FROM STORAGE TO ACCESS: VERIFIABLE ACTIVATION OF PARAMETRIC KNOWLEDGE IN LLMS VIA EXPLICIT PRIMING AND IMPLICIT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Although Large Language Models (LLMs) encode rich factual knowledge in their parameters, reliably recalling and verifying such knowledge remains a key bottleneck in factual question answering. Existing end-to-end methods entangle knowledge elicitation with reasoning, making it difficult to determine whether correct answers arise from parametric knowledge or the input context. To address this challenge, we propose VAKE (***V**erifiable **A**ctivation of Parametric **K**nowledg**E***), a two-stage reinforcement-learning framework that externalizes latent parametric knowledge through explicit Priming and transfers the acquired elicitation capability to implicit Reasoning. Given a query and an insufficient retrieved subgraph, the **Priming** policy explicitly inserts bridging triples as verifiable evidence, with supervision provided by rewards derived from answers generated by a separate frozen model over the augmented subgraph. Building on the policy learned during Priming, the **Reasoning** stage trains the model to answer from the original input, testing whether the capability acquired through explicit knowledge elicitation transfers to implicit reasoning. Across seven benchmarks and two backbone families, VAKE achieves the best average performance on both the in-distribution multi-hop and OOD benchmark groups, outperforming inference-time methods and GRPO. Complementary attribution and probing analyses on 2Wiki suggest that the inserted triples primarily contribute bridging information not derivable from the retrieved context and can surface knowledge missed by direct prompting. These results support the view that VAKE activates latent parametric knowledge rather than copying the input context or memorizing dataset-specific associations.

1 INTRODUCTION

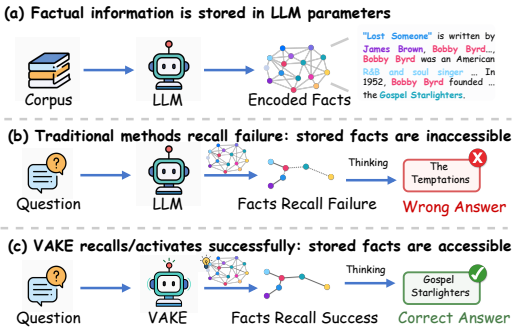

Figure 1: VAKE activates inaccessible parametric knowledge. (a) facts reside in LLM parameters; (b) recall failure yields ungrounded, incorrect answers; (c) Priming surfaces relevant facts before reasoning, supporting correct answers.

To equip large language models (LLMs) with general world knowledge and usable reasoning ability, researchers adopt a standard two-stage training pipeline. Pre-training encodes vast factual knowledge into LLMs' parameters, while post-training methods, such as supervised fine-tuning (SFT) and reinforcement learning (RL), adapt this parametric knowledge to task-specific behaviors and outputs (Xu et al., 2026; Zeng et al., 2026; Yang et al., 2025). Prior literature suggests that frontier models achieve near-saturated encoding of factual knowledge from encyclopedic corpora such as Wikipedia (Calderon et al., 2026; Gekhman et al., 2026). However, a critical bottleneck remains: models may fail to recall factual knowledge within their parameters, an issue defined as "stored but inaccessible" (Zheng et al., 2024a; Wang et al., 2024).

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

Recall-oriented evaluations further expose this limitation: models may fail to answer queries correctly even when the target facts are fully present in the training corpus (Zheng et al., 2024a; Mousavi et al., 2026). Prior analyses suggest that internal parametric representations store richer factual information than what greedy decoding can reliably elicit, yet fact-association recall remains fragile (Wang et al., 2024; Gekhman et al., 2025). Scaling model size and training data can alleviate encoding deficits, but yields limited gains in accessing latent parametric knowledge (Calderon et al., 2026; Mousavi et al., 2026).

This limitation is difficult to diagnose with standard accuracy-based evaluations, which conflate recall failures with encoding failures despite requiring different interventions. Encoding failures motivate scaling or knowledge injection, whereas recall failures call for post-training methods that improve access to knowledge already encoded in model parameters. Existing methods use instruction tuning (See Figure 1) and reinforcement learning to improve knowledge utilization (Yang et al., 2026; Meng et al., 2026). However, their reliance on unstructured generation entangles knowledge elicitation with answer reasoning, making the effect of activation difficult to isolate (Calderon et al., 2026; Mousavi et al., 2026). Crucially, eliciting latent parametric knowledge often requires external query-relevant cues to trigger the model's fact-association recall. To address this, we draw inspiration from the "priming effect" in cognitive science: a phenomenon where exposure to specific cues activates latent memory representations, enhancing subsequent task performance. Transferring this to LLMs, we hypothesize that introducing an independent, cue-conditioned activation phase over an insufficient retrieved subgraph can awaken the model's dormant parametric knowledge.

To this end, we propose **V**erifiable **A**ctivation of Parametric **K**nowledg**e** (VAKE), a two-stage reinforcement-learning framework that separates explicit knowledge elicitation from direct answer reasoning. VAKE consists of **Priming** and **Reasoning**. In Priming, an RL-optimized policy elicits latent parametric knowledge as relational triples and inserts them into a sparse retrieved subgraph. A separate, frozen answerer then evaluates this augmented graph and provides outcome-based rewards. This insert-then-answer intervention ensures that any improvement in the answerer's output is directly attributable to the explicitly injected triples, providing a verifiable reward signal for knowledge activation. In Reasoning, the policy initialized from Priming is optimized via GRPO to answer directly from the original retrieved subgraph without explicitly injected triples. This effectively internalizes the acquired elicitation capability into implicit chain-of-thought (CoT) reasoning. Experiments show that VAKE consistently outperforms non-retrieval activation baselines, standard GRPO, and prior RL-based activation methods across model scales. Models optimized on a single multi-hop dataset transfer effectively to diverse out-of-distribution (OOD) multi-hop and single-hop datasets. Applying GRPO after Priming yields further gains, demonstrating the complementarity of knowledge activation and reasoning optimization. Therefore, VAKE can be seamlessly integrated into existing post-training pipelines as an RL-based complement and SFT-based knowledge activation.

In summary, our main contributions are as follows:

- **A two-stage framework that separates knowledge activation from direct answer reasoning.** Priming explicitly elicits latent parametric knowledge as relational triples, while Reasoning transfers the learned elicitation capability to direct question answering.
- **An observable and attributable representation of activated knowledge.** Relational triples provide discrete and inspectable evidence for evaluating the content and effect of knowledge activation.
- **Compatibility with standard post-training paradigms.** Empirically validated by our ablation results, VAKE serves as an RL-based complement to standard knowledge activation and seamlessly integrates into standard post-training pipelines while retaining full compatibility with reasoning optimization.

2 Related Work

2.1 Access, Not Storage

Factual QA is increasingly framed as an *access* problem rather than a *storage* one, because a model's greedy output only lower-bounds what it encodes. Three lines of evidence support this framing. Ex-

ternal behavior shows that direct prompting underestimates recall (Zheng et al., 2024a; Wang et al., 2024), that models answer inconsistently across paraphrases (Jiang et al., 2020; Elazar et al., 2021), and that fine-tuning on new facts can even hurt existing recall (Gekhman et al., 2024). Internal probing shows that hidden states carry truthfulness signals absent from the decoded answer (Burns et al., 2022; Azaria & Mitchell, 2023; Li et al., 2023; Orgad et al., 2025) and that internal knowledge measurably exceeds what the model explicitly generates (Gekhman et al., 2025). Controlled provenance further shows that an injected corpus can make a fact provably parametric yet still leave the model unable to surface it (Mousavi et al., 2026; Ovadia et al., 2025; Liu et al., 2025). We therefore study multi-hop QA instances limited by access to encoded knowledge, and seek an activation procedure that improves such access.

2.2 Eliciting Activation and Its Confounds

Existing activation methods fall into two families, and both leave the same measurement confound in which the effect of activation cannot be cleanly separated from reasoning or retrieval. **Inference-time** methods produce intermediate content at query time through prompt search (Shin et al., 2020; Zhong et al., 2021), *RECITE* (Sun et al., 2023), *Self-Ask* (Press et al., 2023) and *Step-Back* (Zheng et al., 2024b), all built on chain-of-thought and generated-knowledge prompting (Wei et al., 2022; Liu et al., 2022; Wang et al., 2023). **Training-time** methods use reinforcement learning, where a correctness reward reweights knowledge the model holds rather than new facts (Yang et al., 2026), and token-level and distributional analyses take the same view that RLVR sharpens the base distribution rather than expanding it (Meng et al., 2026; Yue et al., 2025; Wen et al., 2026). Related RL variants either route access through a reasoning chain (Gekhman et al., 2026; Ma & Hewitt, 2026; Chen et al., 2025) or combine access with side objectives such as factuality (Ren et al., 2026; Li & Ng, 2025; Wei et al., 2025). Because both families operate over free-form text and provide no controlled context, activation can only be inferred from output changes rather than directly attributed.

We take a route that removes this confound. We provide the model with an incomplete retrieved subgraph that offers relevant cues without a sufficient answer path, and keep the answerer frozen during Priming, so the inserted bridging triples are the only free variable and their answer-enabling effect is observable and attributable. We treat the subgraph as a medium for query-conditioned augmentation rather than an external structure to traverse, and learn the Priming policy with reinforcement learning.

3 Method

3.1 Problem Formulation

Let M_0 be a frozen model whose parameters encode knowledge that may not be directly accessible for a given query. For a question q with ground-truth answer $a^\star$ and an insufficient context c, our goal is to elicit the required knowledge from model parameters and express it as an explicit, query-conditioned increment $\mathcal{I}$. A parameterized policy π_θ generates this increment as

$$\mathcal{I} \sim \pi_\theta(\cdot \mid q, c). \tag{1}$$

We define successful activation by the following model-relative intervention:

$$M_0(q, c) \neq a^\star, \qquad M_0(q, c \cup \mathcal{I}) = a^\star. \tag{2}$$

Here, the knowledge is *parametric* in source but becomes *explicit* in $\mathcal{I}$. Unlike retrieval augmentation, $\mathcal{I}$ is generated by the policy rather than retrieved from an external source. Its attribution and answer-enabling effect are evaluated through the controlled intervention above.

We implement this activation via two stages: **Priming** and **Reasoning**. Drawing on the cognitive notion of priming, where cues facilitate access to latent information, **Priming** trains the policy to elicit implicit parametric knowledge. We externalize the elicited knowledge as answer-enabling bridging triples $\mathcal{I}$, making activation observable and operationally attributable. **Reasoning** then trains the policy to perform CoT reasoning that exploits this elicitation capability and corrects hallucinations, enabling direct question answering without explicit triple insertion.

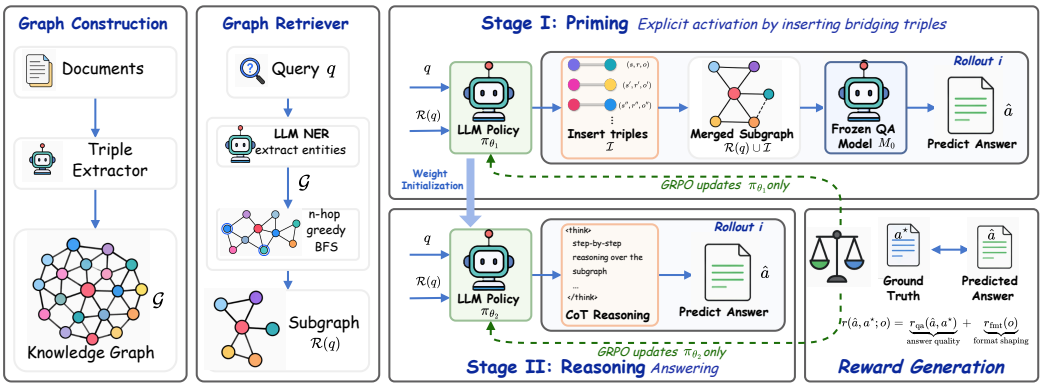

Figure 2: Overview of our proposed VAKE. Given a sparse query-relevant subgraph $\mathcal{R}(q)$, VAKE first performs explicit priming by learning to insert verifiable bridging triples $\mathcal{I}$ for a frozen answerer M_0 (Stage I). It then transfers the learned knowledge-elicitation capability to implicit CoT reasoning through GRPO in the Reasoning stage (Stage II).

3.2 Task Definition

We instantiate the parametric-knowledge activation task defined above on a triple graph $\mathcal{G} = \{(s, r, o)\}$ parsed offline from the associated documents. In the main protocol, raw documents are used only for graph construction and are not exposed during training or standard inference; the model observes only triples. A retriever returns a query-relevant subgraph $\mathcal{R}(q) \subseteq \mathcal{G}$, which serves as the context $c = \mathcal{R}(q)$. A few diagnostic experiments explicitly add raw documents to Priming, but the answerer always receives only triples. The retriever is designed to be shallow and sparse, providing query-relevant cues that can condition the elicitation of parametric knowledge rather than a complete supporting path. The concrete retrieval configuration is described in Section 4.1.

3.3 Two-Stage Activation: Priming and Reasoning

The two stages operate on the same input $(q, \mathcal{R}(q))$ and optimize the same answer-quality objective with respect to $a^\star$, but differ in how the answer is produced. In the **Priming** stage, the policy inserts explicit bridging triples as intermediate evidence for a frozen answerer. In the **Reasoning** stage, the Priming-initialized policy generates the answer directly from the original input, transferring the learned knowledge-elicitation capability to direct reasoning. The two stages are optimized sequentially. Figure 2 gives an overview of this two-stage activation process.

3.3.1 Stage I: Priming (Insert-then-Answer)

In this stage, the policy π_θ acts as a subgraph-augmentation policy. Given the query and the retrieved subgraph, it externalizes bridging facts as a set of inserted triples

$$\mathcal{I} \sim \pi_\theta(\cdot \mid q, \mathcal{R}(q)), \tag{3}$$

after which a frozen answerer M_0 produces

$$\hat{a} = M_0\big(q, \mathcal{R}(q) \cup \mathcal{I}\big). \tag{4}$$

Insert-then-answer makes knowledge activation observable, attributable, and rewardable. Holding M_0 and $\mathcal{R}(q)$ fixed leaves $\mathcal{I}$ as the sole intervention, so a wrong-to-correct transition after insertion isolates its downstream effect. Comparing inserted triples with $\mathcal{R}(q)$ supports attribution: triples absent and non-derivable from the subgraph are operationally attributed to parametric knowledge rather than copied context. Priming therefore needs no SFT labels for insertions; its reward depends only on whether the frozen answerer uses them to produce $a^\star$.

3.3.2 Stage II: Reasoning

The insert-then-answer decomposition in stage I is an instrument for making activation attributable, rather than the intended final usage mode of the model. After Priming, we return the policy to its

native question-answering role. Initialized from the stage I policy,

$$\theta \leftarrow \theta^{(1)}, \tag{5}$$

the model now answers directly,

$$\hat{a} \sim \pi_\theta(\cdot \mid q, \mathcal{R}(q)), \tag{6}$$

without an inserted set or a frozen answerer, and is optimized by GRPO with the same input setting and reward design as Priming, but with a stage-specific format-shaping term. This stage tests whether the explicit activation induced by Priming can be carried into chain-of-thought answering.

This second stage also tests whether gains from Priming are complementary to reasoning gains. If Priming merely reproduces the effect of standard chain-of-thought optimization, continuing with reasoning-oriented GRPO should yield limited additional improvement. Additional gains instead indicate complementarity under sequential optimization. We therefore report Priming followed by Reasoning as a core result rather than as a mere ablation, since it examines whether Priming can be incorporated before reasoning-oriented GRPO in existing post-training pipelines.

3.4 Reinforcement Learning Objective

Both stages in Section 3 are trained by reinforcement learning and optimized with the same GRPO algorithm. They share an answer-quality reward against the gold answer $a^\star$, but differ in how $\hat{a}$ is produced and which output format is required. We keep the reward outcome-driven but shape it into a small number of interpretable components rather than a single hard $0/1$ signal. For stage s, the overall reward is:

$$r^{(s)}(\hat{a}, a^\star; o) \;=\; r_{\mathrm{qa}}(\hat{a}, a^\star) \;+\; r^{(s)}_{\mathrm{fmt}}(o). \tag{7}$$

The answer-quality term r_{qa} is a soft blend of exact match (EM) and token-level F1 between $\hat{a}$ and $a^\star$, weighted so that exact correctness dominates while the F1 component densifies the reward landscape and mitigates the sparsity of a pure $0/1$ signal on hard multi-hop questions. The stage-specific format-shaping term $r^{(s)}_{\mathrm{fmt}}$ depends only on whether the trajectory o is well formed: it grants a small bonus when the output follows the required triple or reasoning format, and imposes a penalty when it is malformed, with severe malformation overriding the quality term so that unparseable trajectories cannot be rewarded.

The two stages instantiate $\hat{a}$ differently. In **Priming**, the trainable policy emits an inserted triple set and the answer is produced by the frozen answerer,

$$\hat{a} = M_0\big(q, \mathcal{R}(q) \cup \mathcal{I}\big), \tag{8}$$

so the reward reaches the policy only through $\mathcal{I}$. In **Reasoning**, the policy answers directly,

$$\hat{a} \sim \pi_\theta(\cdot \mid q, \mathcal{R}(q)), \tag{9}$$

and the reward scores its own output.

The concrete weights, bonuses, and penalty magnitudes in Eq. (7) are detailed in the Appendix A.

3.4.1 Group-Relative Policy Optimization

We optimize both Priming and Reasoning with GRPO. For each question q, the old policy $\pi_{\theta_{\mathrm{old}}}$ samples a group of G trajectories $\{o_i\}_{i=1}^{G}$. In Priming, each trajectory specifies inserted bridging triples, and the frozen answerer M_0 predicts $\hat{a}_i$ from the subgraph after triple insertion. In Reasoning, the trajectory directly contains the reasoning trace and final answer. Each trajectory receives the stage-specific reward $r_i = r^{(s)}(\hat{a}_i, a^\star; o_i)$.

Following GRPO, we compute the group-relative advantage by normalizing rewards within the sampled group: Let $\rho_{i,t}(\theta)$ be the token-level probability ratio between π_θ and $\pi_{\theta_{\mathrm{old}}}$, then define

$$\hat{A}_i = \frac{r_i - \mathrm{mean}(\{r_j\}_{j=1}^{G})}{\mathrm{std}(\{r_j\}_{j=1}^{G}) + \delta}, \qquad c_{i,t}(\theta) = \min\Big\{\rho_{i,t}(\theta)\hat{A}_i,\ \mathrm{clip}_\varepsilon(\rho_{i,t}(\theta))\,\hat{A}_i\Big\}. \tag{10}$$

where δ is a small constant for numerical stability. The same sequence-level advantage is assigned to all tokens in o_i.

The GRPO objective is

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(c_{i,t}(\theta) - \beta D_{i,t}^{\mathrm{KL}}\right)\right], \tag{11}$$

where ε controls the clipping range, β weights the KL penalty, and $D_{i,t}^{\mathrm{KL}}$ denotes the token-level KL penalty to the stage-specific reference policy.

Gradients are propagated only through π_θ. During Priming, M_0 is fixed and non-differentiable, contributing only through the reward induced by its generated answer. Thus, optimization learns to insert more useful bridging triples without adapting the answerer to compensate for weak insertions. During Reasoning, no separate answerer is used, and the policy is rewarded directly by the quality of its generated answer.

4 Experiments

4.1 Experimental Setup

Datasets and backbones. We train and test on three in-distribution multi-hop datasets: 2WikiMultihopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018), and MuSiQue (Trivedi et al., 2022). For OOD evaluation, we use Bamboogle (Press et al., 2023), NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023), all held out from training. OOD results use the HotpotQA-trained checkpoint. We use Qwen2.5-7B-Instruct (Qwen et al., 2025) and Qwen3-8B (Yang et al., 2025). **VAKE-P** is the Priming-only checkpoint; **VAKE** additionally undergoes Reasoning training. VAKE supports direct and insert-then-answer inference, whereas VAKE-P uses a frozen untrained answerer after insertion.

Baselines. All baselines receive the same question and retrieved subgraph, without extra retrieval: direct *Base* and *Base with Reasoning*; inference-time *Self-Ask* (Press et al., 2023) and *RECITE* (Sun et al., 2023); and RL-based *Unlock* (Yang et al., 2026) and *GRPO*. Unlock uses GRPO without insertion and a 72B judge; GRPO matches our RL budget on Qwen2.5-7B.

Graph construction and retrieval. An LLM-based OpenIE pipeline builds a graph from passage entities and triples. We string-match question mentions to graph nodes, then select question-similar edges from a two-hop BFS neighborhood (five per hop). After removing triples containing the gold answer, we retain at most $K = 10$ triples as $\mathcal{R}(q)$, making the retrieved subgraph insufficient by construction. See Appendix F for details.

Implementation and evaluation. For each in-distribution dataset, we train VAKE for one epoch on approximately 14K examples with a batch size of 28 and a learning rate of 1×10^{-6}. Each step samples eight priming trajectories, and all experiments use eight NVIDIA H20 GPUs. Unless otherwise stated, *judge-EM* uses GPT-4o at temperature 0 to assess semantic equivalence from the question, prediction, reference, and aliases, without retrieved or inserted triples; the prompt and rubric are provided in Appendix C. We audited 500 stratified cases across datasets, backbones, and input regimes with two independent annotators; after adjudication, GPT-4o agreed with the human labels with a Cohen's κ of 0.86.

4.2 Main Results

We compare VAKE with inference-time and training-time baselines on in-distribution and OOD QA benchmarks in Table 1, using matched test-time pipelines where applicable.

Priming provides consistent gains With Priming alone and no reasoning at inference, VAKE-P attains the best in-distribution average on both backbones, exceeding the strongest non-reasoning baseline by 2.4 and 2.2 points on Qwen2.5-7B and Qwen3-8B. The largest dataset-level margins appear on HotpotQA, where VAKE-P exceeds the strongest non-reasoning baseline by 2.3 and 2.2 points on the two backbones, respectively. Since VAKE-P keeps the answerer frozen and changes

Table 1: **Answer accuracy (%) on ID and OOD QA benchmarks.** † and ⋆ indicate multi-hop and single-hop QA datasets, respectively. The badges denote the test-time pipeline: ♠ = bridging-triple insertion and ♢ = CoT reasoning. For each backbone group, **best** and second-best results are highlighted.

	In-Distribution Multi-Hop QA				Out-of-Distribution QA				
Method	**2Wk†**	**Hotp†**	**MuS†**	**Avg**	**Bam†**	**NQ⋆**	**Triv⋆**	**Pop⋆**	**Avg**
Qwen2.5-7B-Instruct									
Base	27.2	32.7	13.4	24.4	12.0	28.7	51.6	18.0	27.6
Base ♢	25.2	33.2	13.6	24.0	21.6	28.3	51.2	17.8	29.7
Self-Ask	29.8	38.6	18.4	28.9	**41.6**	30.6	55.4	18.0	36.4
RECITE	30.7	36.3	17.6	28.2	40.8	**34.8**	57.4	19.8	38.2
Unlock	29.4	35.3	16.6	27.1	12.8	28.5	52.0	18.3	27.9
GRPO	32.5	35.4	16.2	28.0	12.8	28.7	51.8	18.2	27.9
GRPO ♢	33.5	31.9	15.8	27.1	30.4	28.0	52.5	18.9	32.5
VAKE-P ♠	33.1	40.9	20.0	31.3	32.0	30.8	54.0	18.8	33.9
VAKE ♠♢	**36.2**	42.3	**21.0**	**33.2**	34.4	31.1	56.2	20.1	35.5
VAKE ♢	35.5	**43.4**	20.4	33.1	**41.6**	33.2	**59.5**	**21.2**	**38.9**
Qwen3-8B									
Base	28.2	36.2	14.8	26.4	14.4	30.8	50.9	19.6	28.9
Base ♢	26.5	33.5	18.4	26.1	40.8	33.8	60.1	22.3	39.3
Self-Ask	34.8	38.5	19.8	31.0	32.8	27.9	53.0	18.5	33.1
RECITE	33.0	40.1	19.6	30.9	33.6	35.3	58.5	21.5	37.2
Unlock	31.4	38.5	21.6	30.5	15.2	32.0	51.9	20.3	29.9
GRPO	34.2	36.8	20.2	30.4	16.8	35.7	61.4	**22.5**	34.1
GRPO ♢	36.2	41.8	19.0	32.3	40.8	34.1	61.2	22.4	39.6
VAKE-P ♠	35.0	42.3	22.4	33.2	32.0	31.3	59.7	21.8	36.2
VAKE ♠♢	37.1	43.7	**23.8**	**34.9**	36.0	34.4	61.0	22.2	38.4
VAKE ♢	**37.9**	**44.9**	20.6	34.5	**43.2**	**37.4**	**62.3**	**22.5**	**41.4**

only its input, the gains reflect the inserted bridging triples, not any adaptation of the answerer. Whether those triples supply retrievable facts or knowledge held in the model's parameters is the question we take up in Section 4.4.

Priming and Reasoning provide complementary gains. Under bridging-triple insertion with CoT at inference, VAKE outperforms VAKE-P across all in-distribution datasets and both backbones, raising the average by 1.9 points on Qwen2.5-7B and 1.7 on Qwen3-8B. This gain isolates the contribution of the Reasoning stage under a fixed test-time pipeline. In the contrasting CoT-only setting, with no triples inserted, VAKE surpasses GRPO, the strongest reasoning baseline, by 6.0 and 2.2 points on the two backbones. Because this gain arises without any inserted triples, it reflects reasoning improved by Priming rather than extra evidence supplied at inference. We test this complementarity more directly through controlled ablations in Section 4.5.

VAKE generalizes across OOD tasks. VAKE achieves the highest OOD average on both backbones, scoring 38.9 on Qwen2.5-7B and 41.4 on Qwen3-8B, and outperforms the strongest inference-time and RL baselines. On Qwen3-8B, it exceeds the best competing method by 1.8 points on average and ranks first on all four datasets. Its largest margin is 2.4 points on Bamboogle, suggesting that the learned activation capability generalizes to held-out tasks, including multi-hop reasoning.

4.3 Robustness: Scaling and General Capabilities

This section tests stability across backbone sizes and broader model capabilities. For the scaling and input-condition analyses, we use VAKE-P with bridging-triple insertion. In document-augmented settings, raw documents are available during triple generation, while the frozen base answerer receives only the augmented subgraph.

Table 2: **General capability evaluation on standard benchmarks.** All scores are reported as percentages using each benchmark's standard evaluation metric. The **best** and second-best results in each column are highlighted.

Method	GSM8K	IFEval	AIME24	AIME25	MMLU	AVG
Base	92.04	75.99	33.33	23.33	70.84	59.11
Unlock	91.28	75.20	30.00	23.33	70.84	58.13
GRPO	**92.42**	76.52	23.33	**30.00**	70.62	58.58
VAKE-P	**92.42**	75.64	**36.67**	20.00	**70.99**	59.14
VAKE	91.74	**76.82**	30.00	26.67	70.87	**59.22**

Priming gains persist across backbone families and sizes. Figure 3 reports judge-EM for bridging-triple insertion on 2Wiki across six Qwen backbones, with triples-only or document-augmented inputs. In the latter setting, raw documents are available to VAKE-P only during triple generation; after insertion, the frozen base answerer predicts from the subgraph alone. VAKE-P outperforms Base across model scales in nearly all settings. The sole exception is Qwen3-0.6B, where limited capacity and longer inputs may impair triple-generation quality. Document augmentation further improves VAKE-P in several settings, suggesting that the added context helps generate more effective bridging triples without exposing documents to the answerer.

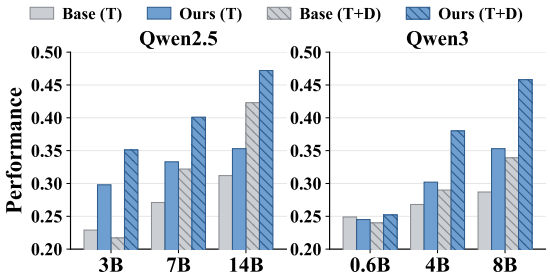

Figure 3: Judge-EM of VAKE-P with bridging-triple insertion on 2Wiki across six Qwen backbones, using triples only (T) or triples with raw documents (T+D).

VAKE preserves general capabilities. Table 2 spans three categories: math, instruction following, and general knowledge. We evaluate on GSM8K (Cobbe et al., 2021), AIME24/25 (Zhang & Math-AI, 2024; 2025), IFEval (Zhou et al., 2023), and MMLU (Hendrycks et al., 2020), reporting pass@10 on AIME24/25 and accuracy on the rest. VAKE-P and VAKE stay on par with Base across all three categories, with near-identical average scores, showing that the QA gains come at no cost to the evaluated general capabilities.

4.4 Attribution of the Activated Knowledge

Table 3: Source attribution of VAKE-P-inserted triples across Qwen2.5/Qwen3 scales. Each cell reports the percentage of inserted triples attributed to model parameters or retrieved context.

	Qwen2.5			Qwen3		
Source	**3B**	**7B**	**14B**	**0.6B**	**4B**	**8B**
Params	84.2	90.8	92.3	95.0	95.8	80.3
Subgraph	15.8	9.2	7.7	5.0	4.2	19.7

We next ask whether the inserted triples restate the retrieved subgraph or expose parametric knowledge.All analyses in this section are conducted on 2Wiki.

Inserted triples mostly originate from parametric knowledge. At each backbone scale, GPT-4o classifies triples inserted by its VAKE-P checkpoint as *subgraph-inferable* if derivable from the retrieved subgraph and *parameter-originated* otherwise; the prompt is provided in Appendix G.1. To validate these labels, two annotators independently audited 300 triples, 50 per scale, with disagreements adjudicated. GPT-4o agreed with the adjudicated human labels with a Cohen's κ of 0.84. As shown in Table 3, over 80% of triples are labeled parameter-originated across scales, while fewer than 20% are attributed to the retrieved subgraph. Thus, most insertions are operationally attributed to parametric knowledge.

Probing reveals a gap between knowledge encoding and access. For the factually correct, parameter-originated triples inserted by VAKE-P (7B), we mask the head (S), relation (R), or tail (O) of each triple (s, r, o) and measure *reproduction accuracy*. We compare VAKE-P with frozen same-family backbones from 7B to 72B (Figure 4). Frozen backbones achieve high head and relation accuracy, indicating that much of the relevant knowledge is encoded in their parameters. However, their tail accuracy stays below 27.0, whereas VAKE-P reaches 35.2 and outperforms even the 72B backbone. This contrast suggests that Priming improves access to encoded answer-bearing information.

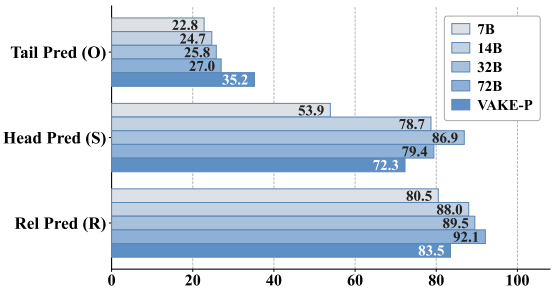

Figure 4: Probing masked-slot reproduction of VAKE-P triples on 2Wiki across VAKE-P(7B) and frozen backbones.

4.5 Ablation Studies

We ablate VAKE on Qwen2.5-7B-Instruct along two axes: its training components and input conditions.

Priming and Reasoning training are complementary. Table 4 ablates VAKE's two training stages. All variants are evaluated without bridging- triple insertion; VAKE and *w/o Priming* use CoT reasoning. VAKE performs best on all datasets, while *w/o Reasoning* surpasses both Base and *w/o Priming* on HotpotQA and MuSiQue, showing that Priming benefits direct answering even without triple insertion.

Table 4: Multi-hop QA ablation without bridging-triple insertion; *w/o Priming* and *w/o Reasoning* omit the corresponding training stage.

Variant	2Wiki	HotpotQA	MuSiQue
VAKE	**35.50**	**43.40**	**20.40**
w/o Priming	<u>33.50</u>	31.90	15.80
w/o Reasoning	28.50	<u>37.20</u>	<u>16.40</u>
Base	27.20	32.70	13.40

The learned activation capability generalizes across input settings. Table 5 compares Base, VAKE-P, and VAKE on 2Wiki with closed-book, triples-only, and document-augmented inputs. All methods use bridging-triple insertion, so differences reflect checkpoint training under a shared inference pipeline. VAKE-P and VAKE outperform Base in every setting, showing that training contributes beyond triple insertion alone. The closed-book gain supports parametric knowledge elicitation, as triples are generated from the question without retrieved context. With document augmentation, VAKE exceeds Base by 15.9 points, showing that Priming also complements external evidence.

Table 5: Information ablation on 2Wiki, with all methods evaluated via bridging-triple insertion.

Method	Closed-book	Triples	Triples+Docs
Base	24.20	27.60	30.70
VAKE-P	30.00	33.10	41.30
VAKE	**30.10**	**36.20**	**46.60**

5 Conclusion

In this paper, we frame factual question answering as accessing, rather than storing, parametric knowledge and introduce VAKE, a two-stage RL framework for explicit knowledge elicitation and direct reasoning. Priming inserts bridging triples into a sparse subgraph using outcome rewards from a frozen answerer, while Reasoning transfers the acquired elicitation capability to direct chain-of-thought answering. Across seven benchmarks and two instruction-tuned backbone families, VAKE achieves the best average performance on both the in-distribution multi-hop and OOD benchmark groups. Further analyses show that Priming gains generally persist across six Qwen sizes and input settings and complement reasoning optimization, while attribution and probing suggest that the inserted triples surface answer-relevant bridging information unavailable from the retrieved context. Overall, VAKE provides an explicit and verifiable RL-based complement to QA-pair construction and SFT-based knowledge activation.

AI USE STATEMENT

In this work, generative AI tools were used solely to assist with language editing, including improving grammar, clarity, and readability. We did not use generative AI tools for other research tasks requiring disclosure, and any disclosure tasks not relevant to this work are not applicable. The authors reviewed all AI-assisted edits and incorporated them only when they accurately reflected the intended meaning and content of the manuscript. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies parametric knowledge elicitation using existing question-answering benchmarks and language models. It does not involve recruiting or interacting with human participants, or collecting new personal or sensitive data. We use benchmark datasets and pretrained models for research purposes under their applicable terms of use. Because VAKE may surface information absent from the retrieved context, its outputs can include inaccurate, outdated, or biased claims and should be independently verified before use in consequential settings. Our evaluation also uses an LLM judge, whose decisions may reflect model-specific biases and are treated as approximate assessments rather than ground truth.

REFERENCES

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 967–976, 2023.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

Nitay Calderon, Eyal Ben-David, Zorik Gekhman, Eran Ofek, and Gal Yona. Empty shelves or lost keys? recall is the bottleneck for parametric factuality. *arXiv preprint arXiv:2602.14080*, 2026.

Xilun Chen, Ilia Kulikov, Vincent-Pierre Berges, Barlas Oğuz, Rulin Shao, Gargi Ghosh, Jason Weston, and Wen-tau Yih. Learning to reason for factuality. *arXiv preprint arXiv:2508.05618*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics (TACL)*, 9:1012–1031, 2021.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning llms on new knowledge encourage hallucinations? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7765–7784, 2024.

Zorik Gekhman, Eyal Ben-David, Hadas Orgad, Eran Ofek, Yonatan Belinkov, Idan Szpektor, Jonathan Herzig, and Roi Reichart. Inside-out: Hidden factual knowledge in LLMs. In *Conference on Language Modeling (COLM)*, 2025.

Zorik Gekhman, Roee Aharoni, Eran Ofek, Mor Geva, Roi Reichart, and Jonathan Herzig. Thinking to recall: How reasoning unlocks parametric knowledge in llms. *arXiv preprint arXiv:2603.09906*, 2026.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics (TACL)*, 8:423–438, 2020.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 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)*, pp. 1601–1611, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Junyi Li and Hwee Tou Ng. Reasoning models hallucinate more: Factuality-aware reinforcement learning for large reasoning models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=Igq7Dyc3OL.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 41451–41530, 2023.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. Generated knowledge prompting for commonsense reasoning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

Kai Liu, Ze Chen, Zhihang Fu, Wei Zhang, Rongxin Jiang, Fan Zhou, Yaowu Chen, Yue Wu, and Jieping Ye. Structure-aware domain knowledge injection for large language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)"*, pp. 29443–29464, 2025.

Melody Ma and John Hewitt. Improving parametric knowledge access in reasoning language models. *arXiv preprint arXiv:2602.22193*, 2026.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 9802–9822, 2023.

Haoming Meng, Kexin Huang, Shaohang Wei, Chiyu Ma, Shuo Yang, Xue Wang, Guoyin Wang, Bolin Ding, and Jingren Zhou. Sparse but critical: A token-level analysis of distributional shifts in rlvr fine-tuning of llms. *arXiv preprint arXiv:2603.22446*, 2026.

Seyed Mahed Mousavi, Simone Alghisi, and Giuseppe Riccardi. What does loss optimization actually teach, if anything? knowledge dynamics in continual pre-training of llms. *arXiv preprint arXiv:2601.03858*, 2026.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. In *International Conference on Learning Representations (ICLR)*, 2025.

Oded Ovadia, Meni Brief, Rachel Lemberg, and Eitam Sheetrit. Knowledge-instruct: Effective continual pre-training from limited data using instructions. *arXiv preprint arXiv:2504.05571*, 2025.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5687–5711, 2023.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Baochang Ren, Shuofei Qiao, Ningyu Zhang, Da Zheng, and Huajun Chen. KnowRL: Exploring knowledgeable reinforcement learning for factuality. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 39640–39658, 2026.

Taylor Shin, Yasaman Razeghi, Robert L Logan Iv, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4222–4235, 2020.

Zhiqing Sun, Xuezhi Wang, Yi Tay, Yiming Yang, and Denny Zhou. Recitation-augmented language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yifei Wang, Yuheng Chen, Wanting Wen, Yu Sheng, Linjing Li, and Daniel Dajun Zeng. Unveiling factual recall behaviors of large language models through knowledge neurons. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7388–7402, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 24824–24837, 2022.

Zhepei Wei, Xiao Yang, Kai Sun, Jiaqi Wang, Rulin Shao, Jingxiang Chen, Mohammad Kachuee, Teja Gollapudi, Yiwei Liao, Nicolas Scheffer, et al. Truthrl: Incentivizing truthful llms via reinforcement learning. *arXiv preprint arXiv:2509.25760*, 2025.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=jGbRWwIidy.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Wanli Yang, Hongyu Zang, Junwei Zhang, Wenjie Shi, Du Su, Jingang Wang, Xueqi Cheng, and Fei Sun. Beyond reasoning: Reinforcement learning unlocks parametric knowledge in llms, 2026. URL https://arxiv.org/abs/2605.07153. arXiv preprint arXiv:2605.07153.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2369–2380, 2018.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=4OsgYD7em5.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Danna Zheng, Mirella Lapata, and Jeff Z Pan. How reliable are llms as knowledge bases? re-thinking facutality and consistency. *arXiv preprint arXiv:2407.13578*, 2024a.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models. In *International Conference on Learning Representations (ICLR)*, 2024b.

Zexuan Zhong, Dan Friedman, and Danqi Chen. Factual probing is [mask]: Learning vs. learning to recall. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 5017–5033, 2021.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Appendix

A Implementation Details

A.1 Reward Design

Priming and Reasoning share the same answer-quality reward r_{qa}, augmented by a stage-specific format-shaping term $r^{(s)}_{\text{fmt}}$. The reward for stage s is

$$r^{(s)}(\hat{a}, a^\star; o) \;=\; r_{\text{qa}}(\hat{a}, a^\star) \;+\; r^{(s)}_{\text{fmt}}(o). \tag{12}$$

Answer-quality term. The answer-quality reward blends exact match (EM) and token-level F1,

$$r_{\text{qa}}(\hat{a}, a^\star) \;=\; w_{\text{EM}} \cdot \text{EM}(\hat{a}, a^\star) \;+\; w_{\text{F1}} \cdot \text{F1}(\hat{a}, a^\star), \tag{13}$$

with $w_{\text{EM}} = 0.6$ and $w_{\text{F1}} = 0.4$. EM returns 1 when the normalized prediction matches any reference answer or its aliases, and 0 otherwise. Token-level F1 is the harmonic mean of precision and recall over whitespace-tokenized unigrams after lowercasing and removing punctuation. The F1 term supplies a continuous gradient that densifies the reward on hard multi-hop questions, where EM alone is too sparse because many near-correct answers receive zero reward.

Format-shaping term for Priming. In the Priming stage the policy outputs a JSON array of INSERT action objects, and the format reward checks three conditions:

1. The output is valid JSON and parseable as an array of INSERT action objects.
2. Each action carries the required fields: `action`, `subject`, `relation`, `object`.
3. The number of inserted triples falls within the allowed range, at least one and at most 20.

Well-formed outputs receive a small bonus of $+0.05$. Mildly malformed outputs, such as those missing one field but still parseable, receive zero format reward. Severely malformed outputs, meaning unparseable JSON or empty output, receive a penalty of -1.0 that overrides the answer-quality term entirely, so unparseable trajectories cannot be rewarded regardless of downstream answer correctness.

Format-shaping term for Reasoning. In the Reasoning stage the policy generates free-form chain-of-thought reasoning followed by a final answer, and the format reward checks two conditions:

1. The output contains a valid `<answer>` tag from which the final answer can be extracted.
2. The reasoning trace is non-empty, so the model has produced some intermediate reasoning before the answer.

The bonus and penalty magnitudes match Priming: $+0.05$ for well-formed output and -1.0 for severe malformation with no extractable answer.

Reward computation in Priming. During Priming the trainable policy emits inserted triples $\mathcal{I}$, which are applied to the retrieved subgraph $\mathcal{R}(q)$ to form an augmented subgraph $\mathcal{R}(q) \cup \mathcal{I}$. A frozen answerer M_0, the same base-model checkpoint held fixed throughout training, then receives the augmented subgraph and produces a predicted answer $\hat{a} = M_0(q, \mathcal{R}(q) \cup \mathcal{I})$. The reward is computed against the gold answer $a^\star$ and propagated to the policy through the GRPO advantage, but gradients flow only through π_θ, since the frozen answerer is non-differentiable and serves purely as a reward oracle.

For a reference point in the reward signal, we also compute the frozen answerer's prediction on the unaugmented subgraph, $\hat{a}_0 = M_0(q, \mathcal{R}(q))$, and cache it. Because M_0, the question, and $\mathcal{R}(q)$ are fixed, $\hat{a}_0$ is deterministic and shared across all rollouts for the same question.

A.2 GRPO Configuration

Both stages use Group Relative Policy Optimization (GRPO) with identical algorithmic hyperparameters. Table 6 summarizes the full configuration.

Hyperparameter	Value
Group size G (trajectories per question)	8
Sampling temperature	0.7
Clipping range ϵ	0.2
KL penalty coefficient β	0.001
KL penalty type	low-variance KL
Advantage normalization δ	10^{-5}
Learning rate	1×10^{-6}
Optimizer	AdamW
Batch size (prompts per update)	28
Mini-batch size	28
Gradient checkpointing	enabled
Max prompt length	2,048 tokens
Max response length	2,048 tokens

Table 6: GRPO hyperparameters shared across Priming and Reasoning stages.

Reference policy. In both stages the KL penalty is computed against the stage-specific initialization. For Priming the reference policy is the pretrained base model π_{θ_0}, and for Reasoning it is the Priming checkpoint $\pi_{\theta^{(1)}}$. Reference-model parameters are offloaded to CPU through FSDP parameter offloading to reduce GPU memory during training.

Rollout generation. At each training step the current policy generates $G = 8$ independent trajectories per question at temperature 0.7. Rollout generation uses vLLM as the inference backend inside the veRL framework, with tensor parallelism size 1 for models up to 7B and size 2 for 14B models. The vLLM engine runs with a maximum model length of 4,096 tokens and GPU memory utilization of 0.6, which leaves headroom for the training processes that share the same GPU cluster.

A.3 Training Setup

Data and epochs. For each in-distribution dataset we train for one epoch over the training split of 12K to 14K examples, which yields roughly 500 gradient steps per stage. We evaluate the 200-example validation set every 2 steps to monitor convergence and save checkpoints every 20 steps.

Checkpoint selection. We use the final checkpoint, the last step of the single epoch, for evaluation rather than early stopping, because convergence is stable and we observe no significant overfitting within one epoch across all datasets and model scales.

Two-stage sequential training. The two stages are trained in sequence. Priming starts from the base-model initialization and produces a Priming checkpoint $\theta^{(1)}$. Reasoning then initializes from $\theta^{(1)}$ and trains for one more epoch with the same hyperparameters but the Reasoning-stage format reward. The total budget is two epochs, one per stage, for roughly 1,000 gradient steps.

Frozen answerer configuration. During Priming the frozen answerer M_0 runs as a separate vLLM inference server on a dedicated GPU that is physically isolated from the training GPUs. It uses greedy decoding at temperature 0 with a maximum output length of 128 tokens, which is enough for short factual answers. The answerer receives only the question and the augmented triple subgraph, never raw documents, so any improvement in the answer is attributable to the inserted triples alone.

B Training Curve

We plot validation accuracy against GRPO updates for every backbone we train, to expose VAKE's training dynamics. All curves use a held-out 2WikiMultihopQA validation split and the greedy-decoding, judge-EM protocol adopted throughout the paper. Each run spans 500 GRPO steps, with

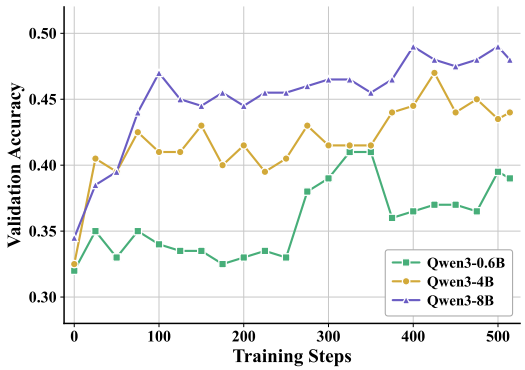

Figure 5: Validation accuracy on 2WikiMultihopQA over 500 GRPO steps of Priming, for Qwen3-{0.6B, 4B, 8B}.

a validation pass every 10 steps reported without smoothing. Two model families appear, both under the shared GRPO configuration in Table 6, and both share the same x-axis for direct cross-model comparison.

Qwen3 Priming across model sizes. Figure 5 plots Priming-stage validation accuracy of Qwen3-{0.6B, 4B, 8B} over 500 GRPO steps. Priming does not update the answerer: the reward comes from a *frozen* copy of the same backbone reading the augmented subgraph (§A). These curves therefore track the policy's ability to inject *useful* bridging triples that make the frozen answerer more likely to be correct. Two features stand out. First, all three sizes rise monotonically from their base-model accuracy and are still improving, or have only just plateaued, at step 500; the answer-quality reward carries a well-shaped gradient even for the 0.6B backbone. Second, larger backbones start higher and converge higher, but the *relative* gain from Priming is largest at 0.6B, consistent with the view that structured RL insertion extracts more value from weaker parametric memory than from an already-strong one.

Qwen2.5 Priming then Reasoning across model sizes. Figure 6 plots the full two-stage trajectory for Qwen2.5-{3B, 7B, 14B}. Each backbone runs 500 GRPO steps of Priming, and the resulting checkpoint warm-starts another 500 steps of Reasoning. We render the two stages as one continuous curve along the training-step axis, with a vertical marker separating the phases. The Priming half mirrors Qwen3: monotone gains at every size, with more headroom at smaller scales. Across the Priming→Reasoning boundary the curve does *not* regress. The policy keeps improving, but the source of the gain shifts from "insert better bridging triples" to "reason over the already-augmented subgraph better", matching the two-stage decomposition VAKE is designed to exploit. All three sizes converge by the end of Reasoning, which motivates our default $500 + 500$ budget for the main-table Qwen2.5 runs.

Policy entropy during Reasoning: with vs. without Priming. Beyond validation accuracy, we monitor *policy entropy* throughout the Reasoning stage, that is, the per-token entropy of the sampling distribution over rollouts, which measures how confidently the policy commits to its next token. Figure 7 contrasts two initializations of Qwen3-8B on HotpotQA, MuSiQue, and 2WikiMultihopQA. The first, **Reasoning**, starts Reasoning-stage GRPO directly from the base model. The second, **VAKE**, starts Reasoning from the Priming-stage checkpoint. The two runs share identical data, prompts, hyperparameters, and evaluation seeds; the only difference is whether Priming has been applied first. The pattern is dataset-dependent but internally consistent. On HotpotQA the VAKE-initialized policy sustains substantially higher entropy throughout training. On MuSiQue it drops more slowly and stabilizes higher toward the end. On 2Wiki it stays at or below the Reasoning-only baseline late in training. Read alongside the accuracy curves, this indicates that Priming does not push the policy toward premature determinism: the accuracy gains in Figure 6 are not bought by collapsing to a narrower output distribution. On HotpotQA and MuSiQue in particular, VAKE preserves more exploration budget deep into training, precisely the regime where Reasoning-stage RL typically loses signal to entropy collapse.

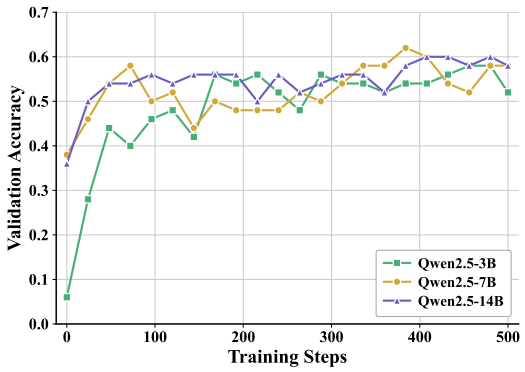

Figure 6: Validation accuracy on 2WikiMultihopQA over the full two-stage training trajectory for Qwen2.5-{3B, 7B, 14B}.

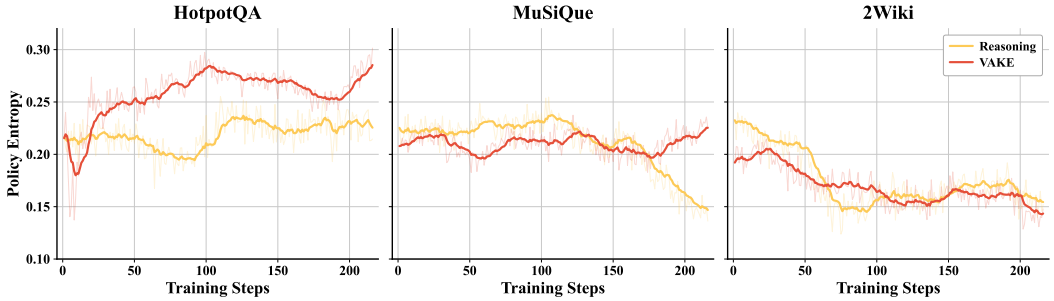

Figure 7: Policy entropy during Reasoning-stage GRPO for Qwen3-8B, comparing initialization from the base model (**Reasoning**) with initialization from the Priming-stage checkpoint (**VAKE**), on HotpotQA, MuSiQue, and 2WikiMultihopQA. Faint lines show raw per-step entropy; solid lines show a moving average.

C Prompt Templates

This section lists the complete prompt templates used in our experiments, shown verbatim as they appear in the training and evaluation pipelines, with placeholders in {`braces`}. Three prompts drive the pipeline. The Priming prompt (Figure 8) uses a single combined template with no separate system and user split. The policy receives a question and a retrieved triple subgraph, and outputs a JSON array of INSERT actions that augment the subgraph with bridging knowledge from its parameters, or an empty array when the subgraph already suffices. The Reasoning prompt (Figure 9) presents the knowledge graph in arrow notation as primary evidence, and instructs the model to produce chain-of-thought reasoning followed by a final answer inside `<answer>` tags. The LLM judge prompt (Figure 10), run with GPT-4o at temperature 0, decides semantic equivalence between the predicted answer and the gold answer under strict rules for edge cases, scoring against the primary answer and all dataset-provided aliases.

D Baseline Details

This section documents the exact procedure and prompt used by each method in the main table. To keep the comparison clean, every method shares the same backbone (Qwen2.5-7B-Instruct or Qwen3-8B), operates on the same retrieved triple subgraph $\mathcal{R}(q)$, uses the same answering interface, and is forbidden from calling any external search. All supporting evidence must come from either $\mathcal{R}(q)$ or the model's own parameters. This treats the prompt-based baselines as pure prompting variants of the same knowledge-activation problem VAKE targets, and isolates the effect of

Prompt for Priming

```
You are a knowledge graph construction expert. You will see a question and a retrieved triple subgraph (with numbered indices).

Your task is to determine whether the current subgraph is sufficient to answer the question. If not, insert new bridging triples that supply the missing factual knowledge needed to reach the answer.

## Available Actions

- INSERT(subject, relation, object) | Add a new factual triple to bridge existing entities with new knowledge

## Output Format

A strict JSON array, where each element is an INSERT action object:
- {"action": "INSERT", "subject": "...", "relation": "...", "object": "..."}

If the current subgraph is already sufficient to answer the question, output an empty array [].
Do not output any explanation.

## Question
{question}

## Retrieved Subgraph
Indexed triples (reference by #id):
  #0 ({subject_0}, {relation_0}, {object_0})
  #1 ({subject_1}, {relation_1}, {object_1})
  ...
  #K ({subject_K}, {relation_K}, {object_K})

## Output INSERT actions (JSON array, output [] if no inserts needed)
```

Figure 8: Complete prompt for the Priming stage. The policy inserts bridging triples from parametric knowledge. `{question}` is filled with the input question; the indexed triples are the retrieved subgraph $\mathcal{R}(q)$ (capped at $K = 10$).

Prompt for Reasoning

```
You are a precise question answering system. Think step by step, then give a short, direct final answer.

Use the following knowledge graph as the primary evidence to answer the question.
If it is insufficient, also draw on your own knowledge to fill the gaps.
Always commit to your best answer; never say it cannot be determined.
Put your final answer inside <answer></answer> tags.
The content inside <answer> must be ONLY the shortest entity name or phrase, with no explanation, no full sentence.

Knowledge graph (reasoning chain):
- {subject_0} → {relation_0} → {object_0}
- {subject_1} → {relation_1} → {object_1}
...
- {subject_K} → {relation_K} → {object_K}

Question: {question}
```

Figure 9: Complete prompt for the Reasoning stage. The knowledge graph triples are presented in arrow notation. The model produces chain-of-thought reasoning followed by a final answer in `<answer>` tags.

how parametric knowledge is elicited: self-decomposition, self-recitation, or RL-learned structured insertion of bridging triples.

D.1 SELF-ASK

Background. Self-Ask elicits multi-hop answers from a language model by making the decomposition explicit in the output. Instead of jumping to the answer, the model is prompted to first ask itself a sequence of atomic `Follow up:` sub-questions, answer each with an `Intermediate answer:` line, and finish with a single `So the final answer is:` `<answer>` line. The reference implementation supplies a 4-shot prompt on Bamboogle-style compositional questions,

Prompt for LLM Judge

```
You are a strict QA grader. Given a question, the gold (reference) answer, and a model's predicted answer, decide whether the prediction is CORRECT: it must give the SAME answer to the question as the gold. The prediction may be longer, phrased as a full sentence, or add harmless context | that is fine ONLY IF its core answer is equivalent to the gold. Mark it INCORRECT if ANY of the following holds:
(a) it names a different entity/value than the gold;
(b) it gives a NARROWER or BROADER term instead of the gold (e.g. a specific member of the group the gold names, a part/subset of the gold, or a parent category of the gold) | the answer must denote the SAME entity/scope the gold does, not merely something related or contained;
(c) it answers a DIFFERENT aspect of the question than what the gold answers (e.g. gold gives WHAT was done, prediction gives WHERE);
(d) it is a refusal, non-answer, or hedges without committing.
Genuine equivalents ARE correct: aliases, abbreviations/expansions, spelling/formatting variants, and the same entity under another common name. When in doubt, mark INCORRECT.
Output STRICT JSON only.

Judge whether the predicted answer is semantically correct.

## Question
{question}

## Gold answer (reference; any of the accepted forms below count as correct)
- {gold_answer}
- {alias_1}
- {alias_2}
...

## Predicted answer
{prediction}

## Output
Strict JSON only, nothing else:
{"correct": true or false, "reason": "one short sentence"}
```

Figure 10: Complete prompt for the LLM judge. The judge applies strict semantic equivalence with explicit rules for edge cases. Gold answers include the primary answer and all dataset-provided aliases. The judge outputs a binary decision with a brief justification.

and in its strongest form delegates each intermediate answer to an external search engine. That external-search dependency is what we ablate away in our port.

Our triples-grounded adaptation. To keep Self-Ask comparable to VAKE we make two adjustments while leaving the 4-shot prompt itself untouched. First, we prepend the retrieved subgraph $\mathcal{R}(q)$ as a hint block (bullet-listed triples in arrow notation, capped at $K = 10$) between the exemplars and the target question, so the decomposition can be anchored on the same evidence VAKE sees. Second, we replace the external-search step with a system message that instructs the model to answer each follow-up from the triples plus its own parametric memory, forbids external search, and requires it to always commit to a best-guess answer instead of hedging. Phrases such as "the information is unavailable" or "cannot be determined" are explicitly ruled out. The system message also asks the final answer to be as short as possible, an entity name, a date, or yes/no, with no explanation, to keep it comparable with the EM/F1 scoring used everywhere else. The full template appears in Figure 11.

Decoding and answer extraction. Following the reference implementation, we run Self-Ask with a single greedy trajectory (temperature 0, max length 512 tokens). We omit self-consistency voting because Self-Ask is a decomposition prompt rather than a sampling method, and adding voting on top would conflate the two contributions. The final answer is extracted by scanning the trajectory tail for the last "`So the final answer is:`" marker and taking the remainder of that line, with the trailing period stripped. A fallback rule takes the substring after the last colon of the last non-empty line if the marker is absent. On Qwen3-8B we additionally set `enable_thinking=false` so the model does not emit a hidden `<think>` block that would displace or truncate the Self-Ask trajectory.

D.2 RECITE

Background. RECITE (Recitation-Augmented Generation) tackles closed-book QA by making the model's implicit knowledge explicit. The model is first prompted to *recite* relevant evidence passages from its own parameters, and then answers conditioned on the self-generated passages. Self-consistency smooths over recitation noise: multiple recitations are sampled and a majority vote is taken over the resulting answers. In its original form, Recite is purely closed-book: the model receives only the question and hallucinates evidence unconditionally.

Our triples-grounded adaptation. To keep Recite comparable to VAKE we again keep the reference implementation's 4-shot HotpotQA recitation exemplars intact, but anchor the recitation on the same retrieved subgraph $\mathcal{R}(q)$ that VAKE consumes. The user message contains the four exemplars, then a bullet-listed hint block of retrieved triples, then the target question, and finally the fixed connector line "`The answer to the above question can be found in the following 2 Wikipedia page, section, and paragraphs:`" followed by "`Answer 1:`". The system message narrows the task from "recall anything relevant" to "recall two Wikipedia paragraphs grounded on the given triples," labeled `Answer 1:` and `Answer 2:`. The recitation template appears in Figure 12. Recited passages are treated exactly as any other context document at answering time. They are not counted as retrieved documents (no real Wikipedia is ever consumed) and the answerer never sees the corpus.

Sampling, answering, and majority voting. We sample the recitation step multiple times with a 256-token cap, then parse each raw generation with a small post-processor that trims any spurious continuation into the next exemplar and re-prepends the "`Answer 1:`" prefix that the prompt consumes. Each recited passage is fed as a *single* context document (title fixed to "Recited evidence") to the same answering template VAKE uses at inference (§C, Figure 9), decoded greedily (temperature 0, 256 tokens). The candidate answers are aggregated by majority voting, breaking ties toward the shorter answer and ignoring empty strings. As with Self-Ask, on Qwen3-8B we set `enable_thinking=false` to keep the recitation and answering steps free of stray `<think>` blocks that would otherwise pollute both parsing and voting.

D.3 RL UNLOCKS PARAMETRIC KNOWLEDGE

Background. Unlock argues that a language model's parametric knowledge can be surfaced by plain outcome-driven RL alone: no supervised fine-tuning, no retrieval augmentation, and no structured intermediate steps such as chain-of-thought or subgraph augmentation. The policy is trained with GRPO to answer the input question directly, and the reward is a single binary signal produced by a large frozen LLM judge that decides whether the model's answer is semantically equivalent to the gold. The claim we test is that this minimal RL recipe already yields non-trivial gains on multi-hop QA, which is the direct point of comparison for VAKE's structured two-stage training.

Our reproduction. We re-implement Unlock inside the same training stack (verl + GRPO) that runs our own method, so any difference in outcome reflects the method rather than the infrastructure. The backbone (Qwen2.5-7B-Instruct / Qwen3-8B), the three multi-hop training and evaluation datasets (2WikiMultihopQA, HotpotQA, MuSiQue, 1000 test each), and the retrieved triple subgraph $\mathcal{R}(q)$ fed to the policy are all identical to those used by VAKE, so the two methods share the same data denominator. The policy consumes "question + $\mathcal{R}(q)$" and directly emits an answer, with no insertion stage, no self-consistency, and no external search.

GRPO configuration. We align algorithmic hyperparameters with the reference implementation: rollout group size $n = 5$ per prompt, sampling temperature 1.0 with $\text{top-p} = 1.0$ and $\text{top-k} = -1$, constant learning rate 1×10^{-6}, KL penalty coefficient $\beta = 0.001$ (low-variance KL), and 8 training epochs on each dataset. On an 8-GPU node the trainer occupies 6 GPUs while the 72B judge takes the remaining 2 (tensor-parallel 2), so the reward and the policy updates run concurrently without cross-node RPC.

Reward: 72B LLM judge (key difference vs. VAKE). The only substantive design difference between Unlock and VAKE lies in the reward. Unlock treats each rollout with a *binary outcome*

Prompt for Self-Ask Baseline

```
[System]
You are a knowledgeable assistant that answers multi-hop questions by decomposing them into simpler follow-up questions and answering each one yourself. The knowledge graph triples provided are only HINTS and may be incomplete or irrelevant -- whenever they do not contain what you need, rely on your own knowledge to answer the follow-up questions. Do NOT use any external search. You must ALWAYS commit to a single best-guess answer; never say the information is unavailable, cannot be determined, or is insufficient. Follow the format of the provided examples exactly: decide if follow-up questions are needed, ask and answer them with 'Follow up:' / 'Intermediate answer:', then finish with a single line 'So the final answer is: <answer>'. The final answer MUST be as short as possible (an entity name, a date, or yes/no) with NO explanation.

[User]
Question: Who lived longer, Muhammad Ali or Alan Turing?
Are follow up questions needed here: Yes.
Follow up: How old was Muhammad Ali when he died?
Intermediate answer: Muhammad Ali was 74 years old when he died.
Follow up: How old was Alan Turing when he died?
Intermediate answer: Alan Turing was 41 years old when he died.
So the final answer is: Muhammad Ali

... (3 more 4-shot exemplars omitted)

Relevant triples (hints, may be incomplete -- use your own knowledge to fill gaps):
- {subject_0} -> {relation_0} -> {object_0}
- {subject_1} -> {relation_1} -> {object_1}
...
- {subject_K} -> {relation_K} -> {object_K}

Question: {question}
Are follow up questions needed here: Yes.
Follow up:
```

Figure 11: Complete prompt used for the Self-Ask baseline. The 4-shot exemplars follow the reference implementation; the system message additionally grounds the decomposition on the retrieved triples and forbids external search. The final answer is extracted from the last "`So the final answer is:`" line of a single greedy trajectory.

reward produced by a frozen **Qwen2.5-72B-Instruct** judge that receives the question, the gold answer (with aliases), and the extracted prediction, and returns 1 if it deems the prediction semantically equivalent and 0 otherwise. Token-level F1 and exact match are logged for monitoring but do not enter the reward. VAKE, by contrast, uses a purely rule-based EM+F1 reward at training time (§A, $w_{\text{EM}} = 0.6$, $w_{\text{F1}} = 0.4$) across all datasets, and reserves the LLM judge exclusively for post-hoc evaluation. This split is deliberate. Training with a cheap, deterministic rule-based reward avoids coupling the RL signal to the idiosyncrasies of a specific judge model, while the LLM judge is still used for reporting because it corrects the well-known under-counting of string EM under alias and paraphrase variation. Any main-table gap between Unlock and VAKE therefore isolates the effect of *how* the RL signal is structured, unstructured answer-only versus structured insert-then-reason, and not the cost of the reward oracle, on which VAKE is strictly cheaper.

E Dataset Details

We evaluate VAKE on seven factual question-answering benchmarks that together span multi-hop compositional reasoning and single-hop factual recall. Three multi-hop datasets serve as in-distribution (ID) benchmarks for both training and testing. Four further datasets are held out for out-of-distribution (OOD) evaluation, probing how well the learned knowledge-activation capability generalizes. Table 7 summarizes the splits used in our experiments, and Table 8 compares them against the original dataset sizes.

E.1 Data Sampling and Split Construction

For each in-distribution dataset we subsample the original training partition into a training split of 12K to 14K examples, and reserve 200 examples as a held-out validation set for checkpoint selection. 2WikiMultihopQA and HotpotQA each contribute 14,400 training examples drawn uniformly from multiple shards, using 15 shards of 1,000 examples with a final shard of 600. MuSiQue contributes

Prompt for Recite Baseline (Recitation Step)

```
[System]
You are a knowledgeable assistant. Given a question and a few
relevant knowledge graph triples as hints, recall from your own
knowledge the relevant Wikipedia paragraphs that contain the
evidence needed to answer the question. Follow the format of the
provided examples exactly: output two evidence paragraphs labeled
'Answer 1:' and 'Answer 2:'.
```

```
[User]
Question: Which magazine was started first Arthur's Magazine or
First for Women?

The answer to the above question can be found in the following 2
Wikipedia page, section, and paragraphs:

Answer 1: Arthur magazine was a bi-monthly periodical that was
founded in October 2002, by publisher Laris Kreslins and editor
Jay Babcock.

Answer 2: First for Women is a woman's magazine published by
Bauer Media Group in the USA. The magazine was started in 1989.

... (3 more 4-shot exemplars omitted)

Relevant triples (hints):
- {subject_0} -> {relation_0} -> {object_0}
...
- {subject_K} -> {relation_K} -> {object_K}

Question: {question}

The answer to the above question can be found in the following 2
Wikipedia page, section, and paragraphs:

Answer 1:
```

Figure 12: Complete recitation prompt used for the Recite baseline. The 4-shot exemplars are the HotpotQA recitation template from the reference implementation; the system message and the "Relevant triples (hints)" block ground the recitation on the retrieved subgraph.

Dataset	Role	Hops	Train	Val	Test	Avg. Ans. Len.
2WikiMultihopQA	ID	2	14,400	200	1,000	1.9
HotpotQA	ID	2	14,400	200	1,000	2.2
MuSiQue	ID	2–3	11,800	200	1,000	2.4
Bamboogle	OOD	2	—	—	125	2.1
NQ	OOD	1	—	—	3,610	2.2
TriviaQA	OOD	1	—	—	11,313	2.5
PopQA	OOD	1	—	—	14,267	1.9

Table 7: Dataset statistics as used in our experiments. "Hops" indicates the number of reasoning steps required. ID datasets are used for training and evaluation; OOD datasets are used for evaluation only. Avg. Ans. Len. denotes the mean number of whitespace-tokenized words in the gold answer.

11,800 training examples from 12 shards of 1,000 examples each. The smaller MuSiQue training set reflects its smaller pool of multi-hop compositions after filtering for 2–3 hop answerable questions with verified reasoning chains.

The test split for each ID dataset holds 1,000 examples drawn from the original development or test partition, kept fully disjoint from the training examples. For each test question we prepare three context variants, triples-only, triples-with-documents, and empty-insert, to support systematic evaluation across input conditions. Section F gives the details of graph construction and retrieval.

OOD datasets are used in full without subsampling: Bamboogle (125 examples), NQ (3,610), TriviaQA (11,313), and PopQA (14,267). All OOD evaluations use the single checkpoint trained on HotpotQA, with no dataset-specific fine-tuning or adaptation. This setup tests whether the learned

Dataset	Total (orig.)	Original Split	Source
2WikiMultihopQA	192,606	167,454 / 12,576 / 12,576	Wikidata + Wikipedia
HotpotQA	112,779	90,564 / 7,405 / 7,405×2	Wikipedia
MuSiQue	24,814	19,938 / 2,417 / 2,459	Composed single-hop
Bamboogle	125	— / — / 125	Hand-crafted
NQ	323,045	307,373 / 7,830 / 7,842	Google Search logs
TriviaQA	95,956	76,764 / 9,596 / 9,596	Trivia enthusiasts
PopQA	14,267	— / — / 14,267	Wikidata triples

Table 8: Original dataset sizes (full release) for reference. The splits in Table 7 are subsampled from these full releases.

Dataset	Question types	Construction pipeline
2WikiMultihopQA	Comparison, Inference, Compositional, Bridge-comparison	Templates + logical rules over Wikidata triples, grounded on Wikipedia text; reasoning paths annotated as ordered triples.
HotpotQA	Bridge, Comparison	Crowd-workers read pairs of Wikipedia introductions and write multi-hop questions with sentence-level supporting facts.
MuSiQue	2–4 hop compositions of single-hop QA	Bottom-up composition of SQuAD/NQ single-hop questions through shared entities; every composed question is verified to require all hops.

Table 9: Question types and construction pipelines of the three in-distribution datasets.

knowledge-activation capability transfers across both dataset distributions and reasoning depths, from multi-hop to single-hop.

E.2 In-Distribution Datasets

Table 9 summarizes the question typology and construction pipeline of the three ID datasets; the paragraphs below focus on provenance and why each dataset is included.

2WikiMultihopQA. 2WikiMultihopQA targets a known weakness in earlier multi-hop benchmarks, where many "multi-hop" questions can be solved through single-hop shortcuts. It combines structured Wikidata triples with unstructured Wikipedia text, and applies logical rules to generate questions that provably require multi-step inference. A distinctive feature is the annotation of evidence as ordered reasoning paths, which allows evaluating whether a model follows the correct intermediate steps rather than reaching the answer through spurious correlations. We choose this dataset because its template-generated structure and evidence annotations align naturally with our triple-based Priming formulation, and its guaranteed multi-hop property rules out correct answers from superficial pattern matching.

HotpotQA. HotpotQA is the first large-scale, crowd-sourced multi-hop QA dataset built over natural Wikipedia paragraphs. Unlike template-based benchmarks, its questions are written by human annotators who read pairs of Wikipedia introductions, giving high linguistic and topical diversity. A hallmark contribution is sentence-level supporting-fact annotations, the minimal set of sentences necessary and sufficient to derive the answer, which supports training and evaluation of explainable multi-hop systems. We use HotpotQA as our primary training dataset and the source of all OOD transfer checkpoints, because its crowd-sourced formulations better represent the distribution of reasoning patterns a model meets in real applications.

MuSiQue. MuSiQue addresses a limitation shared by HotpotQA and 2WikiMultihopQA: models can often exploit disconnected clues or single-hop shortcuts to reach the correct answer without genuine compositional reasoning. Its bottom-up construction chains single-hop questions from SQuAD and Natural Questions through shared entities, and every composed question is verified to satisfy

Dataset	Size	Hops	Built from	Stress test
Bamboogle	125	2	Hand-crafted	Compositional generalization to hand-written, lexically diverse multi-hop questions.
NQ	3,610	1	Google Search logs	Natural query formulations from real information needs.
TriviaQA	11,313	1	Trivia enthusiasts	Large lexical gap between question and evidence; semantic matching, not surface copying.
PopQA	14,267	1	Wikidata triples	Long-tail entity popularity: recall of rarely surfaced parametric knowledge.

Table 10: Out-of-distribution datasets used for zero-shot evaluation. We perform no training or fine-tuning on any OOD dataset; all evaluations use the single HotpotQA-trained checkpoint.

a single-hop disconnectedness property, so no individual sub-question alone leads to the final answer. Each answerable question is also paired with a minimally altered unanswerable counterpart for answerability-detection evaluation. In our experiments we use the answerable subset restricted to 2–3 hop questions, the dominant depth in our KG setting (average 2.85 hops). We select MuSiQue because its verified hop-necessity property gives the strongest guarantee that improvements reflect genuine multi-step knowledge activation rather than shortcut exploitation.

E.3 Out-of-Distribution Datasets

The four OOD datasets probe complementary aspects of generalization. Table 10 summarizes their key features; the paragraphs below describe each dataset's provenance and the specific stress test it applies to our method.

Bamboogle. Bamboogle was introduced as part of a study on the *compositionality gap*, the phenomenon in which a language model can answer each sub-question of a compositional question yet fails on the composed question itself. It consists of 125 hand-written two-hop questions constrained to be (i) hard enough that a popular search engine cannot answer them directly, and (ii) each composed from two facts that are individually present in English Wikipedia. Unlike template-generated multi-hop datasets, Bamboogle has high lexical and topical diversity, no answer aliases, and no supporting-fact annotations, which makes it a purely zero-shot evaluation target. Its small size limits statistical power but provides a qualitative signal on compositional generalization beyond the template distributions seen in training.

Natural Questions (NQ). NQ is built from real, anonymized queries issued to Google Search, which makes it one of the most ecologically valid QA benchmarks. Each query is paired with a Wikipedia page returned as a top search result, and annotators mark a long-answer span (paragraph, table, or list) and one or more short-answer entity spans (or a yes/no judgment). We use the open-domain short-answer formulation with 3,610 test examples that have non-null short answers, following standard open-book QA protocol. We include NQ because its queries come from genuine information needs rather than artificial constructions. As a single-hop dataset, it probes whether the multi-hop Priming capability transfers to direct factual recall without intermediate reasoning.

TriviaQA. TriviaQA collects 95,956 question–answer pairs authored by trivia enthusiasts on quiz-league and trivia websites. Evidence documents are gathered *retrospectively* through web search and Wikipedia retrieval, and question authors never saw these documents when writing the questions. This decoupling produces a large lexical and syntactic gap between questions and evidence passages. Unlike reading-comprehension datasets whose annotators phrase questions using vocabulary from the passage, TriviaQA questions use independent phrasing that often shares few surface tokens with the supporting evidence. We use the Wikipedia-domain human-verified subset (11,313 examples). The lexical gap specifically tests the model's ability to activate parametric knowledge through semantic matching rather than surface-form copying.

Axis	Coverage across the seven datasets
Reasoning depth	Single-hop factual recall (NQ, TriviaQA, PopQA) → 2-hop composition (HotpotQA, 2WikiMultihopQA, Bamboogle) → 2–4 hop chains (MuSiQue).
Construction methodology	Template + logical rules (2WikiMultihopQA), crowd-sourced natural text (HotpotQA), compositionally verified (MuSiQue), hand-crafted for difficulty (Bamboogle), real search queries (NQ), trivia-enthusiast authored (TriviaQA), knowledge-graph sampled (PopQA).
Entity frequency	Head entities that appear often in pretraining (most of NQ, TriviaQA) → deliberately long-tail entities (PopQA), directly testing whether activation improves access to rarely surfaced parametric knowledge.

Table 11: Three orthogonal axes along which the seven-benchmark suite probes knowledge activation.

PopQA. PopQA is an entity-centric factual QA dataset designed to study the relationship between entity popularity and language model memorization. Factual triples are sampled from Wikidata across 16 relationship types (occupation, capital-of, author-of, director-of, and so on), each converted to a natural-language question through a relation-specific template, and the subject entity's Wikipedia page-view count is recorded as a proxy for how often the entity appeared in LLM pretraining corpora. The resulting 14,267 questions follow a deliberately long-tailed popularity distribution. We include PopQA because it lets us analyze VAKE's activation capability as a function of entity popularity: its long tail is exactly the regime where parametric knowledge is most likely stored but inaccessible, and thus most likely to benefit from Priming.

E.4 Design Rationale for Dataset Selection

Our seven-benchmark suite probes knowledge activation along three orthogonal axes summarized in Table 11. This design ensures that observed gains cannot be traced to a single dataset artifact, question style, or entity-frequency regime, and that VAKE's activation capability holds across diverse evaluation conditions.

F Graph Construction and Retrieval

This section details the offline knowledge graph construction and the online subgraph retrieval procedure summarized in Section 4.1 of the main text. The design objective is to provide query-relevant cues that can condition the elicitation of parametric knowledge, rather than a complete reasoning path that would trivialize the task.

F.1 Knowledge Graph Construction

We construct a knowledge graph $\mathcal{G} = \{(s, r, o)\}$ for each dataset using a two-stage LLM-based Open Information Extraction (OpenIE) pipeline. The pipeline processes each passage independently and does not require cross-document coreference or a global relation schema. Both extraction stages use GPT-4o through the OpenAI Responses API with temperature 0.

Stage 1: Entity extraction. For each passage in the supporting corpus, GPT-4o extracts key named entities, concepts, and referential expressions that may serve as subjects or objects in factual triples. The output is a structured list under the `named_entities` field.

Stage 2: Conditional relation extraction. The same GPT-4o model then receives the passage together with the extracted entity list and outputs structured $(\textit{head}, \textit{relation}, \textit{tail})$ triples. The head and tail are constrained to entities in the Stage-1 list, while the relation is represented as a free-form phrase. This decomposition reduces the output space of relation extraction and anchors the resulting triples to passage-level entity mentions.

After extraction, we merge all passage-level triples into a single graph $\mathcal{G}$ for each dataset. Entities form the nodes and triples form directed labeled edges. Duplicate triples, defined as triples with

Table 12: Knowledge graph statistics for the in-distribution datasets. Entities denote unique nodes, and triples denote unique directed edges after deduplication.

Dataset	Entities	Triples
2WikiMultihopQA	∼180K	∼320K
HotpotQA	∼210K	∼380K
MuSiQue	∼150K	∼270K

identical subject, relation, and object after lowercasing, are removed. The resulting graph sizes are summarized in Table 12.

F.2 Subgraph Retrieval

Given a question q and the knowledge graph $\mathcal{G}$, we construct a sparse query-relevant subgraph $\mathcal{R}(q) \subseteq \mathcal{G}$ through entity anchoring, two-hop neighborhood expansion, and answer filtering.

Step 1: Entity anchoring. We identify anchor nodes by matching entity mentions in q to graph nodes using case-insensitive substring matching. An entity is anchored if its name occurs as a substring of the question or if a question span occurs as a substring of the entity name. The resulting anchor set initializes graph traversal. The implementation also supports an optional query-side NER module through a `QAActor`, whose default backbone is Qwen2.5-7B-Instruct; this module is disabled in all main experiments, so the reported results use string-based anchoring only.

Step 2: Two-hop BFS with cosine filtering. Starting from the anchor set, we expand a two-hop breadth-first neighborhood over both incoming and outgoing edges. At each hop, adjacent triples are ranked by their cosine similarity to the query, and the top five triples are retained. The endpoints of the retained triples are then added to the frontier for the next hop.

For similarity computation, we use the local Hugging Face model *Qwen/Qwen3-Embedding-8B*. The query, entity strings, and triple strings are encoded with the final-layer hidden states, followed by attention-mask mean pooling and L2 normalization. Each triple is represented by the concatenated string *subject relation object*, and the query is encoded in the same way. Because the vectors are normalized, their dot product is equivalent to cosine similarity. No query rewriting, iterative refinement, or additional reranking is applied.

Step 3: Answer filtering and capping. For the controlled insufficient-context protocol, we merge the triples collected across both hops and remove any triple whose subject, relation, or object contains the gold answer or one of its dataset-provided aliases as a substring. We then cap the remaining set at $K = 10$ triples to obtain the final retrieved subgraph $\mathcal{R}(q)$. This answer-aware filtering reduces direct answer leakage and defines the intentionally sparse context used to study knowledge activation.

F.3 Design Choices and Rationale

Shallow retrieval is intentional. The retriever is designed to be shallow (two hops, five edges per hop) and deliberately under-powered. A stronger retriever would supply more complete reasoning paths, reducing the need for parametric knowledge activation and confounding the measurement of VAKE's contribution. The sparse subgraph serves as a set of query-relevant cues that can trigger the model's fact-association recall, rather than as a self-sufficient evidence base.

Answer filtering isolates parametric contribution. By removing triples that contain the gold answer, we ensure that any correct answer produced after Priming is attributable to the inserted bridging triples rather than to information already present in the retrieved context. This is the operational basis for the attribution analysis in Section 4.4 of the main text.

Fixed $K = 10$ across all datasets. We use a uniform cap of 10 triples regardless of dataset or question difficulty. This standardization ensures fair comparison across datasets and prevents the

retriever quality from varying as a confound. The cap also keeps the prompt length manageable (10 triples $\approx$ 150–200 tokens), leaving ample generation budget for the policy's INSERT actions.

G Probing Experiment Details

This section documents the two components of the probing analysis reported in the main text: (i) attributing each inserted triple to either the retrieved subgraph or the backbone's parameters, and (ii) measuring the reproduction accuracy of parameter-originated, factually correct triples under masked completion.

G.1 Source Attribution of Inserted Triples

For each backbone scale, we collect all INSERT actions emitted by its own VAKE-P checkpoint on the 2WikiMultihopQA test split, and attribute each inserted triple to one of two sources. Given the question, the retrieved subgraph $\mathcal{R}(q)$, and one inserted triple, a GPT-4o judge decides whether the triple can be found in or reasonably derived from $\mathcal{R}(q)$. Triples judged *subgraph-inferable* include verbatim restatements, paraphrases (e.g. `directed by` $\leftrightarrow$ `director`), and one-hop derivations from an adjacent triple. Everything else is labeled *parameter-originated*. The judge runs at temperature 0 and returns strict JSON; its full prompt appears in Figure 13.

G.2 Probe Set Construction

The probe set targets the parameter-originated fraction because the objective is to test whether a value the policy produced "from its own weights" can be reproduced by a frozen backbone. Starting from all triples inserted by VAKE-P (Qwen2.5-7B) on the 2WikiMultihopQA test split, we apply two successive filters, both scored by the same GPT-4o judge at temperature 0:

1. **Source filter** (§G.1): keep only triples labeled parameter-originated.
2. **Factuality filter**: keep only triples that the judge deems factually correct in the real world, discarding those judged incorrect or uncertain.

The factuality filter is essential. If a factually incorrect triple survives and a frozen backbone fails to reproduce it, the failure is uninterpretable, since it could equally mean "the knowledge does not exist" or "the knowledge is not accessible". Restricting to factually correct triples isolates the second reading. After both filters the probe set contains 490 triples, drawn from $\sim$300 distinct 2Wiki questions.

G.3 Reproduction Accuracy Protocol

For each probe triple (s, r, o) we independently mask one of the three positions and ask a frozen backbone to fill the blank using a knowledge-graph completion prompt (`Subject: s / Relation: r / Object: ?` for tail masking, and symmetrically for head and relation). The reference answer is the *original value produced by VAKE-P*, not the 2Wiki gold answer. This is a self-consistency test that asks whether the frozen backbone can recover what VAKE-P asserted, rather than whether it answers the question correctly. Factual correctness is already guaranteed by the construction of the probe set (§G.2).

Backbones decode greedily at temperature 0 with a 64-token cap. A prediction is scored as correct by the same LLM-as-judge protocol used in the main table (semantic equivalence, model-blind), and reproduction accuracy is the fraction of triples on which the frozen backbone's prediction is judged equivalent to VAKE-P's original value. We compare four frozen same-family backbones (Qwen2.5-7B / 14B / 32B / 72B) against VAKE-P (7B) on the same 490-triple probe set.

H Reward Hyperparameter Ablation

The Priming reward is a convex blend of exact match and token-level F1, $r_{\text{qa}} = w_{\text{EM}} \cdot \text{EM} + w_{\text{F1}} \cdot \text{F1}$ with our default $(w_{\text{EM}}, w_{\text{F1}}) = (0.6, 0.4)$ (§A). This split is not innocuous. EM is a binary indicator that is 1 only on exact-answer match, so it provides no gradient on the many near-miss rollouts

Prompt for Source Attribution of Inserted Triples

```
You are a knowledge graph analyst. For each INSERT triple emitted by an
insertion policy, decide whether its information can be found in, or
reasonably derived from, the retrieved subgraph shown as context.

Rules:
- Mark from_context=true if the retrieved subgraph contains the same
  information as the INSERT triple, or if the INSERT can be obtained by
  a straightforward paraphrase (e.g. "directed by" vs "director"),
  a case/format variant, or a one-hop derivation from an adjacent
  triple in the subgraph.
- Mark from_context=false only if the INSERT introduces information
  that is not present in the subgraph in any equivalent form. Such
  triples reflect the policy's parametric world knowledge rather than
  restatement of the retrieved context.

Output strict JSON only.

Attribute each INSERT triple to its source.

## Retrieved subgraph (indexed triples)
  ({s_0}, {r_0}, {o_0})
  ({s_1}, {r_1}, {o_1})
  ...
  ({s_K}, {r_K}, {o_K})

## Inserted triples
  ({s'_0}, {r'_0}, {o'_0})
  ({s'_1}, {r'_1}, {o'_1})
  ...

## Output
Strict JSON array only, one element per inserted triple:
[
  {"triple": ["s", "r", "o"],
   "from_context": true or false,
   "reasoning": "one short sentence"},
  ...
]
```

Figure 13: Prompt used to attribute each INSERT triple to either the retrieved subgraph (*subgraph-inferable*) or the backbone's parameters (*parameter-originated*). All triples inserted for the same question are attributed in a single batched call, so the judge sees the full retrieved subgraph as one context.

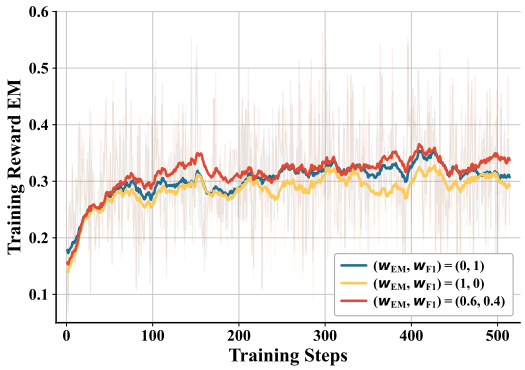

Figure 14: Training-time reward EM on HotpotQA for different reward blends.

that dominate hard multi-hop questions. F1 is continuous and rewards partial word overlap, so it densifies the signal but can also reward answer-adjacent phrasings that share tokens with the gold without ever committing to it. The two terms therefore correct opposite failure modes of each other, and how they trade off directly shapes what the Priming policy learns to insert.

To make the effect of this trade-off explicit, we ablate the two endpoints of the blend on Qwen2.5-7B-Instruct trained on HotpotQA under the same GRPO configuration as the main runs (Table 6). The two settings are $(w_{\text{EM}}, w_{\text{F1}}) = (0, 1)$, in which the reward is pure F1 and the policy is never explicitly told when an answer is exactly right, and $(w_{\text{EM}}, w_{\text{F1}}) = (1, 0)$, in which the reward is pure EM and every near-miss receives the same zero as an outright wrong answer. Both variants keep the format-shaping term (§A), the frozen answerer, and the retrieved subgraph identical to the default run, so any difference isolates the reward blend.

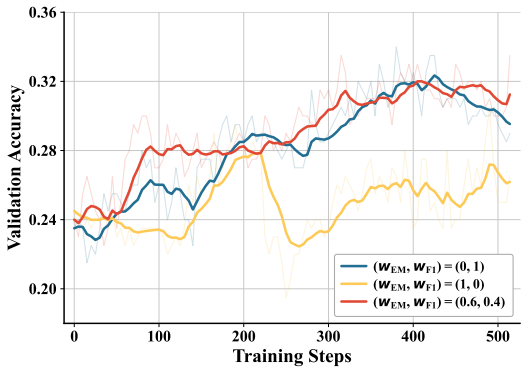

Figure 15: Validation accuracy on HotpotQA for different reward blends.

We report two diagnostics along the training trajectory. The first is the training-time reward EM, that is, the fraction of rollouts that hit gold exactly during GRPO updates, which reads out whether the policy is being pushed toward committed correct insertions or toward token-overlap satisficing. The second is validation accuracy under the same greedy-decoding, judge-EM protocol used elsewhere, which reads out whether that training-time signal translates into downstream answering gains.

Figure 14 shows that the mixed reward consistently yields the highest training-time reward EM across most of the trajectory. Pure F1 also improves exact-match rollouts, but it remains below the mixed reward and shows larger fluctuations. Pure EM is consistently worse, which supports the view that sparse exact-match feedback is insufficient for guiding Priming on hard multi-hop questions. The policy needs dense partial-credit feedback to discover useful bridging triples before it can reliably commit to exact answers.

The validation curve in Figure 15 shows the same trend at the downstream level. The mixed reward reaches the best final accuracy and maintains a more stable advantage in the later stage of training. Pure F1 is competitive in the middle of training, but its accuracy drops near the end, suggesting that token-overlap rewards can overfit to answer-adjacent insertions that do not consistently improve judged correctness. Pure EM underperforms throughout most of training, which indicates that exact-match supervision alone discards too many informative near-miss rollouts. These results justify our default convex blend: F1 provides the dense learning signal needed for exploration, while EM anchors the policy toward fully correct answer insertions.

I Additional Experiments

I.1 Knowledge Accessibility Ceiling via Pass@k

We measure the base model's knowledge accessibility ceiling under repeated stochastic sampling and compare it against VAKE's greedy, deterministic performance. The saturation point of pass@k as k grows is an empirical upper bound on what the base model can access without further training, so VAKE's greedy accuracy exceeding this ceiling is direct evidence of expanded accessibility.

Setup. We evaluate Qwen2.5-7B-Instruct on the 2WikiMultihopQA and HotpotQA test sets, 1,000 questions each, under triples-only and triples+documents settings. For each question we sample up to $K = 128$ responses at temperature $T = 1.0$ using the standard QA prompt, with no CoT and no triple insertion. We apply judge-based early stopping: at each round the same LLM judge used in the main evaluation decides semantic equivalence, and sampling stops for a question once a hit is declared. This cuts total compute by roughly $4\times$, since easy questions exit after 1–3 rounds, while preserving pass@k semantics.

We report two metrics. LLM-judge pass@k is the fraction of questions with at least one judge-correct answer among the first k samples. EM pass@k is a lower bound, because early stopping truncates sampling upon a judge-hit, so EM is computed only on the rounds actually executed.

Dataset	Setting	pass@1	pass@16	pass@32	pass@64	pass@96	pass@128
2Wiki	triples only	24.9	29.6	29.9	30.1	30.3	30.4
2Wiki	triples+docs	31.3	36.9	37.6	38.0	38.5	38.6
HotpotQA	triples only	32.3	37.3	38.4	38.6	39.0	39.2
HotpotQA	triples+docs	51.2	56.0	57.0	57.7	57.8	58.0

Table 13: LLM-judge pass@k (%) of the base model under stochastic sampling ($T = 1.0$, judge-based early stopping).

Dataset	Setting	pass@1	pass@32	pass@64	pass@128
2Wiki	triples only	23.5	28.1	28.2	28.2
2Wiki	triples+docs	27.8	32.5	32.6	32.6
HotpotQA	triples only	29.0	32.7	32.8	32.8
HotpotQA	triples+docs	44.4	49.1	49.2	49.2

Table 14: Normalized EM pass@k (%) of the same base model.

Table 13 reports judge pass@k, Table 14 reports the EM lower bounds, and Figure 16 shows the saturation behavior.

Accessibility ceiling. Base accuracy saturates early and exposes a hard accessibility ceiling. On 2Wiki with triples only, pass@k plateaus by $k = 64$–96, and pass@128 reaches 30.4% while gaining only 0.3 points over pass@64, so roughly 70% of questions stay inaccessible regardless of budget. The span from pass@1 to pass@128, 24.9% to 30.4% in this setting, marks a 5.5-point band of knowledge that is encoded and occasionally reachable but never reliably surfaced.

VAKE breaks through the ceiling under greedy decoding. VAKE reaches 36.2% on 2Wiki and 42.3% on HotpotQA with triples only (Table 1), exceeding the base pass@128 ceilings of 30.4% and 39.2% by 5.8 and 3.1 points. Because greedy decoding removes sampling variance, these gains cannot come from better sampling luck. They reflect genuinely expanded accessibility, closing the base model's reliability gap and extending past it.

External evidence raises the ceiling but does not close it. Adding documents lifts 2Wiki pass@128 from 30.4% to 38.6%, yet VAKE-P still reaches 41.3% on triples+docs (Table 5). Activation therefore goes beyond what external cues alone can trigger.

I.2 Perplexity Analysis of Parametric Belief

We ask whether VAKE changes the model's internal probability assigned to correct answers, beyond the externally observed accuracy gains. We compute the perplexity (PPL) of gold answer tokens under teacher forcing for both the base model and the VAKE checkpoint, given the same input.

Setup. We compare Qwen2.5-7B-Instruct as the base against the full VAKE checkpoint on all three ID test sets, excluding yes/no questions. Both models receive the same input, namely the question plus retrieved indexed triples.

We use two measurement formats:

- **Direct**: PPL of gold answer tokens immediately following the QA prompt.
- **CoT**: the gold answer is wrapped as `<answer>gold</answer>` after the CoT prompt, and PPL is measured only on the gold tokens inside the tags, excluding the tags and the reasoning prefix.

Per-sample PPL is

$$\mathrm{PPL}(a^\star) = \exp\left(-\frac{1}{|a^\star|} \sum_{t \in a^\star} \log P_\theta(x_t \mid x_{<t}) \right). \tag{14}$$

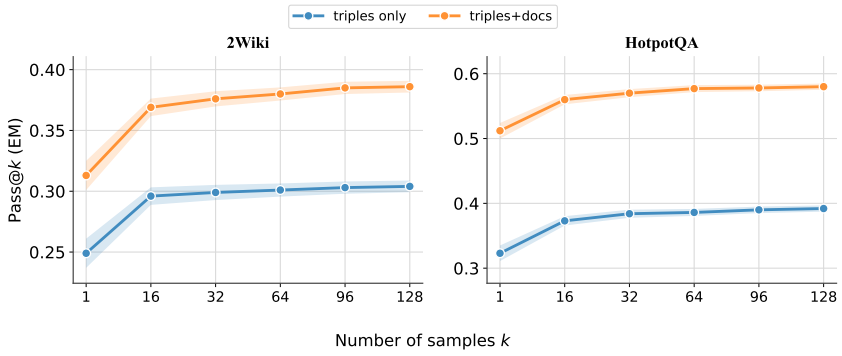

Figure 16: Pass@k (LLM-judge) of the base model versus sampling budget. T = triples only; T+D = triples + documents. All curves saturate by $k \approx 64$, establishing an accessibility ceiling that VAKE's greedy performance exceeds.

Dataset	Format	Base (med.)	VAKE (med.)	VAKE-lower %	n
2WikiMultihopQA	Direct	2.32	**1.72**	63.2%	728
2WikiMultihopQA	CoT	**2.15**	3.40	49.5%	749
HotpotQA	Direct	6.52	**4.08**	57.3%	762
HotpotQA	CoT	6.68	**3.20**	67.9%	748
MuSiQue	Direct	21.52	**18.41**	52.6%	380
MuSiQue	CoT	25.40	**17.20**	62.7%	381

Table 15: Gold answer perplexity (Qwen2.5-7B-Instruct, base vs. VAKE). Lower = higher confidence. Paired VAKE-lower % = fraction of paired samples where VAKE assigns lower PPL.

Answer tokens are located by character-offset matching between the tokenized prompt-plus-answer text and the gold span. We treat PPL ≥ 1000 as outliers and keep only paired samples that are valid under both models. We report median PPL and the paired win rate, the fraction of samples where VAKE PPL is lower than base PPL.

Systematic PPL reduction. As Table 15 shows, VAKE achieves lower median PPL in five of six settings. The reductions are largest on HotpotQA (-37% Direct, -52% CoT) and substantial on MuSiQue (-14% Direct, -32% CoT). Paired win rates exceed 50% in all but one cell, confirming that the shift is systematic at the sample level rather than driven by outliers. Training thus moves internal probability mass toward correct answers, not merely reshaping the output format.

The 2Wiki-CoT exception. The single reversal (2Wiki-CoT) arises because VAKE has learned to distribute probability across more diverse CoT paths before committing to an answer, raising conditional entropy at the measurement point. This does not contradict the accuracy gain: VAKE still outperforms Base on 2Wiki in Table 1, and the Direct-format measurement on the same data shows clear VAKE dominance.

Difficulty and accuracy correlation. Absolute PPL increases monotonically from 2Wiki through HotpotQA to MuSiQue, tracking both reasoning depth and the accuracy ordering. The strongest paired win rate (HotpotQA-CoT) coincides with the largest accuracy gain over Base ($+10.7$ points, Table 4), supporting the interpretation that accuracy improvements are downstream consequences of strengthened parametric belief.

Connection to pass@k. Pass@k (Section I.1) establishes a behavioral ceiling that VAKE breaks through, while PPL exposes the representational mechanism behind it: the model's belief distribution shifts toward correct answers. Together they confirm that VAKE's gains are parametric rather than byproducts of sampling diversity or format learning.

Anchor layer	L0	L6	L8	**L11 (peak)**	L16	L20	L27
Single layer (k=1)	0.067	0.356	0.422	**0.563**	0.504	0.341	0.059
3-layer window (k=3)	0.111	0.459	0.541	**0.593**	0.541	0.348	0.059

Table 16: Forward-patch flip-to-correct rate on the eligible subset (n=135; base wrong, VAKE-P right). Single anchor layer (k=1) vs. 3-layer window (k=3). The curve is a clean inverted-U peaking at L11.

I.3 Cross-Model Activation Patching (CMAP)

The pass@k and PPL analyses show that VAKE's gain is parametric. Here we localize it, asking where in the Priming policy training stores the capability that lets the downstream answerer succeed, and whether that capability can be transplanted. We answer both with a causal intervention, cross-model activation patching (CMAP), which copies a single layer's hidden state from the Priming policy (VAKE-P) into the untrained base policy and measures whether the base then inserts triples that let the frozen answerer answer correctly.

Method. For each question we run the base policy and the VAKE-P policy on the same prompt, consisting of the question, the retrieved subgraph, and the insertion instruction. During the base policy's forward pass we overwrite the residual stream at a single anchor layer L with VAKE-P's residual at the same layer and token positions, a forward patch from VAKE-P to base. We then let the base regenerate its INSERT actions, apply them to the subgraph, and hand the augmented subgraph to the frozen answerer, whose base weights are never patched. Two factors are held fixed: the input, which is the identical question and original subgraph for both policies, and the downstream answerer. Any change in the answer is therefore attributable to the transplanted layer alone. We scan $L = 0 \ldots 27$ to obtain a per-layer flip-to-correct curve.

The capability is concentrated in a single middle layer. The flip-to-correct curve is a clean inverted-U. Its ends (L0, L27) are near zero, while the L8–L16 band holds at 0.50–0.59 and peaks at 0.563 on L11, so transplanting one middle layer recovers 56% of the previously wrong samples. Widening to a 3-layer window raises the peak only to 0.593 and leaves it on L11, so the representation that decides whether a policy can produce an answerable subgraph sits in one mid-stack residual rather than a distributed multi-layer circuit.

The causal layer is not the layer that changed most, and it is necessary as well as sufficient. Weight differences between the base and VAKE-P concentrate in L20–L24, yet the causal peak sits at L11, so inspecting weight deltas alone would mislocate the mechanism. The reverse transplant, base to VAKE-P, confirms necessity. It destroys about 31% of VAKE-P's successes, again peaking mid-stack at about 0.31 on L12, drives the base policy to emit bridging-triple INSERTs at a VAKE-P-like rate, and its downstream gold-answer confidence peaks in the middle and decays at both ends.

A single transplanted layer approaches the full swap. Applied unconditionally to all 1000 samples, L11 lifts the base from 0.268 to 0.316, a 4.8-point gain that recovers about 83% of the whole-policy improvement without matching it. Repairs far outnumber breaks, 103 fixes against 55 breaks for a net of $+48$, and the failures are a faithful transport of behavior. Fixed samples are ones VAKE-P also answers correctly 74% of the time, against only 45% for broken ones, so the transplant carries VAKE-P's mistakes along with its skill, with yes/no questions the main negative-transfer regime at 22% of breaks against 11% globally.

Net gain is positive and approaches the full swap. L11 lifts the base from 0.268 to 0.316, a 4.8-point gain that recovers about 83% of the whole-policy improvement, $(0.316-0.268)/(0.326-0.268)$. A single transplanted layer approaches replacing the entire policy but does not match it.

Repairs far outnumber breaks. L11 fixes 103 samples and breaks 55, a net of $+48$, with the breaks concentrated on base-correct questions (20.5% of them). The failures are a faithful transport of behavior. Fixed samples are ones VAKE-P also answers correctly 74% of the time, whereas

Layer	em_norm	Δ vs base	C→C	C→W (break)	W→C (fix)	W→W	net
L6	0.288	+0.020	216	52	72	660	+20
L11	**0.316**	**+0.048**	213	55	103	629	**+48**
L16	0.295	+0.027	199	69	96	636	+27
L20	0.295	+0.027	225	43	70	662	+27

Table 17: Full-1000 single-layer patch: net accuracy and confusion vs. the base baseline (0.268). Reference: whole-policy swap = 0.326. C/W = correct/wrong before→after patching.

broken samples match VAKE-P only 45% of the time, so the transplant carries VAKE-P's mistakes along with its skill. Yes/no questions are the main negative-transfer regime, accounting for 22% of broken cases against 11% globally.

Case 1. Two-hop bridge: performer → place of death

Question. Where was the place of death of the performer of song "I Can't See Myself Leaving You"?
Gold answer. Detroit *(aliases: Detroit, Michigan; Motor City)*

Base — retrieved subgraph (excerpt).
- (aretha franklin, returned to the top 40 with, a rose is still a rose)
- (aretha franklin, appeared in, the blues brothers)
- (can t remember to forget you, is a, song)
- *no triple links the target song to a performer*
- *no triple mentions a place of death*

Base answer: "Cannot be determined."

Priming — inserted triples.
- (I Can't See Myself Leaving You, performed by, Aretha Franklin)
- (Aretha Franklin, died in, Detroit)

Priming answer: Detroit

Figure 17: **Two-hop bridge from parametric memory.** The retrieved subgraph mentions Aretha Franklin and the target song separately but never links them, and never states her place of death. The Priming policy inserts exactly the two triples the reasoning chain needs. Both are factually correct and neither is derivable from the retrieved subgraph. The frozen answerer, unchanged from the base run, then reads the augmented subgraph and produces the correct answer. The base failure is a *refusal*: without a visible chain the answerer defers, even though the underlying facts are recoverable from parameters.

Transferring the gain across input regimes. Finally we ask whether the L11 transplant improves the end-to-end VAKE-P(patch) pipeline, in which the base policy receives the transplant, inserts triples into the subgraph, and the frozen answerer produces the answer, across the three input regimes of Table 5. Table 18 reports semantic judge-EM on the full 1000 2Wiki samples, comparing the same-harness base baseline against the L11 patch under an identical HF-greedy judge harness, so the Δ isolates the transplant.

Input regime	Base baseline	L11 patch	Δ
Empty-insert	0.278	0.282	+0.004
Triples	0.279	0.328	**+0.049**
Triples+docs	0.329	0.377	**+0.048**

Table 18: End-to-end VAKE-P(patch) transfer across input regimes (2Wiki, full 1000, judge-EM). Δ = L11 patch − same-harness base baseline. The transplant helps only when a subgraph exists to augment with bridging triples.

The transplant yields a stable gain of about $+0.05$ whenever a subgraph is present, both under triples and triples+docs, but the effect is negligible under empty-insert at $+0.004$. This matches the mechanism exactly. VAKE-P's advantage lies in augmenting an existing subgraph, so under empty-insert, where both policies simply recall facts from scratch with no subgraph to modify, there is no insertion skill to transfer. The middle-layer representation is therefore not only causal but transferable. Grafting one layer of VAKE-P into an untrained policy recovers most of the downstream gain whenever the task admits triple insertion.

J Case Study

To make the mechanism behind the aggregate numbers concrete, we walk through cases on which the frozen Qwen2.5-7B answerer flips from wrong to right after Priming. All cases below are drawn from the 2WikiMultihopQA test split under the `triples_only` input condition, with the *same* frozen answerer in both the base and Priming rows. The only variable is whether the retrieved subgraph has been augmented by the Priming policy. We manually verified the inserted triples against external sources and report two representative cases in Figures 17–18. In every case shown, all triples inserted by the policy are factually correct, so the flip reflects genuine parametric knowledge activation rather than a lucky hallucination.

Case 2. Shared-property comparison: two rivers, one country

Question. Are both Cunaú River and Rio Verde (Mato Grosso Do Sul) located in the same country?
Gold answer. yes

Base — retrieved subgraph (excerpt).
- (cunaú river, is a, river)
- (rio verde (mato grosso do sul), is a, river)
- *no triple states either river's country*

Base answer: No

Priming — inserted triples.
- INS `(Cunaú River, is located in, Brazil)`
- INS `(Rio Verde (Mato Grosso do Sul), is located in, Brazil)`

Priming answer: Yes

Figure 18: **Independent attribute recall followed by delegated comparison.** With no country attribute visible for either river, the base answerer commits to the wrong polarity rather than deferring. Priming does not retrieve a chain; it recalls a *scalar attribute* for each entity independently, and hands the equality check back to the answerer. This is the division of labor Priming is designed to enable: the policy contributes atomic facts, the frozen answerer contributes the composition.

What the flips reveal about the mechanism. Across the two figures the base failures partition into two modes: outright refusal when no chain is visible (Case 1), and wrong-polarity commitment when the answerer fills the gap with a guess (Case 2). Priming resolves both modes through a single route, injecting the missing atomic facts as explicit triples so the frozen answerer, already competent at reading triples, can compose the answer without any parameter update. This matches the perplexity and pass@k analyses in §I.2 and §I.1. The parametric knowledge was in the backbone all along but was not surfaced under the base decoding distribution, and Priming's contribution is to make the facts the model already knows visible in the input.

"'latex "'

J.1 Reasoning-stage Case Study: With vs. Without Prior Priming

The cases above isolate what Priming contributes in the Priming stage itself. We next ask what *Priming* contributes to a downstream Reasoning-stage policy: does starting Reasoning-stage GRPO from a Priming checkpoint change the way the model reasons at test time, beyond the aggregate accuracy in Table 6's parent experiments? We fix the backbone (Qwen2.5-7B), the test set (HotpotQA, 1,000 questions), and the inference recipe (triples-only input, self-answered with chain-of-thought), and

Case 3. Relation disambiguation: fictional detective → creator

Question. The fictional private detective that appears in "The Adventure of the Seven Clocks" was written by whom?

Gold answer. Sir Arthur Conan Doyle *(aliases: Arthur Conan Doyle)*

CoT-only reasoning trace (verbatim): Adrian Conan Doyle
CoT-only answer: **Adrian Conan Doyle**

Priming→CoT reasoning trace (verbatim, condensed): Inserted triples:

- (The Adventure of the Seven Clocks, features, Sherlock Holmes)
- (Sherlock Holmes, created by, Sir Arthur Conan Doyle)

Step 1: Identify the fictional private detective in "The Adventure of the Seven Clocks." The detective is Sherlock Holmes.

Step 2: Resolve the target relation in the question. The question asks who created the detective character, rather than who authored the specific story.

Step 3: Identify the creator of Sherlock Holmes. Sherlock Holmes was created by Sir Arthur Conan Doyle.

Final answer: `<answer>Sir Arthur Conan Doyle</answer>`
Priming→CoT answer: **Sir Arthur Conan Doyle**

Figure 19: **Priming→CoT resolves the target relation.** CoT-only returns Adrian Conan Doyle, a coauthor of the specific story, rather than the creator of the detective character. Priming→CoT makes the intended chain explicit by identifying Sherlock Holmes and linking the character to Sir Arthur Conan Doyle. Both variants use the same Reasoning-stage optimization; the difference is whether Priming initialization encourages explicit resolution of the intermediate entity and relation.

compare two training regimes that differ only in whether Priming was applied before Reasoning-stage GRPO: **CoT-only** (Reasoning-stage GRPO trained from the base model) and **Priming→CoT** (Reasoning-stage GRPO warm-started from the Priming checkpoint). Under EM, CoT-only reaches 28.9% and Priming→CoT reaches 40.9%. Of the $1{,}000$ questions, 156 (15.6%) flip from CoT-only-wrong to Priming→CoT-right. We report one representative flip in Figures 19.

From explicit triples to implicit reasoning. Figure 19 illustrates a relation-selection error that Priming helps correct. CoT-only associates the question with an author of the specific story, but fails to resolve the intended chain from the story to its detective character and then to the character's creator. Priming→CoT makes this chain explicit, first identifying Sherlock Holmes and then Sir Arthur Conan Doyle. Under identical Reasoning-stage data and hyperparameters, Priming initialization encourages the policy to resolve intermediate entities and relations before committing to an answer. This behavior transfers from structured triple insertion to free-form reasoning, improving CoT accuracy from 28.9% to 40.9%.