlled progression of difficulty: fact retrieval (L1), compositional reasoning (L2), and ambiguity resolution under fuzziness (L3). By enforcing uniqueness of answers in $\mathcal{G}_{T_1}$, the benchmark remains both rigorous and auditable while reflecting real-world query complexity.

Level-1 (L1): Single-Hop with Uniqueness. Given a source entity $a \in \mathcal{V}$ and a relation $r \in \mathcal{R}$, the task is to identify the unique target b such that

$$|\{b : (a, r, b) \in \mathcal{E}\}| = 1. \tag{5}$$

For example, if the knowledge delta introduces the triple (`ACL2026`, *country*, `America`), the corresponding L1 question is: *"In which country will the ACL2026 conference be held?"* L1 primarily evaluates factual recall of newly introduced triples.

Level-2 (L2): Multi-Hop via Constrained Intersection. To model compositional reasoning, we construct queries where k relational paths ($k \geq 2$) must intersect in exactly one entity. Let $S_i = \{x \mid (a_i, r_i, x) \in \mathcal{E}\}$ be the set of tails for the i-th anchor-relation pair. We require:

$$\left| \bigcap_{i=1}^{k} S_i \right| = 1. \tag{6}$$

For instance, given the triples (`Real Madrid`, *player*, `C. Ronaldo`), (`Juventus`, ...), and (`Al Nassr`, ...), the benchmark synthesizes: *"Which football player has played for Real Madrid, Juventus, and Al Nassr?"* The uniqueness of the intersection ensures the answer is well-defined.

Level-3 (L3): Attribute Fuzzing with an Additional Hop. L3 increases difficulty by generalizing specific entities into

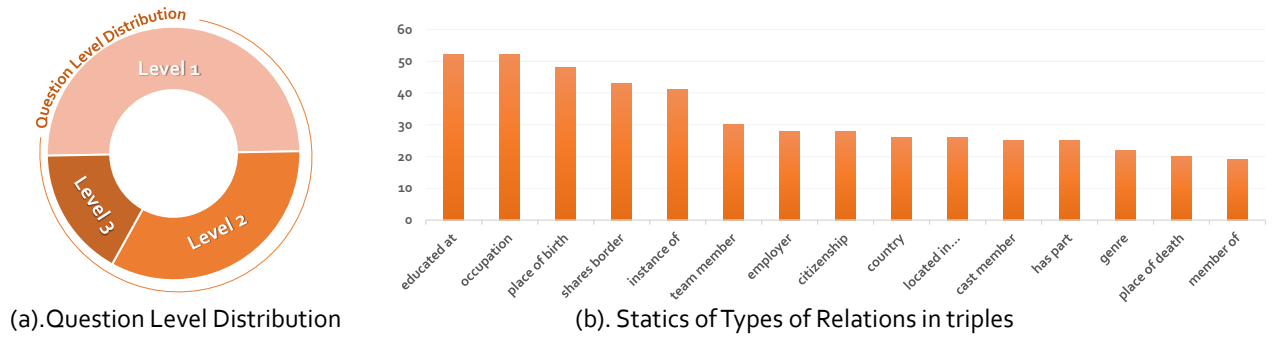

Figure 4. Dataset statistics of LIVESEARCHBENCH. (a) Distribution of questions across difficulty tiers L1–L3. (b) Frequency of the most common relation types in synthesized triples. Together, these plots illustrate both the diversity of reasoning requirements and the breadth of relation coverage in our benchmark.

broader categories (fuzzing), thereby expanding the candidate set before applying constraints. Formally, we replace a specific anchor set S_k with a broadened set S_k^{fuzz}:

$$S_1 = \{x \mid (a_1, r, x) \in \mathcal{E}\}, \tag{7}$$

$$S_2 = \{x \mid (a_2, r, x) \in \mathcal{E}\},$$

$$\text{s.t.} \quad |S_1 \cap S_2 \cap S_3^{\text{fuzz}}| = 1. \tag{8}$$

For example, consider triples linking *player* to anchors `Real Madrid`, `Juventus`, and `Al Nassr`. Instead of querying `Al Nassr` directly, we fuzz it into the broader category "a Saudi Arabian club." The resulting question becomes: *"Which football player has played for Real Madrid, Juventus, and a Saudi Arabian club?"* The intersection implies the unique answer is `Cristiano Ronaldo`.

4.4. Dataset Collection

To build the benchmark, we applied our pipeline to two pairs of Wikidata snapshots. For the recent setting, we used the May 2025 and August 2025 dumps to create LIVESEARCHBENCH-2025; for the historical setting, we used the September 2021 and December 2021 dumps to create LIVESEARCHBENCH-2021. The choice of 2021 vs. 2025 is driven by training cutoff alignment: evaluated models have cutoffs between late 2023 and early 2024, so the 2021 batch falls within their training window as seen knowledge, while the 2025 batch falls after their cutoffs and represents novel knowledge. The 4-year separation avoids ambiguous edge cases near cutoff boundaries. Note that the pipeline accepts any pair of Wikidata snapshots, and can be trivially extended to new time periods. In both cases, all instances are grounded in facts that appeared strictly after the earlier snapshot, ensuring temporal recency and reducing overlap with pretraining data. Starting from 823K raw delta triples in the 2025 batch, the pipeline yields 7,362 verified candidates after filtering, corresponding to a survival rate of approximately 0.89%, with SPARQL uniqueness validation as the main bottleneck. From this verified pool, we sample 1,000 questions: 500 L1, 300 L2, and 200 L3. All instances undergo the same SPARQL uniqueness validation and contextual solvability verification. We report 95% bootstrap confidence intervals across all evaluation tables. Dataset statistics for LIVESEARCHBENCH are shown in Figure 4, illustrating the distribution of questions across difficulty tiers (L1–L3) and the variety of relation types in the synthesized triples.

5. Experiments

5.1. Experimental Setup

Datasets. We evaluate models on two benchmark instances generated by our pipeline, stratified across the three difficulty tiers (L1, L2, L3). The primary evaluation metric is Exact Match (EM) accuracy; we additionally report token-level F1 to verify that conclusions are not driven by metric choice, as detailed in §I. We report 95% bootstrap confidence intervals across all tables. To examine the role of knowledge recency, we construct two batches: 2021 Batch: Derived from updates between Sept. and Dec. 2021. Facts likely overlap with pretraining corpora and thus represent seen knowledge. 2025 Batch: Derived from updates between May and Aug. 2025. Facts post-date training cutoffs and constitute novel knowledge.

Baseline Methods. We group baselines into three categories. Vanilla Prompt Methods include Direct Prompt and Chain-of-Thought (CoT) prompting. RAG-based Methods comprise standard RAG and SEARCH-O1 (Li et al., 2025b). RL-based Methods include SEARCH-R1 (Jin et al., 2025) and SSRL (Fan et al., 2025). To ensure a fair comparison, the number of retrieved passages is capped at $k = 3$ for all RAG approaches. Full implementation details are in Appendix E.

5.2. Main Results

We assess performance across temporal batches (2021 vs. 2025) and difficulty levels. Our analysis centers on recency, retrieval utility, and model scale. Results are detailed in

	LiveSearchBench2021				LiveSearchBench2025			
Model & Method	**L1**	**L2**	**L3**	**Avg.**	**L1**	**L2**	**L3**	**Avg.**
Llama3.2-3B-Instruct								
Direct Answer	22.5	9.0	20.0	17.2	4.5	4.5	2.8	3.9
CoT	23.0	8.5	14.0	15.2	6.0	4.0	1.5	3.8
RAG	41.5	26.0	13.5	27.0	20.0	13.2	9.2	14.1
Search-o1	17.0	17.0	15.5	16.5	7.0	7.5	10.0	8.2
Search-R1-Base	70.0	46.2	20.0	45.4	25.0	22.5	11.2	19.6
Search-R1-Instruct	50.0	43.0	15.5	36.2	19.5	24.8	7.2	17.2
SSRL	66.0	45.8	23.5	45.1	23.0	21.5	13.2	19.2
Qwen2.5-3B-Instruct								
Direct Answer	14.0	9.2	12.0	11.7	3.8	3.2	2.5	3.2
CoT	15.8	8.4	10.2	11.5	3.9	3.6	2.8	3.4
RAG	63.5	31.2	14.8	36.5	24.5	17.0	9.0	16.8
Search-o1	23.0	13.0	11.0	15.7	11.0	4.2	6.8	7.3
Search-R1-Base	56.0	44.0	15.2	38.4	25.2	26.8	13.0	21.7
Search-R1-Instruct	51.0	40.0	11.5	34.2	24.0	24.5	15.0	21.2
SSRL	59.2	36.6	11.0	35.6	20.2	17.6	5.4	14.4
Average across models	40.9	27.0	14.8	27.6	15.5	13.9	7.8	12.4

Table 1. **Exact match accuracy (%) of small models on LiveSearchBench (2021 vs. 2025).** Yellow = best; Blue = second best per model group.

Table 1, with additional results for 7B/14B models provided in Appendix Table 7. A visual summary of absolute and relative retrieval gains across both batches is provided in Appendix Figure 5.

Retrieval vs. Parametric Memory. Dynamic evaluation exposes the critical role of retrieval. On the 2021 batch, retrieval yields moderate gains, as models often recall facts from parametric memory. In contrast, on the 2025 batch, retrieval becomes indispensable. For instance, Llama3.2-3B shows a massive relative gain from RAG on 2025 data compared to 2021. This confirms that while models can "hallucinate" correct answers for old data, they are helpless on LiveSearchBench2025 without external tools.

The Recency Gap. Across all models, performance drops significantly from 2021 to 2025. Since the question generation pipeline is identical, this gap isolates the difficulty of **Novel Knowledge**. It highlights that current LLMs, even when augmented with search, struggle to synthesize answers when the underlying facts contradict or are absent from their pre-training priors.

Model Scale and Families. Scaling from 3B to 14B within the Qwen series yields consistent gains, as shown in Appendix Table 7. Larger models are better at identifying relevant snippets from noisy retrieval results. Interestingly, while Llama3.2-3B outperforms Qwen2.5-3B on the 2021 batch, likely due to different pre-training data distributions,

	2021				2025			
Model	**L1**	**L2**	**L3**	**Avg**	**L1**	**L2**	**L3**	**Avg**
GPT-5.2	85.2	67.3	44.0	65.5	56.8	39.7	24.5	40.3
Claude 4.5	82.4	64.0	41.0	62.5	53.2	37.3	22.0	37.5
Gemini 3 Pro	79.6	60.7	38.0	59.4	50.4	35.0	20.5	35.3

Table 2. **Frontier models (RAG, k=3) on LiveSearchBench.** Despite substantially higher absolute accuracy, the Recency Gap persists (38–42% relative drop from 2021 to 2025).

Qwen models often catch up or lead on the 2025 batch when using RAG. This suggests that some model families may be better optimizers of *context utilization* even if their parametric knowledge is weaker.

5.3. Frontier Model Evaluation

We further evaluate three frontier models, namely GPT-5.2, Claude Sonnet 4.5, and Gemini 3 Pro, under the same RAG setting. Results are shown in Table 2.

Despite substantially higher absolute accuracy, the **Recency Gap persists**, with a 38–42% relative drop from 2021 to 2025. Even GPT-5.2 reaches only 24.5% on L3-2025, compared to an Oracle score of approximately 41.5%, confirming that compositional reasoning over novel evidence remains challenging across model scales.

Retrieval Source	L1	L2	L3	Avg
Google RAG	35.4	26.3	17.5	26.4
Wiki-corpus (BM25)	79.2	46.3	24.0	49.8
Wiki-corpus (Contriever)	84.6	50.7	27.0	54.1
Oracle (triple injection)	96.2	71.3	41.5	69.7

Table 3. **Wikidata-as-corpus and Oracle analyses on Qwen2.5-14B, LSB-2025.** Moving from a noisy search engine to a clean Wikidata corpus closes most of the L1 gap (35.4 % → 84.6 %), nearly matching Oracle (96.2 %); the same move barely helps L3 (17.5 % → 27.0 %), which still trails the Oracle of 41.5 %.

	Sampling budget k					
Batch	**1**	**4**	**16**	**64**	**256**	**1024**
LSB-2021	−12.4	−8.6	−5.9	−4.1	−3.0	−2.3
LSB-2025	−19.6	−16.2	−13.5	−11.8	−10.1	−8.9

Table 4. Δ_k **on LIVESEARCHBENCH, Qwen2.5-14B.** $\Delta_k = \text{Pass@}k_{\text{no-search}} - \text{Pass@}1_{\text{search}}$. On the seen-knowledge 2021 batch Δ_k converges toward zero; on the novel-knowledge 2025 batch it remains strongly negative even at $k = 1024$, confirming a *persistent* Blue Region.

5.4. Disentangling Retrieval vs. Reasoning

To pinpoint whether RAG failures stem from *search-engine coverage* or from *reasoning*, we run two complementary analyses. The **Wikidata-as-Corpus** setting replaces Google search with a corpus built from ground-truth Wikidata triples (retrieved by BM25 / Contriever), eliminating engine coverage as a confound; the **Oracle** setting injects those triples directly into context. Results on Qwen2.5-14B for LSB-2025 are in Table 3.

A clear **dual failure mode** emerges. On L1, accuracy jumps from 35.4 % (Google RAG) to 84.6 % under Contriever and nearly matches the 96.2 % Oracle, pinning the bottleneck on *retrieval coverage* of fresh, long-tail entities. On L3, even a clean corpus only reaches 27.0 %, still far below the 41.5 % Oracle, so the bottleneck has shifted to *reasoning over evidence*: models cannot compose multi-constraint queries even with every supporting triple in context. LIVESEARCHBENCH thus isolates two qualitatively different research targets, entity-aware retrieval for fresh facts and compositional deduction over novel evidence.

5.5. Pass@k: A Persistent Blue Region

A natural concern is whether LIVESEARCHBENCH merely shifts the parametric/retrieval crossover from Section 3 to a larger k, or whether it sustains a regime where retrieval is genuinely necessary. Table 4 reports Δ_k for Qwen2.5-14B on both batches.

On LSB-2021, Δ_k converges toward zero with k, as on legacy benchmarks. On LSB-2025, Δ_k stays at -8.9 even at $k = 1024$, confirming a **persistent Blue Region**: failure on dynamic knowledge cannot be papered over by scaling inference compute.

6. Conclusion

We introduced LIVESEARCHBENCH to shift evaluation from static memorization to dynamic reasoning. By synthesizing complex, multi-constraint queries with formal uniqueness guarantees, we expose a critical **"Recency Gap"** where models falter on novel knowledge despite strong historical performance. Our analysis traces this gap to a dual failure: the inability of retrievers to capture long-tail updates and the struggle of reasoners to deduce connections over new evidence. LIVESEARCHBENCH thus offers a rigorous testbed for Search-RAG agents, benchmarking genuine real-time adaptability beyond parametric memory.

Our findings motivate entity-aware retrieval for emerging facts and architectures for structured deduction over novel evidence; the pipeline supports arbitrary snapshot pairs and thus enables longitudinal study of retrieval-augmented reasoning over time.

Limitations

First, our reliance on structured Wikidata differentials restricts the benchmark to fact-centric queries, excluding unstructured narratives. This is a deliberate trade-off for mathematical verifiability via SPARQL. Second, while Exact Match (EM) may penalize surface-form variations, we additionally report token-level F1 as shown in Appendix I and find method rankings consistent across both metrics.

Ethics Statement

Data derive from *public* Wikidata snapshots under open licenses; the automated pipeline involves no personal data, no human annotation, and no foreseeable ethical risks.

References

Chen, X., Wang, S., Qian, C., Wang, H., Han, P., and Ji, H. Decisionflow: Advancing large language model as principled decision maker, 2025. URL https://arxiv.org/abs/2505.21397.

Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., and Li, Q. A survey on rag meeting llms: Towards retrieval-augmented large language models, 2024. URL https://arxiv.org/abs/2405.06211.

Fan, Y., Zhang, K., Zhou, H., Zuo, Y., Chen, Y., Fu, Y., Long, X., Zhu, X., Jiang, C., Zhang, Y., et al. Ssrl: Self-search reinforcement learning. *arXiv preprint arXiv:2508.10874*, 2025.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. Retrieval-augmented generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Ho, X., Duong Nguyen, A.-K., Sugawara, S., and Aizawa, A. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Scott, D., Bel, N., and Zong, C. (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Jin, B., Zeng, H., Yue, Z., Yoon, J., Arik, S. O., Wang, D., Zamani, H., and Han, J. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Barzilay, R. and Kan, M.-Y. (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Kasai, J., Sakaguchi, K., Le Bras, R., Asai, A., Yu, X., Radev, D., Smith, N. A., Choi, Y., Inui, K., et al. Realtime qa: What's the answer right now? *Advances in neural information processing systems*, 36:49025–49043, 2023.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Li, K., Zhang, Z., Yin, H., Zhang, L., Ou, L., Wu, J., Yin, W., Li, B., Tao, Z., Wang, X., Shen, W., Zhang, J., Zhang, D., Wu, X., Jiang, Y., Yan, M., Xie, P., Huang, F., and Zhou, J. Websailor: Navigating super-human reasoning for web agent, 2025a. URL https://arxiv.org/abs/2507.02592.

Li, X., Dong, G., Jin, J., Zhang, Y., Zhou, Y., Zhu, Y., Zhang, P., and Dou, Z. Search-o1: Agentic search-enhanced large reasoning models, 2025b. URL https://arxiv.org/abs/2501.05366.

Liska, A., Kocisky, T., Gribovskaya, E., Terzi, T., Sezener, E., Agrawal, D., D'Autume, C. D. M., Scholtes, T., Zaheer, M., Young, S., et al. Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. In *International Conference on Machine Learning*, pp. 13604–13622. PMLR, 2022.

Sun, H., Qiao, Z., Guo, J., Fan, X., Hou, Y., Jiang, Y., Xie, P., Zhang, Y., Huang, F., and Zhou, J. Zerosearch: Incentivize the search capability of llms without searching, 2025. URL https://arxiv.org/abs/2505.04588.

Vu, T., Iyyer, M., Wang, X., Constant, N., Wei, J., Wei, J., Tar, C., Sung, Y.-H., Zhou, D., Le, Q., and Luong, T. Freshllms: Refreshing large language models with search engine augmentation, 2023.

Wang, H., Qian, C., Li, M., Qiu, J., Xue, B., Wang, M., Ji, H., and Wong, K.-F. Toward a theory of agents as tool-use decision-makers. *arXiv preprint arXiv:2506.00886*, 2025.

Wei, J., Karina, N., Chung, H. W., Jiao, Y. J., Papay, S., Glaese, A., Schulman, J., and Fedus, W. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024.

Wei, J., Sun, Z., Papay, S., McKinney, S., Han, J., Fulford, I., Chung, H. W., Passos, A. T., Fedus, W., and Glaese, A. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

White, C., Dooley, S., Roberts, M., Pal, A., Feuer, B., Jain, S., Shwartz-Ziv, R., Jain, N., Saifullah, K., Dey, S., Shubh-Agrawal, Sandha, S. S., Naidu, S. V., Hegde, C., LeCun, Y., Goldstein, T., Neiswanger, W., and Goldblum, M. Livebench: A challenging, contamination-free LLM benchmark. In *The Thirteenth International Conference on Learning Representations*, 2025.

Wu, M., Zhang, Z., Dong, Q., Xi, Z., Zhao, J., Jin, S., Fan, X., Zhou, Y., Lv, H., Zhang, M., Fu, Y., Liu, Q., Zhang, S., and Zhang, Q. Reasoning or memorization? unreliable results of reinforcement learning due to data contamination, 2025a. URL https://arxiv.org/abs/2507.10532.

Wu, X., Pan, L., Xie, Y., Zhou, R., Zhao, S., Ma, Y., Du, M., Mao, R., Luu, A. T., and Wang, W. Y. AntiLeakBench: Preventing data contamination by automatically constructing benchmarks with updated real-world knowledge. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18403–18419, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.901. URL https://aclanthology.org/2025.acl-long.901/.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Zhang, G., Geng, H., Yu, X., Yin, Z., Zhang, Z., Tan, Z., Zhou, H., Li, Z., Xue, X., Li, Y., Zhou, Y., Chen, Y., Zhang, C., Fan, Y., Wang, Z., Huang, S., Liao, Y., Wang, H., Yang, M., Ji, H., Littman, M., Wang, J., Yan, S., Torr, P., and Bai, L. The landscape of agentic reinforcement learning for llms: A survey, 2025. URL https://arxiv.org/abs/2509.02547.

Zhou, H., Geng, H., Xue, X., Kang, L., Qin, Y., Wang, Z., Yin, Z., and Bai, L. Reso: A reward-driven self-organizing llm-based multi-agent system for reasoning tasks. *arXiv preprint arXiv:2503.02390*, 2025.

A. Use of Large Language Models (LLMs)

We used the large language model ChatGPT 5.2 as an assistive tool in two ways: first, for writing assistance, including language editing and improving the clarity of the manuscript, and second, for technical support during code environment setup and debugging, particularly when resolving environment-related errors. The model was not used for generating research ideas, designing methodologies, conducting experiments, or analyzing results. All outputs from the LLM were manually verified by the authors, and final decisions regarding both the research content and the manuscript were made by the authors. The authors take full responsibility for the entirety of this work.

B. Quality Assurance and Human Verification

While the pipeline is automated, we conducted a rigorous human verification phase to ensure the dataset's reliability and solvency. Process: We recruited five domain experts, specifically PhD candidates in NLP and Knowledge Graphs who are independent of the author list. **Task**: Reviewers were presented with the triple (s, r, o), the synthesized question Q, and the answer A. They annotated each instance for three criteria: *Factuality*, assessing whether the triple is supported by the snapshot; *Clarity*, assessing whether Q is unambiguous; and *Solvability*, assessing whether Q uniquely leads to A. Protocol: A subset of 100 questions was double-annotated blindly. We calculated an inter-annotator agreement (Cohen's κ) of 0.85, indicating high reliability. Disagreements were resolved by a third senior expert. Following this protocol, the full set of 1,000 questions was verified. This rigorous filtering ensures that LIVESEARCHBENCH serves as a high-precision, noise-free evaluation standard, guaranteeing that any observed performance drops stem from model limitations rather than dataset artifacts.

C. Oracle Analysis

To rigorously investigate the sources of error and the performance gap, we conducted an Oracle Study on a subset of 50 failure cases from the 2025 batch. We isolated the reasoning component by providing models directly with the ground-truth triples converted into natural language text, thereby bypassing the retrieval stage. Result: Under Oracle conditions, L1 accuracy surged to over 95%. This dramatic improvement serves as a strong validation of dataset quality: it confirms that the questions are fundamentally answerable, factually correct, and free from ambiguity when the correct evidence is provided. The primary bottleneck for single-hop dynamic questions is thus confirmed to be Retrieval Failure, namely the inability to access fresh facts, rather than data quality. In contrast, for L3 (Fuzzing) questions, accuracy only improved to $\sim$40%. Since human experts verified these

questions as solvable, this persistent gap indicates that models struggle to deduce correct answers from complex constraints even when the necessary evidence is present. This disentanglement demonstrates that LIVESEARCHBENCH effectively stresses two distinct capabilities: the *retriever's* capacity to index fresh, long-tail information (dominating L1), and the *reasoner's* capacity to perform logical deduction over provided evidence (dominating L3).

D. Pipeline Details

D.1. Filtering Procedure

The filtering procedure consists of three main steps to ensure high-quality and interpretable relations for QA. We maintain a curated filter-list that excludes meta/formatting predicates, focusing on retaining only those relations that yield interpretable QA. A detailed list of the excluded predicates is provided in Table 5. To ensure comprehensive language coverage, we prune entities with incomplete metadata and remove those with highly ambiguous surface forms unless additional qualifiers are available to clarify the context. Additionally, we eliminate deprecated or contradictory statements and deduplicate near-duplicate entries by normalizing keys. The preferred method for normalization is using the statement ID, but if unavailable, we rely on a combination of (s, r) along with label normalization.

D.2. Finalization and Validation Pseudocode and SPARQL Templates

```
SELECT ?b WHERE {
  wd:Q_a wdt:P_r ?b .
  # Optional: Apply filters for rank and time validity.
} LIMIT 2
```

Listing 1. SPARQL sketch for an L1 query. The instance is accepted only if the query returns exactly one result.

```
SELECT ?x WHERE {
  { wd:Q_a  wdt:P_r1 ?x .  FILTER(phi_1(?x)) }
  UNION
  { wd:Q_a' wdt:P_r2 ?x .  FILTER(phi_2(?x)) }
} GROUP BY ?x HAVING (COUNT(?x)=2)
```

Listing 2. SPARQL sketch for an L2 query. The HAVING clause ensures that ?x satisfies both constraints.

```
SELECT ?x WHERE {
  { wd:Q_a  wdt:P_r1 ?x .  FILTER(phi_1_fuzzy(?x)) }
  UNION
  { wd:Q_a' wdt:P_r2 ?x .  FILTER(phi_2_fuzzy(?x)) }
  UNION
  { ?x      wdt:P_r3 C_c . FILTER(phi_3(?x)) }
} GROUP BY ?x HAVING (COUNT(?x)>=3)
```

Listing 3. SPARQL sketch for an L3 query with fuzzy constraints and an additional hop.

```
def generate_benchmark(dump_T0, dump_T1):
    # 1. Differential Knowledge Extraction
    G_T0 = extract_triples(dump_T0)
    G_T1 = extract_triples(dump_T1)
    knowledge_delta = G_T1.difference(G_T0)

    # 2. High-Quality Candidate Filtering
    curated_delta = filter_triples(knowledge_delta,
                                   rules=['relation_type', 'entity_quality', 'statement_rank'])

    # 3. Hierarchical Question Synthesis from recent facts
    benchmark = []
    for seed_triple in curated_delta:
        # Attempt to build questions of increasing difficulty
        question = None
        if not question:
            question = synthesize_question(seed_triple, G_T1, level='L1')
        if not question:
            question = synthesize_question(seed_triple, G_T1, level='L2')
        if not question:
            question = synthesize_question(seed_triple, G_T1, level='L3')

        # 4. Finalization
        if question and is_valid(question):
            final_instance = render_and_finalize(question)
            benchmark.append(final_instance)

    return benchmark

def synthesize_question(triple, graph, level):
    # Builds a SPARQL query based on the level and seed triple.
    # For L2/L3, this involves finding additional constraining triples.
    query = build_sparql_query(triple, graph, level)

    # Validates that the query has a unique answer in the new graph.
    if is_unique_in_graph(query, graph):
        return (query, triple.answer)
    return None
```

Listing 4. End-to-end pipeline for generating benchmark questions from snapshots.

Table 5. The curated filter-list of excluded meta/formatting predicates.

Property ID	Description
P18	image
P31	instance of (often too basic)
P279	subclass of
P373	Commons category
P443	pronunciation audio
P460	said to be the same as
P856	official website
P910	topic's main category
P973	described at URL
P1151	topic's main Wikimedia portal
P1343	described by source
P1424	topic's main template
P1559	name in native language
P1629	Wikidata property
P1630	formatter URL
P1659	related property
P1687	Wikidata property
P1696	inverse property
P1705	native label
P1793	regular expression
P1855	Wikidata property example
P1889	different from
P1921	URI template
P2302	property constraint
P2700	protocol
P2875	property for this type
P2916	source website for the property
P2959	permanent duplicated item
P3254	property usage tracking category
P3709	unit symbol
P3713	pronunciation audio

E. Implementation Details

E.1. Implementation of Baselines

For ZeroSearch, Search-R1, and SSRL, we set the temperature to 0.7, and the max response length to 4096. We do not restrict their max turns for search, so that they can search as many times as they want. We use Exact Match (EM) as our evaluation metric. The prompt we use is listed in Table 6. We use Google Search via Serper API for all all them to ensure fairness.

F. Larger Model Results (Qwen 7B/14B)

Table 7 shows results for larger Qwen models at the 7B and 14B scales. Scaling yields consistent gains, and the Recency Gap persists even at 14B scale.

Table 6. Prompt template. The question is appended at the end during training and inference.

Prompt Template
Answer the given question. You must conduct reasoning inside <think> and </think> first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by <search> query </search>, and you should return the top searched results between <information> and </information>. You can search as many times as you want. If you find no further external knowledge needed, you can directly provide the answer inside <answer> and </answer> without detailed illustrations. For example, <answer> Beijing </answer>. Question:

	2021				2025			
Method	**L1**	**L2**	**L3**	**Avg**	**L1**	**L2**	**L3**	**Avg**
Qwen2.5-7B								
Direct Answer	24.2	10.3	14.0	16.2	7.4	5.2	6.6	6.4
CoT	20.6	10.0	14.2	14.9	5.4	3.5	4.9	4.6
RAG	65.5	37.2	19.4	40.7	34.8	20.0	13.0	22.6
Search-o1	25.6	12.6	22.0	20.1	5.6	7.9	11.2	8.2
Search-R1-Base	68.8	60.8	25.0	51.5	28.5	31.2	17.1	25.6
Search-R1-Inst	69.8	56.0	21.1	49.0	27.6	30.0	18.6	25.4
Qwen2.5-14B								
Direct Answer	30.6	14.2	18.0	20.9	8.5	5.5	8.6	7.5
CoT	31.5	13.9	20.1	21.8	7.4	4.5	10.0	7.3
RAG	73.0	52.5	33.0	52.8	34.8	27.0	19.0	26.9
Search-o1	27.6	18.2	18.6	21.5	6.7	7.9	13.0	9.2
Search-R1-Base	72.5	61.0	31.3	54.9	29.6	38.0	19.2	28.9
Search-R1-Inst	63.5	61.2	25.0	49.9	28.0	34.0	22.8	28.3

Table 7. **Larger Qwen models (7B/14B) on LiveSearchBench.**

G. Model Family Comparison (Llama vs. Qwen)

A clear divergence emerges between the Llama and Qwen families across six benchmarks. On single-hop datasets (NQ, TQ, HQ), both families benefit from increased sampling, but Llama models enter the positive regime of Δ_k earlier and with steeper gains; Qwen retains larger blue regions at small/medium k, indicating a slower shift from retrieval reliance to parametric dominance. The difference is more pronounced on multi-hop datasets such as 2Wiki and Bamboogle: Llama shows deep red saturation across most of the k range, while Qwen improves with k but with smaller margins and a less abrupt transition. MuSiQue shows a slower transition overall, yet the pattern holds. Scaling within each family reinforces this trend: larger Llama models show sharper improvements in Δ_k than their Qwen counterparts, suggesting that static QA benchmarks disproportionately reward Llama's parametric capacity, whereas Qwen requires larger sampling budgets to approach similar performance.

H. Pipeline Filter-out Statistics

Table 8 reports the survival rate at each stage of the pipeline for the 2025 batch. From 823K raw delta triples, 7,362 ver-

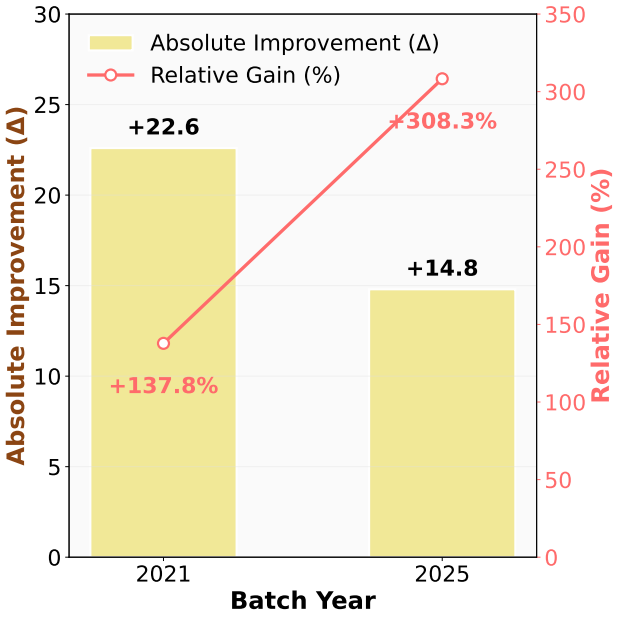

Figure 5. Absolute (Δ) and relative (%) improvements of retrieval-based methods over Direct Answer, averaged across models, on the 2021 vs. 2025 batches.

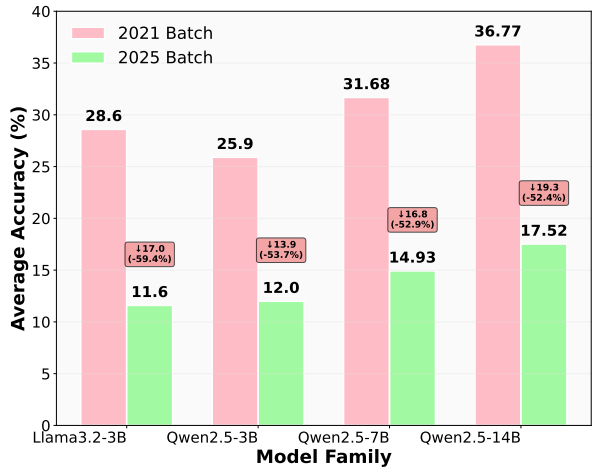

Figure 6. **Family-level comparison.** Averages for different models on 2021 and 2025 batches.

ified candidates survive (∼0.89%), with SPARQL uniqueness validation as the main bottleneck. The verified pool is 7× larger than the final benchmark of 1,000 questions, demonstrating practical scalability.

I. Additional Ablations

Trust-Retrieval Prompt. We test whether explicitly instructing the model to "trust retrieved evidence" improves RAG performance. The improvement is marginal (+0.8% avg), ruling out prompt design as the cause of the Recency Gap.

Pipeline Stage	Count	Survival
Raw delta triples	823,461	100%
Relation allow-list	142,538	17.3%
Entity quality	76,819	53.9%†
Statement validity	45,247	58.9%†
SPARQL-validated	∼10,619	23.5%†
Verified pool	7,362	69.3%†
Final benchmark	1,000	—

Table 8. **Pipeline filter-out statistics** (2025 batch). †Survival rate relative to the preceding stage. Overall: ∼0.89%.

Prompt Variant	L1	L2	L3	Avg
Standard RAG	35.4	26.3	17.5	26.4
"Trust retrieval"	37.0	27.0	17.5	27.2

EM vs. F1. We additionally report token-level F1 alongside EM. F1 is consistently 8–10 points higher, but the **relative ranking of methods is preserved**, confirming conclusions are not driven by metric choice.

	Qwen-14B		Qwen-7B		Llama-3B	
Method	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
Direct Answer	7.0	15.8	6.1	14.3	3.3	11.5
RAG	26.4	35.2	22.4	31.6	13.2	22.0
Search-R1	28.4	37.1	25.0	33.8	18.8	27.5