ONLINE ALIGNMENT OF RANKING MODEL VIA VARIANCE-WEIGHTED MULTI-GENERATOR FEEDBACK

Anonymous authors
Paper under double-blind review

ABSTRACT

We study online alignment of a pointwise ranker to the feedback of multiple downstream generators in retrieval-augmented generation (RAG). Existing work usually aligns the ranker to a *single* generator, which does not reflect the common deployment reality that one ranker serves several downstream models of different capabilities. When extending to scenarios involving multiple generators, how to aggregate the rewards from multiple generators becomes a key issue. A naive fix, averaging the utilities of all generators uniformly, dilutes the learning signal, because it gives equal weight to generators that are largely insensitive to document order. We propose **Variance-Weighted Multi-Generator GRPO** (VW-MG-GRPO). For each query we draw G top-K permutations with a Plackett–Luce policy, send them in parallel to a heterogeneous pool of generators (weak / mid / strong), and aggregate their utilities using each generator's *within-group standard deviation* as its weight. On Natural Questions, TriviaQA, and HotpotQA we validate the method and, more importantly, give a systematic analysis of the variance dynamics during training. We report three transferable findings: (i) variance weighting acts as an implicit per-query router rather than a smoother; (ii) the total variance signal decays over training, so variance weighting behaves as a self-annealing curriculum; and (iii) the effective per-query signal is $2.2\times$ that of uniform aggregation.

1 INTRODUCTION

Retrieval-augmented generation (RAG) couples a retriever/ranker with a downstream generator: the ranker selects and orders documents, and the generator reads them to produce an answer (Lewis et al., 2020; Guu et al., 2020; Izacard & Grave, 2021). The quality of the final answer therefore depends not only on which documents are retrieved but also on the order in which they are presented. In most pipelines, however, the ranker is trained offline with human relevance labels and then frozen. Relevance is a proxy: a document that a human annotator marks as relevant is not necessarily the one that lets a particular generator produce the correct answer, and the ordering that maximizes a labeled relevance metric need not maximize downstream accuracy. As a result, the ranker's notion of a good document order is never reconciled with what the generator actually needs.

Given the wide variety of downstream generators, we are exploring methods that do not require retraining the generators. Two families of methods try to bridge this gap with frozen generators. The first keeps the training of rankers *offline* and replaces or augments the relevance labels with a generator-derived signal, for example by distilling the generator's likelihood or answer quality into the ranker before freezing it (Izacard & Grave, 2020; Sachan et al., 2021). Such supervised fine-tuning (SFT) is stable and cheap, but it is still one-shot: the ranker is optimized against a fixed set of targets and cannot react to how the generator responds to its own, evolving outputs. The second family performs *online alignment*, using a downstream reward, which is attributable to the correctness or quality of the generator's answer, to supervise the ranker directly while it trains. Because the ranker's output is a discrete permutation, this is naturally cast as a reinforcement-learning problem: the ranker is a policy whose action is a document ordering and whose reward comes from the downstream reader. Wu et al. (2026) formulate reranking as a sequential decision process and optimize it against the reader's generation quality with a reference-anchored baseline, while Sun et al. (2026)

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.

model the reranker as an RL agent that adjusts both the order and the number of documents from LLM-output rewards. This line of work shows that aligning a ranker to downstream feedback beats relevance-only supervision. Crucially, though, every existing method, offline or online, assumes a *single*, fixed downstream generator. This assumption rarely holds in deployment: one ranking service typically feeds many generators of different families, sizes, and cost tiers, and the served set changes over time as models are added or retired. A ranker aligned to one generator can overfit to that generator's idiosyncrasies and need not help, or may even hurt, the others, and training a separate ranker per generator is impractical. What is missing is a paradigm that aligns *one* ranker to *several* generators at once.

We study exactly this setting, *multi-generator online alignment*, in which a single ranker is aligned online to a whole pool of downstream generators. We first instantiate a training pipeline that lifts single-generator online alignment to the multi-generator case. The pointwise ranker is turned into a stochastic policy over document orderings with a Plackett–Luce distribution over its relevance scores. For each query it samples a *group* of top-K permutations, and every permutation is sent to *all* generators in the pool, whose answers are scored against the gold answer to give a per-(permutation, generator) utility. The per-permutation rewards then drive a critic-free GRPO update (Guo et al., 2025), whose group-relative advantage removes the need for an extra value network.

This pipeline exposes the one question that the single-generator formulation never had to answer: how should utilities from the different generators be combined into the single scalar reward that GRPO consumes? Averaging with equal weight is a poor choice, because generators differ greatly in how much they respond to document order, so uniform averaging lets uninformative feedback dilute informative feedback. Our key insight is that, for a given query, how much a generator's utility *varies* across the sampled group of permutations is itself a measure of how much that generator consumes the ranker's output: zero variance means the generator ignores the ordering and has nothing to teach, whereas high variance means its feedback is exactly the signal to follow. We therefore weight each generator's contribution to the GRPO reward by its within-group variance, so sensitive generators dominate and insensitive ones are down-weighted per query. The process needs no extra reward model, learned mixture, or annotation, since the weights are pure within-group statistics.

We propose **Variance-Weighted Multi-Generator GRPO**(VW-MG-GRPO), shown in figure 1, which fuses the generators' per-permutation utilities into the GRPO reward using each generator's within-group variance as its weight. Beyond proposing this rule, we analyze the mechanism and find it is far more than a heuristic reweighting. Three properties, observed over a full training run, summarize its behavior:*(i)* the weights concentrate on one generator for most queries, so the rule acts as a per-query *router* rather than a smoother; *(ii)* as the ranker improves and generators increasingly agree, the total signal decays, giving a *self-annealing curriculum* that focuses on the residual hard queries at no cost; and *(iii)* aligning the weights with the generators actually being scored yields about a $2.2\times$ larger effective per-query signal than uniform averaging. These properties explain *why* the rule helps and form the second main contribution. On Natural Questions, TriviaQA, and HotpotQA, VW-MG-GRPO improves the average downstream metric over other baselines and transfers to held-out datasets and generators.

2 Related Works

Bridging the gap between retriever and generator. A recurring theme in RAG is that relevance, as scored by a retriever or ranker, is not the same as *utility* to the generator: the documents an information-retrieval metric prefers are not always the ones that help the reader answer, and language models are moreover sensitive to *where* in the context a document appears (Liu et al., 2024). A body of work therefore tries to close this retriever–generator gap. Some approaches make the reader robust to imperfect context, for example by joint retrieval-augmented instruction tuning of both components (Lin et al., 2024) or by teaching the reader to retrieve and critique on demand (Asai et al., 2024). Others reshape the retriever's input, such as query rewriting driven by the downstream model (Ma et al., 2023), or adapt black-box readers by ensembling over retrieved documents (Shi et al., 2024; Izacard et al., 2023). A third, closely related direction keeps both the retriever and the generator frozen and instead inserts a trainable *middleware* module between them that transforms the retrieved passages into a generator-friendly context: a seq2seq bridge that reranks and selects passages (Ke et al., 2024), a reward-driven contextual adapter that condenses them for a black-

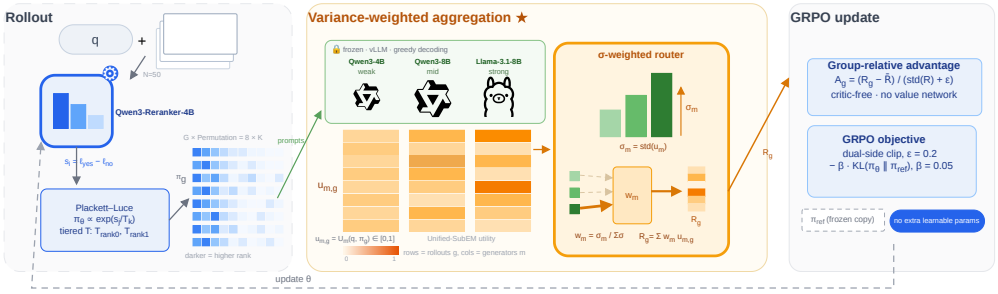


Figure 1: The overall framework of **Variance-Weighted Multi-Generator GRPO**.

box reader (Yang et al., 2023), a context compressor (Xu et al., 2024), or an augmentation-adapted retriever tuned to LM preferences (Yu et al., 2023). These middleware modules are typically trained against a single downstream reader. Our work targets the same gap but from the ranking side and, unlike these single-reader methods, aligns one ranker to the joint utility of *several* readers at once.

Online alignment of rankers to generator feedback. Policy-gradient methods (Williams, 1992; Schulman et al., 2017) have been used to fine-tune retrievers and generators against a downstream signal in RAG (Shi et al., 2024; Izacard et al., 2023). Most relevant to us is a recent line that aligns the *reranker* online to the reader's generation quality instead of static relevance labels: RRPO (Wu et al., 2026) casts reranking as a sequential decision process trained with LLM feedback, and DynamicRAG (Sun et al., 2026) treats the reranker as an RL agent driven by LLM-output rewards. These methods align the ranker to a single downstream generator. We adopt the same online-alignment objective but study the multi-generator case, where the open question is not *whether* to use downstream feedback but *how to aggregate* feedback from several generators of differing sensitivity.

Multi-teacher and multi-reward learning. Aggregating several teachers or reward sources appears in distillation (Hinton et al., 2015; You et al., 2017) and in RLHF (Christiano et al., 2017; Ouyang et al., 2022; Wu et al., 2023; Rame et al., 2023; Wang et al., 2024). These methods typically use a fixed uniform mixture or a separately learned mixture. Our contribution differs: we do *not* learn a fixed mixture. The weights are computed per query from the within-group rollout variance, so they adapt to each query at no extra parameter cost.

3 Method

3.1 Preliminaries

Given a query q, a candidate document set C_q, and a downstream generator pool $\mathcal{M} = \{m_1, \ldots, m_M\}$, the ranker π_θ produces a top-K permutation $\pi \in \mathcal{S}_K$. Let

$$U_m(q, \pi) \;=\; \mathrm{metric}\big(m(q, \pi),\, a_q^\star\big) \in [0, 1] \tag{1}$$

be the downstream utility of generator m under permutation π, where $a_q^\star$ is the gold answer. We wish to solve

$$\max_\theta \; \mathbb{E}_q \, \mathbb{E}_{\pi \sim \pi_\theta} \; \Phi\big(U_1, \ldots, U_M\big), \tag{2}$$

where Φ is a reward-aggregation function that maps the M per-generator utilities to the single scalar reward used for training.

3.2 Single-Generator Online Alignment

The base pipeline aligns the ranker online to *one* downstream generator, i.e. $\Phi = U_m$ for a chosen target m. This is the setting of prior work such as RRPO (Wu et al., 2026) and DynamicRAG (Sun et al., 2026). They used PPO or DPO training approaches, whose stability needed improvement, so we explored using the GRPO training process. We instantiate it as **Single-Gen-GRPO**. The details of the process are as follows:

Pointwise scorer. We use Qwen3-Reranker-4B (Zhang et al., 2025) as the pointwise ranker f_θ. For each pair (q, d_i) we build a prompt that asks the model whether the document is relevant, and take the logit difference of the `yes` / `no` tokens as the relevance score:

$$s_i \;=\; f_\theta(q, d_i) \;=\; \ell_{\text{yes}}(q, d_i) - \ell_{\text{no}}(q, d_i) \in \mathbb{R}. \tag{3}$$

For the top-N candidates (N=50) a chunked forward pass returns the score vector $\mathbf{s} \in \mathbb{R}^N$.

Plackett–Luce policy. We turn the pointwise scores into a sampleable, differentiable policy over top-K permutations (K=10) via a Plackett–Luce (PL) distribution (Plackett, 1975; Luce, 2005; Xia et al., 2008). Position k is sampled, without replacement, from the candidates not yet selected:

$$\pi_\theta(\pi_k = j \mid \pi_{<k}) \;=\; \frac{\exp(s_j/T_k)}{\sum_{j' \notin \pi_{<k}} \exp(s_{j'}/T_k)}, \qquad T_k = \begin{cases} T_{\text{rank0}} & k = 0 \\ T_{\text{rest}} & k \geq 1. \end{cases} \tag{4}$$

The layered temperature $(T_{\text{rank0}}, T_{\text{rest}})$ increases diversity only at the head: the PL denominator shrinks with position, so without a higher rank-0 temperature the G samples would agree at the top and the downstream utility variance, the very signal we exploit, would vanish. For each query we draw G permutations $\{\pi_g\}_{g=1}^G$ and record the old-policy per-position log-probabilities $\log \pi_{\theta_{\text{old}}}(\pi_g)$.

Group-relative advantage. With a scalar reward R_g for each rollout (here $R_g = U_m(q, \pi_g)$; the multi-generator forms appear in Section 3.3), we form a critic-free advantage:

$$A_g \;=\; \frac{R_g - \frac{1}{G}\sum_{g'} R_{g'}}{\text{std}\{R_{g'}\}_{g'=1}^G + \delta}. \tag{5}$$

Clipped objective with KL to a reference. At each update we recompute $\log \pi_\theta(\pi_g)$ under the current θ and form the importance ratio $r_\theta(\pi_g) = \exp[\log \pi_\theta(\pi_g) \log \pi_{\theta_{\text{old}}}(\pi_g)]$. The objective combines PPO-style clipping (Schulman et al., 2017) (ϵ=0.2) with a KL penalty (β=0.05) to a frozen reference ranker:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\,\mathbb{E}_{q,g}\Big[\min\big(r_\theta(\pi_g)A_g,\; \text{clip}(r_\theta(\pi_g), 1{-}\epsilon, 1{+}\epsilon)A_g\big)\Big] + \beta\,\text{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big). \tag{6}$$

Per-position credit assignment. The gradient enters the ranker only through $\log \pi_\theta(\pi_g)$, which the PL model expands as

$$\log \pi_\theta(\pi_g) = \sum_{k=0}^{K-1} \Big[\, s_{\pi_g(k)}/T_k - \underset{j \notin \pi_g(<k)}{\text{logsumexp}}\,\big(s_j/T_k\big)\Big]. \tag{7}$$

Differentiating with respect to a score s_j gives, at step k, $\frac{1}{T_k}\big(\mathbb{1}[j = \pi_g(k)] - \text{softmax}_j(\mathbf{s}_{\text{avail}(k)}/T_k)\big)$. Intuitively, the score of the document chosen at rank k is pushed up (when $A_g > 0$) while competing candidates are pushed down in proportion to their softmax probability. The score s_j then back-propagates through $f_\theta(q, d_j) = \ell_{\text{yes}} - \ell_{\text{no}}$ to the reranker's LM head and the rest of the network (full parameter fine-tuning). Thus the PL expansion yields per-position credit assignment for a pointwise ranker without any extra leave-one-out machinery.

3.3 The multi-generator paradigm

The Uniform Baseline. We now let the ranker serve the whole pool $\mathcal{M}$ at once. The rollout is unchanged except that every sampled permutation is scored by *all* generators: after the PL policy draws the G permutations, all $G \times M$ (permutation, generator) pairs are submitted in parallel to a vLLM server (Kwon et al., 2023), using one batched HTTP call per generator, which yields the utility $u_{m,g} = U_m(q, \pi_g)$ for every generator m and rollout g. This raises the central question of the multi-generator setting: how should the utility vector $(u_{1,g}, \ldots, u_{M,g})$ of each rollout be reduced to the single scalar R_g that GRPO consumes, i.e. how should Φ in Equation (2) be chosen?

The simplest choice averages the generators uniformly. **Uniform-MG-GRPO** sets

$$\Phi = \tfrac{1}{M}\sum_m U_m, \qquad \text{i.e.} \qquad R_g = \tfrac{1}{M}\sum_m u_{m,g}, \tag{8}$$

Uniform averaging treats all generators as equally informative, which, as we show next, wastes reward budget on generators that are insensitive to the ranker's output on a given query.

Variance-weighted multi-generator GRPO. Our method replaces uniform averaging with a per-query, variance-weighted aggregation. For each query, let $\mathbf{u}_m \in [0,1]^G$ collect the utilities of generator m over the G rollouts, and let $\sigma_m = \mathrm{std}(\mathbf{u}_m)$ be its within-group standard deviation. We aggregate with

$$w_m = \frac{\sigma_m}{\sum_{m'} \sigma_{m'} + \delta}, \qquad R_g = \sum_m w_m \, u_{m,g}. \tag{9}$$

If all $\sigma_m = 0$ (the permutations do not discriminate for any generator), we fall back to uniform weights $w_m = 1/M$. This defines Φ for VW-MG-GRPO.

Design intuition. σ_m measures how sensitive generator m is to changes in document order. A low variance means the generator ignores the ranking signal, so its utility carries no useful gradient for the ranker. Variance weighting therefore silences uninformative feedback and preserves the informative one, on a per-query basis. A single step nominally requires $B \times G \times M$ generator calls. Because (i) each generator is queried with one batched call and (ii) generators run concurrently, the measured wall-clock cost is about $1.05\times$ that of the single-generator baseline rather than $M\times$.

4 Main Experiments

4.1 Setup

Datasets and Metrics. Following the open-domain QA setting of Search-R1 (Jin et al., 2025) and RA-DIT (Lin et al., 2024), we mix three benchmarks, Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and HotpotQA (Yang et al., 2018), into a single training/evaluation pool with roughly 20k training and 2k in-distribution evaluation queries (about one third per dataset). To probe out-of-distribution generalization we additionally hold out two eval-only datasets: PopQA (Mallen et al., 2023) (long-tail entities, 1k) and 2WikiMultiHopQA (Ho et al., 2020) (structured two-hop, 1k). For most of datasets (NQ, TriviaQA, Hotpotqa and PopQA) the candidate pool is the top-100 passages retrieved by BM25 (Robertson & Zaragoza, 2009) over the Wikipedia DPR dump (Karpukhin et al., 2020). For 2WikiMultiHopQA, we use the official `distractor` pool of 10 passages (2 gold + 8 distractors). We report SubEM (sub-string exact match) of the generator's answer, both averaged over the three in-pool generators (SubEM/avg) and broken down per generator.

Ranker and Generator Pool. The ranker is Qwen3-Reranker-4B (Yang et al., 2025) used as a pointwise scorer (Section 3.2). All runs start from a single shared warm start: the reranker is pre-trained for 1 epochs on a balanced 500-query subset with a unified-SubEM objective, and every GRPO run (ours and all baselines) is initialized from this same checkpoint. The reference ranker used for the KL penalty is a frozen copy of this warm-started model. The in-distribution pool contains three generators of increasing strength: Qwen3-4B-Instruct (weak), Qwen3-8B (mid) (Yang et al., 2025), and Llama-3.1-8B-Instruct (strong) (Grattafiori et al., 2024). The strength is determined by the sub-em value in the downstream task. Each is served by an independent vLLM (Kwon et al., 2023) OpenAI-compatible endpoint. The trainer submits the $B \times G \times M$ (permutation, generator) pairs as one batched HTTP call per generator, and the generators run concurrently. All generators use greedy decoding (temperature 0) with 1024 new tokens. Two further generators, Qwen3-4B-base (same size as the weak in-pool model but non-instruct) and Qwen3-14B (same family, larger), are held out and served only at evaluation time (Section 4.3), which are never seen during training.

Implementation details. The per-generator utility is the unified-SubEM score $(\min(1,\ SubEM(\hat{a}, a^\star) + \mathrm{F1}(\hat{a}, a^\star)))$, which is denser than plain SubEM while still keyed to the gold answer. Utilities are combined across generators by the aggregation rule under test: `mean` for the single-generator ($\Phi = U_m$) and uniform baselines, and our variance weighting (Equation (9)) for VW-MG-GRPO. All three methods share the same warm start and the same training pool and differ *only* in the reward source and aggregation. The complete implementation details of GRPO is presented in the appendix A.

4.2 Main Results

Table 1 reports in-distribution SubEM over the three-generator pool, organized into three blocks: the untrained ranker, single-generator alignment, and multi-generator alignment. The results show

Table 1: In-distribution evaluation over the three-generator pool (SubEM). "/avg" averages over the three in-pool generators; the remaining columns are per generator.

Method	Reward source	SubEM/avg	Qwen3-4B-ins	Qwen3-8B	Llama3.1-8B-ins
Before alignment					
No-train (raw pointwise)	—	0.4254	0.3483	0.4134	0.5145
No-train (warmup, step 0)	—	0.4240	0.3433	0.4244	0.5045
Single-generator alignment					
Single-Gen-GRPO	Qwen3-4B-ins	0.4244	0.3504	0.4144	0.5085
	Qwen3-8B	0.4351	0.3483	0.4264	0.5305
	Llama3.1-8B-ins	0.4311	0.3483	0.4244	0.5205
RRPO	Qwen3-4B-ins	0.4261	0.3514	0.4204	0.5065
	Qwen3-8B	0.4254	0.3463	0.4284	0.5015
	Llama3.1-8B-ins	0.4254	0.3473	0.4174	0.5115
Multi-generator alignment					
SFT (matched data, 10k)	3-gen (imitation)	0.4341	0.3463	0.4454	0.5105
RRPO extended to pool	3-gen uniform	0.4304	0.3524	0.4264	0.5125
Uniform-MG-GRPO	3-gen uniform mean	0.4458	**0.3604**	0.4474	0.5295
VW-MG-GRPO (ours)	3-gen var-weighted	**0.4511**	0.3533	**0.4655**	**0.5345**

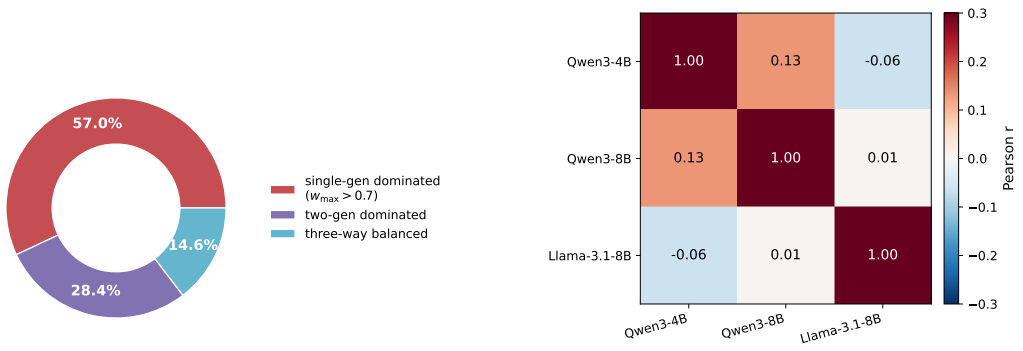
Table 2: Transfer to held-out generators (SubEM).

Method	4B-base	14B
No-train (pointwise)	0.3493	0.4444
No-train (warmup)	0.3483	0.4484
Single-Gen (Qwen3-4B-ins)	0.3693	0.4534
Single-Gen (Qwen3-8B)	0.3713	0.4514
Single-Gen (Llama3.1-8B-ins)	0.3483	0.4424
RRPO (Qwen3-4B-ins)	0.3563	0.4464
RRPO (Qwen3-8B)	0.3523	0.4564
RRPO (Llama3.1-8B-ins)	0.3493	0.4494
SFT (matched data)	0.3723	0.4644
RRPO extended to pool	0.3503	0.4534
Uniform-MG-GRPO	0.3834	0.4714
VW-MG-GRPO (ours)	**0.3913**	**0.4764**

Table 3: Transfer to held-out datasets (SubEM/avg).

Method	PopQA	2Wiki
No-train (pointwise)	0.1570	0.4056
No-train (warmup)	0.1473	0.3979
Single-Gen (Qwen3-4B-ins)	0.1573	0.4039
Single-Gen (Qwen3-8B)	0.1573	0.4073
Single-Gen (Llama3.1-8B-ins)	0.1613	0.4080
RRPO (Qwen3-4B-ins)	0.1580	0.4026
RRPO (Qwen3-8B)	0.1600	0.4070
RRPO (Llama3.1-8B-ins)	0.1586	0.4059
SFT (matched data)	0.1603	0.4053
RRPO extended to pool	0.1600	0.4053
Uniform-MG-GRPO	0.1613	0.4073
VW-MG-GRPO (ours)	**0.1643**	**0.4126**

that: **(1)Single-generator alignment helps its own target but transfers unevenly.** Aligning to one generator mostly benefits that generator, this is evident in both Single-Gen-GRPO and RRPO. The other two generators may or may not see improvements, but in any case, those improvements will not be as significant as those made to the target generator. This aligns with our motivation for proposing multi-generator alignment. We take RRPO as the representative single-generator online-alignment baseline and do not separately compare against DynamicRAG (Sun et al., 2026): it also aligns a reranker to one generator's answer quality, but through a different action space (jointly learning the *number* k and order of documents via iterative, generator-in-the-loop DPO), which is not convenient to make a direct comparison. **(2)Under multi-generator training, GRPO beats the PPO-style RRPO extension.** Uniform-MG-GRPO reaches 0.4458/avg, well above the direct multi-generator extension of RRPO (0.4304), which even falls below the best single-generator baseline. With the reward source held identical (all three generators), this gap isolates the optimization paradigm and shows GRPO's group-relative objective is better suited to this task than RRPO's preference-based updates. **(3)Variance weighting beats uniform averaging.** Replacing the uniform mean with our variance weighting lifts SubEM/avg from 0.4458 to $\mathbf{0.4511}$. This demonstrates the superiority of variance-weighted design, and we will analyze the underlying mechanism in the next section. **(4)RL suits this task better than SFT.** Matched-data SFT (0.4341) trails both GRPO baselines despite the same warm start and comparable data, because it imitates fixed scores and cannot react to how the generators respond to the ranker's own outputs. Optimizing the downstream reward online, in a closed loop, is the more appropriate paradigm.



(a) Per-query weight concentration.

(b) Correlation between the variances.

Figure 2: Per-query weight concentration (left) and cross-generator variance correlation (right).

4.3 Transfer to held-out generators and datasets

Held-out Generators. A ranker aligned to the pool should also help generators it was never trained against. We evaluate every trained ranker, unchanged, on two held-out generators, Qwen3-4B-base (same size as the weak in-pool model but non-instruct) and Qwen3-14B (same family, larger), in Table 2. VW-MG-GRPO transfers best to both (0.3913 and 0.4764), clearly above uniform aggregation and every single-generator baseline. The latter give only small or inconsistent gains, as expected when a ranker is tuned to one generator's idiosyncrasies. Our method produces a document ordering that generalizes across generators rather than overfitting to the training pool.

Held-out Datasets. We further evaluate the same rankers on two out-of-distribution datasets, PopQA (Mallen et al., 2023) (long-tail entities) and 2WikiMultiHopQA (Ho et al., 2020) (structured two-hop), reporting the pool-averaged SubEM in Table 3 (per-generator breakdown in Section B). VW-MG-GRPO again ranks first on both (0.1643 on PopQA, 0.4126 on 2Wiki), showing that the gains carry over to unseen domains and are not an artifact of the training distribution.

5 Analysis: Variance Dynamics of Multi-Generator Feedback

This section shows that variance weighting is not an arbitrary heuristic but a principled mechanism, by establishing three claims: **(i)** it acts as an implicit per-query *router* rather than a smoother; **(ii)** the total variance signal *decays* over training, so the rule behaves as a self-annealing *curriculum*; and **(iii)** it delivers a $2.2\times$ larger effective per-query learning signal than uniform aggregation.

5.1 Claim (i): variance weighting is a per-query router, not a smoother

A smoother would blend all generators on every query, while a router would, per query, concentrate the reward on whichever generator actually responds to the ranker. We give three pieces of evidence, aggregate weights, the per-query weight distribution, and its evolution, plus a worked example.

The weights track generator sensitivity. We first note *why* a router is even possible here: rollout variance is not uniform across generators. Table 4 reports the per-generator variance over all steps: mean σ rises monotonically from the weak to the strong generator, because stronger generators turn changes in document order into changes in the answer more often in our setting. Consequently the aggregation weights, averaged over queries, inherit the same order, weak $0.28 <$ mid $0.31 <$ strong 0.40 (Table 5). So variance weighting reallocates gradient flow toward the more sensitive generators *without any explicit label*. This already rules out the smoother interpretation at the population level: a smoother would assign all three the uniform weight $1/M = 0.33$.

Per query, the weight is concentrated. Population averages could in principle hide a smoother that mildly tilts every query. They do not. Looking at individual queries with a non-degenerate signal, 57.0% place more than 0.7 of the weight on a *single* generator, a further 28.4% split be-

Table 4: Per-generator variance across 800 steps.

Generator	Str.	mean σ	p90	max
Qwen3-4B	weak	0.0746	0.331	0.500
Qwen3-8B	mid	0.0864	0.430	0.500
Llama3-8B	strong	**0.1196**	**0.433**	0.500

Table 5: Mean aggregation weight $\bar{w}_m$.

Generator	$\bar{w}_m$
Qwen3-4B	0.28
Qwen3-8B	0.31
Llama3-8B	**0.40**

Table 6: Weight concentration across training (ESS $= 1/\text{HHI}$).

Phase	mean HHI	median HHI	ESS
Steps 1–100	0.689	0.547	1.72
Steps 401–500	0.750	0.861	1.54
Steps 701–800	0.753	0.858	**1.53**

Table 7: Effective per-query signal: $\boldsymbol{w}$ vs. uniform aggregation.

Aggregation	mean σ	median
Uniform	0.158	0.144
Variance-weighted	**0.307**	**0.335**
Relative gain	**+120%**	**+130%**

tween two (third below 0.1), and only 14.6% are genuinely three-way balanced (Figure 2a). On the majority of queries the reward is sourced from one generator, which is the behavior of a router.

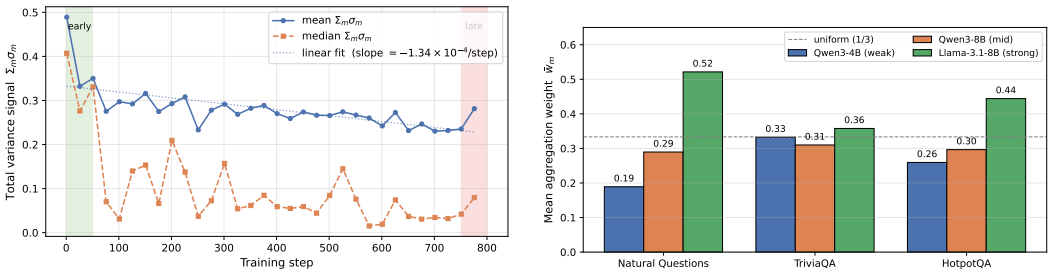
The routing sharpens with training. The concentration is not a transient of initialization. Measuring it with the Herfindahl–Hirschman index $\text{HHI} = \sum_m w_m^2 \in [1/M, 1]$ (equivalently the effective pool size $\text{ESS} = 1/\text{HHI}$), Table 6 show HHI rising from 0.69 to 0.75 (ESS shrinking from 1.72 to 1.53) and then plateauing: the router commits harder as the ranker improves, but stabilizes at an effective pool of about 1.5 rather than collapsing onto one generator.

A worked example. A single query makes the routing concrete and, at the same time, refutes the tempting shortcut "just keep the strongest generator." For the two-hop question *"Who was the actor in the 1979 film Firepower, known to often co-star with his wife Anne Jackson?"* (gold: *Eli Wallach*) at step 54, the variance triple is σ_{q4}=0.484 (weak, responds), σ_{q8}=0.000 (mid, all 8 rollouts identical), and σ_{l8}=0.484 (strong, responds). Uniform aggregation spends a full third of its reward budget on the mid generator, whose flat utility contributes no gradient, thereby diluting the two informative signals. Variance weighting sends that weight to zero and splits the reward $0.5/0/0.5$ between the weak and strong generators, raising the effective signal from 0.323 to 0.484 ($+50\%$). Here the mid generator is uninformative while the weak one is not, so *neither* uniform averaging *nor* a keep-the-strongest rule recovers the right signal, and only per-query routing does. Such cases are not rare: they are the 57% single-generator-dominated mass. Indeed, the generators' variances are nearly uncorrelated across queries (pairwise Pearson $\in [-0.16, 0.24]$; Figure 2b), so each illuminates a largely different subset of hard queries and the full pool is genuinely needed.

5.2 Claim (ii): the signal decays, giving a self-annealing curriculum

We measure the total per-query signal by $\Sigma_m \sigma_m$. A larger value means stronger evidence that at least one generator is still sensitive to the ranker's output on that query. Figure 3a show this signal contracting as training proceeds. A linear fit gives a slope of -1.33×10^{-4} per step(-0.107 over 800 steps). Crucially, the *median* falls far faster than the mean (-83% vs. -37%): the distribution collapses toward zero variance for most queries while a long tail of still-ambiguous queries keeps the mean up. In other words, as the ranker converges toward orderings on which the generators agree, those queries stop producing gradient, and the effective learning signal concentrates on the shrinking residual of genuinely hard queries, which is exactly what a hand-designed curriculum would engineer, here emerging for free from the aggregation rule.

This self-annealing behavior is a property of variance weighting specifically, not of multi-generator training in general. Under uniform aggregation the reward of a rollout is $\frac{1}{M}\sum_m u_{m,g}$, so a query keeps contributing gradient as long as *any* generator's mean utility differs from the group average, including easy queries that one weak generator still gets wrong at random. Concretely, the group-relative advantage that drives learning is nonzero whenever $\text{std}_g(\frac{1}{M}\sum_m u_{m,g}) > 0$, and this quantity does not vanish as the ranker improves: uniform weighting has no term that detects "all gen-



(a) **Total variance signal decays across training.** Mean (solid) and median (dashed) of $\Sigma_m \sigma_m$ with a linear fit.

(b) **Domain-conditional routing.** Mean per-generator weight on NQ / TriviaQA / HotpotQA (dashed line: uniform).

Figure 3: Variance signal decay (left) and domain-conditional weight composition (right).

erators have stopped responding to the ordering" and therefore cannot down-weight solved queries. Its effective signal stays roughly flat over training rather than contracting, so it spends late-stage updates on queries that carry no ordering information, which is the opposite of a curriculum.

5.3 Claim (iii): variance weighting amplifies the effective signal 2.2×

Finally we quantify how much signal actually reaches the ranker. For a query, define the effective signal under each aggregation as the weighted spread of the generator utilities:

$$\text{Uniform: } \bar{\sigma} = \frac{1}{M} \sum_m \sigma_m, \qquad \text{VW: } \sigma_{\text{vw}} = \sum_m w_m \sigma_m = \frac{\sum_m \sigma_m^2}{\sum_m \sigma_m}. \tag{10}$$

These are the (unnormalized) power-mean of order 1 and 2 of $\{\sigma_m\}$, so by the power-mean (Cauchy–Schwarz) inequality $\sigma_{\text{vw}} \geq \bar{\sigma}$ *always*, with equality iff all σ_m are equal; the gap widens as the σ_m become more unequal, and reaches its maximum factor of M when a single generator carries all the signal. Variance weighting is thus guaranteed to convey at least as much per-query signal as uniform averaging, and strictly more whenever the generators differ, which, by Claim (i), is the common case.

Empirically the guaranteed gain is large: averaged over all queries the effective signal rises from 0.158 (uniform) to 0.307 (VW), a $2.2\times$ amplification in the mean ($+120\%$) and $+130\%$ in the median (Table 7). This closes the loop with Claim (i): the amplification is not a coincidence but the direct consequence of routing the weight onto the generators whose utility actually varies.

5.4 Summary

The three claims compose into one statement: variance weighting is a per-query, per-step router (Claim i) that, as the ranker improves, anneals its own learning signal onto the residual hard queries (Claim ii), and that provably and empirically amplifies the effective signal by $2.2\times$ over uniform aggregation (Claim iii), and all with no extra learnable parameters. A further by-product, that the same pool self-organizes into different routing patterns across datasets (Figure 3b: strong-dominant on NQ and HotpotQA, balanced on TriviaQA), shows the mechanism also performs domain-aware routing with no domain label.

6 Conclusion

We proposed VW-MG-GRPO, which uses the within-group variance of downstream generator feedback as the cross-generator aggregation weight, giving a per-query route to the most sensitive generator for online alignment. Through a systematic analysis of the training dynamics we showed that variance weighting is at once an implicit strength-aware router (Section 5.1), a self-annealing curriculum (Section 5.2), and a $2.2\times$ signal amplifier (Section 5.3). Empirically the method improves the average SubEM over the single-generator, uniform, and SFT baselines, and the benefits can transfer to unseen datasets and generators.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs and to provide critical ingredients for proving the mathematical claims in the method section. We have not used generative AI tools to generate synthetic datasets, to propose or refine hypotheses, or to design research methodology or experiments, and the remaining required-disclosure tasks (e.g., developing theoretical models, implementing methods, cleaning or reformatting datasets, qualitative or thematic data analysis) are not applicable to this work. Additionally, we used generative AI tools for recommended-disclosure tasks, namely to draft parts of the paper and to aid or polish the writing. We did not use generative AI tools for retrieval and discovery (e.g., finding related work) or for research ideation and execution. We have reviewed all AI-assisted work: every AI-assisted proof step and mathematical claim was checked and verified by the authors, and all AI-drafted or AI-polished text was read and edited by the authors for correctness and accuracy. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies multi-generator online-alignment and conducts all experiments on publicly available benchmark datasets, using publicly released language models. It does not involve human subjects, the collection or release of new data, or personally identifiable or sensitive information. We do not foresee harmful applications, discrimination, fairness, privacy, security, or legal-compliance concerns arising directly from our methods or findings. We therefore believe this work raises no issues that conflict with the ICLR Code of Ethics.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

Gautier Izacard and Edouard Grave. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584*, 2020.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pp. 874–880, 2021.

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.

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

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

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 *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Bridging the preference gap between retrievers and llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10438–10451, 2024.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. Ra-dit: Retrieval-augmented dual instruction tuning. In *International Conference on Learning Representations*, volume 2024, pp. 19138–19162, 2024.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2005.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 5303–5315, 2023.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 9802–9822, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems*, 36:71095–71134, 2023.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009.

Devendra Sachan, Mostofa Patwary, Mohammad Shoeybi, Neel Kant, Wei Ping, William L Hamilton, and Bryan Catanzaro. End-to-end training of neural retrievers for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6648–6662, 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)*, pp. 8371–8384, 2024.

Jiashuo Sun, Xianrui Zhong, Sizhe Zhou, and Jiawei Han. Dynamicrag: Leveraging outputs of large language model as feedback for dynamic reranking in retrieval-augmented generation. *Advances in Neural Information Processing Systems*, 38:168661–168692, 2026.

Haoxiang Wang, Yong Lin, Wei Xiong, Rui Yang, Shizhe Diao, Shuang Qiu, Han Zhao, and Tong Zhang. Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8642–8655, 2024.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Yuhang Wu, Xiangqing Shen, Fanfan Wang, Cangqi Zhou, Zhen Wu, Xinyu Dai, and Rui Xia. Optimizing rag rerankers with llm feedback via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30474–30490, 2026.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems*, 36:59008–59033, 2023.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pp. 1192–1199, 2008.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, volume 2024, pp. 43478–43502, 2024.

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

Haoyan Yang, Zhitao Li, Yong Zhang, Jianzong Wang, Ning Cheng, Ming Li, and Jing Xiao. Prca: Fitting black-box large language models for retrieval question answering via pluggable reward-driven contextual adapter. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 5364–5375, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1285–1294, 2017.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. Augmentation-adapted retriever improves generalization of language models as generic plug-in. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2421–2436, 2023.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

A Implementation Details

We optimize the GRPO objective of Section 3.2 with group size $G{=}8$, rollout top-$K{=}10$ over $N{=}50$ candidates, and layered Plackett–Luce temperatures $(T_{\text{rank0}}, T_{\text{rest}}){=}(1.5, 1.0)$ to keep the sequence diversity. We use `ppo_epochs`$=1$ (one update per rollout), PPO clip range $\epsilon{=}0.2$, and a KL coefficient $\beta{=}0.05$ to the frozen reference. Optimization uses AdamW at a constant learning rate 5×10^{-6}, per-rank micro-batch 8 with gradient accumulation 4, trained for up to 800 steps with evaluation every 50 steps. Training runs under 4-way DDP (the remaining GPUs host the generator servers). The best checkpoint is selected by the in-distribution SubEM/avg. We fix `seed`$=42$, and the per-generator rollout variances used in Section 5 are dumped online during the main run. At each stage the ranker sees the top-N of this pool: $N{=}50$ candidates during GRPO rollout (from which the Plackett–Luce policy samples top-$K{=}10$ permutations) and $N{=}20$ candidates at evaluation (top-$K{=}5$), matching the Search-R1 evaluation protocol.

B Full per-domain × per-generator results

Table 8 gives the per-generator breakdown of the held-out-dataset evaluation summarized in Table 3.

Table 8: Held-out-dataset transfer, per generator (SubEM). "avg" is the mean over the three in-pool generators.

	PopQA				2WikiMultiHopQA			
Method	avg	Qwen3-4B-ins	Qwen3-8B	Llama3.1-8B-ins	avg	Qwen3-4B-ins	Qwen3-8B	Llama3.1-8B-ins
No-train (pointwise)	0.1570	0.088	0.138	0.245	0.4056	0.395	0.413	0.4269
No-train (warmup)	0.1473	0.071	0.134	0.237	0.3979	0.382	0.394	0.418
Single-Gen (Qwen3-4B-ins)	0.1573	0.080	0.135	0.257	0.4039	0.388	0.411	0.413
Single-Gen (Qwen3-8B)	0.1573	0.085	0.133	0.254	0.4073	0.392	0.417	0.413
Single-Gen (Llama3.1-8B-ins)	0.1613	0.084	0.143	0.257	0.4080	0.387	0.411	0.426
RRPO (Qwen3-4B-ins)	0.1580	0.085	0.142	0.247	0.4026	0.393	0.411	0.404
RRPO (Qwen3-8B)	0.1600	0.085	0.146	0.249	0.4070	0.393	0.417	0.411
RRPO (Llama3.1-8B-ins)	0.1586	0.083	0.143	0.250	0.4059	0.392	0.410	0.416
SFT (matched data)	0.1603	0.080	0.150	0.251	0.4053	0.389	0.399	0.428
RRPO extended to pool	0.1600	0.090	0.139	0.251	0.4053	0.394	0.412	0.405
Uniform-MG-GRPO	0.1613	0.078	0.154	0.252	0.4073	0.397	0.426	0.407
VW-MG-GRPO (ours)	**0.1643**	0.073	0.153	**0.267**	**0.4126**	0.395	0.414	**0.429**