LEARNING TO RETRIEVE VIA REINFORCEMENT LEARNING IN EMBEDDING SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Dense retrieval models are typically trained with contrastive objectives that learn effective representations but do not directly optimize retrieval metrics or downstream task performance. To address this problem, we introduce **RELER** (**RE**inforcement **LE**arning for **R**etrieval), a reinforcement learning framework that enables existing embedding models to *learn to retrieve* directly in embedding space and align to task-specific rewards. We train RELER by sampling unit-length query and document embedding actions from von Mises–Fisher (vMF) distributions centered on normalized encoder outputs, scoring the resulting retrieval or downstream outcomes as rewards, and updating the encoder with REINFORCE using a leave-one-out baseline (RLOO). As exploration in the high-dimensional embedding space is prone to sampling noise, we further propose conditional-mean projection (CMP), which projects each sampled embedding onto the low-dimensional subspace spanned by its encoder output and the candidate embeddings it is compared against, reducing noise in the policy gradient while preserving its expectation. We evaluate RELER on BRIGHT, a benchmark with reasoning-intensive queries that remain challenging for existing embedding models. RELER consistently outperforms InfoNCE and LambdaLoss in average nDCG@10 when post-training BGE-M3 and Qwen3-Embedding backbones. We further evaluate downstream utility through retrieval-augmented generation (RAG), where we adapt only the query encoder while keeping the document index and generator fixed. Across seven QA datasets, jointly optimizing retrieval and answer rewards improves both average retrieval performance and answer quality in RAG.

1 INTRODUCTION

Dense retrieval models enable large-scale semantic retrieval by encoding queries and documents into a shared embedding space, where relevance is modeled by vector similarity (Karpukhin et al., 2020; Zhang et al., 2025). They support applications including web search (Huang et al., 2013), retrieval-augmented generation (RAG) (Lewis et al., 2020), agentic search (Jin et al., 2025a), and memory recall for long-horizon agents (Packer et al., 2023). Despite these varied uses, most text embedding models are trained with large-scale contrastive learning, encouraging relevant query–document pairs to have higher similarity than irrelevant ones (Zhan et al., 2020; Wang et al., 2022). This training objective, however, is only indirectly related to retrieval metrics such as nDCG or downstream task performance such as answer accuracy in RAG. Reducing the contrastive loss therefore does not necessarily translate into better retrieval quality or downstream utility (Zhang et al., 2024).

This objective mismatch is analogous to that in language model alignment: next-token prediction does not directly optimize human preferences or task success. Reinforcement learning (RL) provides a principled approach by optimizing rewards derived from human preferences or task outcomes (Ouyang et al., 2022; DeepSeek-AI, 2025). This motivates using RL to align retrieval models directly with retrieval and downstream metrics, as RL can optimize such non-differentiable objectives by sampling retrieval decisions, evaluating their outcomes with the target metrics, and estimating policy gradients without differentiating through the retrieval process or the downstream evaluator (Williams, 1992). Recent work has explored RL for embedding and retrieval models through policies over generated text (Sun et al., 2026; Jiang et al., 2026) or discrete document selections (Salemi

Code: `https://anonymous.4open.science/r/RELER-F029`

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.

et al., 2024; Liu et al., 2025c; Zhang et al., 2026). These policies explore text or document choices; the query and document vectors used for similarity scoring are not themselves sampled as actions.

In this work, we revisit reinforcement learning for dense retrieval by treating embeddings as learnable retrieval actions. Because their similarities determine which documents are retrieved, exploring alternative embeddings allows task feedback to guide the representations themselves. We introduce **RELER** (REinforcement LEarning for Retrieval), a framework for post-training embedding models to *learn to retrieve*: sample embedding actions, evaluate the retrieval results or downstream answers they produce, and update the encoder to favor actions with higher rewards. The framework explores both query and document embeddings, and also supports query-only adaptation against a fixed document index. Inference retains deterministic single-vector retrieval.

To turn this view into a practical learning algorithm, RELER places a von Mises–Fisher (vMF) policy around each normalized encoder output, enabling exploration on the unit sphere used for cosine retrieval (Bendada et al., 2025). To optimize retrieval, RELER independently samples query and document embeddings and uses product rollouts (Kuntz et al., 2022) to pair every sampled query with every document bundle. Averaging rewards across these pairings gives each action feedback from multiple retrieval outcomes without additional encoder passes. REINFORCE with a leave-one-out baseline (Kool et al., 2019) then converts this feedback into updates to the shared encoder. High-dimensional exploration, however, also introduces random components that leave retrieval scores unchanged but add gradient noise. Building on document-space projection (Wang et al., 2019), we develop *conditional-mean projection* (CMP): for each reward term, sampled actions are projected onto the subspace spanned by their policy mean and the directions that determine the reward. Applied only in gradient computation, CMP averages out this irrelevant randomness while preserving the relevant scores and the expected policy gradient.

We evaluate RELER as a post-training method for existing embedding models, first on reasoning retrieval and then on downstream RAG tasks. Existing embeddings receive extensive training on tasks represented in MTEB (Muennighoff et al., 2023) and already perform strongly (Chen et al., 2024; Zhang et al., 2025). We therefore focus on BRIGHT (Su et al., 2025), whose reasoning-intensive queries expose remaining weaknesses, to test whether limited additional supervision can improve these models. Using fewer than 5k queries, RELER consistently outperforms supervised baselines trained on the same data across BGE-M3 and Qwen3-Embedding backbones. For downstream RAG, we test whether task feedback improves answer quality by adapting only the query encoder with the index and generator frozen. Combining retrieval and answer rewards improves average retrieval and answer quality over the pretrained encoder across seven QA datasets.

2 Related Work

Training dense retrievers and embedding models. Contrastive learning dominates dense retrieval: dual encoders learn a shared geometry by scoring relevant query–document pairs above negatives (Karpukhin et al., 2020), while hard-negative mining and improved sampling sharpen these comparisons (Xiong et al., 2021; Qu et al., 2021). General-purpose embedding models scale this recipe with weak supervision, multi-stage and multi-task training, and multilingual or multi-function objectives (Wang et al., 2022; Li et al., 2023; Xiao et al., 2024; Chen et al., 2024). LLM-based models adapt decoder-only backbones through bidirectional attention or specialized pooling, together with synthetic pairs, instructions, and contrastive fine-tuning (BehnamGhader et al., 2024; Wang et al., 2024; Lee et al., 2025; Muennighoff et al., 2025; Zhang et al., 2025). These methods provide the pretrained dual encoder that RELER subsequently adapts with task-level rewards. ReasonIR, RaDeR, and DIVER develop retrievers for reasoning-intensive queries, while ReasonEmbed uses reasoning-aware synthetic data and adaptive contrastive training (Shao et al., 2025; Das et al., 2025; Long et al., 2025; Chen et al., 2026).

Reward-based adaptation of retrievers and embedding models. Existing RL approaches differ in the action they sample and how rewards reach the encoder. ROPG-RL samples a personal document from a categorical policy and uses downstream generation quality as reward (Salemi et al., 2024). Retrieval-GRPO assigns multi-objective rewards to dynamically retrieved products and optimizes the dense retriever through their similarity scores (Liu et al., 2025c), whereas HARR samples ordered document lists from a Plackett–Luce policy and uses answer-level rewards (Zhang et al.,

2026). These policies reshape retrieval geometry through document selections or scores. Another approach uses generated reasoning: GRACE pools rationale-conditioned states into embeddings, while Embed-RL trains a reasoner using feedback from a separate embedder (Sun et al., 2026; Jiang et al., 2026). For large action sets, Bendada et al. (2025) sample a state embedding from a vMF distribution and select actions through nearest-neighbor search. RELER independently samples both query and document embeddings and uses retrieval rewards with CMP to update a shared encoder. Deployment uses deterministic single-vector encoder outputs. Appendix F discusses additional retrieval methods and the prior work underlying our policy-gradient estimator.

3 METHOD

We present the embedding-space policy, product-rollout RLOO, and conditional-mean projection, then describe fixed-index query adaptation and the training rewards, as illustrated in Figure 1.

3.1 RETRIEVAL AS AN EMBEDDING-SPACE POLICY

Let $h_x = f_\theta(x) \in \mathbb{S}^{d-1}$ be a normalized text representation. The query and document roles share the trainable encoder. At deployment, the similarity of query and document is defined by $h_q^\top h_d$. Small changes in embedding direction can change the score, therefore influencing retrieval outcomes or downstream task performance. Exploring around the encoder outputs lets retrieval or answer feedback guide the representations, even when the reward is not differentiable. To explore on the unit sphere, we let each representation define a von Mises–Fisher (vMF) policy during training (Mardia & Jupp, 2000; Bendada et al., 2025):

$$\pi_\theta(e_x \mid x) = C_d(\kappa) \exp(\kappa h_x^\top e_x), \qquad e_x \in \mathbb{S}^{d-1}. \tag{1}$$

Here C_d normalizes the density and $\kappa > 0$ controls exploration. We specify κ through the expected alignment $\rho = \mathbb{E}[h_x^\top e_x]$; larger ρ concentrates actions around the deterministic representation. We describe the detailed sampling procedure in Appendix B.1.

A training record (q, D, y) contains a query, valid candidates $D = (d_1, \ldots, d_n)$, and task supervision y, such as relevance labels for retrieval or reference answers for RAG. The policy $\pi_\theta(\cdot \mid q, D)$ samples the query action e_q and each document action in $E_D = (e_{d_1}, \ldots, e_{d_n})$ independently. The training objective is

$$\mathcal{J}_\kappa(\theta \mid q, D, y) = \mathbb{E}_{\pi_\theta(\cdot|q,D)}[R(e_q, E_D; y)]. \tag{2}$$

Here R is a task-defined scalar reward that evaluates the retrieved document list or a downstream output against y. It need not be differentiable, so the same objective accommodates both retrieval metrics and downstream task metrics.

3.2 PRODUCT ROLLOUTS WITH REINFORCE-LOO

For each training record, we sample a rollout group consisting of $G \geq 2$ query actions and G document bundles. One-to-one pairing would give each query action feedback from only one document bundle, making its evaluation depend on that particular draw. Product rollouts (Kuntz et al., 2022) evaluate all pairings within the group, so every query action is compared against the same document bundles and every bundle against the same query actions:

$$r^{(i,j)} = R(e_q^{(i)}, E_D^{(j)}; y), \qquad i, j = 1, \ldots, G. \tag{3}$$

Reusing actions yields G^2 correlated rewards without additional encoder passes or action samples, at the cost of more score and reward evaluations.

To evaluate each action, we average over the opposite role's draws. The row mean $m_q^{(i)} = G^{-1} \sum_j r^{(i,j)}$ measures query action i, and the column mean $m_D^{(j)} = G^{-1} \sum_i r^{(i,j)}$ measures document bundle j. These averages make feedback less dependent on any single pairing. We then compare each action's mean reward with those of the other actions in the same role using REINFORCE with a leave-one-out baseline (REINFORCE-LOO, or **RLOO**) (Kool et al., 2019; Ahmadian et al., 2024). For role $c \in \{q, D\}$, the unnormalized advantage is

$$A_c^{(g)} = m_c^{(g)} - \frac{1}{G-1} \sum_{g' \neq g} m_c^{(g')} = \frac{G}{G-1} \big(m_c^{(g)} - \bar{r}\big), \tag{4}$$

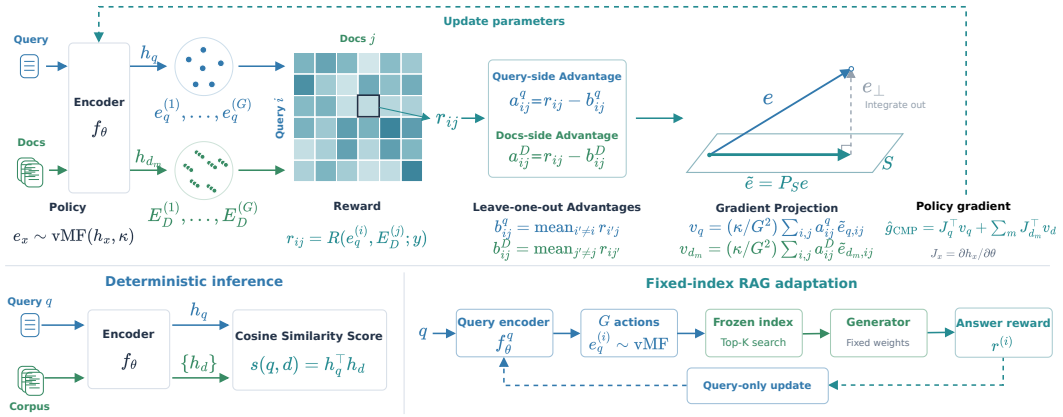

Figure 1: RELER workflow. Top: embedding rollouts yield rewards for RLOO+CMP updates. Bottom left: deterministic cosine retrieval at inference. Bottom right: query-only adaptation with a frozen document index and generator.

where $\bar{r} = G^{-2} \sum_{i,j} r^{(i,j)}$ is the grand mean. These advantages weight the query and document log-densities in the surrogate loss:

$$\mathcal{L}_{\mathrm{RLOO}} = -\frac{1}{G} \sum_i A_q^{(i)} \log \pi_\theta(e_q^{(i)} \mid q) - \frac{1}{G} \sum_j A_D^{(j)} \log \pi_\theta(E_D^{(j)} \mid D), \tag{5}$$

where advantages and sampled actions are held fixed during differentiation. The bundle log-density sums over valid documents. The ascent estimator is $\widehat{g}_{\mathrm{RLOO}} = -\nabla_\theta \mathcal{L}_{\mathrm{RLOO}}$. Conditional on the opposite role's draws, each baseline is independent of the action it evaluates and has zero expected score-weighted contribution. Thus $\mathbb{E}[\widehat{g}_{\mathrm{RLOO}}] = \nabla_\theta \mathcal{J}_\kappa$. Each rollout batch supplies one on-policy update.

To express this update in encoder parameters, let $\mathbf{J}_x = \partial h_x / \partial \theta$ be the encoder Jacobian. For the vMF policy at fixed κ,

$$\nabla_\theta \log \pi_\theta(e_x \mid x) = \kappa \mathbf{J}_x^\top e_x, \qquad \mathbf{J}_x^\top h_x = 0. \tag{6}$$

Unit normalization leaves only the action's tangent component contributing to the gradient; its contribution is weighted by the corresponding advantage. Appendix A.1 gives the gradient derivation and unbiasedness proof.

3.3 Variance Reduction by Conditional-Mean Projection

Sampling in a high-dimensional embedding space introduces randomness along many directions, while a reward based on candidate scores observes only inner products with the candidates. When the candidate span is small relative to the embedding dimension, many sampled components are invisible to the reward but still enter the policy-gradient estimate. They can therefore add gradient noise without providing useful reward information. *Conditional-mean projection* (CMP) averages out this residual randomness while preserving every candidate score. It extends document-space projection (Wang et al., 2019) to vMF actions for both queries and documents.

Projection as conditional averaging. Consider a query action e with fixed candidates. Let $\mathcal{S}$ span those vectors and the policy mean h, and let $P_\mathcal{S}$ be its orthogonal projector. Projection retains every candidate score: $(P_\mathcal{S} e)^\top v = e^\top v$. Let $\mathcal{H}$ contain all other sampled actions. Conditioning on $\mathcal{H}$ and $P_\mathcal{S} e$ fixes these scores, leaving only the residual $(I - P_\mathcal{S})e$ random. Because $h \in \mathcal{S}$, the vMF distribution is symmetric under reflection of this residual, whose conditional mean is therefore zero:

$$\mathbb{E}[e \mid \mathcal{H}, P_\mathcal{S} e] = P_\mathcal{S} e. \tag{7}$$

We call this *conditional-mean projection* because the projection equals the action's conditional mean given the other actions and its retained projection. When the LOO weight is determined by this same information, the weighted gradient term is also conditionally averaged. CMP uses $P_\mathcal{S} e$ only as a gradient coefficient, without renormalization or changes to sampling and reward evaluation.

Projection for product rollouts. Each reward cell (i, j) uses document bundle j and query draw i. Including the corresponding policy means gives the subspaces

$$\mathcal{S}_j^q = \mathrm{span}(h_q, e_{d_1}^{(j)}, \ldots, e_{d_n}^{(j)}), \qquad \mathcal{S}_{im}^d = \mathrm{span}(h_{d_m}, e_q^{(i)}). \tag{8}$$

The query space retains all candidate directions; each document space retains its score against query draw i. Because these spaces vary across cells, we keep cellwise LOO weights until projection:

$$a_{ij}^q = r^{(i,j)} - \frac{\sum_{i' \neq i} r^{(i',j)}}{G-1}, \qquad a_{ij}^D = r^{(i,j)} - \frac{\sum_{j' \neq j} r^{(i,j')}}{G-1}. \tag{9}$$

Each baseline holds the opposite role's draw fixed, and all documents in a bundle share a_{ij}^D. Averaging recovers the marginal advantages: $A_q^{(i)} = G^{-1} \sum_j a_{ij}^q$ and $A_D^{(j)} = G^{-1} \sum_i a_{ij}^D$. For reward R, the projected estimator, **RLOO+CMP**, is

$$\widehat{g}_{\mathrm{CMP}} = \frac{\kappa}{G^2} \sum_{i,j=1}^{G} \left[a_{ij}^q \mathbf{J}_q^\top P_{\mathcal{S}_j^q} e_q^{(i)} + a_{ij}^D \sum_{m=1}^{n} \mathbf{J}_{d_m}^\top P_{\mathcal{S}_{im}^d} e_{d_m}^{(j)} \right]. \tag{10}$$

Replacing the projectors by the identity recovers $\widehat{g}_{\mathrm{RLOO}}$.

For bounded rewards determined by the retained candidate scores, CMP preserves the expected gradient and does not increase any individual summand's variance along any parameter direction (Appendices A.2 and B.3).

3.4 Query Adaptation with a Fixed Index

In practice, dense retrieval is typically served through an approximate nearest neighbor search (ANNS) index built over precomputed document embeddings, which is costly to rebuild whenever the document encoder changes. In this setting, RELER instead adapts only the query encoder. We freeze the pretrained document encoder and keep its precomputed corpus embeddings unchanged. A separate query encoder, initialized from the same pretrained weights, learns to retrieve against these fixed vectors without document re-encoding or index rebuilding.

For each training query, we draw a group of G query vectors from its vMF policy. Each vector retrieves a top-k list from the full index and receives a reward computed from that list. Query-only RLOO uses the mean of the other $G - 1$ rewards as each action's baseline and updates the query encoder through the first term of Eq. (5). No document sampling or product rollouts is needed. We do not apply CMP in this fixed-index setting. Projecting onto the span of the policy mean and retrieved document vectors preserves the scores of those documents, but need not preserve the full-index top-k list: documents outside the retrieved set can cross the retrieval cutoff.

We also anchor the query encoder to its pretrained initialization: for each training query, we penalize the cosine distance between its current deterministic embedding and its initial embedding. This limits drift relative to the frozen index; the loss and its fixed-concentration vMF interpretation are given in Appendix D.3.

The reward can use retrieval quality, downstream performance, or both. For RAG, a frozen generator produces an answer from the original query and each retrieved list. Passage relevance labels and reference answers provide the two sources of feedback (Section 3.5). The same RLOO update then trains the query encoder without differentiating through retrieval or generation. At deployment, the deterministic query embedding h_q replaces sampled actions and searches the unchanged index.

3.5 Training Rewards

We instantiate rewards from relevance labels for retrieval and reference answers for RAG.

Retrieval rewards. From supervision y, we compute R_{nDCG} as nDCG@10 with exponential gains on D and form positive–negative pairs $\mathcal{P}_q$ from its labels. With scores s_p, s_n, we use

$$R = R_{\mathrm{nDCG}} + \lambda R_{\mathrm{pair}}, \qquad R_{\mathrm{pair}} = \frac{1}{|\mathcal{P}_q|} \sum_{(p,n) \in \mathcal{P}_q} \left(\mathbf{1}[s_p > s_n] + \tfrac{1}{2} \mathbf{1}[s_p = s_n] \right), \quad \lambda = 0.5. \tag{11}$$

The nDCG reward emphasizes graded ordering near the cutoff; the pairwise term covers the full candidate set. Appendix B.2 specifies the gains and labels.

Componentwise projection. We apply Eq. (10) to nDCG to obtain $\widehat{g}_{\mathrm{CMP}}^{\mathrm{nDCG}}$. Each pair uses its own LOO weights and a query subspace spanning the policy mean and document-vector difference, since its reward depends only on the corresponding score difference. Averaging the pair estimators gives $\widehat{g}_{\mathrm{pair}}$ (Appendix A.3), yielding

$$\widehat{g}_{\mathrm{RELER}} = \widehat{g}_{\mathrm{CMP}}^{\mathrm{nDCG}} + \lambda \widehat{g}_{\mathrm{pair}}. \tag{12}$$

This is unbiased for the combined reward by linearity; both terms share the actions, candidates, and backward pass.

Answer feedback for RAG. For the fixed-index RAG setting (Section 3.4), let L_{10} be the retrieved top-ten list and $\hat{a}$ the frozen generator's answer from that list. We compare graded retrieval nDCG@10, answer token F1, and their equal mixture:

$$R_{\mathrm{RAG}}^{\mathrm{mix}} = \tfrac{1}{2}\,\mathrm{nDCG@10}(L_{10}; y_{\mathrm{rel}}) + \tfrac{1}{2}\,\mathrm{F1}(\hat{a}, y_{\mathrm{ans}}), \tag{13}$$

where y_{rel} contains passage relevance grades and y_{ans} contains reference answers. The F1 term uses the task's answer-normalization rules; each single-reward control uses one of the two terms alone.

4 Experiments

We evaluate BRIGHT post-training and fixed-index QA adaptation, with ablations of rewards, rollouts, and gradient estimation.

4.1 Reasoning-Intensive Retrieval

4.1.1 Experimental Setup

Baselines. We compare RELER with each pretrained encoder and two supervised objectives: LambdaLoss (Wang et al., 2018), which weights pairwise logistic losses by nDCG@10 swap changes using teacher grades, and InfoNCE (van den Oord et al., 2018), which averages positive-versus-negative contrastive losses over known positives at temperature 0.03.

Training setup. We adapt Qwen3-Embedding-0.6B/4B (Q3E-0.6B/4B) (Zhang et al., 2025) and BGE-M3 (Chen et al., 2024) on 4,963 queries from ReasonRank (Liu et al., 2025b), each with up to 20 candidates. LLM-provided candidate rankings supply the graded relevance labels used by LambdaLoss and RELER. All methods train for 113 steps (approximately 3 epochs), using paired seeds 42, 3407, and 2026. Q3E-0.6B and BGE-M3 are fully fine-tuned at a learning rate of 5×10^{-6}. For Q3E-4B, all methods use LoRA with a learning rate of 2×10^{-4}. The RELER recipe combines product rollouts and RLOO+CMP with $G = 64$, $\rho = 0.70$, and a graded nDCG@10 reward augmented by pairwise feedback ($\lambda = 0.5$). We run all experiments on eight NVIDIA H100 80GB GPUs. Training time is similar to that of contrastive learning; embedding sampling and reward computation reuse the encoder outputs. Full training settings appear in Appendix C.

Benchmark and evaluation. Modern embeddings perform strongly on conventional MTEB/BEIR retrieval but struggle on reasoning-intensive BRIGHT queries (Zhang et al., 2025; Chen et al., 2024; Su et al., 2025). We use BRIGHT to test post-training with limited additional supervision. We report nDCG@10 ($\times 100$), averaged equally over its 12 subsets. Results are means $\pm$ sample SD over three seeds; pretrained models are evaluated once. Each checkpoint uses both original queries and the official GPT-4 reasoning queries.

4.1.2 Main Results

RELER consistently outperforms the baselines across encoders. RELER achieves the highest average nDCG@10 on all three backbones with both original and GPT-4 reasoning queries (Table 1), ahead of InfoNCE, LambdaLoss, and the pretrained encoders. Its consistent advantage over LambdaLoss, which uses the same teacher grades, suggests that graded supervision alone does not explain the gains. The lead persists with GPT-4 queries, showing that embedding post-training remains beneficial when query-side reasoning is already provided.

Table 1: BRIGHT nDCG@10 ($\times 100$). Subsets show three-seed means; Avg. is the macro-average $\pm$ SD. Pretrained models are evaluated once. Bold/underline mark best/second-best scores per model and query setting; shading marks RELER. TheoQ./TheoT.: TheoremQA questions/theorems.

	StackExchange							Coding		Theorem-based			
Method	**Bio.**	**Earth.**	**Econ.**	**Psy.**	**Rob.**	**Stack.**	**Sus.**	**Leet.**	**Pony**	**AoPS**	**TheoQ.**	**TheoT.**	**Avg.**
Original query													
BGE-M3	9.5	15.3	11.9	13.2	12.1	10.6	10.2	**15.8**	**15.2**	1.3	8.4	4.3	10.66
LambdaLoss	17.2	14.1	12.4	16.2	12.8	14.0	16.6	15.0	3.5	**2.7**	8.2	4.2	11.40 $\pm$ 0.08
InfoNCE	18.0	17.0	14.7	20.2	15.2	14.3	16.7	10.0	8.5	2.2	10.1	4.7	12.63 $\pm$ 0.11
RELER	**22.9**	**25.8**	**19.4**	**24.7**	**17.0**	**17.0**	**18.4**	4.9	6.6	1.7	**10.2**	**5.4**	**14.49** $\pm$ 0.04
Q3E-0.6B	13.4	27.8	18.0	16.6	12.0	12.8	13.2	**14.2**	0.7	**3.3**	17.2	32.1	15.10
LambdaLoss	24.3	25.5	23.9	29.0	17.6	22.9	21.9	12.4	1.4	**3.3**	**21.0**	29.0	19.36 $\pm$ 0.28
InfoNCE	34.3	34.1	26.2	33.4	**21.0**	28.8	**27.4**	10.2	1.2	3.1	19.7	**32.3**	22.64 $\pm$ 0.20
RELER	**45.9**	**37.3**	**28.6**	**34.0**	18.4	**29.3**	25.2	9.4	**2.2**	2.3	19.0	28.9	**23.38** $\pm$ 0.17
Q3E-4B	17.2	34.6	16.4	23.0	13.0	15.9	17.1	**21.2**	2.0	3.7	19.6	40.9	18.70
LambdaLoss	32.3	35.5	27.5	36.0	19.9	27.0	30.8	18.5	3.6	**3.8**	23.3	42.0	25.01 $\pm$ 0.37
InfoNCE	48.5	47.0	32.0	42.3	**26.7**	**34.7**	**35.9**	15.2	6.6	**3.8**	**23.5**	44.7	30.08 $\pm$ 0.16
RELER	**48.9**	**50.2**	**32.9**	**44.3**	25.8	33.4	33.6	13.8	**11.3**	3.6	20.7	**46.9**	**30.46** $\pm$ 0.55
GPT-4 reasoning query													
BGE-M3	28.3	38.1	23.1	34.8	15.0	23.9	18.4	**11.6**	24.3	1.5	**14.3**	**24.0**	21.45
LambdaLoss	38.7	42.7	22.2	25.7	15.0	19.3	23.0	9.0	17.7	**2.0**	8.3	6.0	19.13 $\pm$ 0.10
InfoNCE	37.1	42.9	23.5	30.8	17.4	21.8	23.8	4.3	19.8	1.1	12.0	13.0	20.62 $\pm$ 0.09
RELER	**44.7**	**47.3**	**25.5**	**38.2**	**18.0**	**24.7**	**23.9**	1.5	**30.0**	1.1	11.9	19.4	**23.84** $\pm$ 0.07
Q3E-0.6B	27.6	36.3	24.1	28.2	13.5	17.3	19.8	**18.6**	3.2	2.1	16.9	36.8	20.38
LambdaLoss	34.9	38.0	28.1	35.5	18.3	27.7	28.8	12.6	3.5	**3.2**	**20.8**	36.3	23.97 $\pm$ 0.36
InfoNCE	40.1	46.6	29.7	39.8	**21.4**	**34.9**	**29.7**	8.8	6.8	3.1	19.1	38.2	26.51 $\pm$ 0.10
RELER	**48.1**	**48.9**	**30.7**	**41.1**	18.8	33.0	26.5	9.8	**6.9**	2.9	19.7	**39.2**	**27.15** $\pm$ 0.04
Q3E-4B	28.1	39.6	22.6	31.4	15.4	18.8	21.9	**21.1**	2.9	3.1	20.5	38.9	22.01
LambdaLoss	35.4	39.3	25.5	36.3	22.5	29.8	29.8	17.8	7.9	3.3	**20.9**	39.9	25.69 $\pm$ 0.25
InfoNCE	51.3	49.9	31.2	42.2	25.4	**38.8**	**35.5**	14.7	9.7	3.1	20.8	40.8	30.29 $\pm$ 0.35
RELER	**52.5**	**51.2**	**32.2**	**45.3**	**26.8**	38.2	33.3	12.7	**14.7**	**3.9**	19.4	**42.6**	**31.07** $\pm$ 0.39

RELER's margins vary across backbones and baselines. Relative to InfoNCE, RELER has its largest average gains on BGE-M3: 1.86 points with original queries and 3.22 with GPT-4 reasoning. For GPT-4 queries, however, InfoNCE falls below the pretrained BGE-M3 encoder; RELER's improvement over that encoder is 2.39 points. The Qwen backbones gain more over their pretrained encoders but have smaller margins over InfoNCE. Q3E-4B achieves the highest macro-average nDCG@10 in both query settings. The subset scores reveal trade-offs: a higher average does not imply gains on every domain. This variation may partly reflect how closely each BRIGHT subset matches the training distribution.

4.1.3 Ablations and Training Analysis

CMP improves accuracy and reduces seed variation. With the combined reward, BRIGHT rises from 21.91 ± 0.70 to 23.38 ± 0.17 (Table 2, *(b)* to *(a)*). CMP also adds 1.87 points with graded nDCG alone; every paired seed improves in both comparisons. Matched probes at fixed encoder states show 291–590-fold lower gradient variance (Table 9; Figure 2(a)). Since CMP preserves the expected gradient, these gains are consistent with more effective optimization through reduced rollout noise. Appendix C.5 details the protocol.

Embedding exploration benefits from product rollouts. Product rollouts outperform one-to-one pairing by 2.14 points under graded nDCG without CMP (*(d)* vs. *(e)*). Recombining the same actions into G^2 reward cells lets each query draw receive feedback from every document bundle, without an additional encoder forward pass. Query-only and document-only sampling also lower performance (*(f)* and *(g)*), despite updating the shared encoder. These results favor exploring both embedding sides and reusing their samples.

Table 2: BRIGHT ablations (Q3E-0.6B, original queries). Avg. is nDCG@10 (×100), mean ± SD over three seeds; StackEx., Coding, and Theorem average 7, 2, and 3 subsets, respectively. Shading marks full RELER. Query-only and document-only sample one embedding side while updating the shared encoder. Full results are in Appendix C.3.

	Reward	Rollout	CMP	StackEx.	Coding	Theorem	Avg.
(a)	Graded + pairwise	Product	Yes	31.25	5.79	16.73	**23.38** ± 0.17
(b)	Graded + pairwise	Product	No	28.09	5.73	18.27	21.91 ± 0.70
(c)	Graded nDCG	Product	Yes	29.80	6.21	17.24	22.72 ± 0.18
(d)	Graded nDCG	Product	No	26.22	5.79	18.36	20.85 ± 0.68
(e)	Graded nDCG	Paired	No	22.55	6.68	17.78	18.71 ± 0.53
(f)	Graded nDCG	Query only	No	17.92	8.64	15.85	15.85 ± 0.26
(g)	Graded nDCG	Document only	No	22.10	5.98	18.25	18.45 ± 1.56
(h)	Binary nDCG	Product	Yes	21.74	4.49	16.07	17.45 ± 0.65
(i)	Binary MRR	Product	Yes	23.42	4.69	17.83	18.90 ± 0.43

Graded rewards outperform binary rewards. With product rollouts and CMP fixed, replacing binary labels with teacher grades raises the nDCG-based result from 17.45 to 22.72 (*(h)* vs. *(c)*). Binary MRR fares better than binary nDCG but remains well below graded nDCG. Distinguishing levels of candidate relevance matters more here than the choice between the two binary metrics.

Pairwise feedback brings further gains. Adding the pairwise term improves BRIGHT by a further 0.66 points, with gains at all three seeds (*(c)* to *(a)*). The two recipes follow similar training nDCG curves, while the pairwise reward continues to rise (Figure 2(b)). The extra feedback improves the final retrieval score without a clear separation in training nDCG.

A larger pairwise weight hurts performance. Setting $\lambda = 1$ reduces BRIGHT to 22.57, slightly below graded-only training (Figure 2(c)). The best tested weight is 0.5. Because the pairwise reward weights comparisons throughout the candidate list, a larger coefficient may put too much emphasis on ordering away from the top ranks favored by nDCG@10.

Performance is stable across alignment values. BRIGHT varies little over $\rho = 0.60$–0.90 (Figure 2(d)). The best mean occurs at 0.80, but its advantage over the default 0.70 changes sign across seeds. Within this range, the method is not sensitive to a precise alignment value. Lower alignment allows samples to move farther from the encoder output; higher alignment keeps them closer. Neither direction yields a consistent improvement over the default. The full sweep spans only 0.29 points, much less than the effects of CMP or rollout design under the same training budget.

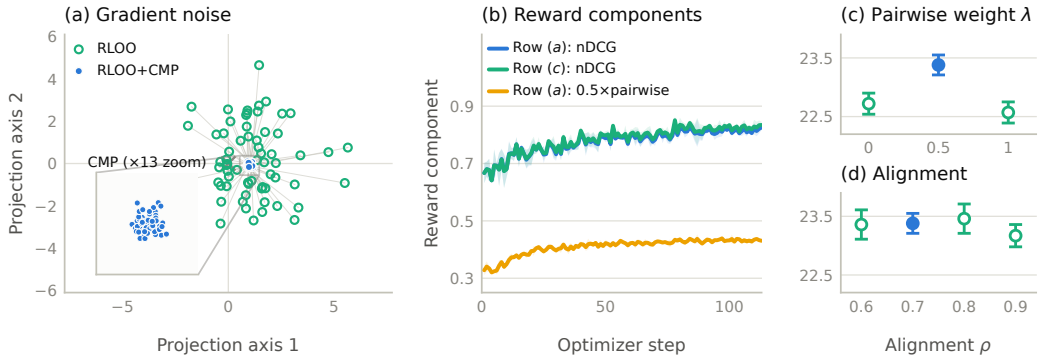

Figure 2: **Training dynamics and sensitivity.** (a) CMP reduces gradient noise before fine-tuning (inset: zoom). (b) Pairwise feedback improves BRIGHT despite similar training nDCG reward curves. (Table 2, rows *(a,c)*). (c,d) BRIGHT nDCG@10 (×100) is highest at $\lambda = 0.5$ and stable across $\rho = 0.60$–0.90. Filled points mark defaults; unsmoothed curves and error bars show three-seed means ± SD. Sweep details are in Appendix C.4.

Table 3: Fixed-index QA (×100), seed 42. In-domain/out-of-domain average 2/5 datasets; Avg. covers all seven. Bold marks column bests. Per-dataset results are in Appendix D.5.

	In-domain			Out-of-domain			Avg.		
Method	EM	F1	Hit@10	EM	F1	Hit@10	EM	F1	Hit@10
Q3E-0.6B	33.79	44.17	65.67	31.71	39.40	51.94	32.30	40.76	55.87
InfoNCE	33.63	44.07	64.30	30.19	37.76	46.72	31.18	39.56	51.74
RLOO, graded nDCG	34.66	45.08	**67.25**	31.38	39.08	51.97	32.32	40.80	56.34
RLOO, answer F1	34.71	**45.16**	66.77	31.38	39.14	**52.61**	32.33	40.86	**56.66**
RLOO, nDCG + F1	**34.82**	45.14	66.87	**32.21**	**39.48**	52.19	**32.96**	**41.10**	56.38

4.2 Retrieval-Augmented Generation

4.2.1 Experimental Setup

We build a Q3E-0.6B index over FlashRAG's 2018 Wikipedia passage corpus (Jin et al., 2025b). The index and Qwen2.5-7B-Instruct generator remain frozen, so all adaptation comes from updating the query encoder. We train on DPR-labeled NQ and FlashRAG HotpotQA for one epoch with seed 42 and query-anchor coefficient 0.5. Human-positive NQ passages and annotated HotpotQA supporting facts are aligned to the corpus as binary relevance labels. The five out-of-domain datasets are PopQA, TriviaQA, 2WikiMultihopQA, MuSiQue, and Bamboogle; none supplies training queries. InfoNCE trains on a fixed offline candidate pool; query-only RLOO searches the full index with graded nDCG@10, answer F1, or their equal mixture as reward, without CMP. InfoNCE negatives are mined from the constructed index; offline mining and online retrieval are part of the respective training methods being compared. Evaluation retrieves once per question, passing the top-10 passages to the generator. We report answer EM/F1 and answer-containing-passage Hit@10, macro-averaged across datasets. Full details are in Appendix D.2.

4.2.2 Results and Analysis

RLOO improves fixed-index retrieval and QA. All three RLOO recipes improve Hit@10 and answer EM/F1 over InfoNCE (Table 3). With the document index and generator frozen, these gains show that adapting the query encoder can improve the context delivered to an existing RAG pipeline. The mixed reward also exceeds the pretrained encoder on all three seven-dataset averages, although its F1 margin is smaller than the gap over InfoNCE. The reference matters because InfoNCE falls below the pretrained encoder, while the RLOO recipes recover or exceed its average scores. RLOO searches the full index, whereas InfoNCE uses a fixed candidate pool, so their gap reflects both objective and candidate access.

Answer feedback captures utility beyond retrieval hits. Answer-F1 reward slightly improves average EM/F1 over graded nDCG, and their mixture gives the best overall answer scores. Answer-F1 has the highest overall Hit@10 but trails the mixture in EM/F1; graded nDCG and the mixture also have nearly identical Hit@10 despite different answer scores. Hit@10 counts any answer-containing top-ten passage and ignores passage order, distracting context, and evidence removed by the generator's input limit. Reward choice can thus improve the utility of the generator's context even when answer containment changes little.

5 Conclusion

RELER aligns embedding models with retrieval and downstream task rewards through vMF exploration in embedding space during training, while retaining deterministic vector retrieval at inference. Conditional-mean projection removes reward-invisible gradient noise without changing the expected fixed-candidate gradient. Across BRIGHT backbones, RELER generally improves average retrieval over InfoNCE, although gains vary by query form and domain. With a fixed document index and generator, adapting only the query encoder improves both retrieval performance and end-to-end answer quality. These results show the potential of embedding-space RL for retrieval post-training, and motivate future work on adaptive exploration policies and multi-step retrieval for search agents.

Reproducibility Statement

Section 3.3 and Appendix A state the estimator assumptions and provide the derivations and proofs. Appendix B specifies the sampling, rewards, and CMP implementation; Appendices C and D document the BRIGHT and fixed-index QA protocols, configurations, and complete results. Appendix E describes direct embedding training from a general-purpose LLM. BRIGHT results use three paired seeds and fixed-budget final checkpoints; fixed-index QA and direct LLM-to-embedding experiments use one seed each. An anonymous snapshot of the training and evaluation code is linked on the first page.

AI Use Statement

We used generative AI assistants to help implement and edit research code, analyze and interpret experimental results, and draft and edit parts of this paper. We reviewed AI-assisted code and manuscript text, checked mathematical arguments and reported results against the derivations and experiment records. The authors take responsibility for the final code, results, claims, and text.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024. URL https://arxiv.org/abs/2402.14740.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. LLM2Vec: Large language models are secretly powerful text encoders. In *Conference on Language Modeling (COLM)*, 2024.

Walid Bendada, Guillaume Salha-Galvan, Romain Hennequin, Théo Bontempelli, Thomas Bouabça, and Tristan Cazenave. Exploring large action sets with hyperspherical embeddings using von mises-fisher sampling. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 3677–3711. PMLR, 2025. URL https://proceedings.mlr.press/v267/bendada25a.html.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

George Casella and Christian P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83 (1):81–94, 1996.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics (ACL)*, 2024.

Jianlyu Chen, Junwei Lan, Chaofan Li, Defu Lian, and Zheng Liu. ReasonEmbed: Enhanced text embeddings for reasoning-intensive document retrieval. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1203–1221, 2026. doi: 10.18653/v1/2026.acl-long.54.

Debrup Das, Sam O'Nuallain, and Razieh Rahimi. RaDeR: Reasoning-aware dense retrieval models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. Hyperspherical variational auto-encoders. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, et al. MMTEB: Massive multilingual text embedding benchmark. In *International Conference on Learning Representations (ICLR)*, 2025.

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2013.

Haonan Jiang, Yuji Wang, Yongjie Zhu, Xin Lu, Wenyu Qin, Meng Wang, Pengfei Wan, and Yansong Tang. Embed-RL: Reinforcement learning for reasoning-driven multimodal embeddings. *arXiv preprint arXiv:2602.13823*, 2026.

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*, 2025a.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. FlashRAG: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM Web Conference 2025*, pp. 737–740, 2025b. doi: 10.1145/3701716.3715313.

Vladimir Karpukhin, Barlas Oğuz, 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 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE samples, get a baseline for free! In *ICLR Workshop on Deep Reinforcement Learning Meets Structured Prediction*, 2019. URL https://wouterkool.github.io/publication/buy-4-samples-free-baseline/.

Juan Kuntz, Francesca R. Crucinio, and Adam M. Johansen. Product-form estimators: exploiting independence to scale up Monte Carlo. *Statistics and Computing*, 32(1):12, 2022. doi: 10.1007/s11222-021-10069-9. URL https://arxiv.org/abs/2102.11575.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. NV-embed: Improved techniques for training LLMs as generalist embedding models. In *International Conference on Learning Representations (ICLR)*, 2025.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

Qi Liu, Yanzhao Zhang, Mingxin Li, Dingkun Long, Pengjun Xie, and Jiaxin Mao. E2Rank: Your text embedding can also be an effective and efficient listwise reranker. *arXiv preprint arXiv:2510.22733*, 2025a.

Tie-Yan Liu. *Learning to Rank for Information Retrieval*. Springer, 2009.

Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. ReasonRank: Empowering passage ranking with strong reasoning ability. *arXiv preprint arXiv:2508.07050*, 2025b.

Xingxian Liu, Dongshuai Li, Jiahui Wan, Tao Wen, Gui Ling, Yuliang Yan, Fuyu Lv, Dan Ou, Haihong Tang, and Bo Zheng. Retrieval-GRPO: A multi-objective reinforcement learning framework for dense retrieval in Taobao search. *arXiv preprint arXiv:2511.13885*, 2025c.

Meixiu Long, Duolin Sun, Dan Yang, Junjie Wang, Yue Shen, Jian Wang, Peng Wei, Jinjie Gu, and Jiahai Wang. DIVER: A multi-stage approach for reasoning-intensive information retrieval. *arXiv preprint arXiv:2508.07995*, 2025.

Kanti V. Mardia and Peter E. Jupp. *Directional Statistics*. John Wiley & Sons, 2000.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2023.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. Generative representational instruction tuning. In *International Conference on Learning Representations (ICLR)*, 2025.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Przemysław Pobrotyn and Radosław Białobrzeski. NeuralNDCG: Direct optimisation of a ranking metric via differentiable relaxation of sorting. *arXiv preprint arXiv:2102.07831*, 2021.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021.

Alireza Salemi, Surya Kallumadi, and Hamed Zamani. Optimization methods for personalizing large language models through retrieval augmentation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 752–762, 2024. doi: 10.1145/3626772.3657783.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen-tau Yih, Pang Wei Koh, and Luke Zettlemoyer. ReasonIR: Training retrievers for reasoning tasks. *arXiv preprint arXiv:2504.20595*, 2025.

Ashudeep Singh and Thorsten Joachims. Policy learning for fairness in ranking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval. In *International Conference on Learning Representations (ICLR)*, 2025.

Jiashuo Sun, Shixuan Liu, Zhaochen Su, Xianrui Zhong, Pengcheng Jiang, Bowen Jin, Peiran Li, Weijia Shi, and Jiawei Han. GRACE: Generative representation learning via contrastive policy optimization. In *International Conference on Learning Representations (ICLR)*, 2026.

Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. SoftRank: Optimizing non-smooth rank metrics. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*, 2008.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Huazheng Wang, Sonwoo Kim, Eric McCord-Snook, Qingyun Wu, and Hongning Wang. Variance reduction in gradient exploration for online learning to rank. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2019. doi: 10.1145/3331184.3331264. URL https://arxiv.org/abs/1906.03766.

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.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The LambdaLoss framework for ranking metric optimization. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2018.

Zeng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. Reinforcement learning to rank with Markov decision process. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2017.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

Andrew T. A. Wood. Simulation of the von Mises Fisher distribution. *Communications in Statistics—Simulation and Computation*, 23(1):157–164, 1994.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general Chinese embeddings. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2024.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations (ICLR)*, 2021.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. Learning to retrieve: How to train a dense retrieval model effectively and efficiently. *arXiv preprint arXiv:2010.10469*, 2020.

Hengran Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. Are large language models good at utility judgments? In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1941–1951, 2024.

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

Yicheng Zhang, Zhen Qin, Zhaomin Wu, Wenqi Zhang, and Shuiguang Deng. Reinforcement fine-tuning for history-aware dense retriever in RAG. *arXiv preprint arXiv:2602.03645*, 2026. URL https://arxiv.org/abs/2602.03645v1.

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. Rank-R1: Enhancing reasoning in LLM-based document rerankers via reinforcement learning. *arXiv preprint arXiv:2503.06034*, 2025.

Appendix A gives the derivations and proofs, and Appendix B describes the implementation. Experimental settings and results are grouped by study: BRIGHT in Appendix C, fixed-index RAG in Appendix D, and training from a general-purpose LLM in Appendix E. Extended related work appears in Appendix F.

A Theoretical Analysis

We first derive the product-rollout policy gradient, then prove CMP's unbiasedness and per-term variance guarantee, and finally specialize the estimator to pairwise rewards. Throughout, the training record, candidate masks, encoder state, and concentration are fixed for each update.

A.1 Gradient Estimation for RELER

Fix a training record (q, D, y) and concentration κ. The objective is $\mathcal{J}_\kappa = \mathbb{E}[R(e_q, E_D; y)]$ under independent query and document vMF actions. The labeled candidate set D is fixed; candidate construction is outside the derivative studied here. For a sampled action e_x, the vMF log-density is $\log \pi_\theta(e_x \mid x) = \log C_d(\kappa) + \kappa h_x^\top e_x$. Since the normalizer is independent of θ at fixed κ,

$$\nabla_\theta \log \pi_\theta(e_x \mid x) = \kappa \mathbf{J}_x^\top e_x, \qquad \mathbf{J}_x = \frac{\partial h_x}{\partial \theta}. \tag{14}$$

Sampled actions are held fixed when differentiating the log-density; gradients pass through the encoder means. The surrogate can omit $\log C_d(\kappa)$ without changing its gradient.

Unit normalization and tangent gradients. The encoder output satisfies $h_x^\top h_x = 1$ for every θ. Differentiating this identity by the chain rule gives

$$0 = \nabla_\theta (h_x^\top h_x) = 2 \left(\frac{\partial h_x}{\partial \theta} \right)^\top h_x = 2 \mathbf{J}_x^\top h_x. \tag{15}$$

Thus $\mathbf{J}_x^\top h_x = 0$: any first-order change of a unit embedding is orthogonal to its current direction. Decompose the sampled action into its radial and tangent components, $e_x = (h_x^\top e_x) h_x + [e_x - (h_x^\top e_x) h_x]$. The radial term vanishes under $\mathbf{J}_x^\top$, so

$$\nabla_\theta \log \pi_\theta(e_x \mid x) = \kappa \mathbf{J}_x^\top \big[e_x - (h_x^\top e_x) h_x \big]. \tag{16}$$

The bracketed vector is orthogonal to h_x and therefore lies in the tangent space of the unit sphere at h_x. This establishes both identities in Eq. (6) and explains why radial action components contribute no encoder gradient.

Query and document contributions. RELER samples the query and every document independently, although their means share encoder parameters. The document-bundle log-density is therefore

$$\log \pi_\theta(E_D \mid D) = \sum_{m=1}^{n} \log \pi_\theta(e_{d_m} \mid d_m). \tag{17}$$

The sum covers valid documents without list-length averaging. For $X \sim \pi_\theta(\cdot \mid q)$ and $Y \sim \pi_\theta(\cdot \mid D)$, the two contributions to $\nabla_\theta \mathcal{J}_\kappa$ are

$$\begin{aligned} g_q &= \mathbb{E}_X[\mathbb{E}_Y[R(X,Y)] \nabla_\theta \log \pi_\theta(X \mid q)], \\ g_D &= \mathbb{E}_Y[\mathbb{E}_X[R(X,Y)] \nabla_\theta \log \pi_\theta(Y \mid D)], \end{aligned} \qquad \nabla_\theta \mathcal{J}_\kappa = g_q + g_D. \tag{18}$$

The reward table's row and column means estimate these inner expectations: each query action is evaluated across document bundles, and each bundle across query actions. Pairing G draws from each role reuses $2G$ component actions for G^2 correlated reward evaluations (Kuntz et al., 2022). Shared encoder parameters do not make the draws dependent, and the reward evaluations are not independent samples.

Leave-one-out feedback. For a query action i, condition on all document-bundle draws. Its baseline uses only the other query draws and is independent of action i; the action's log-density gradient has conditional mean zero. The baseline therefore has zero expected score-weighted contribution. The same argument applies to a document bundle j after conditioning on the query draws and excluding bundle j from its baseline. Applying this argument to the surrogate in Eq. (5) gives

$$\mathbb{E}[\widehat{g}_{\mathrm{RLOO}}] = g_q + g_D = \nabla_\theta \mathcal{J}_\kappa. \tag{19}$$

This is the standard RLOO baseline argument applied to RELER's paired actions (Williams, 1992; Kool et al., 2019; Ahmadian et al., 2024). The statement is conditional on the fixed record, candidate masks, and concentration. RELER uses no sample-dependent reward standardization or document-density averaging; either modification can change the expected gradient. Each rollout batch supplies one on-policy encoder update.

A.2 Conditional-Mean Projection: Unbiasedness and Variance

CMP replaces each sampled action in a weighted gradient term by its projection, retaining the same cellwise LOO weight. For query draw i, document bundle j, and document m, the projection spaces are

$$\mathcal{S}_j^q = \mathrm{span}(h_q, e_{d_1}^{(j)}, \ldots, e_{d_n}^{(j)}), \qquad \mathcal{S}_{im}^d = \mathrm{span}(h_{d_m}, e_q^{(i)}), \tag{20}$$

These spaces contain the policy mean and every direction needed to retain the current action's candidate scores.

Proposition 1 (Unbiasedness and variance reduction)**.** *Fix the encoder state, record, candidate masks, and concentration κ. Assume independent vMF actions and a bounded reward depending on each action only through the candidate inner products retained by Eq.* (20)*. The CMP estimator is unbiased: $\mathbb{E}[\widehat{g}_{\mathrm{CMP}}] = \mathbb{E}[\widehat{g}_{\mathrm{RLOO}}] = \nabla_\theta \mathcal{J}_\kappa$. Each projected query summand (i, j) and document summand (i, j, m) has no greater variance along any parameter direction than its unprojected counterpart.*

Proof of Proposition 1. At a fixed encoder state, the means and Jacobians are deterministic. Consider any one query or individual-document summand of the estimator. Let e be its current action, $\mathcal{S}$ its subspace, $\mathbf{J}$ its Jacobian, and a its cellwise LOO weight. Let $\mathcal{H}$ contain all other sampled actions and define $\mathcal{G} = \sigma(\mathcal{H}, P_{\mathcal{S}} e)$. The subspace is $\mathcal{H}$-measurable and does not depend on e. By independence, conditional on $\mathcal{H}$, e still follows its vMF law.

Write $Q = 2P_{\mathcal{S}} - I$. This orthogonal map fixes $\mathcal{S}$ and negates $\mathcal{S}^\perp$. Because $h \in \mathcal{S}$, $h^\top Q e = h^\top e$, and Q preserves both surface measure and the vMF density. It also leaves $P_{\mathcal{S}} e$ unchanged. Hence the conditional law of $(I - P_{\mathcal{S}})e$ given $\mathcal{G}$ is symmetric about zero, which proves

$$\mathbb{E}[e \mid \mathcal{G}] = P_{\mathcal{S}} e. \tag{21}$$

For a query term, every score in bundle j, including every valid frozen candidate, is determined by $P_{\mathcal{S}_j^q} e_q^{(i)}$. The baseline excludes query action i, so it is already determined by $\mathcal{H}$. For a document term, condition on all other documents in the current bundle as well as all other actions. The current document affects the reward only through $e_q^{(i)\top} e_{d_m}^{(j)}$; the baseline excludes bundle j. Thus a is $\mathcal{G}$-measurable in both cases.

For $Z = \kappa a \mathbf{J}^\top e$, Eq. (21) yields $Z_{\mathrm{CMP}} = \mathbb{E}[Z \mid \mathcal{G}] = \kappa a \mathbf{J}^\top P_{\mathcal{S}} e$. This is the Rao–Blackwell construction (Casella & Robert, 1996) applied to the entire weighted gradient term. Taking expectations and summing the G^{-2}-weighted terms proves $\mathbb{E}[\widehat{g}_{\mathrm{CMP}}] = \mathbb{E}[\widehat{g}_{\mathrm{RLOO}}]$. The preceding LOO derivation identifies their common expectation with $\nabla_\theta \mathcal{J}_\kappa$. For bounded rewards, the law of total covariance also gives

$$\mathrm{Cov}(Z) - \mathrm{Cov}(Z_{\mathrm{CMP}}) = \mathbb{E}[\mathrm{Cov}(Z \mid \mathcal{G})] \succeq 0. \tag{22}$$

The conditioning $\mathcal{G}$ differs between summands. Consequently this identity does not order the covariance of their sums. □

Scope of the guarantee. The result concerns the stated estimator with candidates and concentration fixed within each rollout table. All reward-relevant frozen directions must be retained. In particular, a subspace containing only returned full-corpus top-k hits need not preserve the retrieved list and does not satisfy this condition. The variance guarantee is per term; correlations between terms prevent an automatic ordering of the summed gradient covariance.

A.3 Pairwise Conditional-Mean Projection

For pair $(p, n) \in \mathcal{P}_q$ and document bundle j, let $\delta_{pn}^{(j)} = v_p^{(j)} - v_n^{(j)}$. Each v is a sampled own-query vector $e_d^{(j)}$ or a fixed vector ρh_d, where $\rho = \mathbb{E}[h_d^\top e_d]$. Define $r_{ij}^{pn} = \mathbf{1}[e_q^{(i)\top} \delta_{pn}^{(j)} > 0] + \frac{1}{2}\mathbf{1}[e_q^{(i)\top} \delta_{pn}^{(j)} = 0]$. Its LOO weights are

$$a_{ij}^{q,pn} = r_{ij}^{pn} - \frac{1}{G-1}\sum_{i' \neq i} r_{i'j}^{pn}, \qquad a_{ij}^{D,pn} = r_{ij}^{pn} - \frac{1}{G-1}\sum_{j' \neq j} r_{ij'}^{pn}. \tag{23}$$

Let $\mathcal{I}_{pn}$ denote the pair's sampled own-query endpoints. With $\mathcal{S}_{j,pn}^q = \mathrm{span}(h_q, \delta_{pn}^{(j)})$ and document spaces $\mathcal{S}_{im}^d = \mathrm{span}(h_{d_m}, e_q^{(i)})$, the estimator is

$$\widehat{g}_{\mathrm{pair}} = \frac{\kappa}{G^2|\mathcal{P}_q|}\sum_{(p,n)\in\mathcal{P}_q}\sum_{i,j}\left[a_{ij}^{q,pn}\mathbf{J}_q^\top P_{\mathcal{S}_{j,pn}^q} e_q^{(i)} + a_{ij}^{D,pn}\sum_{m\in\mathcal{I}_{pn}} \mathbf{J}_{d_m}^\top P_{\mathcal{S}_{im}^d} e_{d_m}^{(j)}\right]. \tag{24}$$

For an empty pair set this is zero. Each query projection retains its pair's score difference; each document projection retains that endpoint's score. The symmetry argument of Proposition 1 therefore applies to every term. Actions of documents outside the pair have zero expected score-weighted contribution and can be omitted. It follows that $\mathbb{E}[\widehat{g}_{\mathrm{pair}}] = \nabla_\theta \mathbb{E}[R_{\mathrm{pair}}]$, conditional on the fixed record, detached additions, and κ. For $R = R_{\mathrm{nDCG}} + \lambda R_{\mathrm{pair}}$, combining the listwise and pairwise estimators gives $\mathbb{E}[\widehat{g}_{\mathrm{RELER}}] = \nabla_\theta \mathcal{J}_\kappa$ for the combined reward. Linearity permits shared actions across reward components; the termwise variance guarantee does not order the covariance of their sum.

B Implementation Details

We describe embedding sampling, reward evaluation, and gradient projection, then combine them into the complete training algorithm.

B.1 Exploration and Embedding Sampling

RELER specifies exploration through the expected alignment of a sampled embedding with its encoder output:

$$\rho = A_d(\kappa) = \mathbb{E}[h_x^\top e_x] = \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)}, \tag{25}$$

where I_ν is the modified Bessel function of the first kind. We solve for κ at the embedding dimension and use it in the sampler below.

Exact vMF sampling. We sample vMF actions using the rejection sampler of Wood (1994), shown in Algorithm 1. Each action decomposes as $e = wh + \sqrt{1-w^2}\, v$, where $w = h^\top e$ sets its alignment with the mean and v is a uniformly sampled unit direction orthogonal to h. Rejection sampling generates w; projecting and normalizing an independent Gaussian vector generates v.

We apply this procedure independently to each query and document mean. The implementation batches all pending draws, evaluates the rejection test in FP64, and constructs tangent directions and actions in FP32 before casting to the input dtype. It raises an error if any draw remains unaccepted after 256 rejection rounds. Sampled actions are held fixed in the policy-gradient update.

In contrast, $\mathrm{normalize}(h + \sigma\epsilon)$ follows a projected-normal law. Scoring such samples with a vMF density does not give the score-function gradient of their sampling law. We include that approximation only as a sampling/scoring mismatch diagnostic, using the nominal local scale $\sigma = 1/\sqrt{\kappa}$.

B.2 Reward Definitions

In our implementation, y collects teacher-assigned relevance grades and the original binary relevance annotations. Let $c_1, \ldots, c_M$ be the candidates for a query, with grades ℓ_m and scores s_m. Sorting

Algorithm 1 Sampling a group of vMF embedding actions

Require: Unit mean $h \in \mathbb{S}^{d-1}$, $d \geq 3$; concentration $\kappa > 0$; group size G
Ensure: Independent actions $e^{(1)}, \ldots, e^{(G)} \sim \mathrm{vMF}(h, \kappa)$

$a \leftarrow (d-1)/2$
$b \leftarrow \dfrac{d-1}{2\kappa + \sqrt{4\kappa^2 + (d-1)^2}}$
$w_0 \leftarrow (1-b)/(1+b)$; $c \leftarrow \kappa w_0 + (d-1)\log(1-w_0^2)$
for $g = 1, \ldots, G$ **do**
  repeat ▷ Sample the alignment coefficient
    $z \sim \mathrm{Beta}(a, a)$; $u \sim \mathrm{Uniform}(0, 1)$
    $w \leftarrow \dfrac{1-(1+b)z}{1-(1-b)z}$
  until $\kappa w + (d-1)\log(1 - w_0 w) - c \geq \log u$
  $\xi \sim \mathcal{N}(0, I_d)$ ▷ Sample an independent tangent direction
  $v \leftarrow \xi - (h^\top \xi)h$; $v \leftarrow v/\|v\|$
  $e^{(g)} \leftarrow wh + \sqrt{1-w^2}\, v$
end for
return $e^{(1)}, \ldots, e^{(G)}$

by score gives a permutation π. For cutoff k and $k' = \min(k, M)$,

$$\mathrm{nDCG@}k = \frac{\sum_{t=1}^{k'} (2^{\ell_{\pi_t}} - 1)/\log_2(t+1)}{\max_{\pi'} \sum_{t=1}^{k'} (2^{\ell_{\pi'_t}} - 1)/\log_2(t+1)}. \tag{26}$$

Zero ideal DCG gives zero reward. The pool contains only valid own-query candidates. Teacher ranks map to grades ℓ_m: 3 for rank 1, 2 for ranks 2–5, 1 for ranks 6–10, and 0 otherwise. The main reward is $R_{\mathrm{nDCG}} + \lambda R_{\mathrm{pair}}$ with $\lambda = 0.5$. The pairwise term averages a score of 1 for a positive ranked above a negative, $1/2$ for a tie, and 0 otherwise. Positive membership comes from the original binary annotations in y, not a threshold on teacher grades. A query with no valid pair receives zero pairwise reward and remains in the batch average. For binary nDCG and MRR controls, labels indicate all annotated positives. MRR@10 is the reciprocal rank of the first positive within the top ten, or zero if none occurs. All rewards are evaluated exactly; no differentiable rank relaxation is used.

Exploration and discrete feedback. For the retrieval rewards used by RELER, changing embeddings provides feedback only when it changes a reward-relevant ordering. If every pairing in the reward table has the same value, all marginal and cellwise LOO weights vanish. Both RLOO and CMP then give zero updates for that reward component: projection can remove gradient noise but cannot create a missing reward difference. Concentrating actions too tightly around their means can make such constant reward tables more frequent; broader exploration changes candidate orderings more often but can move retrieval behavior farther from deterministic deployment.

RELER uses graded nDCG together with binary pairwise feedback. Graded labels make more swaps reward-relevant, while an individual positive–negative pair can change order even when nDCG remains constant. The pairwise term can therefore supply feedback in those cases.

B.3 Conditional-Mean Projection Implementation

Projection bases use the current encoder means and sampled actions. We detach the bases and coefficients, then apply the resulting gradient through the live unit means.

For listwise rewards, the query subspace spans the policy mean, sampled documents in the current bundle. The implementation projects the query weighted sum separately for each bundle. With at most 20 own-query candidates and the policy mean, the listwise query subspace has dimension at most 21, compared with $d = 1024$. We construct the basis with reduced SVD in FP32, setting padded and masked candidate columns to zero. For a $d \times w$ basis-input matrix with largest singular value $\sigma_{\max}$, we retain singular vectors whose singular values exceed $\max(d, w)\epsilon_{\mathrm{FP32}}\sigma_{\max}$. This handles duplicate or linearly dependent columns without adding arbitrary directions. For documents, write $u_{im} = e_q^{(i)} - (h_{d_m}^\top e_q^{(i)})h_{d_m}$. The tangent projection is $u_{im}u_{im}^\top e_{d_m}^{(j)}/\|u_{im}\|^2$; the radial component vanishes under the normalized mean's Jacobian. We form this tangent direction in FP64 to

avoid cancellation for nearly parallel vectors, normalize it when its norm exceeds 10^{-12}, and set it to zero otherwise; the subsequent contractions use FP32. The implemented coefficient retains the radial component, which normalization removes during backpropagation. CMP requires no additional encoder forwards or reward evaluations.

For each pairwise reward, the query subspace spans the policy mean and the positive–negative document-vector difference, giving dimension at most two. We project each pair before averaging, rather than applying one listwise projector to the aggregate pair reward. The implementation processes pairs in small blocks, reuses the product scores, and accumulates query and document coefficients before one encoder backward pass. Linearity allows us to sum the query coefficients over query draws before projecting within each document bundle, and to sum document coefficients over document draws before projecting for each query draw. These contractions implement Eq. (10) without materializing a $B \times G \times G \times n \times d$ tensor, where B is the microbatch size. Actions, LOO weights, and projectors are detached throughout; gradients flow only through the live normalized encoder means. Projected coefficients are not renormalized to unit length.

B.4 Query–Document Post-Training Algorithm

Algorithm 2 assembles the main recipe into one training loop using each query's labeled documents. Each $\widehat{g}^{(q)}$ estimates the full shared-encoder gradient for record q. In the pseudocode, sg denotes stop-gradient, and SampleVMF is Algorithm 1.

LOO subtracts other-row means for query weights and other-column means for document weights. ListCMP and PairCMP form the listwise and individual pair gradients using the current actions and means. The combined batch gradient requires one backward pass; AdamW receives its negative for ascent. An empty pair set leaves only the listwise contribution.

Algorithm 2 RELER: embedding post-training with retrieval feedback

Require: Encoder f_{θ_0}; minibatches of records (q, D, y)
Require: $G = 64$, $\rho = 0.70$, $\lambda = 0.5$
Ensure: Deterministic retriever f_θ

$\theta \leftarrow \theta_0$; $\kappa \leftarrow A_d^{-1}(\rho)$
for each minibatch $\mathcal{B}$ **do**
  $\{h_x\} \leftarrow f_\theta(\mathcal{B})$ ▷ Encode queries and documents once
  for each $(q, D, y) \in \mathcal{B}$ **do**
    $e_q^{(1:G)} \leftarrow \mathrm{SampleVMF}(h_q, \kappa, G)$
    $e_d^{(1:G)} \leftarrow \mathrm{SampleVMF}(h_d, \kappa, G)$ for each $d \in D$
    $S = (s_m^{(i,j)}) \leftarrow \mathrm{Scores}(e_q^{(1:G)}, E_D^{(1:G)})$
    $\mathbf{R} \leftarrow [\mathrm{nDCG@10}(S_{ij,:}, y)]_{i,j=1}^{G}$
    $(\mathbf{a}^q, \mathbf{a}^D) \leftarrow \mathrm{LOO}(\mathbf{R})$ ▷ Cellwise leave-one-out weights
    $\widehat{g}^{(q)} \leftarrow \mathrm{ListCMP}(\mathbf{a}^q, \mathbf{a}^D)$ ▷ Listwise gradient
    $\mathcal{P}_q \leftarrow \mathrm{Pairs}(y, D)$
    for each $(p, n) \in \mathcal{P}_q$ **do**
      $\mathbf{R}^{pn} \leftarrow [\mathbf{1}[s_p^{(i,j)} > s_n^{(i,j)}] + \frac{1}{2}\mathbf{1}[s_p^{(i,j)} = s_n^{(i,j)}]]_{i,j=1}^{G}$
      $(\mathbf{a}^{q,pn}, \mathbf{a}^{D,pn}) \leftarrow \mathrm{LOO}(\mathbf{R}^{pn})$
      $\widehat{g}^{(q)} \leftarrow \widehat{g}^{(q)} + \dfrac{\lambda}{|\mathcal{P}_q|}\,\mathrm{PairCMP}(\mathbf{a}^{q,pn}, \mathbf{a}^{D,pn}; p, n)$
    end for
  end for
  $\theta \leftarrow \mathrm{AdamW}\left(\theta, -\dfrac{1}{|\mathcal{B}|}\sum_{q\in\mathcal{B}} \widehat{g}^{(q)}\right)$ ▷ One optimizer step
end for
return f_θ

C BRIGHT EXPERIMENTS

This section collects the BRIGHT training and evaluation protocol, reference baselines, complete ablation results, and gradient measurements.

C.1 EXPERIMENTAL SETUP

C.1.1 DATA AND PREPROCESSING

ReasonRank preprocessing removes 133 possible BRIGHT overlaps, 1,546 MS MARCO records, 61 relevance-conflict records, 17 records without negatives, and one duplicate query from 6,721 records. The remaining 4,963 queries contain 15,830 positive occurrences. Text deduplication preserves all known positives; annotations and teacher ranks remain separate supervision sources. Source-wise batching retains 4,896 queries at a per-device batch size of 16, reshuffling within sources each epoch. Training seeds pair the data order across methods; embedding exploration and negative sampling use separate random-number streams. There is no internal development split. Overlap screening does not establish complete semantic decontamination.

C.1.2 REPRESENTATION AND EVALUATION SETTINGS

Training and evaluation use the same model and tokenizer initialization. For Qwen3-Embedding, we reserve one position when truncating text and append a single terminal token, then pool its hidden state. Last-token pooling, normalization, and similarity computation use FP32. Queries contain a task instruction followed by the query text; documents use raw text. The backbone uses BF16 during training and FP16 during evaluation. Training query/document limits are 512/1024 tokens; external evaluation permits 8192 tokens. Deterministic retrieval uses unscaled cosine similarity.

BGE-M3 uses its dense encoder with CLS pooling, right padding, no appended terminal token, and no query/document prefixes. Training lengths and the 8192-token evaluation limit follow the same budget as Qwen3. Its nine training runs use the same prepared ReasonRank data, global batch 128, 113 steps, learning rate 5×10^{-6}, and three seeds as the Q3E-0.6B comparison, with full-encoder fine-tuning. Only dense retrieval is evaluated; BGE-M3's other retrieval modes are not used.

Both query settings use the same BRIGHT corpus, relevance labels. [1] For GPT-4 reasoning, we use the released query field without human-annotated reasoning.

C.1.3 BASELINE OBJECTIVES

InfoNCE treats each positive separately, excluding other positives from its denominator, then averages across positives and queries:

$$\mathcal{L}_{\mathrm{CL}}(q) = -\frac{1}{|P_q|} \sum_{p \in P_q} \log \frac{\exp(s(q,p)/\tau)}{\exp(s(q,p)/\tau) + \sum_{n \in N_q} \exp(s(q,n)/\tau)}, \qquad \tau = 0.03. \tag{27}$$

InfoNCE uses the full valid cross-device pool, with cross-query document gradients. LambdaLoss uses teacher grades and normalizes its weighted logistic sum by the total swap weight within each query. Swap weights are detached absolute nDCG@10 changes, with logistic scale $1/0.03$ on cosine scores.

C.1.4 TRAINING HYPERPARAMETERS

Table 4 summarizes the training configurations.

BRIGHT adaptation disables gradient clipping, reference-policy KL, and auxiliary contrastive losses. Its main reward uses $\lambda = 0.5$. The shared step budget fixes query exposure across methods.

Exploration settings. The main BRIGHT recipe fixes alignment at $\rho = 0.70$ and solves $A_d(\kappa) = \rho$ for the model's embedding dimension d. The alignment sweep in Figure 2(d) uses

[1] BRIGHT dataset.

Table 4: Training settings for Q3E-0.6B BRIGHT adaptation. All 4B BRIGHT methods use LoRA (rank 16) with a learning rate of 2×10^{-4}.

Setting	Q3E-0.6B BRIGHT adaptation
Trainable parameters	Full shared encoder
Optimizer / learning rate	AdamW / 5×10^{-6}
Global / per-device batch	128 / 16 (8 devices)
Training steps	113
Schedule / warm-up fraction	Linear / 0.03
Weight decay	0.01
Training seeds	42, 3407, 2026
RL group size / alignment	64 / 0.70
RL reward	Graded nDCG + pairwise
Checkpoint selection	Final step

$\rho \in \{0.60, 0.70, 0.80, 0.90\}$, fixed throughout each run. The implementation logs reward spread, near-degenerate groups, and mean alignment.

C.1.5 Per-Seed Main Results

Table 5 reports the individual runs underlying Table 1. With GPT-4 reasoning queries, RELER exceeds InfoNCE at every paired seed on all three backbones. Group means and SDs are computed before rounding the displayed per-seed values.

Table 5: Per-seed BRIGHT nDCG@10 ($\times 100$), averaged over 12 subsets. The pretrained encoders are evaluated once. GPT-4 reasoning queries use the released `gpt4_reason.query` inputs.

		Original query			**GPT-4 reasoning query**		
Model	**Method**	42	3407	2026	42	3407	2026
Q3E-0.6B	Q3E-0.6B	15.10	–	–	20.38	–	–
	InfoNCE	22.50	22.56	22.87	26.40	26.59	26.56
	LambdaLoss	19.13	19.30	19.67	23.61	23.96	24.34
	RELER	23.27	23.57	23.29	27.15	27.19	27.10
Q3E-4B	Q3E-4B	18.70	–	–	22.01	–	–
	InfoNCE	30.17	30.17	29.89	30.12	30.70	30.05
	LambdaLoss	25.44	24.76	24.85	25.96	25.49	25.61
	RELER	29.98	31.06	30.33	30.77	31.52	30.92
BGE-M3	BGE-M3	10.66	–	–	21.45	–	–
	InfoNCE	12.67	12.51	12.71	20.69	20.63	20.52
	LambdaLoss	11.38	11.33	11.49	19.12	19.03	19.24
	RELER	14.48	14.46	14.54	23.77	23.91	23.84

C.2 BRIGHT Reference Baselines

Table 6 shows original-query BRIGHT results from Table 2 of Chen et al. (2026), excluding their MS MARCO fine-tuning controls and pretrained Qwen3-Embedding rows, and including their ReasonEmbed-Qwen3-4B Redapter model. It also includes our pretrained 4B baseline and trained recipes from Table 1. Published results were not rerun under our evaluation protocol; their scores provide context rather than controlled comparisons with RELER.

Table 6: Original-query BRIGHT nDCG@10 ($\times 100$). Literature rows are from Table 2 of Chen et al. (2026); MS MARCO fine-tuning controls and pretrained Qwen3-Embedding rows are omitted. ReasonEmbed-Qwen3-4B uses Redapter. Our 4B rows repeat Table 1; Avg. reports three-seed mean $\pm$ sample SD for trained recipes. Literature and our results use different training and evaluation protocols. Abbreviations follow the main table.

	StackExchange							Coding		Theorem-based			
Method	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	Avg.
General-purpose methods													
BM25	18.9	27.2	14.9	12.5	13.6	18.4	15.0	24.4	7.9	6.2	10.4	4.9	14.5
OpenAI-3-Large	23.3	26.7	19.5	27.6	12.8	14.3	20.5	23.6	2.4	8.5	23.5	11.7	17.9
Google-Gecko-1B-768	22.7	34.8	19.6	27.8	15.7	20.1	17.1	29.6	3.6	9.3	23.8	15.9	20.0
GritLM-7B	24.8	32.3	18.9	19.8	17.1	13.6	17.8	29.9	22.0	8.8	25.2	21.2	21.0
gte-Qwen2-7B-instruct	34.1	42.6	18.2	27.4	13.2	17.3	20.9	30.4	2.2	13.3	30.6	32.6	23.5
Tailored methods for reasoning-intensive retrieval													
ReasonIR-8B	26.2	31.4	23.3	30.0	18.0	23.9	20.5	35.0	10.5	14.7	31.9	27.2	24.4
RaDeR-gte-Qwen2-7B	34.6	38.9	22.1	33.0	14.8	22.5	23.7	37.3	5.0	10.2	28.4	35.1	25.5
Seed-1.5-Embedding	34.8	46.9	23.4	31.6	19.1	25.4	21.0	43.2	4.9	12.2	33.3	30.5	27.2
DIVER-Retriever (4B)	41.8	43.7	21.7	35.3	21.0	21.2	25.1	37.6	13.2	10.7	38.4	37.3	28.9
ReasonEmbed-Q3E-4B (Redapter)	55.4	54.5	34.9	46.9	34.0	36.1	37.4	34.5	13.6	11.3	41.4	45.1	37.1
Our Q3E-4B adaptation													
Q3E-4B	17.2	34.6	16.4	23.0	13.0	15.9	17.1	21.2	2.0	3.7	19.6	40.9	18.70
LambdaLoss	32.3	35.5	27.5	36.0	19.9	27.0	30.8	18.5	3.6	3.8	23.3	42.0	25.01 $\pm$0.37
InfoNCE	48.5	47.0	32.0	42.3	26.7	34.7	35.9	15.2	6.6	3.8	23.5	44.7	30.08 $\pm$0.16
RELER	48.9	50.2	32.9	44.3	25.8	33.4	33.6	13.8	11.3	3.6	20.7	46.9	30.46 $\pm$0.55

C.3 Complete BRIGHT Ablation Results

The BRIGHT ablation and sensitivity tables use Q3E-0.6B with original queries, $G = 64$, 113 training steps, and seeds 42, 3407, and 2026. Subset entries are three-seed means of nDCG@10 ($\times 100$); Avg. is the 12-subset macro-average with sample SD across seeds. Subset means are computed from the rounded per-subset exports; Avg. and SD retain the grouped export precision. Bold marks the highest displayed mean within each block, and shading marks the main RELER setting.

Table 7 gives the 12-subset breakdown of Table 2, retaining its row labels and order. All rows use own-query candidates and $\rho = 0.70$. The first block compares full RELER with its unprojected RLOO control, disabling CMP in both reward components while retaining the other settings. The second compares CMP, rollouts, and one-sided policies under standalone graded nDCG; the third reports binary rewards with product rollouts and RLOO+CMP. Paired rollouts use one-to-one query–bundle pairings, whereas product rollouts evaluate all G^2 pairings. The one-sided policies sample only the named embeddings while updating the shared encoder. These rollout comparisons all use graded nDCG@10 on the own-query candidate set.

Table 7: Complete subset results for Table 2. RELER uses $\lambda = 0.5$. Subset abbreviations follow Table 1.

	StackExchange							Coding		Theorem-based			
Training variant	**Bio.**	**Earth.**	**Econ.**	**Psy.**	**Rob.**	**Stack.**	**Sus.**	**Leet.**	**Pony**	**AoPS**	**TheoQ.**	**TheoT.**	**Avg.**
Graded nDCG + pairwise													
(a) RELER	**45.9**	**37.3**	**28.6**	**34.0**	**18.4**	**29.3**	**25.2**	9.4	**2.2**	2.3	19.0	28.9	**23.38** ± 0.17
(b) RELER w/o CMP	33.6	35.3	27.1	32.0	18.0	27.0	23.7	**10.2**	1.3	**2.7**	**19.7**	**32.4**	21.91 ± 0.70
Graded nDCG													
(c) RLOO+CMP / product	**40.5**	**38.0**	**28.0**	**34.7**	**18.2**	**25.4**	**23.7**	10.1	**2.3**	2.7	19.4	29.6	**22.72** ± 0.18
(d) RLOO / product	27.7	31.9	27.3	32.4	17.3	23.8	23.1	10.2	1.3	3.0	**19.7**	32.4	20.85 ± 0.68
(e) RLOO / paired	20.6	29.4	24.6	27.8	14.2	20.6	20.7	12.2	1.2	**3.4**	18.5	31.4	18.71 ± 0.53
(f) RLOO / query only	15.5	21.3	19.2	20.0	14.2	16.5	18.8	**16.2**	1.1	2.8	18.0	26.7	15.85 ± 0.26
(g) RLOO / document only	21.0	32.0	24.6	25.7	14.4	18.7	18.1	10.8	1.1	2.5	17.5	**34.8**	18.45 ± 1.56
Binary rewards (product rollouts, RLOO+CMP)													
(h) Binary nDCG	27.0	24.1	21.3	21.2	12.2	**29.2**	**17.1**	7.5	**1.4**	2.0	16.1	30.1	17.45 ± 0.65
(i) Binary MRR	**32.9**	**28.0**	**22.9**	**22.2**	**13.8**	27.7	16.5	**8.2**	1.2	**2.8**	**16.7**	**34.0**	**18.90** ± 0.43

C.4 Pairwise Weight and Alignment

Table 8 gives the full subset results behind Figure 2(c,d). Both sweeps use own-query candidates and product rollouts with RLOO+CMP. The first varies λ at $\rho = 0.70$; the second varies ρ at $\lambda = 0.5$. The main setting ($\lambda = 0.5$, $\rho = 0.70$) is repeated in both blocks. The gain from $\lambda = 0.5$ comes from StackExchange; Coding and Theorem-based retrieval score lower than with $\lambda = 0$. Changing alignment has a smaller effect on the average. Although $\rho = 0.80$ scores 0.08 points above the main setting, it does not improve every paired seed.

Table 8: Complete BRIGHT subset results for the pairwise-weight and alignment sweeps in Figure 2(c,d). The first column is λ in the upper block and ρ in the lower block.

	StackExchange							Coding		Theorem-based			
Setting	**Bio.**	**Earth.**	**Econ.**	**Psy.**	**Rob.**	**Stack.**	**Sus.**	**Leet.**	**Pony**	**AoPS**	**TheoQ.**	**TheoT.**	**Avg.**
Pairwise weight λ ($\rho = 0.70$)													
0	40.5	**38.0**	28.0	**34.7**	18.2	25.4	23.7	**10.1**	**2.3**	**2.7**	**19.4**	**29.6**	22.72 ± 0.18
0.5	**45.9**	37.3	**28.6**	34.0	**18.4**	29.3	**25.2**	9.4	2.2	2.3	19.0	28.9	**23.38** ± 0.17
1.0	43.6	34.9	27.2	32.8	18.1	**29.7**	23.9	9.0	2.0	1.9	18.9	28.9	22.57 ± 0.18
Alignment ρ ($\lambda = 0.5$)													
0.60	**46.2**	**38.3**	**28.8**	**34.3**	18.2	**29.6**	24.9	8.8	2.1	**2.7**	18.9	27.5	23.36 ± 0.25
0.70	45.9	37.3	28.6	34.0	18.4	29.3	**25.2**	9.4	**2.2**	2.3	19.0	28.9	23.38 ± 0.17
0.80	45.1	37.5	28.6	**34.3**	**19.6**	28.7	24.9	10.0	2.0	2.2	19.0	29.6	**23.46** ± 0.25
0.90	42.4	37.1	28.2	33.8	19.1	28.2	24.6	**10.3**	2.0	2.2	**19.5**	**30.6**	23.17 ± 0.19

C.5 Gradient Variance: Protocol and Results

Matched main-recipe probes. We fix Q3E-0.6B before fine-tuning and at steps 50 and 113 of the seed-3407 RELER run ($\lambda = 0.5$). At each state, three fixed training microbatches (Stack Overflow, math-theorem, and biology) receive 64 independent rollout draws. Both estimators use the same actions, own-query candidate shortlist, graded reward table, and pairwise inputs. The unprojected control replaces CMP in both reward components with RLOO score-function terms; it does not leave the pairwise CMP path active. Draw-level hashes and exact pairwise-reward agreement check the pairing. Dropout is disabled and no optimizer update is made. The measurements include every trainable encoder parameter.

For gradients $g_1, \ldots, g_N$ in one fixed state and microbatch, define $\bar{g} = N^{-1} \sum_k g_k$ and $\widehat{V} = (N-1)^{-1} \sum_k \|g_k - \bar{g}\|^2$, with $N = 64$. The bias-corrected squared signal estimate is $\widehat{S}^2 = \|\bar{g}\|^2 - \widehat{V}/N$, and we report the single-draw signal-to-noise ratio $\sqrt{\max(\widehat{S}^2, 0)/\widehat{V}}$. For paired differences $\delta_k = g_k^{\mathrm{CMP}} - g_k^{\mathrm{RLOO}}$, we compare $\|\bar{\delta}\|^2 - \widehat{V}_\delta/N$ with zero as a descriptive Monte Carlo resolution check, not a formal significance test.

Complete-gradient measurements. The variance reduction holds in all nine probes (Table 9): CMP lowers full-gradient variance by 291–590 times, and raises the single-draw SNR above one throughout. The cost is an 11–14% increase in forward/backward time per draw, so the reduction remains large when measured by variance times runtime. These are fixed-state measurements of rollout noise; the BRIGHT seed SD also reflects the subsequent optimization trajectory.

Table 9: Complete-encoder gradient measurements for the combined reward. Each row uses 64 paired rollout draws at one fixed encoder state and microbatch. V is the sample gradient variance and t is mean forward/backward time per draw. Ratios compare CMP to RLOO; SNR uses the bias-corrected signal estimate.

		SNR		CMP / RLOO		
Encoder state	**Microbatch**	RLOO	CMP	V	t	Vt
Before fine-tuning	Stack Overflow	0.25	5.35	0.00226	1.14	0.00258
	Math-theorem	0.27	6.00	0.00182	1.11	0.00203
	Biology	0.20	4.85	0.00169	1.13	0.00191
Step 50	Stack Overflow	0.23	4.07	0.00322	1.14	0.00367
	Math-theorem	0.22	3.81	0.00321	1.11	0.00358
	Biology	0.23	4.53	0.00248	1.13	0.00280
Step 113	Stack Overflow	0.23	3.95	0.00343	1.14	0.00392
	Math-theorem	0.19	3.30	0.00305	1.12	0.00341
	Biology	0.19	3.64	0.00283	1.13	0.00318

After correcting for sampling noise, the estimated squared difference between the two mean gradients is nonpositive in five probes. In the remaining four, it is at most 2.81% of CMP's estimated squared signal. These finite-sample measurements supplement the unbiasedness proof in Appendix A.2. The proof bounds each summand's variance; Table 9 measures the variance after summing all contributions to the encoder gradient.

Gradient-cloud visualization. Figure 2(a) uses the math-theorem microbatch before fine-tuning. We project each complete gradient onto eight fixed random sign vectors (seed 20260925), shared by both estimators, then display the axes with the largest and smallest absolute CMP mean coordinates. Both axes use the same scale, the absolute CMP mean on the first axis, and its sign is oriented so that the CMP mean has first coordinate one. The inset zooms into the CMP cloud. This selected two-dimensional view illustrates the paired draws; all quantitative variance and SNR comparisons use the complete parameter gradients in Table 9.

D FIXED-INDEX RAG EXPERIMENTS

D.1 CORPUS AND TRAINING DATA

The fixed-index QA protocol in Section 4.2 builds a Q3E-0.6B index over FlashRAG's `wiki18_100w` release: 21,015,324 100-word passages from December 2018 English Wikipedia (Karpukhin et al., 2020).[2] The index and Qwen2.5-7B-Instruct generator stay frozen while the query encoder adapts. All trained rows use NQ and HotpotQA for one epoch, seed 42, and anchor coefficient 0.5. InfoNCE uses once-mined candidates, whereas query-only RLOO searches the full index. The recipes differ in both candidate access and objective. Training labels use DPR NQ human-positive passages and HotpotQA supporting facts mapped to this corpus; queries without complete evidence mappings are excluded.

D.2 TRAINING OBJECTIVES AND HYPERPARAMETERS

Passage judgments and rewards. The prepared NQ and HotpotQA records associate corpus passage IDs with three judgments: external binary qrels, supporting-evidence membership, and reference-answer containment. Qrels come from DPR human-positive NQ passages and corpus-aligned HotpotQA supporting facts. For graded nDCG, qrel positives receive grade 3, other supporting-evidence passages grade 2, and remaining answer-containing passages grade 1; all others receive grade 0. The highest-priority applicable grade is used. Gains are $2^{\ell} - 1$. Retrieved IDs are matched against these fixed passage judgments; IDs absent from the judgment set receive zero gain. Ideal DCG is computed from the highest grades in the query's complete prepared judgment set, independently of the sampled retrieved list. Zero ideal DCG gives zero reward. Answer-F1 rewards use the maximum normalized token F1 over reference answers; the mixed reward averages this value and graded nDCG@10.

InfoNCE candidates and masking. The offline pool is mined once with the pretrained query encoder at depth 1,000, retaining all mapped qrel positives within this fixed-size pool. InfoNCE uses the binary qrel positives in its numerator, at temperature 0.03. Answer-containing passages that are not qrel positives are excluded from the negative denominator, rather than treated as negatives. Query-only RLOO instead searches the frozen full-corpus index for each sampled action. All four trained recipes use the cosine anchor in Appendix D.3; Table 10 lists the common optimizer and rollout settings.

Table 10: Fixed-index RAG training and generation settings. The query encoder is fully fine-tuned; the document index and generator remain frozen.

Setting	Value
Training budget / seed	One epoch / 42
Optimizer / learning rate	AdamW / 5×10^{-6}
Global / per-device batch	128 / 16
Schedule / warm-up / weight decay	Cosine / 0.03 / 0.01
Training query limit	128 tokens
InfoNCE temperature / anchor coefficient	0.03 / 0.5
RL group size / alignment	32 / 0.80
RL baseline / advantage normalization	Leave-one-out / none
Retrieval reward / generator context depth	Top 10 / top 10
Generator	Qwen2.5-7B-Instruct
Decoding / input limit / output limit	Greedy / 2,048 / 32 tokens

[2]Corpus: `https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets`.

D.3 Fixed-Index Query Anchor

Let $h_0(q)$ denote the normalized query embedding produced by the pretrained encoder, and let $h_\theta(q)$ be the current deterministic query embedding. For a training batch $\mathcal{B}$, the fixed-index experiments add the following term to either the InfoNCE or query-only RLOO loss:

$$\mathcal{L}_{\text{anchor}}(\theta) = \alpha_{\text{anc}} \frac{1}{|\mathcal{B}|} \sum_{q \in \mathcal{B}} \left(1 - h_\theta(q)^\top h_0(q)\right), \qquad \alpha_{\text{anc}} = 0.5. \tag{28}$$

The initial embeddings are computed once, before the first optimizer step, using the same query prompt, tokenizer, pooling, and normalization as the trainable encoder. They are stored by training-row position in FP16 and treated as constants; gradients pass only through $h_\theta(q)$. The term changes neither the retrieval or answer reward nor the frozen document index and generator. Inference uses $h_\theta(q)$ directly and requires no anchor lookup. Thus the penalty directly constrains training queries; any effect on unseen queries comes through the learned encoder parameters.

The directional penalty also has a policy interpretation when both query vMF distributions use the same fixed concentration κ. Writing $A_d(\kappa)$ for the vMF mean resultant length, their KL divergence is

$$D_{\text{KL}}(\text{vMF}(h_\theta(q), \kappa) \| \text{vMF}(h_0(q), \kappa)) = \kappa A_d(\kappa) \left(1 - h_\theta(q)^\top h_0(q)\right). \tag{29}$$

Hence Eq. (28) is proportional to a KL penalty for fixed κ; α_{anc} is the coefficient on cosine distance, not the coefficient on the KL itself. This equivalence does not apply if the two policies use different concentrations.

D.4 Generation and Evaluation

We present passages in retrieval order, with numbered document blocks and their titles, followed by the original question, using the generator's chat template. The system prompt is: *"Answer the question based on the given documents. Only give me the answer and do not output any other words."* Prompts exceeding the input budget drop the lowest-ranked passages first; if one remaining passage is still too long, its prefix is retained together with the question. Greedy decoding uses temperature zero. Evaluation caches generated answers by query, ordered passage IDs, context, and generation configuration. EM/F1 lower-case text, remove punctuation and English articles, and normalize whitespace; F1 uses token overlap and requires an exact normalized match for yes/no/noanswer references or predictions. Hit@10 checks normalized reference-answer containment in any of the ten passages, independently of generator input truncation.

D.5 Per-Dataset Results

The averages in Table 3 conceal differences across datasets. InfoNCE remains best on NQ, reaching 47.98 F1 despite its lower overall average. Among the RL rewards, answer F1 exceeds graded nDCG in F1 on five of seven datasets; NQ and Bamboogle are the exceptions. The mixed reward adds 0.40 out-of-domain F1 points over graded nDCG, alongside a 0.22-point Hit@10 gain. Its F1 margin over graded nDCG falls from $+0.30$ points across seven datasets to $+0.13$ without Bamboogle. Relative to answer-F1 training, the mixed reward's $+0.24$-point seven-dataset F1 margin becomes -0.25 without Bamboogle.

Table 11 reports the dataset results and query counts behind Table 3. Each dataset has equal weight in the group means despite the different query counts.

Table 11: Per-dataset fixed-index QA results (×100), seed 42. n is the number of evaluation queries; 2Wiki denotes 2WikiMultihopQA. The same document index, generator, and query counts are used for all recipes.

Method	NQ	HotpotQA	PopQA	TriviaQA	2Wiki	MuSiQue	Bamboogle
Queries (n)	3,610	7,405	14,267	11,313	12,576	2,417	125
Answer EM							
Q3E-0.6B	34.99	32.59	42.22	61.00	28.76	7.36	19.20
InfoNCE	37.01	30.26	42.07	60.26	23.81	5.63	19.20
RLOO, graded nDCG	36.01	33.32	41.82	60.54	29.03	6.48	19.02
RLOO, answer F1	35.94	33.49	42.40	61.43	28.98	7.27	16.80
RLOO, nDCG + F1	36.12	33.52	42.70	61.20	28.75	6.80	21.60
Answer F1							
Q3E-0.6B	45.68	42.67	49.25	69.67	35.07	14.93	28.07
InfoNCE	47.98	40.17	49.20	69.08	30.12	12.43	27.97
RLOO, graded nDCG	46.69	43.48	48.96	69.29	35.16	14.26	27.73
RLOO, answer F1	46.66	43.67	49.53	69.92	35.34	14.96	25.94
RLOO, nDCG + F1	46.63	43.65	49.01	69.72	34.87	14.73	29.09
Hit@10							
Q3E-0.6B	76.84	54.50	76.74	79.24	46.25	27.89	29.60
InfoNCE	79.86	48.73	77.61	79.08	34.18	21.14	21.60
RLOO, graded nDCG	78.78	55.71	76.83	79.60	46.73	27.10	29.60
RLOO, answer F1	77.61	55.93	76.90	79.53	46.14	29.29	31.20
RLOO, nDCG + F1	78.09	55.65	76.72	79.76	46.26	27.60	30.60

E Embedding Training from a General-Purpose LLM

We also train embeddings directly from `Qwen3-0.6B`, a general-purpose post-trained LLM, without embedding-specific initialization. Both InfoNCE and RELER use the E2Rank training data (Liu et al., 2025a), full fine-tuning for 1,200 steps, learning rate 5×10^{-6}, global batch size 128, and seed 42. RELER uses graded nDCG@10 plus pairwise feedback ($\lambda = 0.5$), own-query candidates, product rollouts, RLOO+CMP, $G = 64$, and $\rho = 0.70$. We evaluate the final checkpoints on MTEB English v2 (Muennighoff et al., 2023; Enevoldsen et al., 2025), covering 41 tasks in seven types.

Table 12: Direct embedding training from `Qwen3-0.6B` on MTEB English v2 (×100), seed 42. Each type column averages its tasks; Task avg. weights all 41 tasks equally, while Type avg. weights the seven task-type means equally. Classif.: classification; Clust.: clustering; Pair class.: pair classification; Rerank.: reranking; Summ.: summarization. Bold marks the higher score in each column.

Objective	Classif.	Clust.	Pair class.	Rerank.	Retrieval	STS	Summ.	Task avg.	Type avg.
InfoNCE	72.84	44.19	**82.11**	44.80	**51.52**	75.94	**29.49**	60.99	**57.27**
RELER	**73.41**	**45.81**	79.76	**45.13**	50.04	**76.81**	26.61	**61.01**	56.80

RELER and InfoNCE are nearly tied in the all-task average (Table 12), but InfoNCE leads in retrieval by 1.48 points and also has the higher task-type average. This experiment provides no clear advantage for RELER when starting from a general-purpose LLM. The comparison uses one seed and complete recipes with different supervision and candidate handling.

F Extended Related Work

Retriever training and learning to rank. Contrastive training underlies dense retrieval and general-purpose embedding adaptation (Karpukhin et al., 2020; Wang et al., 2022; BehnamGhader et al., 2024). Learning-to-rank methods complement this representation-learning perspective with pairwise and listwise surrogates, including differentiable approximations to retrieval metrics (Burges et al., 2005; Liu, 2009; Taylor et al., 2008; Pobrotyn & Białobrzeski, 2021). Policy gradients have also been used to optimize retrieval utility through MDP-based ranking policies and stochastic rankings with fairness constraints (Wei et al., 2017; Singh & Joachims, 2019). RELER studies a one-step embedding action with offline relevance feedback. Its contribution is the continuous policy over query and document embeddings and its gradient estimator; matched InfoNCE and LambdaLoss controls assess its value beyond supervised adaptation.

Reward-based retrieval and reasoning. ROPG-RL samples a profile document from a categorical policy over a fixed candidate subset and uses a frozen LLM's generation metric as reward (Salemi et al., 2024). RELER's fixed-index mode instead samples a query vector and searches the full index, using retrieval or answer feedback in our QA experiments. Retrieval-GRPO updates dense retrieval using rewards on dynamically retrieved products (Liu et al., 2025c), while Rank-R1 and Search-R1 optimize generated reranking or search-and-reasoning sequences (Zhuang et al., 2025; Jin et al., 2025a). GRACE optimizes rationale tokens with representation-based rewards and pools rationale-conditioned states into embeddings (Sun et al., 2026); Embed-RL trains a separate reasoner with feedback from a frozen multimodal embedder (Jiang et al., 2026). The latter two approaches use generated reasoning to support retrieval. RELER directly samples query and document embeddings during training and retains deterministic single-vector inference.

Embedding policies and gradient estimation. vMF distributions also have precedents in hyperspherical representation learning (Davidson et al., 2018). REINFORCE baselines, multi-sample LOO, and product-form recombination are established estimation tools (Williams, 1992; Greensmith et al., 2004; Kool et al., 2019; Ahmadian et al., 2024; Kuntz et al., 2022). Their use in RELER's query and document gradient estimator is detailed in Section 3.2 and Appendix A.1. The closest antecedent of CMP is document-space projection: Wang et al. (2019) project exploration directions after interleaved feedback in linear rankers. Our conditional-expectation construction includes vMF means and accounts for each query–document reward cell and its LOO baseline before updating a shared encoder. It applies Rao–Blackwell averaging (Casella & Robert, 1996), with a per-term variance guarantee; variance of the summed encoder gradient is evaluated empirically. The projection must retain every reward-relevant direction, so the guarantee does not extend to using only returned full-corpus top-k hits.