POLICY-INTEGRATED QUERY ENCODING FOR FIXED-INDEX AGENTIC RAG: FEASIBILITY AND LIMITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Dense-retrieval agentic RAG usually runs two neural models at inference time: a policy language model that writes search queries, and a separate dense encoder that embeds every query for the passage index. We ask whether the policy's own search-token representations can act as the query-side interface to a fixed passage index, so that the external encoder is not needed online. We mean-pool the search-token hidden states, map them into the passage-embedding space with a 6.55M-parameter linear projection, retrieve a top-M candidate set with the projected query, define a stochastic policy over that candidate set, and optimize token and selected-passage log-probabilities jointly under the same final-answer reward. We refer to this architecture as policy-integrated query encoding (PIQE). PIQE removes the external 4B query encoder from the online retrieval path while the corpus index stays fixed. Under a controlled comparison with a matched Search-R1 baseline over three seeds, PIQE reaches 36.11 ± 0.34 macro and 42.36 micro exact match on the frozen reporting partition against 36.77 ± 0.72 and 43.16 for the baseline, and 56.55 against 57.17 micro exact match out of domain. PIQE is within one exact-match point of the baseline on each aggregate, but remains below it on average. Retrieval diagnostics associate the deficit with candidate generation: the hidden-state interface puts an answer-containing passage into the candidate set less often than the text-query interface (Pool@20 45.67 against 47.14), and passage-level credit assignment does not compensate through better selection inside the candidate set. PIQE is thus architecturally feasible but not lossless, and the results motivate learning signals that improve candidate generation without establishing that better selection could not close the gap.

1 INTRODUCTION

Agentic retrieval-augmented generation systems interleave reasoning with search: the policy language model writes a search query, a retriever returns passages, and the model continues from what it has read (Jin et al., 2025; Chen et al., 2025; Xiong et al., 2025; Li et al., 2025; Zheng et al., 2025; Yao et al., 2023). In the dense-retrieval configuration studied here, this pipeline runs two neural models online. The policy emits a search string, an external query encoder embeds that string, and the embedding is matched against a fixed passage index. To isolate the retrieval interface from a backbone-family mismatch, we study a matched Qwen3-based Search-R1 configuration in which Qwen3-4B-Instruct-2507 is the policy and Qwen3-Embedding-4B is the query encoder, so that each search invokes two separate 4B networks; the original implementation instead retrieves with the much smaller e5-base-v2 encoder (Wang et al., 2022), and the symmetry is a property of our setup rather than of agentic RAG in general.

The separate online encoding pass may be avoidable. While generating the search tokens, the policy computes contextual representations of those tokens, conditioned on the question, on its own prior reasoning, and on the evidence it has read so far. The external encoder, in contrast, sees only the surface string. This motivates the question we study: *can the policy provide the query-side interface to a fixed passage index by itself?* If it can, the retrieval interface collapses into the policy, and the only remaining retrieval component at inference is the fixed index itself. Our goal is not to demonstrate an optimized serving implementation, but to isolate whether the policy representation can function

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

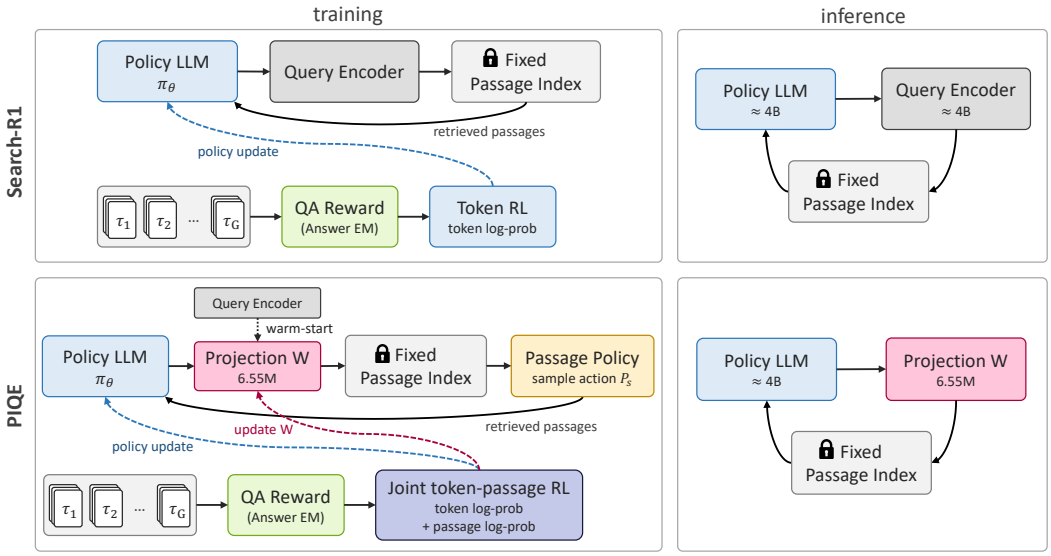

Figure 1: Training and inference interfaces of Search-R1 and PIQE. Search-R1 embeds a generated search string with an external query encoder and optimizes token actions. PIQE replaces that encoder with a projection of policy hidden states and jointly optimizes token and passage actions. Both use the same fixed passage index.

as the query-side interface to a fixed passage index.[1] Producing retrieval embeddings inside the reasoning model is not new: Search-R3 trains an LLM to emit them from its own hidden states under retrieval-level supervision, updating document embeddings as it goes (Gui & Cheng, 2025). Our question is more constrained, in that an existing agentic policy reuses a corpus representation it never updates and, after a warm-start, learns from its final-answer reward alone (Section 2).

Integration is not trivial. Search-token hidden states are not naturally aligned with the fixed passage-embedding space and may shift as reinforcement learning updates the actor, so we let the query interface co-adapt rather than freezing it after warm-start; and because candidate generation is a discrete nearest-neighbour operation, any signal defined after retrieval can redistribute credit among retrieved passages but cannot directly reward passages outside the candidate set.

We study this question with the architecture in Figure 1, which we call PIQE, for policy-integrated query encoding. At each search step the hidden states of the generated search tokens are mean-pooled and mapped by a single 2560×2560 linear projection into the passage-embedding space; the fixed index returns the top-M candidates for that query, and a stochastic policy samples the k passages inserted into the context. The projection is adapted by joint token–passage reinforcement learning under a single final-answer reward, which is the adaptation mechanism for the integrated interface rather than the object of study. The external encoder is still used offline to encode the corpus and as the warm-start teacher, but is not called during inference.

The answer is partial: removing the online query encoder produces mean decreases of less than one exact-match point on each reported aggregate. Over three seeds the integrated system reaches 36.11 ± 0.34 macro exact match on the frozen reporting partition against 36.77 ± 0.72 for Search-R1, a mean cost of 0.65 macro and 0.80 micro points in domain and 0.61 micro points out of domain. The difference is not uniform across seeds, but its mean favours the baseline in every aggregate, so we report the integrated system as close to, but below, the two-model architecture.

Retrieval diagnostics associate the remaining gap with candidate generation: the hidden-state interface puts an answer-containing passage into the candidate set less often than the text query (Pool@20 45.67 against 47.14) and selection inside the candidate set does not offset that loss. A passage policy

[1] Our prototype recovers the search-token hidden states with an additional policy forward pass, so the experiments establish architectural feasibility rather than realized latency or compute savings (Section 3.5).

can only redistribute credit among candidates that the truncated retrieval step has already returned, and under the training configuration and budget we evaluate it does not remove the deficit.

Our contributions are the following.

- We study an architectural question, whether an agentic RAG policy can absorb the online query-encoding function of an external dense retriever using its own search-token hidden states, and instantiate it as policy-integrated query encoding over a fixed index, stating precisely what is and is not unified.
- We show, in a controlled comparison with a matched Search-R1 baseline, that removing the external online query encoder retains most of the baseline accuracy (mean decreases of 0.80 in-domain and 0.61 out-of-domain micro exact-match points) while remaining below it on average: the replacement is feasible, but it is not lossless.
- We associate the remaining gap with candidate generation rather than with selection, using a retrieval decomposition, factorial controls, a headroom diagnostic, and recovery controls, and show that within-candidate passage-level credit assignment does not remove it under the training configuration and budget we evaluate.

2 Related Work

Agentic RAG with reinforcement learning. Search-R1 trains a policy to interleave reasoning, search-query generation, and answering, inserting retrieved passages as observations and learning from a final-answer reward (Jin et al., 2025); related systems vary the trajectory formulation, add process supervision, or use multi-reward schemes (Chen et al., 2025; Xiong et al., 2025; Li et al., 2025; Zhao et al., 2025; Zheng et al., 2025). In these systems the policy decides when to search and what text to search for, and in a dense-retrieval configuration a separate encoder turns that text into a query vector at every search step. Our question concerns the step after query generation: whether that online encoding can be carried out by the policy's own representations instead.

Adapting retrieval to downstream utility. A second line makes retrieval serve the generator rather than generic relevance. RA-DIT fine-tunes the language model to use retrieved evidence and separately updates the retriever towards what the model prefers (Lin et al., 2024); reinforced contrastive learning optimizes a RAG retriever from downstream signal (Zhou & Chen, 2025); JSA-RAG treats retrieved passages as discrete latent variables in end-to-end training (Cao et al., 2025); and query rewriting improves the text of the query before retrieval (Wang et al., 2025; Chan et al., 2024). Closest to our setting, Agentic-R trains a retriever specifically for agentic search, measuring passage utility by both query–passage relevance and final-answer correctness and optimizing the agent and the retriever iteratively (Liu et al., 2026). These methods keep the two-model division of labor and make the external retriever better; we instead ask what remains when the online query encoder is removed from it.

Embeddings produced by the model itself. Search-R3 is the closest architectural precedent (Gui & Cheng, 2025). It adapts an LLM to emit a search embedding from the hidden state of a dedicated embedding token at the end of its reasoning. A supervised stage combines language-modelling, KL regularization, InfoNCE, and triplet-margin losses, followed by GRPO with a scaled discounted cumulative-gain retrieval reward. Document embeddings are pre-computed with the same model and selectively updated during training, and the method is evaluated with ranking metrics on retrieval benchmarks. The question we ask is complementary and more constrained. Rather than training the policy and corpus representation as a new embedding model, PIQE retrofits an existing multi-step agentic RAG policy to a fixed external passage space: corpus vectors and the index are produced once by an external encoder and never updated, the query interface is a single linear projection over search-token states, and learning uses the binary final-answer reward rather than a retrieval reward. What we report is therefore not a stronger embedding model, but how far this minimal retrofit gets under those constraints and where it falls short. Training only the query side against fixed passage representations is a standard device in dense retrieval (Zhan et al., 2021; 2020) and is what makes the retrofit feasible over a 21M-passage index; Section D covers the remaining related work.

3 METHOD

3.1 AGENTIC RAG WITH AN EXTERNAL QUERY ENCODER

We consider agentic question answering in which a policy language model interleaves reasoning, search, and answer generation. Given a question x, the model produces an answer y through a trajectory $\tau = (o_0, u_1, \ldots, u_L)$, where $o_0 = x$ and each u_ℓ is either a generated token action a_t^{tok} or a retrieval-action/observation pair (a_s^{ret}, o_s). Search-query tokens are placed between `<search>` and `</search>`, retrieved passages are inserted between `<information>` and `</information>`, and answer tokens between `<answer>` and `</answer>` (Jin et al., 2025).

In the standard configuration, the retrieval step at search step s runs a second neural model. The policy generates a search string r_s, an external query encoder E_q embeds it, and the fixed index returns the candidate set $\mathcal{C}_s^{\text{text}} = \text{TopM}(E_q(r_s), \mathcal{D})$, where $\mathcal{D} = \{d_i\}_{i=1}^N$ is the corpus and each passage d_i has a fixed embedding e_i computed offline by the passage encoder. The top-k candidates are inserted as observations, and both E_q and the policy are called for every search.

3.2 POLICY-INTEGRATED QUERY ENCODING

The alternative studied here removes E_q from the inference path. Let $H_s = (h_{s,1}, \ldots, h_{s,m_s})$ denote the hidden states associated with the m_s generated search tokens. A trainable projection maps their mean into the passage-embedding space,

$$q_s = \text{Normalize}(W\,\text{MeanPool}(H_s))\,, \qquad \mathcal{C}_s = \text{TopM}(q_s, \mathcal{D})\,, \tag{1}$$

with $W \in \mathbb{R}^{2560\times 2560}$ a single linear map without bias and $M = 20$ candidates retrieved by exact inner-product search.

Given the candidate set, the score of a candidate and the passage policy are

$$z_{s,i} = \frac{q_s^\top e_i}{\tau_{\text{temp}}}, \qquad \pi_\phi^{\text{ret}}(d_i \mid H_s, \mathcal{C}_s) = \frac{\exp(z_{s,i})}{\sum_{d_j\in\mathcal{C}_s}\exp(z_{s,j})}, \quad d_i \in \mathcal{C}_s, \tag{2}$$

where τ_{temp} is a temperature. The retrieval action a_s^{ret} is an ordered list $P_s = (d_{s,1}, \ldots, d_{s,k})$ of $k = 5$ passages sampled without replacement from this distribution, and we record its ordered log-probability $\log \pi_\phi^{\text{ret}}(P_s \mid H_s, \mathcal{C}_s)$. Sampling is used during training rollouts; at evaluation time the k highest-scoring candidates are selected deterministically. Because the same score $q_s^\top e_i$ ranks the index and the candidate set, this deterministic selection returns the top k passages of the index for q_s, and Pool@20 and the conditional selection rate of Section 4.4 are two cutoffs of one ranking rather than two independent scorers. Selected passages are inserted in list order. The text inside `<information>` is an observation and is masked from the token-generation loss; what the policy contributes is the probability of having selected that information.

Optimization Boundary. Because top-M retrieval is a discrete, fixed-index, stop-gradient truncation, the passage-action likelihood assigns credit only among candidates already in $\mathcal{C}_s$; passages outside it receive no direct gradient at that retrieval event. Updates to the policy or the projection may change future candidate sets, but the current event provides no direct comparison between included and excluded passages (Section E.2), which is what makes the distinction between candidate generation and within-candidate selection an empirical question rather than a definitional one, measured separately in Section 6.

3.3 REWARD AND JOINT OBJECTIVE

The reward depends only on final-answer correctness. Following Search-R1 (Jin et al., 2025), we extract the last `<answer>` span and use binary exact match after standard answer normalization, $R(\tau) = \text{EM}(y, \mathcal{Y}^*) \in \{0, 1\}$, where $\mathcal{Y}^*$ is the set of gold answer strings. There is no format or process reward, and every system compared here uses the same reward.

A projection that is merely distilled from the text encoder inherits that encoder's behavior rather than the policy's, so we optimize token and passage-action log-probabilities together with a GRPO-style

group-normalized policy gradient (Shao et al., 2024; Schulman et al., 2017; Williams, 1992):

$$\mathcal{L}_{\mathrm{RL}}(\theta, \phi) = -\mathbb{E}_{\tau}\Big[A(\tau)\Big(\sum_{t} \log \pi_{\theta}^{\mathrm{tok}}(a_t^{\mathrm{tok}} \mid c_t) + \alpha \sum_{s} \log \pi_{\phi}^{\mathrm{ret}}(P_s \mid H_s, \mathcal{C}_s)\Big)\Big], \tag{3}$$

where c_t is the language-model context at token step t, θ denotes language-model parameters, ϕ the projection, $A(\tau)$ the group-relative advantage, and α the weight of the passage-action term. The implementation uses PPO-style clipped ratios and a token-policy KL penalty; the clipped passage surrogate is written out in Section E.5 and the remaining settings are listed in Section F.

Gradient flow. In the default configuration H_s is not detached when the passage log-probability is computed, so the gradient of $\log \pi_{\phi}^{\mathrm{ret}}$ updates both the projection and the upstream search-state representation in θ. The top-M operator, the passage embeddings, the sampled passage identities, and the inserted `<information>` text are stop-gradient or non-differentiable.

3.4 Projection Warm-Start

A randomly initialized projection would retrieve unrelated candidates and receive almost no learning signal from a sparse answer reward, so the projection is warm-started before RL by distilling the external encoder on the search events of 65,536 rollouts of the initial policy. The objective aligns projected hidden states with the teacher query embeddings and candidate rankings. Its two-stage training procedure and exploratory variants are given in Section F.2.

3.5 Scope of the Integration and of the Implementation

PIQE integrates into the policy the online query representation, the scoring of candidates, and the selection of passages, which is an action optimized by the same reward as token generation. It does not integrate the offline encoding of the corpus, the fixed index that both architectures share, or the teacher that warm-starts the projection, all of which use the external encoder; the corpus vectors stay in that teacher's embedding space, which is what allows the index to stay fixed. The diagnostic systems of Section 6.4 restore the text-query ranking and therefore call the encoder at inference by construction.

The implementation is equally partial: vLLM 0.8.5 does not expose the hidden states of generated tokens, so our prototype recovers H_s with an additional policy forward pass over the generated prefix. PIQE thus removes a separate online model from the inference path, but the prototype demonstrates no wall-clock speedup and we make no latency or throughput claim.

4 Experimental Setup

4.1 Primary Systems

The main comparison contains two systems, because only these two differ in the architectural property under study.

Search-R1 is the matched baseline (Jin et al., 2025): token-only RL in which the generated search string is embedded online and the top-5 passages from the fixed index are inserted as observations. We run it with Qwen3-Embedding-4B rather than the e5-base-v2 retriever of the original implementation, so that the baseline and the integrated system share one passage encoder, one embedding space, and one index. Sharing the index makes the comparison a test of the query interface rather than of the encoder family; it is not a deployment study, and a smaller external encoder may offer a preferable trade-off that we do not measure.

PIQE (policy-integrated query encoding) is the integrated system: the projection of Equation (1) produces the query, the fixed index returns the top-20 candidates, the passage policy selects all five inserted passages, and token and passage log-probabilities are optimized jointly.

Both systems start from Qwen3-4B-Instruct-2507, use the same prompt and special-token protocol, allow at most four search steps per trajectory, insert $k = 5$ passages per search, query the same exact inner-product index of wiki-18 encoded with Qwen3-Embedding-4B, and share the token-RL component together with the same rollout configuration, learning rate, KL coefficient, reward, budget,

and checkpoint-selection rule. They differ in the query interface and in the additional passage-action objective and training-time passage selection of PIQE. PIQE adds a passage-action weight $\alpha = 0.03$, an annealed retrieval temperature, and a separate projection learning rate (Table 12); Section G reports the sweeps behind these choices.

4.2 Diagnostic Controls

Four diagnostic systems appear only in the analysis, because each reintroduces something the integrated architecture is meant to remove or holds one factor fixed. **Text-query passage RL** keeps the external text query and adds the passage-action term, so the passage log-probability updates an identity-initialized projection of that text query and not the language model, which isolates passage-level credit from the hidden-state interface. **Frozen projection** is PIQE with the projection held at its warm-started state while the actor is still updated by token-only RL, which removes passage-loss-based adaptation of the projection. **PIQE + RRF** fuses the text and projection rankings at inference from the PIQE checkpoints, and **Hybrid** is a separate training run that fixes the text retriever's top two passages and lets the policy select the other three, so its actor differs from the PIQE actor. Both of the last two call the external encoder online and do not satisfy the integration objective, but they show how much of the accuracy gap that ranking accounts for; Section F gives their settings.

4.3 Data, Index, and Protocol

We train on 125,385 questions from the processed training splits of HotpotQA, 2WikiMultiHopQA, and MuSiQue in the distribution of Table 10 (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022), converted to the Search-R1 prompt schema with no further filtering, subsampling, or deduplication after download; Table 11 gives the counts per dataset. The retrieval corpus is the 2018 Wikipedia dump used by Search-R1, with 21,015,324 passages (Karpukhin et al., 2020; Jin et al., 2025), encoded once and left unchanged.

The in-domain evaluation pool is the union of the official development sets, 22,398 questions, split once, before the experiments reported here, into fixed partitions identified by question ID: Dev1200 with 400 questions per dataset for checkpoint and configuration selection, a frozen reporting partition of 11,262 questions for reporting, and a reserve partition for development diagnostics. The reporting partition is not unseen: 11,197 of its 11,262 questions appear in exploratory evaluations run before the partition was frozen, and after freezing it was used only to report selected checkpoints (Section J.1). Out-of-domain evaluation uses the test splits of Natural Questions, TriviaQA, and Bamboogle, 15,048 questions in total (Kwiatkowski et al., 2019; Joshi et al., 2017; Press et al., 2023).

Every run of the main comparison uses 200 optimizer steps and seeds 13, 21, and 42; diagnostics and exploratory runs use single seeds or the earlier budget (Section G). Dev1200 is evaluated at step 0 and every 10 steps with greedy decoding and deterministic retrieval, checkpoints are saved every 10 steps, and the selected checkpoint is the saved checkpoint between step 50 and step 200 with the highest Dev1200 macro exact match, ties going to the earlier step. The frozen reporting partition and the out-of-domain sets are then evaluated for that checkpoint.

4.4 Metrics

For answer quality we report exact match after standard answer normalization: in domain the macro average over the three datasets and the micro average over all questions, out of domain the micro average over all 15,048 questions, since a simple macro average would give Bamboogle, which is 0.83% of the questions, one third of the weight. Paired comparisons between systems evaluated on the same questions use McNemar's exact test. Tables abbreviate the frozen reporting partition as ID and out-of-domain as OOD.

For retrieval we decompose each search event, labelling a candidate as positive when it contains a gold answer string. *Pool@20* is the fraction of events whose candidate set holds a positive and measures candidate generation; *AC@5* is the fraction whose five selected passages hold one in full text, before the observation limit truncates the block, and so measures selection rather than what the policy reads; and the conditional quantity $P(\mathrm{AC@5} \mid \mathrm{Pool@20})$ measures selection inside the candidate set. These labels are a proxy for supporting evidence, so we speak of answer-containing

Table 1: Online query-side architecture and answer accuracy of Search-R1 and PIQE, averaged over seeds 13, 21, and 42 (standard deviations in parentheses). PIQE replaces the external 4B online query encoder with a 6.55M-parameter projection and stays within one exact-match point of the baseline on every aggregate, below it in each case.

Method	Online query encoder	Online query-side parameters	ID macro EM	ID micro EM	OOD micro EM
Search-R1	Qwen3-Embedding-4B	≈4B	**36.77** (0.72)	**43.16** (0.79)	**57.17** (0.26)
PIQE	none	6.55M	36.11 (0.34)	42.36 (0.22)	56.55 (0.08)

Table 2: Seed-level accuracy of the two primary systems, with the paired difference PIQE − Search-R1 on the same questions and McNemar's exact test. The difference is not uniformly directional: it is detectable in two of three seeds in domain, and in one of three out of domain.

	Seed (ID macro / ID micro / OOD micro)		
Method	13	21	42
Search-R1	37.31 / 43.57 / 57.10	37.04 / 43.66 / 57.46	35.95 / 42.26 / 56.94
PIQE	35.73 / 42.12 / 56.64	36.23 / 42.52 / 56.52	36.39 / 42.45 / 56.50
Paired Δ, ID micro	-1.46 ($p = 0.0002$)	-1.14 ($p = 0.0030$)	$+0.20$ ($p = 0.63$)
Paired Δ, OOD micro	-0.46 ($p = 0.087$)	-0.94 ($p = 0.0006$)	-0.45 ($p = 0.10$)

retrieval rather than of gold evidence; Section B defines the auxiliary columns and states how events are counted. All exact-match and retrieval quantities are percentages, and differences between them are percentage points. For architecture we report whether an external online query encoder is required and the number of query-side neural parameters it adds at inference.

5 How Much Accuracy Survives the Replacement?

5.1 Accuracy Retained After Removing the Query Encoder

Producing the query from policy hidden states, rather than from a separate online query encoder, costs 0.65 macro and 0.80 micro exact-match points in domain and 0.61 micro points out of domain, in each case in favour of the two-model architecture; the decreases are numerically within one point on all three aggregates. Token-level F1 gives the same ordering and gaps of comparable size: PIQE trails the baseline by 0.74 macro and 0.68 micro points in domain and by 0.81 micro points out of domain (Tables 5, 6). We do not claim equivalence or non-inferiority, neither of which we have tested; the parameter column of Table 1 counts what each system runs online beside the shared policy, index, and offline corpus encoding, not measured inference cost.

The seed-level view in Table 2 qualifies this: in domain the integrated system is below the baseline for seeds 13 and 21 and above it for seed 42, so the in-domain difference is not uniformly directional even though its mean favours the baseline, while out of domain it is lower on all three seeds. The paired tests follow the same pattern: the in-domain micro difference is detectable on seeds 13 and 21 (-1.46, $p = 0.0002$; -1.14, $p = 0.0030$) and not detectable on seed 42 ($+0.20$, $p = 0.63$). The largest of these paired gaps, 1.46 micro points, is comparable to the 1.40-point spread of the baseline's own micro exact match across seeds (42.26 to 43.66). Out of domain the decrease in mean micro exact match likewise stays below one point, so a projection warm-started on in-domain rollouts does not collapse when the question distribution changes.

6 Why Does Policy-Integrated Retrieval Fall Short?

6.1 Candidate Generation and Evidence Acquisition

Table 3 separates the stages of the pipeline over the 23,403–25,806 search events logged per run for these two systems, and it shows that the answer-accuracy gap is accompanied by a retrieval gap. The two systems are different actors, so the decomposition localizes where the retrieval difference

Table 3: Retrieval decomposition of Search-R1 and PIQE on the in-domain reporting partition, averaged over three seeds (standard deviations in parentheses) and measured on the logged search events; the columns are defined in Section 4.4. The retrieval deficit of PIQE is already visible at candidate generation and is not offset by selection inside the candidate set.

Method	Pool@20 (generation)	AC@5 | Pool@20 (selection)	AC@5 (selected)	First rank (positives)	Searches per question	Micro EM
Search-R1	**47.14** (0.47)	**82.31** (0.28)	**38.80** (0.48)	**3.29** (0.01)	2.28	**43.16**
PIQE	45.67 (0.39)	81.43 (1.03)	37.19 (0.57)	3.39 (0.11)	2.18	42.36

Table 4: Factorial diagnostic on seed 13 under the 200-step budget; single seed, exploratory. Within these four configurations, accuracy is lower in the two that use the hidden-state query, and not in the one that only adds passage-level credit; the conditions differ in more than the two factors, as Section G details.

Hidden-state query	Passage RL	Condition	Dev1200 macro EM	ID macro EM	OOD micro EM
No	No	Search-R1	37.00	**37.31**	57.10
No	Yes	Text-query passage RL	36.75	37.08	**58.09**
Yes	No	Frozen projection	34.58	35.76	56.45
Yes	Yes	PIQE	36.17	35.73	56.64

appears rather than isolating a single cause, and it does not separate downstream use of the evidence from retrieval itself.

The loss appears at candidate generation. The projected query puts an answer-containing passage into the candidate set less often than the text query (Pool@20 45.67 against 47.14), and when such a passage is present it sits slightly lower in the candidate ranking (first rank 3.39 against 3.29). Selection inside the candidate set does not compensate: the integrated system keeps a present positive slightly less often than the baseline (81.43 against 82.31), so the deficit propagates to the passages it selects (AC@5 37.19 against 38.80). The mean number of searches per question is similar (2.18 against 2.28).

Passage-level credit assignment operates strictly inside $\mathcal{C}_s$, and by the optimization boundary of Section 3.2 it cannot promote a passage that the truncated retrieval step did not return. The passage objective therefore gives no direct comparison with excluded passages; updates can still change later candidate sets, but under this configuration and budget the coverage deficit remains.

6.2 Is the Deficit Caused by the Hidden-State Query or by Passage-Level RL?

Table 4 compares four configurations on seed 13, which we read as a comparison of configurations rather than as an isolation of causes. Adding passage-level credit while keeping the text query stays close to Search-R1 in domain (37.08 against 37.31 macro exact match; paired micro difference -0.45, $p = 0.22$) and is the strongest of the four conditions out of domain (58.09). Replacing the text query with the hidden-state projection instead costs accuracy whether the projection is frozen or adapted (35.76 and 35.73; paired micro differences -1.25, $p = 0.0013$ and -1.46, $p = 0.0002$). Adaptation improves Dev1200, the split on which checkpoints are selected (36.17 against 34.58), but gives no visible benefit on the reporting partition (35.73 against 35.76; paired micro difference -0.20, $p = 0.61$).

6.3 How Much Headroom Is There Inside the Candidate Set?

If within-candidate selection were the binding constraint, a much better selector should be worth a large amount of accuracy. Replacing the baseline's selection with one that maximizes gold supporting-document title coverage inside its own candidate sets gains 1.78 macro and 1.80 micro exact-match points on average over the three seeds, between $+1.58$ and $+1.98$ macro points in every seed and with $p < 0.0001$ on seed 13 (Table 7). That is the gain of one gold-label-informed rule rather than a bound over selection policies, and it exceeds the 0.80-point micro gap between

PIQE and the baseline, so selection is not a negligible lever; but it is measured on the baseline's candidate sets, which hold an answer-containing passage more often than PIQE's (Table 3), so the same rule need not transfer.

6.4 Recovering Accuracy by Reintroducing the External Ranking

PIQE + RRF applies reciprocal rank fusion to the PIQE checkpoints at inference, while Hybrid is a separate training run whose actor differs from the PIQE actor, so its accuracy difference is not the effect of a ranking change alone. PIQE + RRF improves mean in-domain accuracy relative to PIQE (36.85 against 36.11 macro and 43.28 against 42.36 micro), slightly exceeding Search-R1 numerically on both aggregates (36.77 macro and 43.16 micro), and also recovers candidate coverage (Pool@20 46.71 against 45.67; AC@5 38.01 against 37.19); Hybrid does not (35.90 macro and 41.94 micro). Out of domain, PIQE + RRF improves micro exact match over the baseline in all three seeds (57.94 ± 0.20 against 57.17 ± 0.26), but the seed-13 attribution controls associate most of that gain with the PIQE actor rather than with the trained projection: with the PIQE actor and the same fusion, replacing the RL-trained projection by the warm-started one costs 0.24 points (57.99 against 57.75), whereas replacing the baseline actor by the PIQE actor while the projection stays at its warm start adds 0.73 (57.02 against 57.75). Per-seed values and the attribution controls are in Table 8, Section C.

Both use the external query encoder that the integrated system removes, and both improve out-of-domain micro exact match over PIQE, while only PIQE + RRF also improves the in-domain aggregates; they support the diagnosis rather than the architecture, and we do not present them as instances of policy-integrated retrieval.

7 Limitations

Our evidence is limited to one policy family, one passage-embedding space, and one fixed corpus and index, so how far a policy can absorb query encoding may depend on how that space was trained and on the size of the policy relative to the encoder. We evaluate online model composition and answer accuracy; because the prototype recovers hidden states with an additional policy forward pass (Section 3.5), latency and computation are left to a single-pass study, and we do not compare against a smaller external encoder. In the main comparison, training uses a sparse binary final-answer reward, a candidate set truncated at $M = 20$, and a 200-step budget; M and the budget vary only in the exploratory single-seed runs of Section G, and two of the three PIQE seeds are still improving on the selection split at the end of the budget (Figure 2). Some diagnostics are narrower than the three-seed main comparison or rely on imperfect answer-containing and label-informed proxies. Finally, the frozen reporting partition is not a pristine unseen test set, because most of its questions were evaluated during exploratory development before the split was frozen.

8 Conclusion

We asked whether an agentic RAG policy can absorb the online query encoding of an external dense retriever using its own search-token hidden states, and it partially can: a 6.55M-parameter projection removes the external 4B encoder from the online retrieval path, with the corpus index unchanged, at mean costs of 0.80 in-domain and 0.61 out-of-domain micro exact-match points against a matched Search-R1 baseline.

The replacement is not lossless, and the diagnostics point at one stage: the hidden-state interface generates candidate sets that contain an answer-containing passage less often than the text-query interface, and passage-level credit did not recover that loss, since it applies only to passages the truncated retrieval step returned. Controls that restore the external ranking regain part of the accuracy, at the cost of the architecture the paper set out to test, so whether the trade-off is worthwhile depends on the value of removing an online 4B model from a deployment. Our results point to better alignment between policy representations and candidate generation, without ruling out better selection.

AI USE STATEMENT

Generative AI tools were used to assist with brainstorming and refining the framing of the work, providing feedback on experimental methodology, interpreting and presenting experimental results, identifying and summarizing relevant literature, organizing the manuscript, and drafting, translating, and editing text. They were also used to assist with the development and revision of research code and analysis scripts. The authors made all final decisions regarding the research questions, methodology, experiments, interpretation, and claims. Citations were verified against the original sources, AI-assisted code was reviewed and tested, and reported results and numerical claims were verified against the corresponding experimental outputs. The authors reviewed all AI-assisted content and take full responsibility for the final manuscript.

ETHICS STATEMENT

This work uses publicly available question-answering datasets and a public Wikipedia snapshot and collects no new human-subject data; those public resources may themselves mention people, and we use them only as benchmark material. We see no specific ethical concerns beyond those common to retrieval-augmented language models, such as propagating errors or biases present in the corpus.

REPRODUCIBILITY STATEMENT

Answer-quality metrics are recomputed from stored per-question predictions; retrieval metrics are computed from the search-event logs of the same runs; wall-clock and hardware statistics come from run metadata. Section 4.1 and Section 4.3 describe the systems, data, index, fixed partitions, and checkpoint-selection protocol; Section 4.4 defines the metrics; Section F provides implementation details, hyperparameters, compute, and per-run selection records. Training and evaluation code, configuration files, and the list of question identifiers that defines each partition will be released; the link is omitted here to preserve anonymity.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/25f7be9694d7b32d5cc670927b8091e1-Abstract-Conference.html.

Hongyu Cao, Yuxuan Wu, Yucheng Cai, Xianyu Zhao, and Zhijian Ou. Improving end-to-end training of retrieval-augmented generation models via joint stochastic approximation, 2025. URL https://arxiv.org/abs/2508.18168.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. RQ-RAG: Learning to refine queries for retrieval augmented generation. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tzE7VqsaJ4.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. ReSearch: Learning to reason with search for LLMs via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/7b3a0d3a0864b66f229c0a9c84f9e950-Abstract-Conference.html.

Yunmo Chen, Tongfei Chen, Harsh Jhamtani, Patrick Xia, Richard Shin, Jason Eisner, and Benjamin Van Durme. Learning to retrieve iteratively for in-context learning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7156–7168, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.406. URL https://aclanthology.org/2024.emnlp-main.406/.

Yuntao Gui and James Cheng. Search-R3: Unifying reasoning and embedding in large language models, 2025. URL https://arxiv.org/abs/2510.07048v2. arXiv:2510.07048v2.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/guu20a.html.

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

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251): 1–43, 2023. URL https://jmlr.org/papers/v24/23-0037.html.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

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

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

Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Bridging the preference gap between retrievers and LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10438–10451, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.562. URL https://aclanthology.org/2024.acl-long.562/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP

tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Yuan Li, Qi Luo, Xiaonan Li, Bufan Li, Qinyuan Cheng, Bo Wang, Yining Zheng, Yuxin Wang, Zhangyue Yin, and Xipeng Qiu. R3-RAG: Learning step-by-step reasoning and retrieval for LLMs via reinforcement learning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10491–10507, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.554. URL https://aclanthology.org/2025.findings-emnlp.554/.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. RA-DIT: Retrieval-augmented dual instruction tuning. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/536d18fbb454f80221465f1a42c6f389-Abstract-Conference.html.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Yuchen Li, Daiting Shi, Dawei Yin, and Zhicheng Dou. Agentic-R: Learning to retrieve for agentic search, 2026. URL https://arxiv.org/abs/2601.11888.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

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. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Yujing Wang, Hainan Zhang, Liang Pang, Binghui Guo, Hongwei Zheng, and Zhiming Zheng. MaFeRw: Query rewriting with multi-aspect feedbacks for retrieval-augmented large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):25434–25442, 2025. doi: 10.1609/aaai.v39i24.34732. URL https://ojs.aaai.org/index.php/AAAI/article/view/34732.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. Supervising the search process produces reliable and generalizable information-seeking agents, 2025. URL https://arxiv.org/abs/2502.13957.

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 Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Fuda Ye, Shuangyin Li, Yongqi Zhang, and Lei Chen. R^2AG: Incorporating retrieval information into retrieval augmented generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 11584–11596, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.678. URL https://aclanthology.org/2024.findings-emnlp.678/.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 121156–121184. Curran Associates, Inc., 2024. doi: 10.52202/079017-3850. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/db93ccb6cf392f352570dd5af0a223d3-Abstract-Conference.html.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. Learning to retrieve: How to train a dense retrieval model effectively and efficiently, 2020. URL https://arxiv.org/abs/2010.10469.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing dense retrieval model training with hard negatives, 2021. URL https://arxiv.org/abs/2104.08051.

Qingfei Zhao, Ruobing Wang, Dingling Xu, Daren Zha, and Limin Liu. R-Search: Empowering LLM reasoning with search via multi-reward reinforcement learning, 2025. URL https://arxiv.org/abs/2506.04185.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.22. URL https://aclanthology.org/2025.emnlp-main.22/.

Jiawei Zhou and Lei Chen. Optimizing retrieval for RAG via reinforcement learning, 2025. URL https://arxiv.org/abs/2510.24652v2. arXiv:2510.24652v2.

A PER-DATASET AND PER-SEED RESULTS

Tables 5, 6 give the per-dataset and per-seed values behind the aggregates of Tables 1, 2, for the two primary systems and the two recovery controls. For the paired in-domain comparison of Table 2, the discordant pairs are 860 against 1,024, 854 against 982, and 960 against 938 for seeds 13, 21, and 42, counting questions answered correctly by PIQE only against the baseline only. Out of domain the discordant pairs are 754 against 823, 766 against 907, and 786 against 853 for seeds 13, 21, and 42, in the same order. For the headroom diagnostic of Table 7, the seed-13 gains by dataset are $+2.44$ on HotpotQA, $+1.55$ on 2WikiMultiHopQA, and $+1.34$ on MuSiQue.

Table 5: In-domain results on the frozen reporting partition (11,262 questions) for all four systems, including the two recovery controls. Dataset columns are means over seeds 13, 21, and 42 (HotpotQA $n = 3{,}733$; 2WikiMultiHopQA $n = 6{,}259$; MuSiQue $n = 1{,}270$). Per-seed micro exact match is 43.57 / 43.66 / 42.26 for Search-R1, 42.12 / 42.52 / 42.45 for PIQE, 43.19 / 43.50 / 43.15 for PIQE + RRF, and 42.98 / 40.86 / 41.98 for Hybrid. Differences and summary statistics quoted in the paper are computed from unrounded values, so they can differ from the difference of two displayed values by 0.01. The last two columns give token-level F1 (Section E.5) on the same predictions, averaged over the same three seeds; their standard deviations are 0.89 / 1.03 for Search-R1, 0.25 / 0.19 for PIQE, 0.17 / 0.19 for PIQE + RRF, and 0.99 / 1.08 for Hybrid (macro / micro).

	Dataset EM (mean over seeds)			Macro EM		Micro	Token F1 (mean)	
Method	HQA	2W	MQ	per seed	mean	mean	macro	micro
Search-R1	42.55	48.35	19.40	37.31/37.04/35.95	36.77 (0.72)	43.16	45.29	51.10
PIQE	42.51	47.06	18.77	35.73/36.23/36.39	36.11 (0.34)	42.36	44.55	50.43
PIQE + RRF	43.20	48.23	19.11	36.84/37.07/36.63	36.85 (0.22)	43.28	45.39	51.47
Hybrid	41.94	46.55	19.21	36.76/35.02/35.93	35.90 (0.87)	41.94	44.74	50.34

Table 6: Out-of-domain results for the same selected checkpoints (Natural Questions $n = 3{,}610$; TriviaQA $n = 11{,}313$; Bamboogle $n = 125$; 15,048 questions in total). Micro averaging is the primary aggregate because a simple macro average over the three datasets would weight Bamboogle, which is 0.83% of the questions, at one third; token-level F1 is therefore also reported as a micro average.

	Dataset EM (mean over seeds)			Micro EM		Micro F1
Method	NQ	TriviaQA	Bamboogle	per seed	mean	mean
Search-R1	36.12	63.99	47.20	57.10 / 57.46 / 56.94	57.17 (0.26)	65.72 (0.37)
PIQE	35.78	63.26	48.80	56.64 / 56.52 / 56.50	56.55 (0.08)	64.91 (0.14)
PIQE + RRF	36.97	64.71	50.93	57.99 / 58.10 / 57.72	57.94 (0.20)	66.47 (0.18)
Hybrid	36.24	64.14	50.93	57.80 / 56.56 / 57.66	57.34 (0.68)	65.78 (0.76)

Table 7: Headroom diagnostic on the frozen reporting partition, three seeds. Both rows use the same Search-R1 actor, retriever, and index, and both retrieve twenty candidates per search; the second row replaces the actor's selection with one that first takes the highest-scoring candidate of each gold supporting-document title group, in descending score order, and fills the remaining slots by score, so it consumes gold labels at evaluation time. At the first search of a question, 98.8% of the ordered top-20 candidate lists are identical between the two rows on seed 13; the lists then diverge, because inserting different passages changes the context that produces later searches (58.1%, 35.5%, and 27.3% identical at the second, third, and fourth search). It measures the gain obtained by one gold-label-informed selection rule on the baseline's candidate sets, not a bound over selection policies.

	Seed (macro EM / micro EM)			Mean (SD)
Condition	13	21	42	macro / micro
Search-R1	37.31 / 43.57	37.04 / 43.66	35.95 / 42.26	36.77 (0.72) / 43.16 (0.79)
Label-informed selection	39.09 / 45.39	39.02 / 45.51	37.52 / 44.00	38.54 (0.88) / 44.97 (0.84)
Difference	+1.78 / +1.82	+1.98 / +1.85	+1.58 / +1.74	+1.78 / +1.80

B Retrieval Decomposition for All Systems

Table 8: Per-seed retrieval decomposition on the frozen reporting partition, measured on the search events logged during evaluation. PPR@5 is pool-positive recall@5, the fraction of the candidate set's answer-containing passages that the five inserted passages cover; the other columns are defined in Section 4.4. Bold marks the best value in each column over all twelve runs; the event count is a property of the run, not a quality measure, and is not marked. The first rank is not marked either, because for Hybrid it is a rank in a list whose first two entries are fixed by the text retriever rather than a rank under a single query score, so the column is not comparable across systems. Means over seeds are 47.14 / 82.31 / 38.80 / 22.80 / 3.29 for Search-R1, 45.67 / 81.43 / 37.19 / 21.98 / 3.39 for PIQE, 46.71 / 81.38 / 38.01 / 22.38 / 3.39 for PIQE + RRF, and 44.72 / 84.61 / 37.84 / 25.02 / 3.10 for Hybrid. Search events from the same question are not independent, so uncertainty is quantified by seed only. For Hybrid the candidate set is the union of the text retriever's top two passages, placed first, and the projected top-20, truncated to twenty, so the fixed passages are themselves candidates and the conditional decomposition remains well defined; because those two are always inserted, its within-pool selection is the highest of the four systems while its candidate pool is the weakest.

Method	Seed	Events	Pool@20	AC@5 | Pool@20	AC@5	PPR@5	First rank
Search-R1	13	25,806	**47.57**	82.62	**39.30**	22.93	3.28
	21	24,447	46.64	82.23	38.35	22.65	3.29
	42	24,450	47.20	82.08	38.74	22.83	3.30
PIQE	13	23,661	45.99	82.29	37.84	22.37	3.30
	21	24,071	45.80	80.29	36.77	21.70	3.52
	42	23,403	45.23	81.70	36.95	21.86	3.35
PIQE + RRF	13	23,434	47.10	81.49	38.38	22.69	3.35
	21	23,915	46.76	80.97	37.86	22.36	3.42
	42	23,093	46.27	81.68	37.80	22.10	3.41
Hybrid	13	24,604	44.79	**85.55**	38.31	**25.41**	3.06
	21	24,506	44.94	84.12	37.80	24.75	3.10
	42	22,841	44.44	84.17	37.40	24.89	3.14

Table 8 reports the decomposition for all four systems and all three seeds. Beyond the three quantities of Section 4.4 it lists the rank of the first answer-containing candidate among events that have one, for which a lower value is better; pool-positive recall@5, the fraction of the candidate set's positives that the inserted passages cover, averaged over events and counting an event without a positive as zero; the searches-per-question column of Table 3 is the mean number of valid search actions per question in the evaluation run. Positives are labelled on the full text of a candidate, before the observation length limit truncates what is inserted, so AC@5 measures the evidence selected rather than the text the policy finally reads. The retrieval columns are computed on logged search

events, of which there are fewer than valid search actions because an event whose generated span cannot be matched to a search segment is discarded: over the six runs of the two primary systems, 692–946 of the 24,230–26,616 valid search actions per run are dropped, that is 2.8–3.8%. The two recovery controls behave differently from each other. PIQE + RRF restores part of the candidate coverage lost by the integrated interface (Pool@20 46.71 against 45.67 for PIQE and 47.14 for the baseline) and correspondingly recovers evidence acquisition (AC@5 38.01). Hybrid instead keeps the weakest candidate pool of the four systems (44.72) while achieving the strongest within-pool selection (84.61), because two of its five inserted passages are taken directly from the text retriever's ranking; the improved selection does not translate into better end-to-end acquisition (AC@5 37.84) or accuracy (micro EM 41.94). Separating candidate generation from within-candidate selection is therefore necessary to describe these systems correctly.

C Recovery and Attribution Controls

Table 9: Attribution controls for the out-of-domain fusion result, seed 13, micro exact match over 15,048 questions. All fused rows use reciprocal rank fusion of the text-retriever ranking with a projection ranking (rank constant 60, fusion weight 1); "warm-start" is the projection obtained by distilling the text retriever, before any RL. Differences are against the reference row, with paired tests giving $p = 0.73$, $p = 0.0090$, and $p = 0.0004$ from top to bottom. The Full row is the seed-13 entry of PIQE + RRF in Table 6, whose three-seed mean is 57.94; the same checkpoint without fusion scores 56.64 on this seed. The corresponding in-domain micro values are 43.57, 43.09, 43.13, and 43.19.

Condition	Actor	Retrieval at evaluation	OOD micro EM	Δ
Reference	Search-R1	text retriever only, no fusion	57.10	—
A	Search-R1	fused with warm-start projection	57.02	−0.08
A′	PIQE	fused with warm-start projection	57.75	+0.65
Full	PIQE	fused with RL-trained projection	57.99	+0.90

Read as sequential contrasts, applying the fusion operation under the baseline actor changes micro exact match by -0.08 points; replacing the baseline actor by the PIQE actor, with the warm-started projection held fixed, adds $+0.73$; and replacing the warm-started projection by the RL-trained one, with the PIQE actor held fixed, adds a further $+0.24$. The fusion operation alone therefore does not improve accuracy, and most of the out-of-domain gain appears in conditions that use the PIQE actor. Because these contrasts depend on the order in which the factors are changed, and because the PIQE actor differs from the baseline actor in several ways at once, this analysis does not isolate a single cause and does not establish that the projection update is responsible for the gain.

Both controls call the external query encoder at inference, by construction: fusion needs the text ranking, and Hybrid inserts text-retrieved passages directly. They are therefore evidence about where the accuracy gap comes from, not instances of the integrated architecture.

D Additional Related Work

This appendix places the paper among neighboring lines of work; Section 2 states the relation to the closest ones.

Retrieval-augmented generation and agentic search. Classical retrieval-augmented models retrieve once or a fixed number of times rather than letting the model decide when to search (Lewis et al., 2020; Guu et al., 2020; Izacard et al., 2023), on the dense-retrieval foundation of Karpukhin et al. (2020). Agentic systems interleave retrieval with reasoning (Yao et al., 2023; Trivedi et al., 2023; Press et al., 2023; Jiang et al., 2023), and reinforcement learning is now the standard way to train the policy that drives them (Jin et al., 2025; Chen et al., 2025; Xiong et al., 2025; Li et al., 2025; Zheng et al., 2025). Iterative retrieval decisions have also been learned outside question answering, for instance to assemble in-context examples rather than knowledge passages (Chen et al., 2024). All of these place retrieval outside the language-model policy; in the dense-retrieval configurations

among them, that means a separate neural query encoder, which is the assumption this paper tests, whereas systems built on sparse or web search do not use one (Jiang et al., 2023).

Improving what is retrieved. Beyond the query rewriting and retriever adaptation discussed in Section 2, a further line addresses the mismatch between retriever relevance and generator utility, by ranking context inside the generator (Yu et al., 2024), by learning what the generator prefers or by critiquing what has been retrieved (Ke et al., 2024; Asai et al., 2024), or by passing retrieval information itself into the generator as features (Ye et al., 2024). These interventions act before or after retrieval while leaving the two-model inference path intact. Our results speak to this literature in a specific way: in our setting the retrieval deficit is already visible at candidate generation and is not offset by the observed top-5 ranking, so interventions that operate on the retrieved set start from a weaker candidate pool.

Query-side training with a fixed index. Keeping passage embeddings fixed and updating only the query side is an established design in dense retrieval (Zhan et al., 2021; 2020), and it is what makes our setting practical: updating passage representations during reinforcement learning would require re-encoding and re-indexing 21M passages. The same choice is what bounds the method, because the projection has to map policy states into a space that was built for text queries.

E Mathematical Details

This appendix restates the definitions of Section 3 in full and adds the details omitted there: ordered sampling without replacement, the old-policy ratio used in the update, and the reductions applied to each term.

E.1 Trajectory notation

For an input question x, the agent produces an ordered trajectory

$$\tau = (o_0, u_1, \ldots, u_L), \tag{4}$$

where $o_0 = x$, each u_ℓ is either a token action a_t^{tok} generated by the LLM or a retrieval-action/observation pair (a_s^{ret}, o_s). Retrieval actions a_s^{ret} are ordered sampled passage lists P_s, and observations o_s are the inserted `<information>` contexts. The generated token stream contains reasoning tokens, search-query tokens, and answer tokens.

At search step s, let $H_s = (h_{s,1}, \ldots, h_{s,m_s})$ denote the hidden states corresponding to the tokens between `<search>` and `</search>`. The query-side module maps these hidden states to

$$\boldsymbol{q}_s = \frac{W \,\mathrm{MeanPool}(H_s)}{\|W \,\mathrm{MeanPool}(H_s)\|_2}, \tag{5}$$

where W is the trainable projection matrix. Other pooling functions are possible, but the main experiments use mean pooling for stability.

E.2 Top-M candidate approximation

The full passage set is $\mathcal{D} = \{d_i\}_{i=1}^N$, with fixed passage embeddings $\boldsymbol{e}_i$. The default implementation stores L_2-normalized passage embeddings, matching the normalized query embedding in Equation (5). Exact normalization over N passages is expensive. We therefore first obtain a candidate set $\mathcal{C}_s$ of size M from the fixed index. Unless otherwise stated, $M = 20$. Scores are computed as

$$z_{s,i} = \frac{\boldsymbol{q}_s^\top \boldsymbol{e}_i}{\tau_{\text{temp}}}, \qquad d_i \in \mathcal{C}_s. \tag{6}$$

The passage policy is

$$\pi_\phi^{\text{ret}}(d_i \mid H_s, \mathcal{C}_s) = \frac{\exp(z_{s,i})}{\sum_{d_j \in \mathcal{C}_s} \exp(z_{s,j})}. \tag{7}$$

This top-M normalization is an approximation. It is biased relative to full-corpus normalization because the candidate set depends on the query embedding and because passages outside $\mathcal{C}_s$ receive

zero probability in the recorded objective. We treat candidate generation as a fixed-index, stop-gradient operation and optimize the conditional policy within $\mathcal{C}_s$. This makes RL training feasible while preserving a well-defined sampled-action log-probability for the selected candidate set. The resulting policy should therefore be interpreted as a candidate-set passage policy, not as an exact full-corpus retrieval policy.

E.3 Without-replacement passage sampling

During training, we sample k passages without replacement from $\mathcal{C}_s$. Let $P_s = (d_{s,1}, \ldots, d_{s,k})$ be the ordered sampled sequence. The probability of this ordered sequence is

$$\pi_\phi^{\mathrm{ret}}(P_s \mid H_s, \mathcal{C}_s) = \prod_{r=1}^{k} \frac{\exp(z_{s,d_{s,r}})}{\sum_{d_j \in \mathcal{C}_s \setminus \{d_{s,1},\ldots,d_{s,r-1}\}} \exp(z_{s,j})}. \tag{8}$$

The implementation records the ordered sampling log-probability. The selected passages are serialized in their sampled order, and we assume no permutation invariance of the downstream model. At evaluation time, we take the top-k candidates under the policy scores deterministically.

E.4 Unified trajectory probability

The token policy is $\pi_\theta^{\mathrm{tok}}$ and the passage policy is π_ϕ^{ret}. The trajectory log-probability is

$$\log \pi_{\theta,\phi}(\tau \mid x) = \sum_{t=1}^{T} \log \pi_\theta^{\mathrm{tok}}(a_t^{\mathrm{tok}} \mid c_t) + \sum_{s=1}^{S} \log \pi_\phi^{\mathrm{ret}}(P_s \mid H_s, \mathcal{C}_s), \tag{9}$$

where c_t is the LLM context at token step t. In the optimization loss, we use the weighted score

$$S_{\theta,\phi}(\tau \mid x) = \sum_{t=1}^{T} \log \pi_\theta^{\mathrm{tok}}(a_t^{\mathrm{tok}} \mid c_t) + \alpha \sum_{s=1}^{S} \log \pi_\phi^{\mathrm{ret}}(P_s \mid H_s, \mathcal{C}_s), \tag{10}$$

where α controls the relative weight of passage-action log-probabilities.

The information text inserted between `<information>` and `</information>` is not treated as generated output. It is an observation. Only the probability of selecting the inserted passages contributes to the retrieval-action log-probability. This is the point at which selected passages enter the same credit-assignment objective as generated tokens.

Gradient path. For a sampled trajectory, the candidate set $\mathcal{C}_s$ is held fixed during the gradient update. We stop gradients through the nearest-neighbor top-M operator, fixed passage embeddings, sampled passage identifiers, and retrieved text. We do not detach H_s in the default configuration. Consequently, $\nabla_\phi \log \pi_\phi^{\mathrm{ret}}$ updates the query-side projection and $\nabla_\theta \log \pi_\phi^{\mathrm{ret}}$ updates the language model only through the search-token hidden states H_s. The token log-probability term provides the usual policy-gradient update to generated tokens, while the passage log-probability term provides an additional reward-weighted signal to the representation used for retrieval. This implementation makes the trajectory objective unified at both the probability level and the differentiable parameter-update level, while still treating retrieved passages as observations rather than generated text.

E.5 Reward and advantage

The reward is the binary exact match of the last `<answer>` span against the gold answers,

$$R(\tau) = \mathrm{EM}(y, \mathcal{Y}^*) \in \{0, 1\}. \tag{11}$$

There is no format reward and no process reward. Token-level F1 is computed on the last `<answer>` span after the same normalization as exact match, taking the maximum token-overlap F1 over the accepted gold answers; it is reported in Tables 5, 6.

We sample a group of $G = 8$ trajectories $\{\tau_g\}_{g=1}^{G}$ for the same question. The group-relative advantage is

$$A_g = \frac{R(\tau_g) - \mathrm{mean}_{g'} R(\tau_{g'})}{\mathrm{std}_{g'} R(\tau_{g'}) + \epsilon}. \tag{12}$$

This follows the GRPO-style normalization used in recent RL training of reasoning models (Shao et al., 2024), with $\epsilon = 10^{-6}$; the advantage of a trajectory is assigned to every token of its response. Answer normalization lowercases the string, removes punctuation and the articles *a*, *an*, and *the*, and collapses whitespace; the same function defines exact match, token-level F1, and the answer-containing labels of Section 4.4. The core objective is

$$\mathcal{L}_{\mathrm{RL}}(\theta, \phi) = -\mathbb{E}_g \left[A_g \left(\sum_t \log \pi_\theta^{\mathrm{tok}}(a_{g,t}^{\mathrm{tok}} \mid c_{g,t}) + \alpha \sum_s \log \pi_\phi^{\mathrm{ret}}(P_{g,s} \mid H_{g,s}, \mathcal{C}_{g,s}) \right) \right]. \quad (13)$$

Here $P_{g,s}$ is the ordered passage list sampled at search step s in trajectory g. This objective is a policy-gradient objective over a mixed action space of tokens and ordered passage lists, building on classical policy-gradient ideas (Williams, 1992) and modern regularized RL practice (Schulman et al., 2017).

Clipped surrogate used in the implementation. As in the token term, the passage term uses a PPO-style clipped ratio with respect to the rollout policy ϕ_{old}. The denominator is the ordered selected-list log-probability recorded during the rollout, under the projection, temperature, hidden states, and candidate set of that rollout; it is stored with the trajectory and not recomputed from the current states. For every valid search step,

$$\begin{aligned} \ell_{g,s}^{\mathrm{ret}} &= -\min\Big(\rho_{g,s} A_g,\ \mathrm{clip}(\rho_{g,s}, 1-\epsilon_c, 1+\epsilon_c) A_g\Big), \\ \rho_{g,s} &= \exp\Big(\log \pi_{\theta,\phi,\tau_{\mathrm{temp}}}^{\mathrm{ret}}\big(P_{g,s} \mid H_{g,s}(\theta), \mathcal{C}_{g,s}^{\mathrm{old}}\big) - \mathrm{stopgrad}\big(\ell_{g,s}^{\mathrm{old}}\big)\Big), \end{aligned} \quad (14)$$

with $\epsilon_c = 0.2$, where $\ell_{g,s}^{\mathrm{old}}$ is the stored rollout log-probability of the same ordered list under the actor, projection, temperature, and candidate set of that rollout, and the numerator is recomputed with the current actor and projection on the same candidate set. The sum over search steps in Equation (3) is realized as a mean over search events in the implementation: the trajectory advantage A_g is shared by all search steps of trajectory g, and the step losses are averaged over the valid search events of the micro-batch, so every valid search event carries equal weight and a trajectory's total passage-side weight is proportional to its number of valid searches, while a trajectory with no search contributes nothing to this term. The result is added to the token loss, whose own reduction is the usual token mean, with weight α. The log-ratio is clamped to $[-20, 20]$ before exponentiation for numerical safety.

E.6 Regularization

Token-policy drift is controlled by a KL penalty to the initialization policy,

$$\mathcal{L}_{\mathrm{KL}} = \beta_{\mathrm{tok}}\, \mathrm{KL}(\pi_\theta^{\mathrm{tok}} \| \pi_{\theta_0}^{\mathrm{tok}}), \quad (15)$$

with $\beta_{\mathrm{tok}} = 0.02$ and the low-variance estimator $\exp(\delta) - \delta - 1$ for $\delta = \log \pi_{\theta_0}^{\mathrm{tok}} - \log \pi_\theta^{\mathrm{tok}}$, clamped to $[-10, 10]$ per token. The final optimization loss is

$$\mathcal{L} = \mathcal{L}_{\mathrm{RL}} + \mathcal{L}_{\mathrm{KL}}. \quad (16)$$

The implementation also supports an entropy bonus $-\beta_{\mathrm{ret}} \sum_s \mathcal{H}(\pi_\phi^{\mathrm{ret}}(\cdot \mid H_s, \mathcal{C}_s))$ on the passage policy, but $\beta_{\mathrm{ret}} = 0$ in all reported runs.

E.7 Temperature schedule

The retrieval temperature is annealed linearly from τ_{start} to τ_{end} over the first U_{a} RL steps and then held constant:

$$\tau_{\mathrm{temp}}(u) = \tau_{\mathrm{end}} + (\tau_{\mathrm{start}} - \tau_{\mathrm{end}}) \left(1 - \frac{\min(u, U_{\mathrm{a}})}{U_{\mathrm{a}}}\right), \quad (17)$$

with $\tau_{\mathrm{start}} = 0.05$, $\tau_{\mathrm{end}} = 0.02$, and $U_{\mathrm{a}} = 100$ in the default configuration. Configurations labeled "constant" use $\tau_{\mathrm{temp}} = 0.02$ throughout. The purpose is to allow more exploration early and a sharper sampling distribution later in training.

Table 10: Models, data, and tools. The question sets and the corpus are used at the revisions given here; the model snapshots are the ones the runs actually loaded, since our configurations name the models without pinning a revision.

Component	Choice	Source
Policy LLM	Qwen3-4B-Instruct-2507, snapshot `cdbee75`	`https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507`
Passage encoder (fixed)	Qwen3-Embedding-4B, snapshot `5cf2132`, last-token pooling, 2,560 dims, L_2-normalized	`https://huggingface.co/Qwen/Qwen3-Embedding-4B`
Corpus	wiki-18, December 2018 Wikipedia in DPR format (21,015,324 passages), revision `69c1c00`	`https://huggingface.co/datasets/PeterJinGo/wiki-18-corpus`
Index	FAISS exact inner product (`IndexFlatIP`)	`https://github.com/facebookresearch/faiss`
Training data	HotpotQA (90,447), 2WikiMultiHopQA (15,000), MuSiQue (19,938) processed `train` splits from `RUC-NLPIR/FlashRAG_datasets`, revision `bcafb8d`	`https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets`
In-domain evaluation data	Official development sets of the same datasets, split as in Section 4.3	same as above
Out-of-domain evaluation data	Natural Questions, TriviaQA, and Bamboogle `test` splits from `RUC-NLPIR/FlashRAG_datasets`, revision `bcafb8d`, converted to the Search-R1 evaluation format	`https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets`
Software	veRL (Search-R1 fork), vLLM 0.8.5.post1, Transformers 4.51.3, PyTorch 2.6.0+cu124 (CUDA runtime 12.4, cuDNN 9.1.0), FAISS 1.8.0 with GPU search	`https://github.com/volcengine/verl`, `https://github.com/vllm-project/vllm`

Table 11: Question counts by dataset. The training splits are used as released; the in-domain pool is the union of the official development sets, split once into Dev1200, a reserve, and the frozen reporting partition; the out-of-domain sets are evaluation only. Sources and revisions are in Table 10.

	In-domain				
Dataset	Train	Dev pool	Dev1200	Reserve	Reporting
HotpotQA	90,447	7,405	400	3,272	3,733
2WikiMultiHopQA	15,000	12,576	400	5,917	6,259
MuSiQue	19,938	2,417	400	747	1,270
Total	125,385	22,398	1,200	9,936	11,262
	Out-of-domain (test splits, evaluation only)				
Natural Questions			3,610		
TriviaQA			11,313		
Bamboogle			125		
Total			15,048		

F Implementation Details

F.1 Models, data, and tools

The query-side module is a single linear map $W \in \mathbb{R}^{2560\times2560}$ on the mean of the search-token hidden states of the policy, followed by L_2 normalization. The passage encoder is used only to

encode the corpus once before training and to embed text queries for the Search-R1 baseline and the warm-start teacher; it is never updated.

F.2 PROJECTION WARM-START

The warm-start data are the search events of 65,536 rollouts of the initial policy (with the text retriever) on training questions. The teacher embedding of an event is the Qwen3-Embedding-4B embedding of its search string with the instruction "Given a web search query, retrieve relevant passages that answer the query". Stage v2 trains W from scratch for at most 40 epochs (learning rate 10^{-3}, patience 5) with the loss $\mathcal{L}_{\cos} + 1.0 \cdot \mathcal{L}_{\text{rank}}$, where $\mathcal{L}_{\cos} = 1 - \langle q_s, q_s^{\text{teacher}} \rangle$ and $\mathcal{L}_{\text{rank}}$ is $\text{KL}(p^{\text{teacher}} \,\|\, p^{\text{student}})$ between softmax distributions (temperature 0.05) over the teacher's top-20 candidates. Stage v3 starts from v2 and trains for at most 80 epochs (learning rate 10^{-4}, weight decay 10^{-4}, patience 10) with ranking weight 0.5 over a fixed union of the teacher's and the v2 student's top-20 candidates. Both stages use batch size 128 and seed 13. The 65,536 logged rollouts yield 81,035 search events, since 32,590 trajectories issue no search; they are split by example, using a hash of seed, split, data source, and example index so that events of one question cannot cross the split, into 77,153 training and 3,882 validation events (a validation fraction of 0.05). Stage v3 monitors the total validation loss, runs all 80 epochs with patience 10, and keeps the checkpoint of the epoch with the lowest validation loss, which is epoch 74 (zero-based) at a validation loss of 0.333; its ranking term uses coefficient 0.5 at temperature 0.05. Layer-specific projections (Table 16) repeat both stages on the hidden states of the corresponding layer.

Projection-only warm-up. We also tried inserting a short reinforcement-learning phase between the distillation and the joint objective, in which the actor is frozen and only W is updated by the passage-action term (25 steps, projection learning rate 10^{-5}, temperature 0.02). The observed changes are small. Evaluated before joint training on a fixed $11{,}136$-question development subset with seed 13, it moves macro exact match from 20.47 to 20.39 for the 64k warm-start and from 20.24 to 20.21 for the 32k one, against 20.72 for the text query in the same evaluation; after the full 150-step budget the corresponding Dev1200 macro exact match is 34.33 against 34.25 without the phase (Table 15). This evaluation is a weak instrument, because the policy before joint training issues only 0.86 searches per question and 51.5% of its trajectories contain no retrieved passage at all, and it uses one seed; we report it to record that the phase was tried, not as a measurement of its effect. The primary runs therefore use the distilled projection directly, with no projection-only phase.

F.3 SEARCH AND PROMPTING FORMAT

The agent uses the following special-token protocol:

- `<think>...</think>`: internal reasoning tokens.
- `<search>...</search>`: search-query tokens used to construct H_s.
- `<information>...</information>`: retrieved passages inserted as observations.
- `<answer>...</answer>`: final answer tokens.

The hidden states for the tokens inside `<search>` are pooled to define the query embedding. Retrieved text inside `<information>` is masked from the token-generation loss because it is not generated by the model.

F.4 OPTIMIZATION AND GENERATION

The actor is optimized with AdamW ($\beta = (0.9, 0.999)$, weight decay 10^{-2}) at learning rate 10^{-6} with 10 warm-up steps and a constant schedule afterwards, the warm-up length being the configured ratio of 0.01 applied to the scheduler's nominal horizon of 1,005 optimizer steps, while training is stopped at 200; the projection uses its own learning rate (Table 12) and the same optimizer. Each optimizer step takes 128 questions with 8 rollouts per question, performs one PPO epoch over that batch with a mini-batch of 256 sequences, and uses the per-device micro-batch sizes of Table 13. Training rollouts are generated with vLLM at temperature 1.0 with top-k disabled and no nucleus truncation; validation decodes greedily. A trajectory may generate at most 2,048 tokens before its

first search, 500 tokens per subsequent segment, and 500 tokens per inserted observation, within a 6,144-token prompt window, and it may issue at most four search steps. Trajectories that issue no search contribute to the token term only (Section E.5).

F.5 Retrieval and preprocessing

The retriever encodes queries with last-token pooling in bfloat16 under the instruction "Given a web search query, retrieve relevant passages that answer the query", truncating the query at 512 tokens, and searches the exact inner-product index on GPU. An inserted observation lists the passages as `Doc` i `(Title:` t_i`)` x_i, where the title is the first line of the corpus entry and the body is the remainder; passages are inserted in the order of the selected list and are not shortened beyond the observation length limit. Candidate sets are deduplicated by passage identifier, which is what makes the Hybrid union of Section 4.2 well defined. The five inserted passages are concatenated into one `<information>` block, tokenized, and truncated to the first 500 tokens, so a long block loses its last passages rather than being shortened passage by passage. Reciprocal rank fusion combines the text ranking and the projection ranking, each of length twenty, as $\sum_r w_r/(60 + \mathrm{rank}_r)$ with unit weight on both lists, so a passage that appears in only one list accumulates only that list's term, and the fused list is truncated to twenty candidates. A search whose generated span cannot be matched to a `<search>` segment is dropped from the event log, which is why the event counts in Table 8 are slightly below the number of search actions.

F.6 Hyperparameters

Table 12 lists the settings shared by all runs and those specific to PIQE.

Table 12: Hyperparameters. Micro-batch sizes depend on the cluster (Table 13).

Group	Hyperparameter	Value
Shared	Questions per step / rollouts per question	128 / 8
	Optimizer steps	200 (150 in the exploratory runs of Section G)
	Actor learning rate	10^{-6}, 10 warm-up steps, then constant
	KL coefficient / estimator	0.02 / low-variance KL
	PPO clip ratio / gradient clipping	0.2 / 1.0
	Log-ratio clamp (token, KL, passage)	20
	Max search steps / passages per search	4 / 5
	Reward	EM of the last answer span (no format or process reward)
	Validation	Dev1200 at step 0 and every 10 steps, greedy decoding
	Checkpoints	every 10 steps from step 50
PIQE (current)	Passage-action weight α	0.03
	Candidates M	20
	Temperature	$0.05 \to 0.02$ linearly over 100 steps, then 0.02
	Projection learning rate η_ϕ	5×10^{-5}
	Entropy bonus β_{ret}	0
	Hidden states	final layer, mean over search tokens, not detached
	Projection initialization	v3 warm-start (Section F.2)

F.7 Compute

Table 13 summarizes the hardware and the per-device micro-batch settings used in the reported runs.

F.8 Per-run results

Figure 2 shows the Dev1200 selection metric over training for the six runs of the main comparison, and Table 14 lists those runs individually; exploratory runs on other clusters and under the earlier budget are summarized in Section G. We plot the selection metric rather than the training reward or

Table 13: Clusters (NVIDIA GPUs) and per-device micro-batch sizes for rollout generation, actor update, and old/reference log-probability computation. The global batch (128 questions × 8 rollouts), learning rates, warm-up, and number of optimizer steps are identical on all clusters; only the micro-batch size is adapted to the data-parallel size. The retrieval server shares the node with training.

		Per-device micro-batch			
Cluster	GPUs	rollout	actor	old log-p.	ref log-p.
Cluster B	4× H100 (94 GB)	8	4	8	8
Cluster A (4-GPU runs)	4 of 8× H200 (141 GB)	8	4	8	8
Cluster A (8-GPU runs)	8× H200 (141 GB)	8	8	16	16

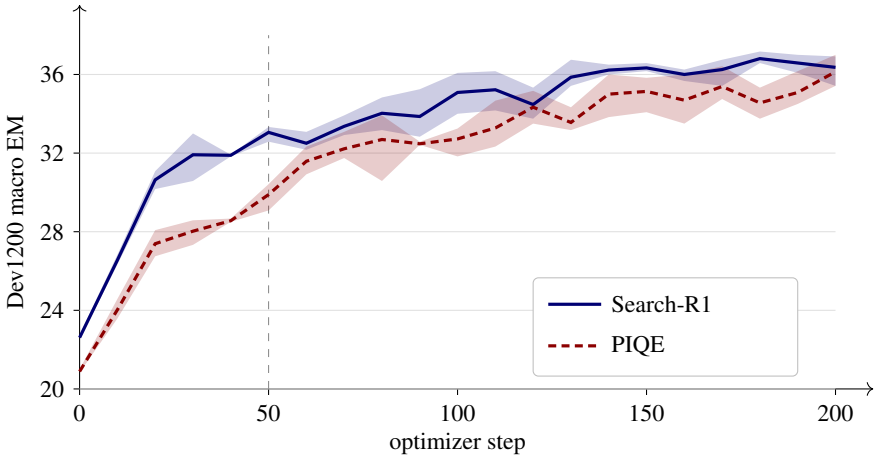

Figure 2: Dev1200 macro exact match against optimizer step for the two primary systems, seeds 13, 21, and 42 on Cluster A. Lines are the mean over the three seeds and the shaded regions span the minimum and maximum seed at each step. Both systems are evaluated on the same 1,200 questions with greedy decoding and deterministic retrieval, and checkpoints become eligible for selection at step 50. The two curves have the same shape and neither collapses over the evaluated training interval, with PIQE tracking below the baseline throughout. PIQE also rises more slowly: from about step 130 onward the baseline mean stays within half a point of its final value, whereas for seeds 21 and 42 the best PIQE checkpoint is the last one in the budget, so a longer budget could still change its endpoint (Section 7). Per-run values for these runs are in Table 14; exploratory runs on other clusters and under the earlier budget are summarized in Section G.

the training loss: the two objectives differ by the passage-action term, so their losses are not on a common scale, whereas Dev1200 macro exact match is the same quantity for both systems and is the quantity used to select checkpoints.

Table 14: Runs of the two primary systems and of the Hybrid control, all on Cluster A, with Dev1200 macro exact match at step 0 and at the selected checkpoint; checkpoint selection is restricted to steps 50–200, and the seed-13 Hybrid run continued to step 250 for runtime reporting. Wall-clock is the elapsed time of the logged training run, including checkpointing and the periodic Dev1200 evaluations, of which there are 21 under the 200-step budget and 26 for the seed-13 Hybrid run that continued to step 250, as recorded by the experiment tracker. Per optimizer step it is 0.09 h for Search-R1, 0.14 h for PIQE, and 0.14 h for Hybrid, so the totals differ mainly through the objective and, for Hybrid seed 13, through the additional 50 steps. Step-0 values differ slightly across seeds although the initial model and the evaluation protocol are identical: at step 0, 97.5% of the generated responses agree between the baseline runs of seeds 13 and 21, so generation is not bitwise reproducible across runs in our setup, and we did not isolate the cause. We did not re-evaluate a single checkpoint to quantify that evaluation-run variability, so differences of a few tenths of a point in the single-seed diagnostics of Tables 4, 9 are not separated from it. [a]Continued beyond the 200-step budget of the main comparison; selection is restricted to steps 50–200. PIQE + RRF is an inference-time variant of the PIQE checkpoints and is not trained separately, so it has no row of its own. The exploratory runs under the earlier 150-step budget, on other clusters, are summarized in Section G and are not repeated here.

Method	Seed	Step 0	Selected step	Dev macro EM	Last step	Wall-clock
Search-R1	13	22.75	190	37.00	200	18.6 h
Search-R1	21	22.58	200	36.92	200	18.9 h
Search-R1	42	22.50	180	37.17	200	18.9 h
PIQE	13	20.83	190	36.17	200	27.2 h
PIQE	21	20.83	200	36.00	200	27.2 h
PIQE	42	21.00	200	37.00	200	27.7 h
Hybrid	13	21.17	200	39.08	250[a]	36.5 h
Hybrid	21	21.25	180	36.75	200	28.8 h
Hybrid	42	21.25	160	37.00	200	28.8 h

G Exploratory Ablations

The four conditions of Table 4 differ in more than the two tabulated factors: adding passage-level RL also makes training-time selection stochastic rather than deterministic, and the passage gradient reaches the actor in PIQE but is detached in text-query passage RL, where it updates only an identity-initialized projection of the text query.

The other runs in this appendix use the earlier 150-step budget on Dev1200 with seed 13 unless stated otherwise. They were used to choose the configuration of the joint objective and are not part of the main comparison. The Search-R1 reference under the same budget and split is 36.50 macro EM, and none of the variants below exceeds it. They cover the training choices behind the current configuration (Table 15), the hidden-state representation used as the query (Table 16), and the candidate-set size and the passage-selection interface (Table 17). The passage-action weight was chosen earlier, under a 512-question monitor protocol that is not comparable to the tables here: a unit weight was clearly harmful (35.74 against 42.77 micro exact match at $\alpha = 0.03$), while the small weights were within that split's noise, so we fixed $\alpha = 0.03$; the weight was not swept again under the 200-step budget, and Table 15 reports the α comparison run on Dev1200.

Table 15: Training-choice sweeps on seed 13 (Dev1200, $\alpha = 0.03$ and $M = 20$ unless varied). [c]The Cluster B runs marked with this symbol stopped before step 150: temperature $0.1 \to 0.02$ at step 120, $\alpha = 0.01$ at step 130, $\alpha = 0.1$ at step 110, detached H_s at step 100; selection is over the steps reached. Single runs on one seed; the variability of these differences was not estimated.

Factor	Setting	Other settings	Cluster	Sel. step	Dev macro EM
Temperature	constant 0.02	$\eta_\phi = 10^{-5}$	B, 4 GPUs	100	33.42
	$0.05 \to 0.02$	$\eta_\phi = 10^{-5}$	B, 4 GPUs	80	35.25
	$0.05 \to 0.01$	$\eta_\phi = 10^{-5}$	B, 4 GPUs	90	31.92
	$0.1 \to 0.02$	$\eta_\phi = 10^{-5}$	B, 4 GPUs	100[c]	34.17
	constant 0.02	$\eta_\phi = 10^{-5}$	A, 4 GPUs	150	34.25
	$0.05 \to 0.02$	$\eta_\phi = 10^{-5}$	A, 4 GPUs	130	34.50
	$0.05 \to 0.01$	$\eta_\phi = 10^{-5}$	A, 4 GPUs	120	33.33
	$0.1 \to 0.02$	$\eta_\phi = 10^{-5}$	A, 4 GPUs	140	34.75
Projection LR η_ϕ	3×10^{-5}	$\tau : 0.05 \to 0.02$	A, 8 GPUs	110	33.83
	5×10^{-5}	$\tau : 0.05 \to 0.02$	A, 8 GPUs	150	35.83
	10^{-4}	$\tau : 0.05 \to 0.02$	A, 8 GPUs	150	34.25
Passage weight α	0.01	constant τ	B, 4 GPUs	80[c]	33.75
	0.03	constant τ	B, 4 GPUs	100	33.42
	0.1	constant τ	B, 4 GPUs	110[c]	32.17
Detach H_s	$\alpha = 0.03$, detached	constant τ	B, 4 GPUs	70[c]	31.33
Initialization	v2	constant τ	A, 4 GPUs	140	34.25
	v3	constant τ	A, 4 GPUs	150	34.25
	v3 + 25-step projection-only RL	constant τ	A, 4 GPUs	150	34.33

Table 16: Hidden-state representations on seed 13 (Dev1200, constant $\tau = 0.02$, $\eta_\phi = 10^{-5}$, $\alpha = 0.03$). Layer numbers count transformer blocks of the 36-layer policy. RET variants insert a deterministic marker token before or after the search query and use its hidden state, optionally mixed with the search-token mean with weight β on RET; "search mean" is the mean over the search tokens. Every variant has its own v2→v3 warm-start.

Representation	Pooling	GPUs	Step 0	Sel. step	Dev macro EM
Final layer (36)	search mean	4 GPUs, Cluster B	21.58	100	33.42
Layer 35	search mean	4 GPUs, Cluster B	21.33	90	33.25
Layer 30	search mean	4 GPUs, Cluster B	20.67	120	32.50
Layer 24	search mean	4 GPUs, Cluster B	19.33	140	29.17
Layer 18	search mean	4 GPUs, Cluster B	19.17	120	29.50
Layer 35 policy, layer-35 index	search mean	8 GPUs, Cluster A	19.83	150	34.92
Final layer	RET before query	8 GPUs, Cluster A	18.25	140	33.00
Final layer	RET after query	8 GPUs, Cluster A	16.75	150	32.42
Final layer	$0.3{\cdot}\text{RET} + 0.7{\cdot}\text{mean}$	8 GPUs, Cluster A	20.75	150	33.50
Final layer	$0.5{\cdot}\text{RET} + 0.5{\cdot}\text{mean}$	8 GPUs, Cluster A	20.17	130	35.00
Final layer	$0.7{\cdot}\text{RET} + 0.3{\cdot}\text{mean}$	8 GPUs, Cluster A	19.00	120	32.08

Table 17: Candidate-set size and passage-selection interfaces on Dev1200, seed 13, 150-step budget; learning-rate, temperature, initialization, layer, and RET-token variants are in Tables 15, 16. The three candidate-set sizes are not matched: the $M = 20$ row is the main configuration, with projection learning rate 5×10^{-5}, whereas $M = 50$ and $M = 100$ use the base rate 10^{-5} on the same hardware, so only those two form a controlled contrast; there the median wall-clock per optimizer step is 600 s at $M = 50$ and 913 s at $M = 100$. The candidate union keeps the text retriever's top five in the candidate set; the residual query retrieves and scores with the text embedding plus a scaled projection. The Search-R1 reference under the same conditions is 36.50.

Factor	Setting	Dev1200 macro EM
Candidate-set size	$M = 20$	35.83
	$M = 50$	34.92
	$M = 100$	35.92
Passage-selection interface	candidate union (text top-5)	35.75
	residual query, $\gamma = 0.3$	33.83

H OFFLINE QUERY-REPRESENTATION REPLAY

Table 18: Offline replay of the 3,882 search events of the warm-start validation split, the same split on which the projection's epoch was selected, so these events are not held out from the warm-start; the count is search events, not questions. The events were produced by the policy before any RL, and the projection is the warm-started projection, also before RL. For each event the search string, the preceding context, the cached hidden states, the corpus, and the candidate budget are held fixed, and only the query representation changes. GDR@k is gold-document recall, that is, the fraction of the question's gold supporting documents found in the top k; Any@5 is the fraction of events with at least one gold document in the top 5; the two are different quantities. AC@5 is the fraction of events with an answer-containing passage in the top 5. The last two rows are context-enriched text-retrieval controls rather than Search-R1 systems, and they are favoured by a metric that counts all gold documents of a question.

Query representation	GDR@5	GDR@20	Any@5	AC@5
Search string (text retriever)	20.70	25.11	38.10	28.90
Warm-start projection	19.23	24.16	35.21	28.44
Text $+ 0.5\times$ projection	21.08	25.58	38.64	29.21
Text $+ 0.5\times$ shuffled projection	20.25	24.62	37.30	28.41
Previous evidence + search string	25.59	31.92	45.62	30.71
Question + search string	42.03	52.23	69.50	46.81

Before RL, the warm-started projection retrieves less gold evidence than the text retriever it was distilled from, and adding it to the text embedding changes coverage by less than half a point, while a shuffled control changes it in the opposite direction by a similar amount. Prepending the question to the search string, with the same fixed retriever, roughly doubles gold-document recall at both cutoffs (2.0 times at 5 and at 20), while the other columns rise by less. This offline diagnostic is consistent with the candidate-generation deficit in Section 6.1, and it also indicates that the fixed text retriever has unexploited headroom when it is given more context, which is a different intervention from the one studied here.

Table 19: Same-query replay after RL, on the search events that the seed-13 step-190 actors produced on the frozen reporting partition (28,329 events for Search-R1 and 26,401 for PIQE, the validation side of a 95/5 split of the cached events). For every event the search string, the preceding context, the corpus, and the candidate budget are held fixed, and only the query representation changes. Text denotes retrieval obtained by encoding the actor-generated search string with the external Qwen3-Embedding-4B query encoder and querying the same fixed index; the projected columns replace that query by the projection of the same event's hidden states, either at its warm start or as trained in the seed-13 PIQE run at step 190, which is also the projection applied to the baseline's events since the baseline trains none of its own. Values are gold-document recall at 5 (GDR@5), the same metric as Table 18, measured on different events.

		Projected hidden states	
Event source (seed 13, step 190)	Text	warm-start	RL-trained
Search-R1	25.84	25.39	25.50
PIQE	24.79	25.64	25.54

Table 19 repeats the replay after RL, on the events that the selected seed-13 checkpoints produced on the reporting partition. On the baseline's own search strings the projection still retrieves slightly less gold evidence than the external encoder (25.39 at its warm start and 25.50 after RL, against 25.84), so training the projection moves it by about a tenth of a point. On PIQE's search strings the ordering reverses: the projected query recovers more gold evidence than the text embedding of the same string (25.64 and 25.54 against 24.79). The text column is itself lower on PIQE's events than on the baseline's (24.79 against 25.84), so the coverage difference of Section 6.1 involves the search strings the policy writes and not only the mapping applied to them; the two rows come from different actors and therefore from different event sets, and this diagnostic uses one seed.

Why the hidden-state query may fall short. We can only offer hypotheses, none of which this paper tests. The fixed passage space was trained to match text-query embeddings, so a policy state has to be mapped into a geometry that was not built for it, and a single linear map may be too weak to perform that alignment; we have not compared nonlinear projections. The reward is a sparse binary signal at the end of the trajectory, and the top-M truncation further narrows what it can influence. Finally, warm-start distillation transfers the teacher's behavior on the search string only, and the replay above is consistent with it preserving just part of the retrieval geometry that makes the teacher effective.

I Search Behavior and Cost

At the selected seed-13 checkpoints, the three systems issue a similar number of searches per trajectory on Dev1200: 2.29 for Search-R1, 2.07 for PIQE, and 2.16 for Hybrid, with 3.55, 3.34, and 3.46 generation rounds respectively. A round is one generation segment of the loop, and the loop also re-prompts a trajectory whose segment produced neither a valid search nor an answer, so rounds exceed searches by one final answering segment plus these retries; the search budget caps retrievals at four. Training cost differs more: on Cluster A, including the periodic Dev1200 evaluations, the 200-step baseline runs took 18.6–18.9 hours and the PIQE runs 27.2–27.7 hours, about 1.5 times as long per optimizer step. The Hybrid runs cost 28.8 hours at 200 steps, and 36.5 hours for seed 13, which was continued to 250 steps at the same cost per step. The additional cost of the joint objective comes from computing the hidden-state queries, retrieving candidates with them, and the passage-policy forward and backward passes; we have not broken it down further.

J Reproducibility Checklist

J.1 Data and model availability

All datasets, models, and tools are publicly available from the sources in Table 10. The in-domain partitions are derived from a previously evaluated subset of 11,136 questions of the 22,398-question development pool: Dev1200 keeps the 512 questions of an earlier monitor split and fills each dataset to 400 in an order fixed by a hash of the question identifier, the remaining 9,936 questions form the reserve, and the frozen reporting partition is the complement of that subset. Of its 11,262 questions, 11,197 appear in exploratory evaluations run before the partition was frozen and 65 do not. The question sets carry the licenses of their original releases: CC BY-SA 4.0 for HotpotQA, CC BY 4.0 for MuSiQue, CC BY-SA 3.0 for Natural Questions, and Apache 2.0 for TriviaQA; the 2WikiMultiHopQA and Bamboogle repositories are released under Apache 2.0 and MIT without a separate data license, and the wiki-18 corpus distribution carries no license metadata. Because the redistribution terms of the last three are not explicit, we release revisions, checksums, and download scripts instead of the data themselves. The policy and passage encoders were loaded from the Hugging Face snapshots `cdbee75` and `5cf2132`; our configurations name the models but do not pin a revision, so we report the snapshots that were actually used rather than a configured pin.

J.2 Training and evaluation procedure

1. Encode wiki-18 once with Qwen3-Embedding-4B and build an exact inner-product FAISS index.
2. Warm-start the query projection by distilling the text retriever on search events of the initial policy (Section F.2).
3. For each question, sample 8 trajectories from the current policy. After the search query has been completed and `</search>` has been generated, pool the hidden states of the tokens between the search tags, compute the query embedding, retrieve the top-M candidates from the fixed index, and sample an ordered list of k passages without replacement.
4. Insert the sampled passages into `<information>` and continue generation.
5. Compute the exact-match reward of the final answer and group-normalized advantages.

6. Update the policy and the projection with the joint objective. Search-token hidden states are not detached for the passage term, while candidate generation and retrieved text remain stop-gradient.
7. Evaluate Dev1200 every 10 steps with deterministic retrieval, select the checkpoint by Dev1200 macro EM, and evaluate the reporting partition once for the selected checkpoint.

J.3 Seeds and reporting

The main comparison uses seeds 13, 21, and 42, with the baseline and the joint run of each seed on the same cluster. Results on the frozen reporting partition are reported as means and standard deviations over these seeds. Exploratory sweeps use seed 13 only and are labeled as such. Answer-quality metrics are recomputed from saved per-question predictions, retrieval metrics from search-event logs, and runtime and hardware statistics from run metadata.

J.4 Artifact release

The planned release contains the training and evaluation code, the configuration file of every run reported here, the question-identifier lists that define Dev1200, the reserve partition, and the frozen reporting partition, the warm-started and RL-trained projection checkpoints, the stored per-question predictions and search-event logs, and the aggregation scripts that turn them into every table in this paper. A manifest stored with every run records the model names, the dataset and index paths, the retrieval and passage-policy settings, the tool versions, the container image, and the code revision. We give dataset revisions, split names, question counts, preprocessing scripts, and SHA-256 checksums rather than redistributing the question files or the corpus, so that the inputs can be reconstructed from their original sources. One artifact is only partly reproducible: the warm-started projection was produced from a working tree with uncommitted changes, so its configuration, metrics, and checkpoint are preserved but its exact code state is not. The release link is omitted here to preserve anonymity.