Evaluating Reasoning over Novel Facts Beyond Parametric Memory

Anonymous ACL submission

Abstract

Static benchmarks often conflate memorization with reasoning, failing to capture the dynamic nature of world knowledge. We present LIVESEARCHBENCH, an automated pipeline constructing retrieval-dependent benchmarks from knowledge graph differentials. Unlike prior dynamic evaluations focused on simple fact updates, our method synthesizes complex, multi-constraint questions guaranteed to have unique answers via strict SPARQL validation. Experiments reveal a pronounced "Recency Gap": models struggle significantly with facts post-dating their pretraining, particularly on multi-hop queries. While retrieval-augmented generation (RAG) offers partial gains, it fails to close this gap, limited by distinct failures in both indexing novel entities and reasoning over evidence. LIVESEARCHBENCH thus shifts reasoning evaluation from static memorization toward rigorous, real-time 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., 2025b; 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 pretraining 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 (e.g., "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 Sec3 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 1 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 (L2/L3) 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

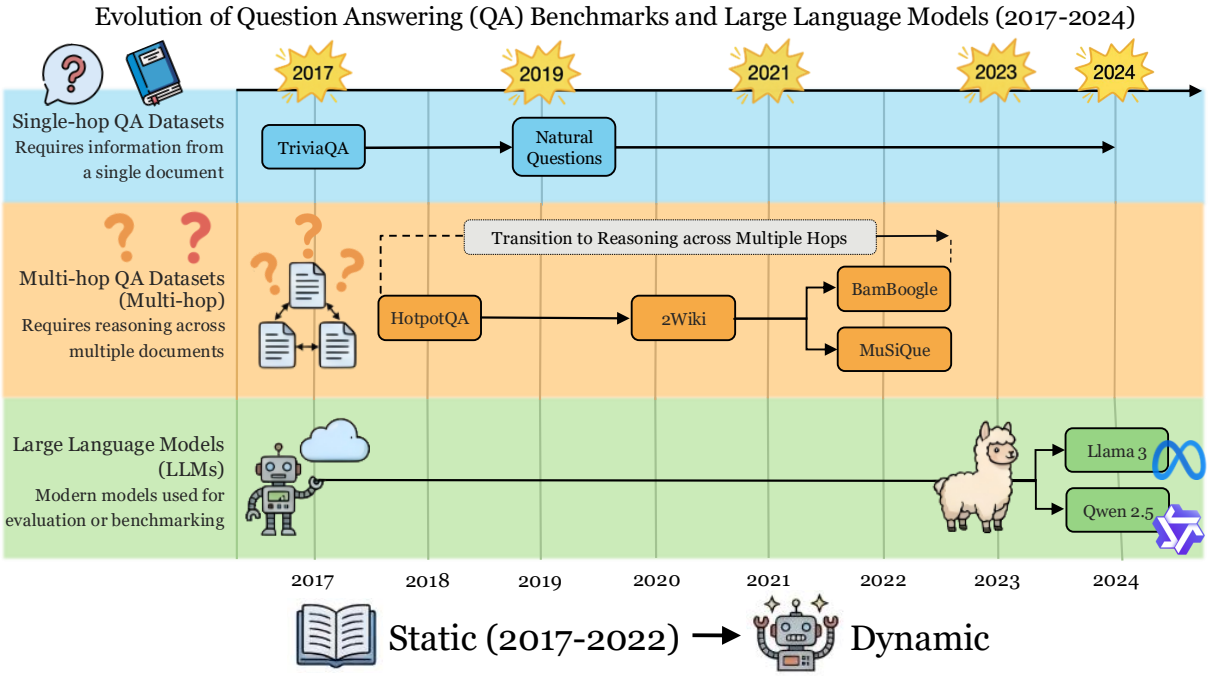


Figure 1: A timeline of major QA benchmarks and model releases. The figure illustrates the historical reliance on static benchmarks, motivating the need for dynamic evaluation resources.

recall.

Our evaluation yields three key insights. First, by systematically testing LLMs and retrieval-augmented generation (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 (intersection, 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., 2025b; 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 snapshots fail to capture evolving knowledge.

Dynamic and Continual Benchmarks. Recent work addresses this via dynamic evaluation on

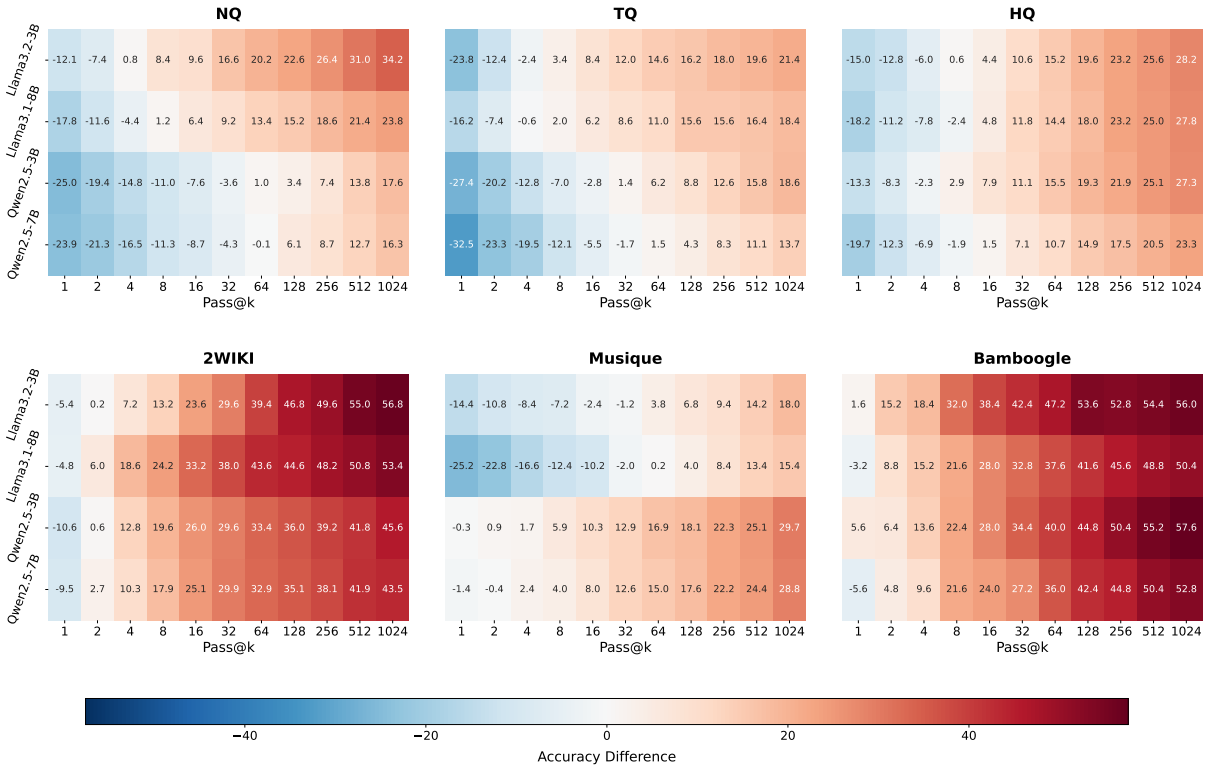


Figure 2: Accuracy difference Δ_k = Pass@$k_{\text{no-search}}$ − Pass@1_{search} across six QA benchmarks. **Red regions** indicate that scaling parametric sampling (without search) outperforms standard retrieval-augmented generation. **Blue regions** indicate where retrieval is essential. The dominance of red areas in static benchmarks suggests that correct answers are often accessed via memorization rather than necessary retrieval.

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 relies on fixed relation templates, limiting diversity to anticipated schemas. Our method employs data-driven topology synthesis. We automatically identify compositional structures (e.g., intersections) directly from knowledge deltas, enabling open-ended reasoning beyond rigid templates. Furthermore, we enforce strict answer uniqueness via SPARQL validation, ensuring mathematical rigor absent in heuristic template filling.

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 (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 (greedy decoding). We define the performance gap as Δ_k = Pass@$k_{\text{no-search}}$ − Pass@1_{search}. A positive Δ_k (Red in Figure 2) implies that internal memory scaling is superior to retrieval, while a negative Δ_k (Blue) indicates that retrieval provides essential information absent from parameters.

Benchmark-Level Patterns. Figure 2 reveals a systematic trend across widely used datasets. On single-hop benchmarks (NQ, 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 (HotpotQA, 2Wiki, 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 (the "distractor effect"), while parametric inference remains robust.

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.

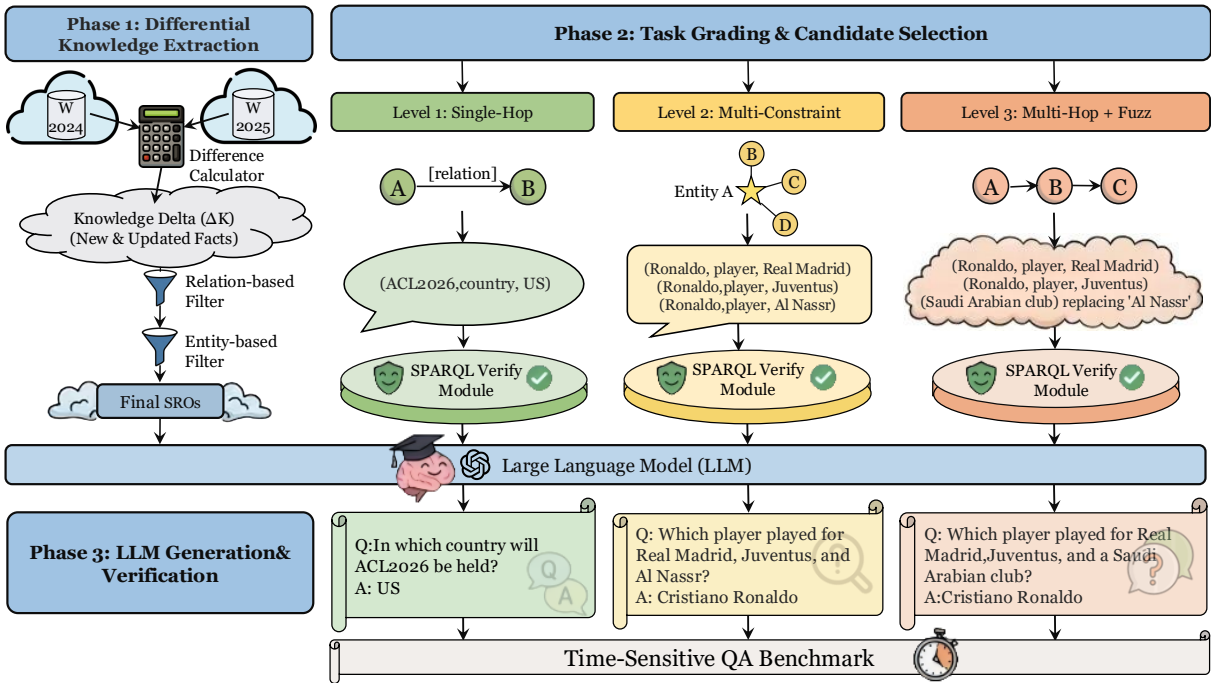


Figure 3: Overview of the generation pipeline. We compute a *knowledge delta* between two Wikidata snapshots to obtain new or updated subject–relation–object (SRO) triples. After relation and entity based filtering, candidate triples are used to synthesize questions at three difficulty tiers: (L1) single-hop,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

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 (e.g., occupation, country, affiliation), checking after each addition whether uniqueness is achieved; we stop when `COUNT=1`. L3 (multi-hop with fuzz): extend L2 by (a) relaxing an attribute to a broader type/hypernym ("fuzzing") and (b) 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 (serving 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.

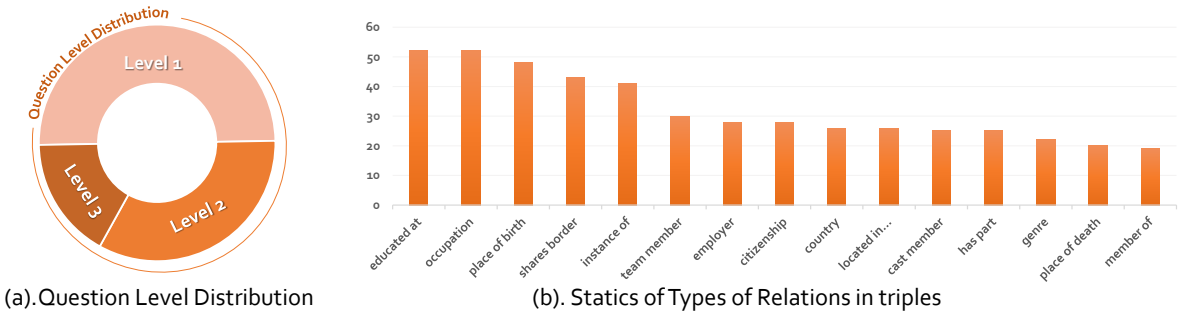
4.3 Question Complexity Levels

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

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

For example, if the knowledge delta introduces the triple (`ACL2026`, *country*, `America`), the corresponding L1 question is: *"In which country will*



(a).Question Level Distribution (b). Statics of Types of Relations in triples

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.

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 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 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. 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. While the pipeline can generate much larger datasets, we opted for a cost-efficient representative subset: 150 L1, 100 L2, and 50 L3 questions. This stratified sample balances reasoning diversity with evaluation efficiency and suffices for robust comparative analysis. 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. 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 (seen knowledge). 2025 Batch: Derived from updates between May and Aug. 2025. Facts post-date training cutoffs (novel knowledge).

Baseline Methods. We group baselines into three categories. Vanilla Prompt Methods include Direct

	LiveSearchBench2021				LiveSearchBench2025			
Model & Method	**L1**	**L2**	**L3**	**Avg.**	**L1**	**L2**	**L3**	**Avg.**
Llama3.2-3B-Instruct								
Direct Answer	22.0	9.0	20.0	17.0	4.0	4.0	2.0	3.3
CoT	23.3	8.0	14.0	15.1	6.0	4.0	0.0	3.3
RAG	42.0	26.0	12.0	26.7	18.7	13.0	8.0	13.2
Search-o1	16.0	17.0	16.0	16.3	6.0	7.0	10.0	7.7
Search-R1-Base	70.7	46.0	18.0	44.9	25.3	21.0	10.0	18.8
Search-R1-Instruct	48.7	43.0	14.0	35.2	18.0	25.0	6.0	16.3
SSRL	66.0	46.0	24.0	45.3	23.3	22.0	12.0	18.4
Qwen2.5-3B-Instruct								
Direct Answer	14.0	9.0	12.0	11.7	3.3	3.0	2.0	2.8
CoT	16.0	8.0	10.0	11.3	3.3	3.0	2.0	2.8
RAG	64.0	31.0	14.0	36.3	23.3	17.0	8.0	16.1
Search-o1	23.0	13.0	10.0	15.3	11.0	3.0	6.0	6.7
Search-R1-Base	56.0	44.0	14.0	38.0	24.0	27.0	12.0	21.0
Search-R1-Instruct	49.3	40.0	10.0	33.1	24.0	24.0	14.0	20.7
SSRL	60.0	37.0	10.0	35.7	20.0	18.0	4.0	14.0
Average across models	43.9	27.6	14.9	28.8	15.6	13.5	7.4	12.2

Table 1: **Exact match accuracy (%) of small models on LiveSearchBench (2021 vs. 2025).**

Prompt and Chain-of-Thought (CoT) prompting. RAG-based Methods comprise standard RAG and SEARCH-O1 (Li et al., 2025c). RL-based Methods include SEARCH-R1 (Jin et al., 2025a) 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 Table 1 and Table 2.

Retrieval vs. Parametric Memory. As shown in Figure 5, 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

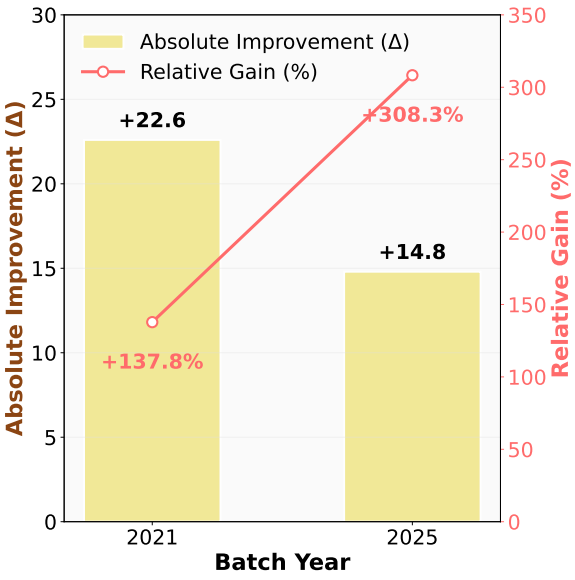


Figure 5: Absolute (Δ) and relative (%) improvements of retrieval-based methods over Direct Answer, averaged across models, on the 2021 vs. 2025 batches.

	LiveSearchBench2021				LiveSearchBench2025			
Model & Method	**L1**	**L2**	**L3**	**Avg.**	**L1**	**L2**	**L3**	**Avg.**
Qwen2.5-7B-Instruct								
Direct Answer	24.0	10.0	14.0	16.0	7.3	5.0	6.0	6.1
CoT	20.7	10.0	14.0	14.9	5.3	3.0	4.0	4.1
RAG (Standard)	66.0	37.0	18.0	40.3	35.3	20.0	12.0	22.4
Search-o1	25.3	12.0	22.0	19.8	4.7	7.0	10.0	7.2
Search-R1-Base	68.7	61.0	24.0	51.2	28.0	31.0	16.0	25.0
Search-R1-Instruct	68.7	55.0	20.0	47.9	27.3	29.0	18.0	24.8
Qwen2.5-14B-Instruct								
Direct Answer	30.7	14.0	18.0	20.9	8.0	5.0	8.0	7.0
CoT	31.3	14.0	20.0	21.8	7.3	4.0	10.0	7.1
RAG (Standard)	73.3	53.0	32.0	52.8	34.7	27.0	18.0	26.6
Search-o1	27.3	18.0	18.0	21.1	6.0	7.0	12.0	8.3
Search-R1-Base	73.3	61.0	30.0	54.8	29.3	38.0	18.0	28.4
Search-R1-Instruct	62.7	61.0	24.0	49.2	28.0	33.0	22.0	27.7
Average across models	49.5	34.6	21.3	35.1	18.8	21.0	11.0	16.9

Table 2: **Accuracy of larger Qwen models (7B/14B) on LiveSearchBench.**

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 (Qwen series) yields consistent gains (Table 2). 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), 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.

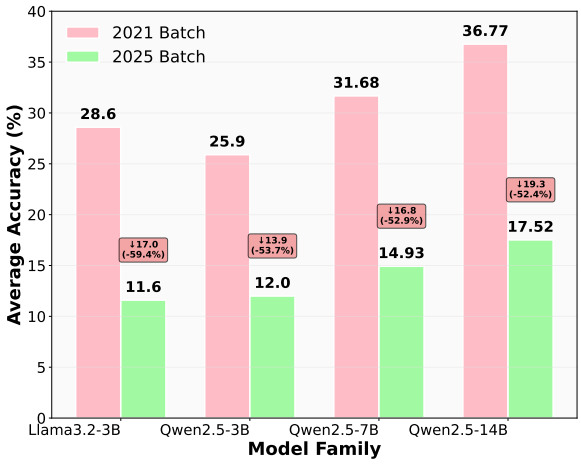


Figure 6: **Family-level comparison.** Averages for different models on 2021 and 2025 batches.

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.

Limitations

First, our reliance on structured Wikidata differentials restricts the benchmark to fact-centric queries, excluding unstructured narratives found in raw news streams. However, this constraint is a deliberate trade-off to ensure the mathematical verifiability and deterministic solvability of every question via SPARQL, which is difficult to guarantee with unstructured text. Second, our use of Exact Match (EM) as the primary metric imposes a high penalty on surface-form variations. While potentially conservative compared to model-based evaluation, we prioritize EM to rigorously distinguish precise retrieval from hallucination, ensuring that the benchmark serves as a trustworthy standard for dynamic reasoning.

Ethics Statement

This work leverages *publicly available* Wikidata snapshots as the sole knowledge source. Wikidata is collaboratively maintained under open licenses, and our pipeline only processes structured triples that are already public. No personal, sensitive, or proprietary data are involved, and all derived benchmark questions are grounded in verifiable facts with explicit provenance. We emphasize that LIVESEARCHBENCH is intended purely for the evaluation of large language models, not for deployment in real-world decision-making scenarios. To the best of our knowledge, this study raises no ethical concerns regarding human subjects, animal welfare, or data misuse.

References

Xiusi Chen, Shanyong Wang, Cheng Qian, Hongru Wang, Peixuan Han, and Heng Ji. 2025. Decisionflow: Advancing large language model as principled decision maker. *Preprint*, arXiv:2505.21397.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. *Preprint*, arXiv:2405.06211.

Yuchen Fan, Kaiyan Zhang, Heng Zhou, Yuxin Zuo, Yanxu Chen, Yu Fu, Xinwei Long, Xuekai Zhu, Che Jiang, Yuchen Zhang, and 1 others. 2025. Ssrl: Self-search reinforcement learning. *arXiv preprint arXiv:2508.10874*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *Preprint*, arXiv:2403.07974.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025a. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025b. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, Kentaro Inui, and 1 others. 2023. Realtime qa: What's the answer right now? *Advances in neural information processing systems*, 36:49025–49043.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025a. Websailor: Navigating super-human reasoning for web agent. *Preprint*, arXiv:2507.02592.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025b. Search-o1: Agentic search-enhanced large reasoning models. *Preprint*, arXiv:2501.05366.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025c. Search-o1: Agentic search-enhanced large reasoning models. *Preprint*, arXiv:2501.05366.

Adam Liska, Tomas Kocisky, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, Cyprien De Masson D'Autume, Tim Scholtes, Manzil Zaheer, Susannah Young, and 1 others. 2022. Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. In *International Conference on Machine Learning*, pages 13604–13622. PMLR.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. 2025. Zerosearch: Incentivize the search capability of llms without searching. *Preprint*, arXiv:2505.04588.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and Thang Luong. 2023. Freshllms: Refreshing large language models with search engine augmentation. *Preprint*, arXiv:2310.03214.

Hongru Wang, Cheng Qian, Manling Li, Jiahao Qiu, Boyang Xue, Mengdi Wang, Heng Ji, and Kam-Fai Wong. 2025. Toward a theory of agents as tool-use decision-makers. *arXiv preprint arXiv:2506.00886*.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. 2025. Livebench: A challenging, contamination-free LLM benchmark. In *The Thirteenth International Conference on Learning Representations*.

Mingqi Wu, Zhihao Zhang, Qiaole Dong, Zhiheng Xi, Jun Zhao, Senjie Jin, Xiaoran Fan, Yuhao Zhou, Huijie Lv, Ming Zhang, Yanwei Fu, Qin Liu, Songyang Zhang, and Qi Zhang. 2025a. Reasoning or memorization? unreliable results of reinforcement learning due to data contamination. *Preprint*, arXiv:2507.10532.

Xiaobao Wu, Liangming Pan, Yuxi Xie, Ruiwen Zhou, Shuai Zhao, Yubo Ma, Mingzhe Du, Rui Mao, Anh Tuan Luu, and William Yang Wang. 2025b. AntiLeakBench: Preventing data contamination by automatically constructing benchmarks with updated real-world knowledge. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18403–18419, Vienna, Austria. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Yue Liao, Hongru Wang, Mengyue Yang, and 6 others. 2025. The landscape of agentic reinforcement learning for llms: A survey. *Preprint*, arXiv:2509.02547.

Heng Zhou, Hejia Geng, Xiangyuan Xue, Li Kang, Yiran Qin, Zhiyong Wang, Zhenfei Yin, and Lei Bai. 2025. Reso: A reward-driven self-organizing llm-based multi-agent system for reasoning tasks. *arXiv preprint arXiv:2503.02390*.

A Use of Large Language Models (LLMs)

We used large language model (ChatGPT 5.2) as an assistive tool in two ways: (1) for writing assistance, including language editing and improving the clarity of the manuscript, and (2) 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 (PhD candidates in NLP and Knowledge Graphs, 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 (1) *Factuality* (Is the triple supported by the snapshot?), (2) *Clarity* (Is Q unambiguous?), and (3) *Solvability* (Does Q uniquely lead 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 600 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 (i.e., the inability to access fresh facts), not data quality. In contrast, for L3 (Fuzzing) questions, accuracy only improved to ∼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 3. 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.

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 4. We use Google Search via Serper API for all all them to ensure fairness.

F Additional Analysis: 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 (2Wiki, 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.

```
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 3: 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

Table 4: 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: