REVELA+: ADAPTING MULTI-STAGE LANGUAGE MODELING TRAINING FOR REASONING RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning-intensive retrieval is difficult because useful documents may support an answer through multi-step inference rather than resemble the query. Dense retrievers are mostly trained contrastively on annotated or synthetic query–document pairs, which reduces each document to a binary label on a similarity score: it records which document is relevant, but not how, or how much, it helps. A language model's next-token loss offers a finer signal, and Revela (Cai et al., 2026) uses it to train a standalone dense retriever by letting the retriever weight the in-batch sequences that the model attends to. Revela, however, applies this only to pre-training, whereas language models are trained through a pipeline of pre-training, supervised fine-tuning (SFT), and preference tuning. Can a dense retriever be trained through this full pipeline, and what does each stage contribute? We introduce **Revela+**, which carries the retriever and the language model through all three stages: while attending to retriever-weighted sequences in its batch, the model predicts document text, then answers to queries, and then compares answers grounded in relevant and mined negative documents. No stage adds a document-ranking loss. Across five encoders from two model families, mean retrieval performance improves at both stage transitions on BRIGHT and BEIR. For Llama-3.1-8B, mean BRIGHT NDCG@10 rises from 25.1% after pre-training to 29.8% after SFT and 32.1% after preference tuning, surpassing contrastively trained ReasonIR-8B (Shao et al., 2025) (24.4%) and RaDeR-7B (Das et al., 2025) (26.0%) with 6.6–7.7$\times$ less training compute than ReasonIR-8B.

1 INTRODUCTION

Dense retrievers provide efficient access to external knowledge by representing queries and documents in a shared embedding space. For reasoning-intensive queries, however, a relevant document may supply a premise, an analogous example, or an intermediate result without resembling the query (Su et al., 2025). A retriever must therefore learn how a document helps solve a problem, not only what it is about.

Most retrievers, including reasoning-oriented ones such as ReasonIR (Shao et al., 2025) and RaDeR (Das et al., 2025), are trained contrastively, pulling each query's embedding toward a labeled relevant document and away from others. General-purpose embedders follow the pre-train–fine-tune paradigm but apply this same objective in both stages (Li et al., 2023; Zhang et al., 2025). Contrastive learning suits semantic retrieval, where relevance and similarity largely coincide, but its supervision is a binary label on a similarity score: it records which document is relevant, not how, or how much, it helps.

A second line of work derives the signal from a language model, either trained alongside the retriever or kept frozen as a black box: the retriever learns to favor documents under which the model better predicts a target (Guu et al., 2020; Izacard et al., 2023; Shi et al., 2024; Lin et al., 2024). This signal is graded rather than binary, but obtaining it requires retrieving from a corpus throughout training. Revela (Cai et al., 2026) avoids this search: the retriever weights the other sequences in the training batch that the language model attends to, so the loss on every predicted token trains the retriever. It applies this mechanism to pre-training alone.

Language models themselves are trained through a pipeline of pre-training, supervised fine-tuning (SFT), and preference tuning (Ouyang et al., 2022; Rafailov et al., 2023), yet neither line has adapted

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

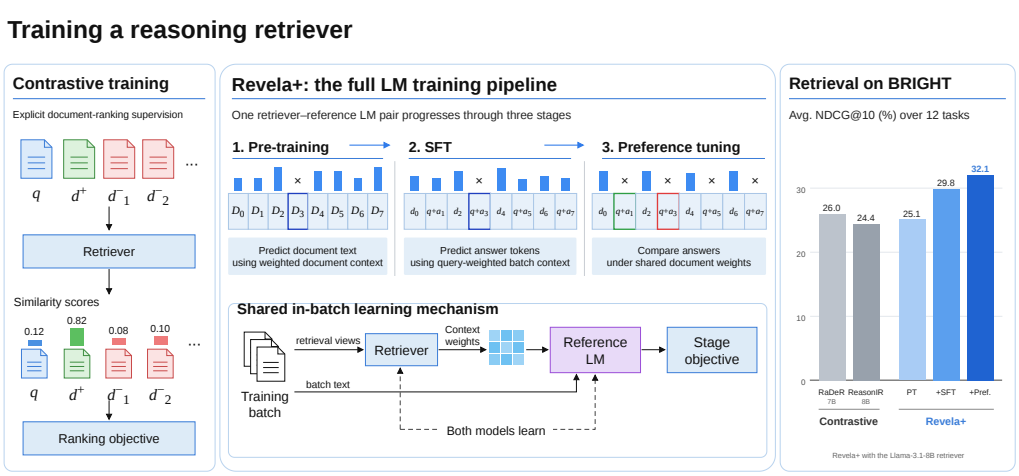


Figure 1: Contrastive retriever training versus Revela+.

this full pipeline to train a retriever. We therefore ask: *Can a dense retriever be trained through the full language-model training pipeline, learning at every stage only from the language model's objective, and what does each stage contribute?*

Because Revela's retriever learns from the language model's own training loss, every stage of language-model training can supply its signal. We introduce **Revela+** (Figure 1), which carries the retriever and the language model together through all three stages. At every stage, the documents come from the training batch itself rather than from a corpus: each batch groups related sequences, and the language model predicts each target while attending to the other sequences, weighted by the retriever's scores, so the retriever learns to favor those that make the target predictable. Pre-training predicts a document chunk, with the retriever scoring the other chunks from the chunk's opening text. SFT predicts the answer to a query (in a control, the query itself), with the retriever scoring the batch's documents from the query alone, as at inference. Preference tuning compares two answers to a query, one grounded in its relevant document and one in a mined negative, and rewards retriever weights under which the first is the more likely. Each stage continues from the models of the previous one, and no stage adds a document-ranking loss. At inference, the language model is removed and the retriever performs ordinary dense retrieval.

Across five encoders from two model families, retrieval improves at both stage transitions on BRIGHT (Su et al., 2025) and BEIR (Thakur et al., 2021). The Llama-3.1-8B retriever rises from 25.1% NDCG@10 on BRIGHT after pre-training to 29.8% after SFT and 32.1% after preference tuning, above the contrastively trained ReasonIR-8B (24.4%) and RaDeR-7B (26.0%; Table 1), with 6.6–7.7$\times$ less training compute than ReasonIR-8B on the same backbone. Controlled comparisons show that each stage builds on the one before it: SFT gains from the pre-trained retriever–LM pair, and the comparative term of preference tuning improves retrieval beyond continued answer prediction on the same examples. For the 8B retriever on BRIGHT-Pro (Zhao et al., 2026), which annotates the aspects of each query's information need, later stages retrieve evidence for more of these aspects, and preference tuning also ranks that evidence earlier.

Our contributions are as follows:

- We introduce Revela+, which trains a dense retriever through the full language-model training pipeline of pre-training, SFT, and preference tuning, learning at every stage only from the language model's objective.
- We show across five encoders and two benchmarks that each stage improves retrieval, with Revela+-8B surpassing contrastively trained reasoning retrievers on BRIGHT.
- Through controlled comparisons and aspect-level analysis, we show how each stage builds on the previous one and how it changes the evidence the retriever finds.

2 Related Work

Contrastive retriever training. Dense retrievers are usually trained with InfoNCE (van den Oord et al., 2018), which scores each query's labeled document above in-batch and mined negatives. Unsupervised pre-training pairs two crops of one document (Izacard et al., 2022), and general-purpose embedders, increasingly built on decoder-only LLMs (Ma et al., 2024; Wang et al., 2024), train on weakly supervised pairs before labeled ones (Li et al., 2023; Zhang et al., 2025).

Reasoning-intensive retrieval. BRIGHT tests queries whose relevant documents connect through inference rather than lexical similarity (Su et al., 2025). ReasonIR and RaDeR train contrastive encoders on synthesized or derived relevance supervision; DIVER adds query expansion, sparse–dense fusion, and reranking (Shao et al., 2025; Das et al., 2025; Sun et al., 2025). These methods change the training data but keep the contrastive objective; Revela+ changes the objective.

Language-model and preference supervision. REALM (Guu et al., 2020), with a masked-language-model encoder, and RAG (Lewis et al., 2020), with an encoder–decoder generator, marginalize the target likelihood over retrieved documents. Atlas (Izacard et al., 2023) distills document targets from an encoder–decoder reader, ART (Sachan et al., 2023) from a frozen LM's likelihood of the question, and the decoder-only REPLUG (Shi et al., 2024) and RA-DIT (Lin et al., 2024) from continuation likelihoods, REPLUG keeping the language model frozen as a black box. All of them retrieve from a corpus during training, whereas Revela backpropagates next-token likelihood through encoder-weighted attention over the other sequences in the batch (Cai et al., 2026). For preference learning, Syntriever ranks LLM-preferred passages with a partial Plackett–Luce objective and PORTS trains tool retrieval from preferences and frozen-LM scores (Kim & Baek, 2025; Molfetta et al., 2025); Revela+ instead routes a CPO-style preference gradient (Xu et al., 2024) through the predictive LM rather than a document-ranking loss.

3 Revela+: Multi-stage Retriever Learning

3.1 Shared Retriever–LM Learning Mechanism

We follow Revela (Cai et al., 2026) in training a dense retriever through a predictive language model (LM). During training, sequences are grouped into batches so that they can provide context for one another. The context therefore comes from the batch itself; no corpus is searched during training. The retriever scores the available context, and these scores determine how strongly it contributes to the LM's predictions. Because the predictions depend on the retrieval scores, the LM's training objective also provides a learning signal to the retriever.

The retriever f_ϕ uses a pretrained decoder-only Transformer to encode text as normalized final-layer end-of-sequence (EOS) embeddings (Ma et al., 2024; Wang et al., 2024). The predictive model p_θ is a separate autoregressive language model, termed the *reference LM* following Revela (Cai et al., 2026). For a sequence i whose tokens are being predicted, the retriever assigns a similarity score s_{ij} to each candidate context sequence j. These scores are computed using cosine similarity between text embeddings; Section 3.2 specifies the texts encoded at each stage. We convert the scores into context weights:

$$W_{ij} = \frac{\exp(s_{ij}/\tau)}{\sum_{k\in\mathcal{E}_i}\exp(s_{ik}/\tau)}, \qquad j \in \mathcal{E}_i.$$

Here, $\mathcal{E}_i$ contains the indices of sequences eligible to provide context to sequence i, excluding i itself. We set $W_{ij} = 0$ otherwise. The temperature $\tau > 0$ controls the concentration of the weights. We denote the resulting weight matrix by W_ϕ to indicate its dependence on the retriever parameters.

The reference LM uses this context through Revela's cross-sequence attention mechanism (Cai et al., 2026). It first processes each sequence x_j independently and caches its keys K_j and values V_j at every layer. When it predicts token t of sequence i, each attention layer then computes

$$o_{i,t} = \mathrm{Attn}\big(Q_{i,t}, K_{i,\le t}, V_{i,\le t}\big) + \sum_{j\in\mathcal{E}_i} W_{ij}\,\mathrm{Attn}\big(Q_{i,t}, K_j, V_j\big),$$

where $Q_{i,t}$ is the token's attention query and each Attn applies its own softmax over the tokens of one sequence; the cross-sequence terms are additionally normalized (Appendix A). We write

$\mathcal{C}_i = (x_j)_{j \in \mathcal{E}_i}$ for this eligible context. The retriever therefore determines how much each context sequence contributes, while the LM determines what to read within it.

The training objective updates both models: gradients reach the retriever through the context weights and also update the reference LM's token embeddings. Its transformer layers stay frozen, which limits how much of a stage's prediction task the reference LM can absorb by itself; we also train them with LoRA for comparison (Section 4.2). This makes the LM's prediction task a source of retrieval supervision, without a separate document-ranking objective. Curated relevance still shapes training, but only through the data: query–document pairs determine which documents condition each synthesized answer and which sequences share a batch, while the loss scores LM tokens alone. The following section explains how pre-training, SFT, and preference tuning each supply a learning signal through this shared mechanism.

3.2 Training across Stages

We apply the shared mechanism to the three stages of language-model training—pre-training, SFT, and preference tuning—which predict documents, predict answers, and compare answers, respectively. Each stage specifies the training examples, the text encoded by the retriever, the context available to the reference LM, and the objective. Both models carry their parameters forward.

Because pre-training has no explicit queries and later stages include answers that are unavailable at retrieval time, we distinguish the full LM inputs from their *retrieval views*, the text used to compute the similarity scores in Section 3.1; each stage below defines both.

In the batch illustrations, underlining marks tokens scored by next-token prediction, and the bracket marks the two answer spans the comparative term compares.

3.2.1 Pre-training: Predicting Document Text

Pre-training starts both models from pretrained language-model checkpoints, rather than from scratch, and trains them on document prediction. Following Revela (Cai et al., 2026), we group related document chunks—contiguous spans of document text—within a batch, treating each chunk as one sequence. These sequences can provide context for one another. We apply this construction to a mixture of general, mathematical, and code text (Appendix B.1).

Without explicit queries, retrieval views are constructed from the documents themselves. To predict a sequence D_i, the retriever uses a prefix of that sequence to score the other full sequences in the batch. The reference LM processes the full sequence and uses the resulting retrieval weights to incorporate context from the other sequences. The retriever is therefore rewarded for weighting chunks that make the predicted text more predictable, such as other chunks of the same document.

Row	Retriever inputs: Pseudo-query	Retriever inputs: Candidate sequence	Reference-LM input
0	$\text{prefix}(D_0)$	D_0	$\underline{D_0}$
1	$\text{prefix}(D_1)$	D_1	$\underline{D_1}$
⋮	⋮	⋮	⋮

Gray: document.
Blue: query or pseudo-query text.
Underline: next-token prediction term.

For $D_i = (x_{i,0}, \ldots, x_{i,T_i-1})$, let $\mathcal{C}_i^{\text{pre}}$ denote its eligible context. The objective is

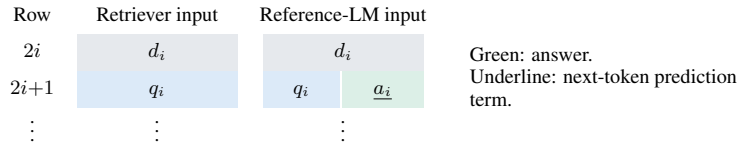
$$\mathcal{L}_{\text{pre}} = -\frac{1}{N_{\text{tok}}} \sum_{i,t} \log p_\theta(x_{i,t} \mid x_{i,<t}, \mathcal{C}_i^{\text{pre}}; W_\phi)\,, \tag{1}$$

where N_{tok} is the number of valid prediction targets, excluding padding. The objective includes tokens throughout the sequence; the prefix restriction applies only to the retriever's input. This stage supplies document-prediction supervision without annotated query–document pairs.

3.2.2 Supervised Fine-tuning: Predicting Answers

SFT makes the prediction task question-specific. Continuing from the models of the first stage (Section 3.2.1), we use document–query–answer examples (d_i, q_i, a_i), where the answer is synthesized using the query and its associated document (Appendix B.2). Predicting these answers provides supervision through the LM's use of context to address the query.

Each example contributes two LM sequences: the document d_i and the concatenated query and answer $q_i + a_i$. Their retrieval views are d_i and q_i, respectively. The answer therefore supplies prediction supervision while the retriever scores context using the query alone. The retriever is therefore rewarded for weighting, from the query alone, the documents that make its answer predictable, such as the document the answer was synthesized from. For each query–answer sequence, all other batch sequences are eligible context, including document and query–answer sequences; a query–answer context sequence is represented to the retriever by its query.



Let $\mathcal{C}_i^{\mathrm{SFT}}$ denote the eligible context for example i's answer. We optimize

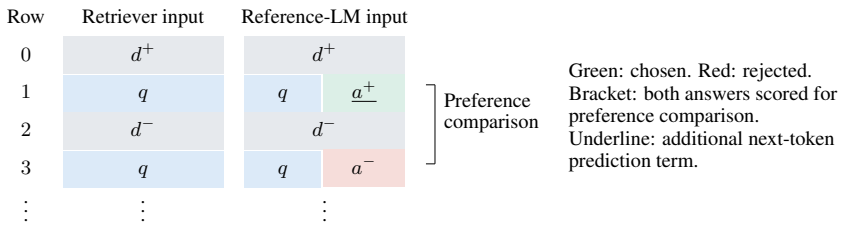
$$\mathcal{L}_{\mathrm{SFT}} = -\frac{1}{N_{\mathrm{ans}}} \sum_{i,t} \log p_\theta\big(a_{i,t} \mid q_i, a_{i,<t}, \mathcal{C}_i^{\mathrm{SFT}}; W_\phi\big)\,, \tag{2}$$

where N_{ans} counts the scored answer tokens. Document and query tokens receive no direct prediction loss. The change from pre-training is thus the task supplying supervision: context is now used to predict an answer to a particular question. We call this objective Answer NTP. As a control, Query NTP keeps the same document–query batches but predicts the query tokens instead, so training needs no synthesized answers (control **(B)**, Section 4).

3.2.3 Preference Tuning: Comparing Answers

Preference tuning introduces comparative supervision beyond predicting an answer alone. We continue the SFT models using chosen and rejected answers, synthesized from a query paired with a positive document d^+ and a mined negative document d^-, respectively (Appendix B.3). Each example supplies the LM with both documents and both query–answer sequences. The retriever encodes the documents and uses the query alone to represent either query–answer sequence. A rejected answer is not judged incorrect; it is the answer synthesized from the mined negative, so each preference pair encodes which document an answer is grounded in.

During training, both answers are evaluated using the same query-derived weights over the document sequences in the batch, including the positive and mined negative documents. The document associations used during answer synthesis therefore do not restrict the context available when scoring either answer. Query–answer sequences are excluded from the available context, preventing either answer from attending to the other. The mined negative remains available as competing document context.



Let $\mathcal{C}$ denote this shared document context. For each answer, define its summed token log-probability:

$$\ell_\pm = \sum_t \log p_\theta\big(a_t^\pm \mid q, a_{<t}^\pm, \mathcal{C}; W_\phi\big)\,.$$

We combine a pairwise comparative term with continued chosen-answer prediction:

$$\mathcal{L}_{\mathrm{pref}} = -\log \sigma(\beta(\ell_+ - \ell_-)) - \lambda \frac{\ell_+}{T_+}, \tag{3}$$

where σ is the sigmoid function, T_+ is the number of scored chosen-answer tokens, $\beta > 0$ scales the preference margin, and $\lambda \geq 0$ weights the chosen-answer prediction term. We average the loss over preference pairs. Only the chosen-answer prediction term is normalized by answer length.

The comparative term favors the chosen answer relative to its rejected alternative, while the prediction term continues to support the chosen answer's likelihood. This objective follows the CPO-style combination of pairwise preference learning and chosen-answer prediction (Xu et al., 2024). Unlike DPO (Rafailov et al., 2023), it uses no fixed-reference-policy likelihood ratio. The reference LM remains the predictive model carried forward from SFT. Because both answers are scored under the same weights, the comparative term rewards the retriever for weighting documents that make the chosen answer more likely than the rejected one, such as d^+ over d^-, while the prediction term continues to reward documents that support the chosen answer.

At inference, the reference LM is discarded: the final retriever f_ϕ encodes queries and documents independently and ranks documents by cosine similarity, so document embeddings can be computed in advance.

4 Experiments

4.1 Experimental Setup

Benchmarks and metrics. We use BRIGHT (Su et al., 2025) as the primary benchmark to assess whether retriever learning through language modeling supports reasoning-intensive retrieval. Its 12 tasks span StackExchange, coding, and theorem-based retrieval. We additionally evaluate on the 13-dataset BEIR subset (Thakur et al., 2021) used by Revela, hereafter BEIR, to examine whether the benefits extend to broader text retrieval. For both benchmarks, we report NDCG@10 as a percentage and its unweighted mean across tasks, with task-level results providing a view of variation across domains. The selected BEIR datasets and detailed evaluation protocol are provided in Appendix D.

For additional analysis, we use BRIGHT-Pro (Zhao et al., 2026), whose judgments decompose each query's information need into aspects and record which aspects each document supports. This lets us ask whether later stages improve retrieval by covering previously missing evidence, by ranking already-covered evidence earlier, or both. We report importance-weighted α-nDCG@25 for the three evaluated encoders and analyze top-25 aspect coverage for the primary Llama-3.1-8B pipeline. Appendix G describes the corpus and judgments, metric definitions, and aggregation.

Baselines. Our main BRIGHT comparisons are with ReasonIR-8B (Shao et al., 2025), RaDeR-7B (Das et al., 2025), and DIVER-4B's dense retriever (Sun et al., 2025), which use contrastive training for reasoning-intensive retrieval, and with Revela (Cai et al., 2026) on two of our backbones, Llama-3.2-1B and Llama-3.2-3B. We also include BM25 (Robertson & Zaragoza, 2009), general-purpose open-source retrievers, and proprietary embedding models to provide broader context. Baseline details and score sources, including our evaluation of RaDeR-7B and DIVER-4B, appear in Appendix D.1.

Model configurations. We study five retriever backbones from two model families: Llama-3.2-1B and Llama-3.2-3B (Meta, 2024), Llama-3.1-8B (Grattafiori et al., 2024), and Qwen3-0.6B-Base and Qwen3-4B-Base (Yang et al., 2025). Each encoder is trained with LoRA (Hu et al., 2022) (Appendix C) and paired with Llama-3.2-1B as the reference LM, which trains only its token embeddings. These configurations examine whether the training progression is effective across encoder sizes and model families while keeping the reference LM backbone fixed. As for every baseline, all models start from pretrained checkpoints; ours are base language models, whereas RaDeR and DIVER start from embedders already trained contrastively.

Training data. Pre-training uses a 1,000-update mixture of general, mathematical, and code text drawn from Wikipedia, Proof-Pile-2 arXiv (Azerbayev et al., 2024), NuminaMath (Li et al., 2024), CodeSearchNet (Husain et al., 2019), synthetic text-to-SQL data, and Revela's code corpus (Cai et al., 2026). For SFT, we augment query–document pairs from ReasonIR-VL (Shao et al., 2025) and RaDeR's MATH training set (Das et al., 2025; Hendrycks et al., 2021) with document-conditioned answers. Preference tuning uses chosen and rejected answers derived from RaDeR's MATH queries, grounded in positive and mined negative documents, respectively. Both the SFT answers and the preference answers are generated with Qwen3-235B-A22B-Instruct (Yang et al., 2025). Dataset sizes and construction details appear in Appendix B.

Table 1: BRIGHT results with original queries, reported as NDCG@10 (%). Avg. averages 12 tasks; Top-1/2 counts task-wise first/second places. Bold/underline indicate best/second-best values, including ties. Baseline details, score sources, and further baselines appear in Appendix D.1; *evaluated with our harness, all other baseline scores are published.

Model	StackExchange							Coding		Theorem-based			Avg.	Top-1/2
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.		
Sparse model														
BM25	19.2	27.1	14.9	12.5	13.5	16.5	15.2	24.4	7.9	6.0	13.0	6.9	14.8	0 / 0
Open-source models ($<$ 3B)														
SBERT	15.1	20.4	16.6	22.7	8.2	11.0	15.3	26.4	7.0	5.3	20.0	10.8	14.9	0 / 0
BGE	11.7	24.6	16.6	17.5	11.7	10.8	13.3	26.7	5.7	6.0	13.0	6.9	13.7	0 / 0
Inst-XL	21.6	34.3	22.4	27.4	18.2	21.2	19.1	27.5	5.0	8.5	15.6	5.9	18.9	0 / 0
Revela-1B	16.7	29.4	13.2	15.0	10.5	15.5	15.4	26.1	3.0	8.8	24.1	8.9	15.6	0 / 0
Revela+-0.6B	39.4	50.3	23.8	25.4	17.1	23.2	23.7	32.3	6.6	13.2	22.9	11.7	24.1	0 / 0
Revela+-1B	45.6	48.9	22.9	27.9	17.4	25.2	23.5	26.8	7.6	10.3	16.6	14.1	23.9	0 / 0
Open-source models ($\geq$ 3B and $<$ 7B)														
DIVER-4B*	38.8	42.5	**29.5**	36.0	**26.9**	27.8	28.9	19.6	9.3	6.8	37.5	**41.7**	28.8	3 / 2
Revela-3B	24.9	40.3	17.2	21.4	15.3	16.9	19.2	26.6	6.5	11.6	27.1	14.2	20.1	0 / 0
Revela+-3B	46.6	53.3	28.1	34.5	19.9	30.1	27.3	34.1	3.4	8.9	30.7	21.1	28.2	0 / **3**
Revela+-4B	42.6	50.5	24.1	33.8	21.5	28.5	**29.6**	**37.3**	2.7	**18.6**	**38.4**	27.6	29.6	**4** / 0
Open-source models ($\geq$ 7B)														
E5	18.6	26.0	15.5	15.8	16.3	11.2	18.1	28.7	4.9	7.1	26.1	26.8	17.9	0 / 0
SFR	19.1	26.7	17.8	19.0	16.3	14.4	19.2	27.4	2.0	7.4	24.3	26.0	18.3	0 / 0
GritLM-7B	25.0	32.8	19.0	19.9	17.3	11.6	18.0	29.8	**22.0**	8.8	25.1	21.1	20.9	1 / 0
gte-Qwen1.5-7B	30.6	36.4	17.8	24.6	13.2	22.2	14.8	25.5	9.9	14.4	27.8	32.9	22.5	0 / 0
RaDeR-7B*	29.8	39.1	21.1	30.6	18.1	26.3	20.5	37.1	4.3	15.0	35.2	35.1	26.0	0 / **3**
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	0 / 1
Revela+-8B	**51.9**	**56.4**	28.8	**41.1**	22.9	**33.9**	29.1	32.1	10.0	13.4	37.2	28.2	**32.1**	**4 / 3**
Proprietary models														
Cohere	18.7	28.4	20.4	21.6	16.3	18.3	17.6	26.8	1.9	6.3	15.7	7.2	16.6	0 / 0
OpenAI	23.7	26.3	20.0	27.5	12.9	12.5	20.3	23.6	2.5	8.5	23.8	12.3	17.8	0 / 0
Voyage	23.6	25.1	19.8	24.8	11.2	15.0	15.6	30.6	1.5	7.4	26.1	11.1	17.7	0 / 0
Google	23.0	34.4	19.5	27.9	16.0	17.9	17.3	29.6	3.6	9.3	21.5	14.3	19.5	0 / 0

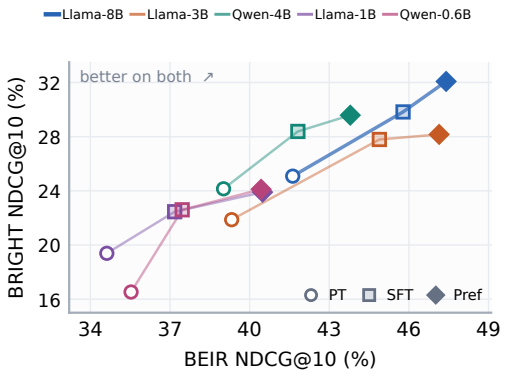
Controlled comparisons. Five controls on the Llama-3.1-8B pipeline each change one decision while keeping the rest of the stage recipe fixed. Controls **(A)**, **(B)**, and **(E)** test how each stage builds on the one before it; **(C)** and **(D)** test the SFT learning signal. **(A)** starts Answer-NTP SFT from the base encoder and reference LM instead of the pre-trained pair. **(B)** replaces Answer NTP with Query NTP and continues preference tuning from the resulting models with the same preference data and hyperparameters. **(C)** shuffles answers across queries while keeping every document and query in place. **(D)** replaces reference-LM NTP with encoder-only in-batch InfoNCE on the same document–query batches and update budget. **(E)** removes the comparative term ($\beta = 0$), continuing only chosen-answer prediction on the same preference examples at the same attention temperature. Recipes appear in Appendix F. We also compare which reference-LM parameters are trained (Section 4.2).

Training and evaluation protocol. We train each configuration through pre-training, SFT, and preference tuning, carrying the learned models forward as described in Section 3. We evaluate retrieval after each stage and compare the final encoders with the external baselines. Each stage runs for one training round: pre-training uses a fixed budget of 1,000 optimizer updates, while SFT and preference tuning each make one epoch over their data (909 and 670 updates, respectively). We report each stage's terminal checkpoint. Stage settings are fixed on the 8B pipeline and reused for the other four encoders without per-encoder tuning. Preference tuning raises the attention temperature from 10^{-4} to 0.02: after SFT, the retriever weights are already concentrated, so the higher temperature avoids a near-saturated softmax and preserves a usable gradient through those weights. Appendix C provides complete optimization settings.

4.2 Experimental Results

Does every stage improve retrieval? Yes. All five encoder configurations improve from pre-training to SFT and again from SFT to preference tuning on both BRIGHT and BEIR (Figure 2(a); per-task results in Tables 5 and 6). For Llama-3.1-8B, BRIGHT increases from 25.1% to 29.8% to 32.1%, while BEIR increases from 41.6% to 45.8% to 47.4%. SFT gives the larger BRIGHT gain; preference tuning adds 0.4 to 2.3 points on BRIGHT and 1.6 to 3.3 points on BEIR.

(a) Cross-benchmark trajectories



(b) Llama-3.1-8B recipes

Run	Starts from	BRIGHT	BEIR
Pre-training			
Base models, no Revela+ training	–	0.2	3.6
Revela+ PT	base models	25.1	41.6
SFT			
Revela+ SFT (Answer NTP)	Revela+ PT	29.8	45.8
(A) Answer NTP	base models	20.1	41.2
(B) Query NTP	Revela+ PT	28.2	46.9
(C) Answer NTP, shuffled answers	Revela+ PT	1.3	2.2
(D) Encoder-only InfoNCE	Revela+ PT	25.7	46.0
Preference tuning			
Revela+ preference tuning	Revela+ SFT	32.1	47.4
(B) Preference tuning	(B) SFT	27.3	44.8
(E) No comparative term ($\beta = 0$)	Revela+ SFT	30.2	45.8

Figure 2: Mean NDCG@10 (%). **(a)** Stage trajectories on BRIGHT and BEIR. **(b)** Llama-3.1-8B recipes. Each lettered row is a control from the text: the Revela+ recipe of its stage with one change; control (B) spans both stages. Query NTP predicts query tokens; Answer NTP predicts paired-answer tokens. Details appear in Appendix F.

How do the final retrievers compare with existing systems? Revela+ with Llama-3.1-8B obtains the highest mean BRIGHT NDCG@10 among the retrievers in Table 1, reaching 32.1% compared with 26.0% for RaDeR-7B (Das et al., 2025) and 24.4% for ReasonIR-8B (Shao et al., 2025), which shares its Llama-3.1-8B backbone. Revela+-8B uses an estimated 6.2×10^{18} training FLOPs, 6.6–7.7$\times$ fewer than ReasonIR-8B, because its retriever trains on 12–14% of ReasonIR-8B's tokens and 9% of its query–document examples, and it adds no inference cost, since the reference LM is discarded after training (Appendix H). At smaller scales, Revela+-4B reaches 29.6%, above DIVER-4B's 28.8% in our evaluation, and Revela+-3B reaches 28.2%. RaDeR-7B and DIVER-4B, moreover, start from contrastively trained embedders, gte-Qwen2-7B-instruct and Qwen3-Embedding-4B, the latter trained on about 169M pairs (Li et al., 2023; Zhang et al., 2025). Against Revela on the same backbones, Revela+-1B and Revela+-3B score 8.3 and 8.1 points higher (23.9% versus 15.6% and 28.2% versus 20.1%), of which SFT and preference tuning supply 4.5 and 6.3.

Does each stage build on the previous one? Yes. Every alternative scores below the Revela+ recipe on BRIGHT (Figure 2(b)). SFT gains from the pre-trained pair **(A)**, and preference tuning improves retrieval after Answer NTP but not after Query NTP **(B)**, mainly through its comparative term **(E)**. Within SFT, the answers must match their queries **(C)**, and learning through the language model outperforms encoder-only InfoNCE on the same batches **(D)**. Section 5 discusses them.

Does every stage still improve retrieval when the reference LM's layers adapt? We also train the reference LM's layers with LoRA, with and without its token embeddings, through the same three stages (Table 7). Pre-training and SFT carry over to these configurations: SFT improves BRIGHT by 6.5 and 5.5 points, at least as much as when only the embeddings are trained (4.7). Preference tuning does not: it changes BRIGHT by -0.2 and $+0.9$ points, against $+2.3$ with embeddings only. A reference LM whose layers adapt may separate chosen from rejected answers by itself, without relying on the documents the retriever weights; we leave testing this to future work.

5 Discussion and Analysis

We read the results by stage; each control fixes a stage's data and changes one decision (Table 5).

What pre-training contributes. Pre-training's main contribution is the starting point it gives SFT. Started from the base models instead of the pre-trained pair **(A)**, the same SFT recipe reaches 20.1% on BRIGHT, below pre-training alone (25.1%), and scores lower on all 12 tasks than SFT from the pre-trained pair (29.8%). On BRIGHT-Pro, the pre-trained 8B retriever already finds support for at least one annotated aspect on 93.8% of queries but covers every critical aspect on only 59.1%, leaving the later stages to fill in missing evidence.

What SFT contributes. SFT supplies the larger of the two stage gains on BRIGHT for every encoder; on BRIGHT-Pro it improves mean α-nDCG@25 for all three evaluated encoders (Table 9) and raises the 8B retriever's all-critical coverage from 59.1% to 70.0%. The retriever learns from how well the target fits its example: pairing each query with another query's answer collapses BRIGHT to 1.3% **(C)**. Most of the gain needs no synthesized answers, since Query NTP reaches 28.2% **(B)**; answers add the rest, most on AoPS (11.5% to 20.1%). Learning through the language model also builds on what pre-training learned. Encoder-only InfoNCE on the same batches reaches 25.7% **(D)**. It scores higher on LeetCode and both TheoremQA tasks, but it falls below the pre-trained starting point on Biology (51.4% to 33.2%) and Earth Science (48.6% to 30.7%), where Answer NTP rises above it (53.2% and 56.0%). SFT also has a cost from its data rather than its target: Pony falls after SFT for every encoder and under Query NTP (for 8B, 22.7% to 9.3% and 11.0%).

What preference tuning contributes. Preference tuning adds a smaller gain for every encoder, and its comparative term supplies most of it: continued chosen-answer prediction on the same examples reaches 30.2%, against 29.8% before the stage and 32.1% with the comparative term **(E)**. The two terms improve different tasks. Chosen-answer prediction mainly improves theorem retrieval (TheoremQA-Theorems 20.3% to 27.2%), while the comparative term helps the StackExchange domains most (mean 34.7% to 37.7%; Psychology 33.0% to 41.1%), although every preference pair is mathematical. Comparing an answer grounded in the positive document with one grounded in the mined negative teaches the retriever to prefer documents that support an answer over documents that are merely related. This costs AoPS, whose relevant documents are similar problems (18.2% to 13.4%); AoPS falls during preference tuning for every encoder. On BRIGHT-Pro, preference tuning adds 2.0 points of α-nDCG@25 for the 8B retriever, raising its all-critical coverage from 70.0% to 72.9% and ranking evidence for already-covered aspects earlier (Appendix G); for 3B and 4B it stays within 0.3 points, as do their BRIGHT scores on BRIGHT-Pro's seven domains.

Preference tuning also depends on what comes before it. After Answer NTP it gains most on Psychology, TheoremQA-Questions, and TheoremQA-Theorems (32.0% to 41.1%, 29.9% to 37.2%, and 20.3% to 28.2%); after Query NTP, the same three tasks fall (34.2% to 29.3%, 27.6% to 20.5%, and 20.4% to 16.3%), and BRIGHT drops from 28.2% to 27.3% **(B)**. It also depends on the reference LM's layers staying frozen: training them removes most of the stage's BRIGHT gain (Section 4.2).

Scope and limitations. The SFT and preference stages reuse existing query–document pairs from ReasonIR and RaDeR, adding synthesized answers and simple batching; we leave richer data and batch construction to future work. We evaluate the dense retriever, not the reference LM, whose layers stay fixed by design, and test transfer beyond BRIGHT only on BEIR. Each configuration is trained once; the stage-wise gains recur across all five encoder configurations.

6 Conclusion

Revela+ trains a dense retriever through the full language-model training pipeline, learning at every stage only from the language model's objective. Retrieval improves at both stage transitions for all five encoders on BRIGHT and BEIR, each stage builds on the one before it, and Revela+-8B reaches 32.1% on BRIGHT, above contrastively trained ReasonIR-8B and RaDeR-7B. When a retriever learns through a language model's objective, each stage of the language-model pipeline becomes a stage of retrieval supervision.

AI USE STATEMENT

In this work, we used generative AI to generate synthetic data: the SFT and preference answers are generated by Qwen3-235B-A22B-Instruct (Section 4.1 and Appendix B). Coding agents were used to assist with implementation and analysis. Additionally, generative AI tools were used to assist with computational infrastructure management, error correction, scripting, and improving the schematic figure in the paper (Figure 1), and to provide light writing assistance for this manuscript. The authors reviewed and verified all AI-assisted outputs and take full responsibility for the content of the paper.

REFERENCES

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4WnqRR915j.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, et al. SmolLM2: When smol goes big – data-centric training of a fully open small language model. In *Proceedings of the 2nd Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=3JiCl2A14H.

Fengyu Cai, Tong Chen, Xinran Zhao, Sihao Chen, Hongming Zhang, Sherry Tongshuang Wu, Iryna Gurevych, and Heinz Koeppl. Revela: Dense retriever learning via language modeling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=e7pAjJZJWb.

Debrup Das, Sam O'Nuallain, and Razieh Rahimi. RaDeR: Reasoning-aware dense retrieval models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 19970–19997. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1011. URL https://aclanthology.org/2025.emnlp-main.1011/.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. URL https://arxiv.org/abs/2103.03874.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. CodeSearchNet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*, 2019. URL https://arxiv.org/abs/1909.09436.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=jKN1pXi7b0.

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL https://arxiv.org/abs/2001.08361.

Minsang Kim and Seung Jun Baek. Syntriever: How to train your retriever with synthetic data from LLMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2523–2539. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.136. URL https://aclanthology.org/2025.findings-naacl.136/.

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*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/AI-MO/NuminaMath-CoT.

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. URL https://arxiv.org/abs/2308.03281.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. RA-DIT: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.01352.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning LLaMA for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2421–2425, 2024. URL https://arxiv.org/abs/2310.08319.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. SFR-Embedding-Mistral: Enhance text retrieval with transfer learning. Salesforce AI Research Blog, 2024. URL https://www.salesforce.com/blog/sfr-embedding/.

Meta. Llama 3.2 model card. Model card, 2024. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md.

Lorenzo Molfetta, Giacomo Frisoni, Nicolò Monaldini, and Gianluca Moro. PORTS: Preference-optimized retrievers for tool selection with large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10007–10030. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.507. URL https://aclanthology.org/2025.emnlp-main.507/.

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*, pp. 45544–45613, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/70cf215430492f7d34830a24e744b3f1-Abstract-Conference.html.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike,

and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2203.02155.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. doi: 10.52202/075280-2338. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Devendra Singh Sachan, Mike Lewis, Dani Yogatama, Luke Zettlemoyer, Joelle Pineau, and Manzil Zaheer. Questions are all you need to train a dense passage retriever. *Transactions of the Association for Computational Linguistics*, 11:600–616, 2023. doi: 10.1162/tacl_a_00564. URL https://aclanthology.org/2023.tacl-1.35/.

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. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=kkBCNLMbGj.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.463. URL https://aclanthology.org/2024.naacl-long.463/.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. Repetition improves language model embeddings. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2402.15449.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023. URL https://arxiv.org/abs/2212.09741.

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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ykuc5q381b.

Duolin Sun, Meixiu Long, Dan Yang, Junjie Wang, Yecheng Luo, Yue Shen, Jian Wang, Hualei Zhou, Chunxiao Guo, Peng Wei, Jiahai Wang, and Jinjie Gu. DIVER: A multi-stage approach for reasoning-intensive information retrieval. *arXiv preprint arXiv:2508.07995*, 2025. URL https://arxiv.org/abs/2508.07995v5. Version 5, revised April 2, 2026.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. URL https://arxiv.org/abs/1807.03748.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11897–11916. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.642. URL https://aclanthology.org/2024.acl-long.642/.

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 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. URL https://arxiv.org/abs/2309.07597.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 55204–55224. PMLR, 2024. URL https://proceedings.mlr.press/v235/xu24t.html.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

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. URL https://arxiv.org/abs/2506.05176.

Yilun Zhao, Jinbiao Wei, Tingyu Song, Siyue Zhang, Chen Zhao, and Arman Cohan. Rethinking reasoning-intensive retrieval: Evaluating and advancing retrievers in agentic search systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36776–36806, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1705. URL https://aclanthology.org/2026.acl-long.1705/.

A Detailed Retriever–LM Coupling

Let f_ϕ be the retriever and p_θ the reference LM. For a batch row i, let u_i be the text encoded by the retriever when that row receives information; for a possible source row j, let v_j be its text encoded when it supplies information. These retrieval views depend on the training stage, but both use the same retriever. As at inference, $f_\phi(z)$ denotes the L2-normalized final-layer EOS embedding of text z (Cai et al., 2026; Ma et al., 2024; Wang et al., 2024). The score for source j relative to row i is $s_{ij} = f_\phi(u_i)^\top f_\phi(v_j)$. We normalize scores over the set $\mathcal{E}_i$ of eligible source-row indices:

$$W_{ij} = \frac{\exp(s_{ij}/\tau)}{\sum_{k\in\mathcal{E}_i}\exp(s_{ik}/\tau)} \quad \text{for } j \in \mathcal{E}_i, \qquad W_{ij} = 0 \text{ otherwise.}$$

The temperature $\tau > 0$ controls how concentrated the weights are. Every stage excludes the target row itself from $\mathcal{E}_i$.

Let x_i be the full text of row i processed by the reference LM. Following Revela (Cai et al., 2026), the LM first processes each x_j independently and caches its keys and values at every layer. In a second pass over x_i, it retains causal self-attention and separately attends to the cached representation of each eligible source row. At each layer, the resulting source outputs are weighted by W_{ij}, summed, and added to the causal self-attention output before the output projection. Before this weighting, each source output is normalized by its attention-weighted value norm: if target token t attends to token k of source row j with weight $\alpha_{t,k}$ and that token has value $v_{j,k}$, the source output is $\sum_k \alpha_{t,k} v_{j,k} \big/ \sum_k \alpha_{t,k}\|v_{j,k}\|$, computed per attention head. The same row weights apply across layers, while attention over tokens within a source varies by target token and layer. The indexed collection $\mathcal{C}_i = (x_j)_{j\in\mathcal{E}_i}$ denotes the eligible full source texts used in Section 3.2.

The loss mask determines which target tokens contribute to the stage objective; it does not determine source eligibility. A row with no scored tokens may still supply information to another row. Excluding the target row from cross-sequence context leaves its causal self-attention intact. Gradients from the second-pass LM objective pass through W to update ϕ and also update the trainable LM parameters θ. Figure 3 illustrates each stage's batch, eligibility mask, and scored tokens with constructed examples.

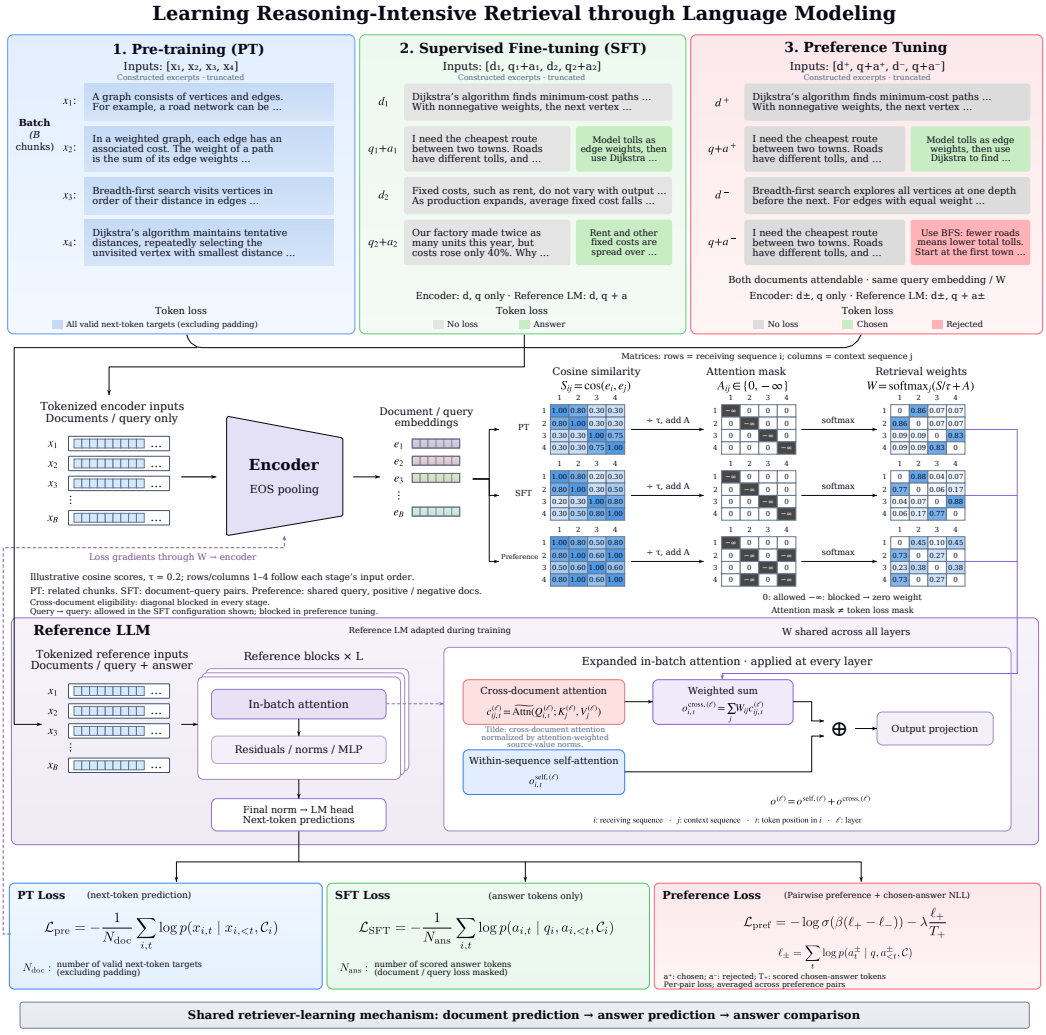


Figure 3: **Retriever learning across three training stages.** Top: constructed, truncated examples show the reference-LM inputs and scored tokens for pre-training, SFT, and preference tuning; the encoder sees documents and queries without answer continuations. Middle: encoder similarities and stage-specific eligibility masks produce retrieval weights W. The reference LM uses these weights at every layer to combine cross-document context with causal self-attention. Bottom: the objectives in Equations 1–3 train the encoder through W and update the trainable reference-LM parameters. The four-row batches, similarity scores, and weights are illustrative ($\tau = 0.2$ for visualization). Token-loss colors indicate scored targets and are distinct from cross-document eligibility masks.

B Training Data Construction

B.1 Pre-training Corpus

Pre-training uses a fixed budget of 1,000 optimizer updates. At 32 local attention batches per update, this round consumes 32,000 batches; each batch contains eight document chunks, for 256,000 chunk presentations in total. The batches are sampled from a larger shuffled pool containing Wikipedia, ProofPile-2 arXiv (Azerbayev et al., 2024), NuminaMath (Li et al., 2024), CodeSearchNet (Husain et al., 2019), synthetic text-to-SQL data, and Revela-code (Cai et al., 2026). The builder normalizes the sources to eight-chunk batches, shuffles them with seed 42, and keeps related chunks contiguous.

B.2 SFT Data

SFT uses 29,119 local attention batches, each containing four document–query–answer examples (116,476 examples total). Of these batches, 18,317 come from ReasonIR-VL (Shao et al., 2025) and 10,802 from RaDeR MATH (Das et al., 2025).

The ReasonIR-VL ingredient comes from the VL split of `reasonir/reasonir-data`, retaining document–query pairs with a positive passage of at least 50 words. The RaDeR ingredient pairs a MATH (Hendrycks et al., 2021) problem with its first ProofWiki positive passage. The answer-generation instruction is, in brief: `Use the document as central evidence; explain how it supports the query; do not force unsupported links; give a standalone answer without referring to ``the document.''` We generate the answers with `Qwen/Qwen3-235B-A22B-Instruct-2507-FP8` (Yang et al., 2025) at temperature 0.6 and top-p 0.95, with a 1,024-token limit. We shuffle the source-homogeneous batches together with seed 42. In the RaDeR ingredient, 323 of 10,802 batches repeat a document (1.11% of all SFT batches), so the within-batch alternatives are not universally distinct.

B.3 Preference Data

Preference tuning uses 21,468 local attention batches, each containing two chosen–rejected answer pairs (42,936 pairs total) derived from the RaDeR source (Das et al., 2025).

For each RaDeR MATH source example, the first positive passage grounds a synthesized chosen answer and the first mined negative passage grounds a synthesized rejected answer. At training time, each pair forms the four rows $[d^+, q{+}a^+, d^-, q{+}a^-]$; both answer rows are scored against the same eligible document pool (Section 3.2). The source-provided passage labels define answer-generation conditions but never enter the loss as document-ranking labels. Both answers use the SFT instruction, generator, and sampling settings (Appendix B.2), each with its own grounding document, served with vLLM 0.16.0 with thinking disabled. Of the 43,210 RaDeR source examples with a usable positive passage, 42,944 retain a mined negative passage of at least 20 words; after answer-length filtering and batching, 42,936 pairs remain. Both answers are then trimmed at sentence boundaries to fit the 512-token reference-LM cap, with the query prefix capped at 300 tokens; trimming shortens 39.1% of chosen and 35.7% of rejected answers. The pairs cover 27,785 distinct queries because some queries recur across RaDeR source examples. Answers are not filtered for correctness.

Under the Llama-3.2-1B reference-LM tokenizer, chosen and rejected answers average 370.7 and 368.5 tokens (medians 389 and 386), and the chosen answer is longer in 49.8% of pairs, so neither side of the preference is systematically longer. Because the preference margin compares summed log-probabilities, an individual pair's margin can still depend on its length difference.

C Training Configuration

Table 2 lists the stage settings, and Table 3 lists the per-encoder layouts. All five pipelines share one update geometry: each GPU processes one eight-row local attention batch per step, and gradient accumulation makes every optimizer update cover 32 local batches. The encoder uses rank-256 LoRA (Hu et al., 2022) with $\alpha = 64$ and dropout 0.1 on the query, key, value, output, up, down, and gate projections. The reference LM updates its token embeddings while its remaining parameters

stay frozen. Between stages, we merge the encoder adapter into its base weights before initializing a fresh adapter, and retain the learned reference-LM embeddings. Training uses bfloat16, DeepSpeed ZeRO-3, AdamW without weight decay, a linear learning-rate schedule after a 10% warmup, and gradient clipping at 1.0. Each stage ends at its terminal checkpoint. Preference tuning continues from already-trained models with a tenfold lower learning rate and a higher attention temperature than the earlier stages. At $\tau = 10^{-4}$, the SFT retriever places more than 0.99 of its weight on a single in-batch document for 91% of preference queries, so the comparative term sends almost no gradient to the retriever; at $\tau = 0.02$ the mined negative keeps a median weight of 0.21. The earlier stages keep $\tau = 10^{-4}$, where sharper weighting strengthens the signal from small score differences. In an otherwise identical 8B preference run, keeping $\tau = 10^{-4}$ reaches 30.3% on BRIGHT, matching continued chosen-answer prediction alone (30.2%, control **(E)**), compared with 32.1% at 0.02 (Table 5).

Table 2: Stage settings. Each stage runs for one training round. Updates are optimizer updates to the terminal checkpoint; each update covers 32 local attention batches. Sequence caps are in tokens.

Setting	PT	SFT	Pref
Optimizer updates	1,000	909	670
Local batches per update	32	32	32
Rows per local batch	8	8	8
Learning rate	10^{-4}	10^{-4}	10^{-5}
Attention temperature	10^{-4}	10^{-4}	0.02
Encoder length cap	157	512	512
Reference-LM length cap	157	512	512
Reference-LM query-prefix cap	–	300	300
Preference coefficients	–	–	$\beta = 0.1, \lambda = 1$

Table 3: Per-encoder training layouts. Every configuration reaches 32 local batches per update. GPU types come from the training logs; the 0.6B and 1B SFT and preference runs used a separate cluster with the shared pipeline launcher's layout, and their GPU types were not logged.

Encoder	GPUs	Accumulation	Batches/update	GPU type (PT / SFT / Pref)
Qwen3-0.6B-Base	4	8	32	H100 / – / –
Llama-3.2-1B	4	8	32	H100 / – / –
Llama-3.2-3B	4	8	32	H200 / H200 / H200
Qwen3-4B-Base	4	8	32	H200 / H200 / H200
Llama-3.1-8B	8	4	32	H100 / H200 / H200

D Evaluation Protocol and Baselines

The selected BEIR (Thakur et al., 2021) subset comprises ArguAna, Climate-FEVER, DBPedia-Entity, FEVER, FiQA, HotpotQA, NFCorpus, Natural Questions, Quora, SciDocs, SciFact, TREC-COVID, and Touche-2020. This is the subset Revela evaluates on (Cai et al., 2026): the publicly available BEIR test collections other than MS MARCO and CQADupStack. We report the unweighted mean of dataset NDCG@10 values. The evaluation harness truncates each input to 2,048 tokens, including the EOS token it appends, pools the final token, normalizes embeddings, and ranks by cosine similarity. It retains the top 1,000 candidate documents; BRIGHT excludes query-specific disallowed documents before metric computation.

D.1 Baseline References and Score Sources

The published BRIGHT scores for BM25, SBERT, BGE, Inst-L/XL, E5, SFR, GritLM, gte-Qwen1.5-7B, ReasonIR-8B, and the API models follow Su et al. (2025); Shao et al. (2025). Revela scores follow its published BRIGHT evaluation (Cai et al., 2026). For DIVER, we compare its dense-retriever component. Table 1 uses original queries and

excludes query expansion, sparse–dense fusion, and reranking. We evaluate RaDeR-7B and DIVER-4B (marked *) ourselves, with the harness described above: the released checkpoints `Raderspace/RaDeR_gte_Qwen2-7B_MATH_LLMq_CoT_lexical` and `AQ-MedAI/Diver-Retriever-4B-1020`, original queries, final-token pooling in bfloat16, and inputs truncated to 2,048 tokens. Each model uses its released prompts: `Query:` and `Passage:` prefixes for RaDeR, and for DIVER the Qwen3 query instruction "Instruct: Given a web search query, retrieve relevant passages that answer the query" followed by `Query:`, with no document prefix. RaDeR reports 25.5% (Das et al., 2025); DIVER's model card lists 31.9% for this checkpoint without stating the evaluation setting, and 28.9% for the earlier DIVER-Retriever-4B with original queries.

The baselines are BM25 (Robertson & Zaragoza, 2009), SBERT (Reimers & Gurevych, 2019), BGE (Xiao et al., 2024), Inst-L/XL (Su et al., 2023), E5 (Wang et al., 2024), SFR (Meng et al., 2024), GritLM (Muennighoff et al., 2025), gte-Qwen1.5 (Li et al., 2023), DIVER (Sun et al., 2025), Revela (Cai et al., 2026), RaDeR (Das et al., 2025), and ReasonIR (Shao et al., 2025). Table 4 reports two further baselines and Revela+ with three compact encoders.

Table 4: Additional BRIGHT results with original queries, NDCG@10 (%), in the column order of Table 1. Inst-L (Su et al., 2025) and Revela-0.5B (Cai et al., 2026) scores are published. The Revela+ rows use compact encoders, SmolLM2-135M and SmolLM2-360M (Ben Allal et al., 2025) and Gemma-3-270M (Gemma Team, 2025), trained with the same three-stage recipe and Llama-3.2-1B reference LM as the main encoders; scores are at the preference-tuning endpoint.

Model	**Bio.**	**Earth.**	**Econ.**	**Psy.**	**Rob.**	**Stack.**	**Sus.**	**Leet.**	**Pony**	**AoPS**	**TheoQ.**	**TheoT.**	**Avg.**
Published baselines													
Inst-L	15.2	21.2	14.7	22.3	11.4	13.3	13.5	19.5	1.3	8.1	20.9	9.1	14.2
Revela-0.5B	16.9	29.0	13.9	13.3	9.1	10.8	13.9	27.6	3.9	12.5	24.8	6.7	15.2
Revela+ with compact encoders													
Revela+-135M (SmolLM2)	27.5	33.9	16.7	17.9	10.2	15.4	18.8	25.6	1.8	9.8	11.4	4.5	16.1
Revela+-270M (Gemma 3)	22.7	39.0	17.8	18.5	20.2	14.5	18.9	29.7	10.5	9.6	10.4	3.8	18.0
Revela+-360M (SmolLM2)	29.6	36.4	20.6	18.9	12.2	16.7	20.8	22.7	7.5	11.2	14.8	6.6	18.2

E Per-Task Results

Table 5 gives per-task BRIGHT scores and Table 6 per-dataset BEIR scores at the terminal checkpoint of each stage for all five encoders; Table 5 also lists the Llama-3.1-8B controls. Table 7 compares which reference-LM parameters are trained.

Table 5: Per-task BRIGHT NDCG@10 (%) at the fixed terminal checkpoint of every reported run. PT, SFT, and Pref denote pre-training, supervised fine-tuning (Answer NTP), and preference tuning. Controls **(A)**–**(E)** follow Appendix F; **(A)**–**(D)** replace the 8B SFT stage, the second **(B)** row continues preference tuning from the Query-NTP model, and **(E)** replaces the preference stage. The last row repeats 8B preference tuning with the earlier stages' attention temperature $\tau = 10^{-4}$ instead of 0.02. Each model uses the Llama-3.2-1B reference LM. Means are unweighted across the 12 tasks and computed before rounding.

Run	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	Avg.
Qwen3-0.6B encoder													
PT	26.6	32.2	16.8	15.3	10.2	14.6	14.4	26.4	9.7	14.3	13.4	4.4	16.5
SFT	43.9	49.0	21.6	22.2	11.3	20.3	21.4	30.7	5.0	16.7	20.3	8.8	22.6
Pref	39.4	50.3	23.8	25.4	17.1	23.2	23.7	32.3	6.6	13.2	22.9	11.7	24.1
Llama-3.2-1B encoder													
PT	41.4	43.0	18.7	23.3	11.4	21.9	19.1	20.3	10.9	10.0	10.8	1.8	19.4
SFT	48.2	50.4	20.4	25.5	13.8	24.8	22.5	24.2	6.6	10.6	13.8	8.6	22.5
Pref	45.6	48.9	22.9	27.9	17.4	25.2	23.5	26.8	7.6	10.3	16.6	14.1	23.9
Llama-3.2-3B encoder													
PT	44.3	44.2	20.2	23.8	14.1	23.3	21.2	24.8	16.4	12.0	14.1	3.9	21.9
SFT	53.3	57.9	25.9	29.9	18.4	28.0	26.6	33.0	2.6	13.1	25.5	19.2	27.8
Pref	46.6	53.3	28.1	34.5	19.9	30.1	27.3	34.1	3.4	8.9	30.7	21.1	28.2
Qwen3-4B encoder													
PT	42.0	46.1	19.6	20.2	13.1	22.8	20.1	31.0	30.4	15.1	19.9	9.6	24.1
SFT	49.9	54.9	22.4	29.4	18.8	26.9	29.4	35.9	2.3	20.5	30.2	20.0	28.4
Pref	42.6	50.5	24.1	33.8	21.5	28.5	29.6	37.3	2.7	18.6	38.4	27.6	29.6
Llama-3.1-8B encoder													
PT	51.4	48.6	22.4	25.3	13.5	28.2	19.5	23.5	22.7	14.0	20.6	11.3	25.1
SFT	53.2	56.0	27.9	32.0	19.3	31.7	26.3	32.0	9.3	20.1	29.9	20.3	29.8
Pref	51.9	56.4	28.8	41.1	22.9	33.9	29.1	32.1	10.0	13.4	37.2	28.2	32.1
Llama-3.1-8B controls													
(A) SFT from base models	32.5	41.8	20.0	23.2	12.9	20.5	20.0	27.5	2.8	10.1	15.2	15.0	20.1
(B) Query NTP	51.5	51.5	24.3	34.2	20.0	32.4	22.6	31.9	11.0	11.5	27.6	20.4	28.2
(B) Query NTP $\rightarrow$ Pref	57.9	54.8	22.6	29.3	16.9	31.3	21.1	28.6	15.8	12.1	20.5	16.3	27.3
(C) Shuffled answers	2.0	3.0	0.3	0.8	0.0	0.2	0.2	0.4	8.4	0.0	0.0	0.0	1.3
(D) Encoder-only InfoNCE	33.2	30.7	27.9	31.1	18.2	26.2	24.7	35.2	3.7	11.1	36.7	29.7	25.7
(E) Chosen-answer prediction ($\beta = 0$)	52.1	53.7	28.0	33.0	18.9	30.9	26.1	32.0	10.1	18.2	32.5	27.2	30.2
Pref, $\tau = 10^{-4}$	52.4	54.5	28.9	33.1	20.6	31.1	26.5	33.2	8.6	18.9	33.8	22.5	30.3

Table 6: Per-dataset NDCG@10 (%) on the selected 13 BEIR datasets at the fixed terminal checkpoint of each stage. PT, SFT, and Pref denote pre-training, supervised fine-tuning, and preference tuning. Each model uses the Llama-3.2-1B training-only reference LM. Means are unweighted across datasets and computed before rounding.

Stage	Arg.	C-FEVER	DBP.	FEVER	FiQA	HotpotQA	NFC.	NQ	Quora	SciDocs	SciFact	TREC	Touche	Avg.
Qwen3-0.6B encoder														
PT	17.68	23.60	22.78	61.88	24.60	41.92	21.93	26.97	52.98	15.00	65.72	63.25	23.60	35.53
SFT	22.30	19.40	20.11	51.49	25.45	30.56	26.95	31.19	75.93	15.80	62.26	75.92	29.57	37.46
Pref	25.30	22.72	22.84	56.87	26.81	35.53	29.29	34.60	80.15	17.15	64.19	79.71	30.45	40.43
Llama-3.2-1B encoder														
PT	12.90	18.86	30.15	50.36	24.03	47.62	25.60	35.31	37.75	16.02	64.60	66.28	20.56	34.62
SFT	19.29	17.87	22.91	63.60	27.84	32.46	28.60	28.58	64.34	16.38	64.51	73.82	23.06	37.17
Pref	23.23	18.98	27.16	64.03	28.67	38.80	30.33	34.31	76.12	17.98	67.05	76.50	23.18	40.49
Llama-3.2-3B encoder														
PT	15.26	21.68	31.16	58.57	26.16	57.45	25.00	39.84	62.81	17.20	67.41	67.02	21.65	39.32
SFT	25.72	22.61	29.59	74.78	33.15	45.88	31.99	44.30	80.83	19.23	71.00	78.52	25.92	44.89
Pref	28.28	23.12	32.47	76.97	34.80	51.44	35.57	46.55	84.54	20.11	72.12	80.67	26.16	47.14
Qwen3-4B encoder														
PT	18.77	21.22	31.39	63.93	31.25	50.91	30.10	43.06	26.91	18.31	70.65	74.23	26.47	39.02
SFT	27.87	16.96	25.62	58.44	33.62	34.49	34.81	41.89	71.85	19.71	71.32	78.74	28.29	41.82
Pref	32.34	18.49	29.43	58.08	35.52	37.79	36.15	45.43	79.21	21.12	71.29	80.58	23.84	43.79
Llama-3.1-8B encoder														
PT	14.80	24.51	31.25	76.79	30.55	50.71	30.85	42.00	56.34	18.69	70.96	70.52	23.13	41.62
SFT	22.61	22.97	33.52	76.09	36.48	49.72	35.05	45.91	75.21	21.00	71.83	78.03	26.71	45.78
Pref	25.43	21.81	34.93	74.18	39.47	52.87	37.73	48.48	81.77	21.88	72.71	78.18	26.77	47.40

Table 7: Mean BRIGHT NDCG@10 (%) for Llama-3.1-8B pipelines that differ only in which reference-LM parameters are trained, with the same data, update counts, and stage settings. The first row is the Revela+ pipeline reported throughout the paper.

Reference LM trains	PT	SFT	Pref
Token embeddings (Revela+)	25.1	29.8	32.1
LoRA on layers	21.9	28.4	28.2
LoRA on layers + token embeddings	21.9	27.4	28.3

F Controlled Comparisons

All five controls change one decision in the Llama-3.1-8B pipeline and are evaluated at their terminal checkpoints. Scores are mean NDCG@10 (%) on BRIGHT, the primary benchmark, and on BEIR (Figure 2(b); per-task BRIGHT scores in Table 5). Answer NTP, the Revela+ SFT objective, predicts the answer paired with each document and query. The Base LM row of Figure 2(b) describes the pretrained starting point before any Revela+ training: its BRIGHT score uses bare query and document inputs, whereas every trained checkpoint and the BEIR Base LM use the standard query and passage prefixes.

(A) Pre-training initialization. The Answer-NTP SFT recipe starts from the base encoder and reference LM instead of the pre-trained pair. It reaches 20.1% on BRIGHT and 41.2% on BEIR, against 29.8% and 45.8% from the pre-trained pair. This comparison tests the preceding stage as a unit and does not separate its effects on the encoder and the reference LM.

(B) Query NTP. Query NTP removes the answer text and predicts the query tokens instead. It retains the source batches, document–query pairs, pre-trained parent, update budget, and eight-row layout, but changes both the reference input and the prediction target and uses a shorter reference cap, so it compares complete SFT recipes rather than isolating the target span. It reaches 28.2% on BRIGHT and 46.9% on BEIR, against 29.8% and 45.8% for Answer NTP. Its preference arm

continues the Query-NTP checkpoint with the canonical preference recipe (the same data, 670 updates, learning rate, attention temperature, and β), accumulating the effective batch of 32 as 4 GPUs $\times$ 8 steps rather than 8 $\times$ 4, as in the chosen-answer-only arm of control **(E)**. It reaches 27.3% on BRIGHT and 44.8% on BEIR, below its SFT parent, whereas preference tuning after Answer NTP reaches 32.1% and 47.4%.

(C) Shuffled answers. Every document, query, local batch, and row position stays fixed, but each answer is replaced with a content-distinct answer from the same source and a nearby reference-token-length neighborhood. BRIGHT falls to 1.3% and BEIR to 2.2%.

(D) Encoder-only InfoNCE. In-batch query-to-document InfoNCE on the encoder replaces reference-LM NTP, with the same pre-trained encoder, document–query batches, and update budget. Each local eight-row batch contains four document–query pairs, giving at most three non-duplicate negatives per query; the loss gathers no cross-device negatives and uses a conventional temperature of 0.05 rather than a tuned scale. It therefore tests objective substitution within the Revela+ training setup rather than the performance ceiling of contrastive training. InfoNCE reaches 25.7% on BRIGHT and 46.0% on BEIR, against 29.8% and 45.8% for Answer NTP.

(E) Comparative term. From the same Answer-NTP SFT checkpoint, with the same preference examples and document context, the canonical arm uses $\beta = 0.1$ and the control sets $\beta = 0$, continuing chosen-answer prediction alone. Chosen-answer prediction alone reaches 30.2% on BRIGHT and 45.8% on BEIR; adding the comparative term reaches 32.1% and 47.4%, 1.9 and 1.6 points higher, and 2.3 BRIGHT points above the SFT parent.

F.1 Query-Level Analysis

We analyze original-BRIGHT rankings for controls **(D)** and **(E)**. Queries are grouped by the rank of the first judged-relevant document under the common parent checkpoint: the pre-trained checkpoint for Answer NTP versus InfoNCE, and the SFT checkpoint for preference tuning versus continued chosen-answer prediction. The same queries appear in both arms of each comparison. Each query initially receives its domain's equal share divided by that domain's query count; weights are then normalized within a cohort. Cohorts reflect the parent model's retrieval state, not intrinsic difficulty or reasoning intensity. Saved rankings extend to 1,000 documents, so absent relevant evidence is censored rather than absent from the corpus.

Table 8: Selected BRIGHT control comparisons on fixed-parent query cohorts. NDCG@10 and top-ten success are multiplied by 100. Within-cohort averages use equal-dataset query weights.

Parent cohort	Metric	Answer NTP	InfoNCE
PT: first relevant at rank 1	NDCG@10	70.7	50.4
PT: first relevant at rank 1	Top-ten success	97.5	81.6
Parent cohort	**Metric**	**Pref**	**$\beta = 0$**
SFT: first relevant at rank 11–100	NDCG@10	8.7	4.2
SFT: first relevant at rank 11–100	Top-ten success	25.6	19.0

Answer NTP's net 4.1-point BRIGHT advantage over InfoNCE decomposes into +7.3 points from queries with relevant evidence already in the pre-trained top ten and -3.2 from the remaining queries, using their full-suite weights. InfoNCE has higher cohort NDCG for parent ranks 11–100 and 101–1000 and for queries with no relevant document in the saved top 1,000. Biology and earth science account for approximately 91% of the net advantage; omitting both leaves +0.4 points across the other ten tasks. This post-hoc concentration check does not replace the full benchmark.

For the preference comparison, the SFT rank-11–100 cohort contributes 62% of the 1.9-point overall BRIGHT advantage, with the within-cohort outcomes shown in Table 8. The initially rank-one cohort has a -1.6-point NDCG difference, while queries without relevant evidence in the saved SFT ranking contribute only +0.04 points to the suite-wide advantage. This diagnostic thus localizes the gain to reranking already-retrieved evidence rather than broad recovery.

G BRIGHT-PRO ASPECT COVERAGE AND RANKING ANALYSIS

Evaluation protocol. BRIGHT-Pro (Zhao et al., 2026) covers biology, earth science, economics, psychology, robotics, Stack Overflow, and sustainable living. We evaluate the terminal pre-training, SFT, and preference checkpoints used throughout the paper (1,000, 909, and 670 updates, respectively), with original queries and independent dense retrieval over each domain's BRIGHT-Pro corpus; no intermediate checkpoints are selected. The analysis follows the benchmark's static retrieval protocol, using aspect weights 1 (minor), 2 (important), and 3 (critical), normalized to sum to one within each query. These labels measure annotated importance, not reasoning difficulty. For α-nDCG, each additional supporting document for an already encountered aspect receives the factor $(1-\alpha)$ relative to the previous support, with $\alpha = 0.5$, and gains are discounted by rank and normalized by the benchmark's ideal DCG. We report α-nDCG multiplied by 100 and averaged equally across domains.

Aspect coverage requires an annotated supporting document in the top k, with primary cutoff $k = 25$. All-critical and all-aspect coverage require support for every corresponding annotated aspect. Query coverage percentages in Table 10 pool the 739 queries, unlike the equal-domain averages of α-nDCG. Aspect recovery rates pool counts of aspects missing before each transition. These analyses are exploratory readings of existing evaluations: coverage does not establish answer correctness or sufficiency, recovery rates and score decompositions describe these checkpoints rather than controlled comparisons or causal attributions, and the bootstrap intervals condition on these checkpoints, annotations, and domains without measuring training-seed variation.

Table 9: Mean BRIGHT-Pro α-nDCG@25 ($\times 100$), averaged equally across seven domains. PT, SFT, and Pref are each stage's terminal checkpoint. Δ is Pref minus SFT in percentage points, calculated before rounding. All encoders use the Llama-3.2-1B reference LM.

Encoder	PT	SFT	Pref	Δ
Llama-3.1-8B	53.01	60.20	62.22	+2.02
Llama-3.2-3B	51.19	62.18	61.91	−0.27
Qwen3-4B-Base	52.66	61.59	61.38	−0.20

Table 10: Top-25 aspect coverage for the primary Llama-3.1-8B pipeline. Counts and percentages pool all 739 queries; each criterion requires at least one annotated supporting document per corresponding aspect.

Query criterion	PT	SFT	Pref
Any aspect covered	693 (93.8%)	706 (95.5%)	708 (95.8%)
Every critical aspect covered	437 (59.1%)	517 (70.0%)	539 (72.9%)
Every annotated aspect covered	285 (38.6%)	360 (48.7%)	391 (52.9%)

Paired coverage changes. Among queries gaining at least one aspect, 193 of 210 at SFT and 102 of 110 at preference tuning already covered another aspect. Across all aspects, SFT gains 259 and loses 77; preference tuning gains 123 and loses 71. Of the 259 aspects first covered at SFT, 234 remain covered after preference tuning; 99 of the 123 preference-stage gains were uncovered at both preceding checkpoints. Critical-aspect gains are 154 and 61, while critical-aspect losses are 36 and 33, respectively. Thus the recovery rates in Table 11 should be read alongside the losses rather than as net coverage changes.

Table 11: Newly covered aspects divided by aspects missing before each transition, pooled over queries at cutoff 25. The two transitions start from different sets of missing aspects.

Importance	PT → SFT	SFT → Pref
Minor	14/58 (24.1%)	6/47 (12.8%)
Important	91/334 (27.2%)	56/281 (19.9%)
Critical	154/394 (39.1%)	61/276 (22.1%)

Decomposing ranking changes. For each query, we partition α-DCG into first-support and repeated-support contributions for each annotated aspect, using the same official ideal-DCG denominator for both checkpoints. First support is the highest-ranked annotated supporting document for an aspect; its identity may change between checkpoints. The change partitions into first support for entering aspects, lost first support, movement of first support for retained aspects, and changes in repeated support (Table 12). Repeated support is a metric category and need not be semantically redundant. Table 12 pools all annotated aspects in the equal-domain mean. A separate within-domain decomposition splits retained first-support movement by aspect importance. In robotics, economics, and psychology, the three domains with the largest preference-stage gains, earlier first support for critical aspects already covered after SFT contributes 2.594 of 4.717 points (55.0%), 2.217 of 3.432 points (64.6%), and 1.721 of 3.802 points (45.3%) of the net α-nDCG@25 increase, respectively. In the other four domains, this critical-aspect term is at most 0.678 points and is negative in sustainable living (-0.128 points).

Table 12: Exact additive decomposition of the 8B mean α-nDCG@25 change, in percentage points. First support is the highest-ranked supporting document for an aspect; repeated support is subsequent supporting documents. Displayed entries are rounded.

Contribution	PT $\rightarrow$ SFT	SFT $\rightarrow$ Pref
First support for newly covered aspects	+3.456	+1.402
Lost first support	-0.866	-0.778
First-support movement for retained aspects	+3.194	+1.048
Repeated support	+1.408	+0.349
Total	+7.192	+2.022

Completion examples and losses. Nine robotics queries gain support for every annotated aspect during preference tuning, each by filling one remaining gap, while two lose complete coverage. Table 13 lists all nine newly covered aspects. Sustainable living has 11 newly complete queries and four lost-complete queries, yielding 42 to 49 complete queries out of 106. Queries covering all critical aspects change from 75 to 74, while six critical-aspect gains and six losses leave the number of covered critical aspects at 169. Eight of the eleven newly complete queries already covered all critical aspects. The one-query drop in all-critical coverage thus reflects offsetting changes rather than a domain-wide decline.

Table 13: All nine robotics queries becoming complete at cutoff 25 during preference tuning. Descriptions paraphrase the annotated missing aspect. Ranks identify the best support at each checkpoint, which may be different documents.

Query	Newly covered aspect	SFT	Pref
24	Collision Monitor's safety polygons and robot-relative frame	27	21
26	Mecanum controller alternative and velocity interface	28	8
38	NavSat sensor declaration, link attachment, and output topic	74	17
45	PlannerSelector's required QoS settings	77	17
48	ROS2 actions: goal/result/feedback and service/topic implementation	29	13
72	Location of Crazyflie firmware parameter configuration	32	22
74	Tutorial's ROS distribution and Ubuntu prerequisite	37	16
75	SDK compatibility with the queried operating system and kernels	26	12
78	Meaning of `use_sim_time` and the `/clock` source	31	18

Cutoff sensitivity and uncertainty. Mean 8B α-nDCG improves at both transitions for cutoffs 10, 25, and 50. The paired 95% percentile intervals in Table 14 use 2,000 within-domain query-bootstrap replicates and retain equal domain weighting.

Table 14: Changes in equal-domain mean α-nDCG ($\times 100$) with paired 95% query-bootstrap intervals for the 8B pipeline.

Cutoff	PT → SFT	SFT → Pref
10	+6.67 [5.55, 7.76]	+2.15 [1.23, 3.06]
25	+7.19 [6.17, 8.20]	+2.02 [1.24, 2.81]
50	+7.04 [6.11, 7.98]	+1.90 [1.20, 2.62]

H Training Compute

We estimate each retriever's training compute in model FLOPs: matrix-multiplication FLOPs over non-padding tokens, following the accounting of Kaplan et al. (2020). A weight matrix costs two FLOPs per parameter per token in the forward pass, two in the backward pass to propagate gradients to its inputs, and two more to compute its own gradient when it is trained. Full fine-tuning therefore costs $6N$ FLOPs per token for N matrix parameters, and LoRA on a frozen base costs $4N+6N_{\text{LoRA}}$. Revela+'s reference LM runs two forward passes per step and backpropagates through both, since the second pass attends to the keys and values cached by the first. With its token embeddings tied to the output layer and trained, it costs $8N_{\text{ref}}$ per token, where N_{ref} includes the output layer. Attention adds $4d$ FLOPs per query–key pair per layer in the forward pass and twice that in the backward pass, for attention width d. Bidirectional encoders attend over all token pairs, causal models over preceding tokens, and Revela+'s second reference pass also over every token of the other rows in its batch. Attention contributes 2–6% of each total. Embedding lookups, activation recomputation (gradient checkpointing and GradCache), padding, and synthetic-data generation are excluded for all methods.

Token counts are measured on each method's training data with its own tokenizer, input format, and length caps. For Revela+, we apply each stage's training-time truncation rules to its data (Appendix C) and multiply by the local batches consumed; during pre-training, the retriever embeds each chunk in both its passage and query views. For ReasonIR-8B, we sample up to 20,000 examples from each training source, format them as in the GritLM training code (Muennighoff et al., 2025), and count one query, one positive, and one negative per example, each capped at 2,048 tokens, over $1{,}000 \times 2{,}048$ examples. Its public subset is the echo-embeddings release (Springer et al., 2025), which the ReasonIR authors identify as the source. No combination of its files matches the reported 1,383,877 examples, so we report a range over two plausible selections. For RaDeR-7B, we count all 35,898 training queries, each with one positive and eleven negatives, under the length caps of its released training command. DIVER-4B does not report its training schedule, so we count one pass over the RaDeR mathematical data it names (121,528 of the roughly 200k examples it reports), with a query and a positive each.

RaDeR and DIVER start from embedding models that were already contrastively trained. Qwen3-Embedding-4B, the initialization of DIVER-4B, was trained on about 150M synthetic pairs followed by 19M fine-tuning pairs (Zhang et al., 2025). A single pass over these pairs at the frozen-base cost of $4N$, with at least 40 tokens per pair (the shortest pair type we measured), gives a lower bound of 9.8×10^{19} FLOPs. The contrastive training of gte-Qwen2-7B-instruct (Li et al., 2023), the initialization of RaDeR-7B, is not quantified, so RaDeR-7B's total is also a lower bound.

Against the baselines, Revela+-8B's 6.2×10^{18} FLOPs are 6.6–7.7$\times$ fewer than ReasonIR-8B's, which shares its Llama-3.1-8B backbone, while it scores 7.7 points higher on BRIGHT. Revela+-3B surpasses both ReasonIR-8B and RaDeR-7B with less than a tenth of ReasonIR's training compute. RaDeR's fine-tuning is slightly cheaper than Revela+-8B's (5.5×10^{18}); counting its contrastively trained initialization, DIVER-4B's retrieval training exceeds Revela+-8B's by more than an order of magnitude. The saving comes from training on fewer tokens, 0.13B retriever tokens against ReasonIR's 0.95–1.11B, rather than from cheaper steps.

Table 15 summarizes the estimates. Revela+-8B spends 2.05, 2.39, and 1.73×10^{18} FLOPs on pre-training, SFT, and preference tuning, and its reference LM accounts for 30% of the total. Its measured training time is 32.5 GPU-hours, against 24.0 for Revela+-3B and 34.7 for Revela+-4B (GPU types in Table 3). These runs reach 3–6% of peak BF16 throughput, reflecting eight-row

Table 15: Estimated training compute in model FLOPs (Appendix H). Pairs are labeled or synthetic query–document training examples; Revela+ additionally pre-trains on 256k unlabeled chunks, and its SFT pairs are drawn from ReasonIR's and RaDeR's data. Revela+ tokens are retriever plus reference-LM tokens, with the Llama-3.2-1B reference LM for every encoder. Inherited FLOPs cover the contrastive training of the initialization. $\geq$ marks lower bounds.

Model	**Initialization**	**Tuning**	**Pairs**	**Tokens**	**Training FLOPs**	**Inherited FLOPs**
ReasonIR-8B	Llama-3.1-8B	Full	1.73M	0.95–1.11B	4.1–4.8×10^{19}	–
RaDeR-7B	gte-Qwen2-7B-instruct	LoRA (r=16)	35.9k	0.20B	5.5×10^{18}	not reported
DIVER-4B	Qwen3-Embedding-4B	not reported	$\sim$200k	$\geq$0.05B	$\geq 7.8 \times 10^{17}$	$\geq 9.8 \times 10^{19}$
Revela+-0.6B	Qwen3-0.6B-Base	LoRA (r=256)	159k	0.13B + 0.17B	2.2×10^{18}	–
Revela+-1B	Llama-3.2-1B	LoRA (r=256)	159k	0.13B + 0.17B	2.5×10^{18}	–
Revela+-3B	Llama-3.2-3B	LoRA (r=256)	159k	0.13B + 0.17B	3.7×10^{18}	–
Revela+-4B	Qwen3-4B-Base	LoRA (r=256)	159k	0.13B + 0.17B	4.3×10^{18}	–
Revela+-8B	Llama-3.1-8B	LoRA (r=256)	159k	0.13B + 0.17B	6.2×10^{18}	–

attention batches and ZeRO-3 sharding, and the baselines report no training time, so we compare FLOPs rather than GPU-hours.