LYCHEE-RERANKER: FAST BUT NOT LATE INTERACTION FOR COMPRESSED DOCUMENT RERANKING

Anonymous authors
Paper under double-blind review

ABSTRACT

As retrieval systems scale, effective and efficient reranking becomes increasingly important. However, most existing encoder- and decoder-based rerankers jointly process every query–passage pair, tightly coupling their online computation and limiting deployment efficiency and flexibility. We present `Lychee-Reranker`, a *fast but not late-interaction* (FBNL) reranker that decouples query and passage computation while retaining expressive relevance modeling. Built on an encoder–decoder architecture, `Lychee-Reranker` pre-encodes passages using Matryoshka embedding pooling, while its decoder models system and user instructions together with query intent; cross-attention then captures fine-grained relevance between the resulting query context and passage representations. Together, these designs offer four key advantages: (i) *efficiency* from offline passage encoding, (ii) *expressiveness* from cross-attention, (iii) *compactness* from Matryoshka embedding pooling, and (iv) *test-time compute* through an adjustable compute budget. We instantiate `Lychee-Reranker` in three sizes, *Nano*, *Small*, and *Large*, with 0.27B, 1B, and 4B activated parameters, respectively. Extensive experiments on BEIR, MIRACL, and LMEB demonstrate strong reranking performance with superior efficiency. On BEIR and MIRACL, our models achieve competitive performance in multi-domain and multilingual reranking, on par with strong industrial rerankers such as the Qwen3/BGE-Reranker series. On LMEB-Dialogue, a compact embedding model paired with our Nano reranker, which has only 0.27B activated parameters, remains competitive with 7–12B embedding models. We will release our code, data, and models to facilitate academic research.

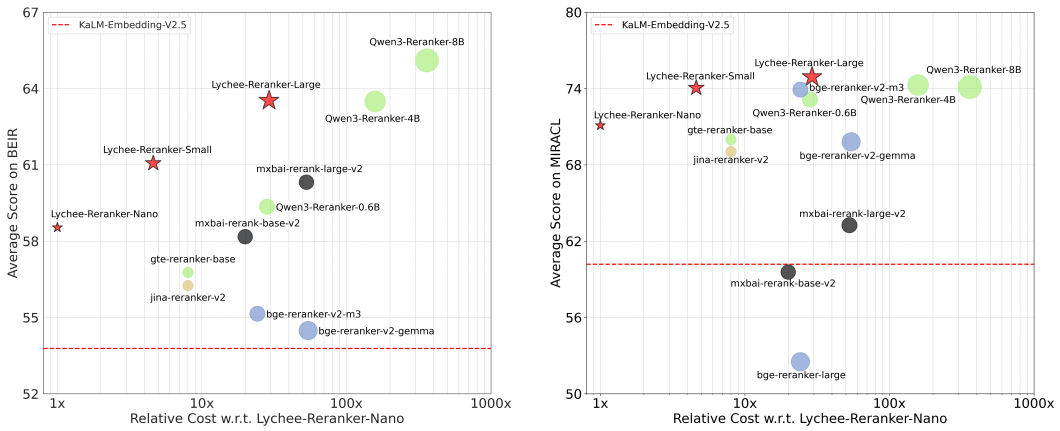
1 INTRODUCTION

At scale, neural retrieval systems typically follow a two-stage pipeline: a retriever first recalls a small candidate set from a large corpus, and a reranker then performs fine-grained relevance modeling to produce the final ranking (Nogueira & Cho, 2019; Karpukhin et al., 2020). The reranked passages can then be provided directly to large language models (LLMs) or further processed through context engineering (Lewis et al., 2020; Xu et al., 2024). Reranking quality is therefore crucial to end-to-end performance in search (Nogueira & Cho, 2019), recommendation (Liu et al., 2022; Zhao et al., 2024a), and retrieval-augmented generation (RAG) (Lewis et al., 2020; Zhao et al., 2024b; Chen et al., 2026).

Most existing rerankers estimate relevance by jointly processing query–passage pairs, thereby coupling query and passage computation (Zhang et al., 2025c; Wang et al., 2025; Chen et al., 2024). Consequently, passage representations must be recomputed for each query, preventing offline precomputation and incurring substantial online cost at scale. Late-interaction models offer a more efficient alternative by encoding queries and passages independently (Khattab & Zaharia, 2020). However, their reliance on token-level similarity functions restricts query–passage interaction to predefined scoring operations, limiting the expressiveness of relevance modeling. More broadly, both paradigms estimate relevance using a fixed computational budget and lack a mechanism for flexibly allocating computation during inference. This makes it difficult to flexibly adjust test-time compute (Snell et al., 2025), where additional inference-time computation progressively refines relevance modeling.

In this work, we introduce the `Lychee-Reranker` series, which features a *fast but not late-interaction* (FBNL) reranking design that decouples query and passage computation while preserving expressive relevance modeling. `Lychee-Reranker` is built on an encoder–decoder architecture,

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.



(a) BEIR. (b) MIRACL.

Figure 1: Comparison between the `Lychee-Reranker` series and other competitive reranking models on BEIR and MIRACL *w.r.t.* performance and cost. The cost is estimated following the analysis in Appendix A.1; the x-axis is plotted on a logarithmic scale. Marker sizes are proportional to model sizes, and points closer to the upper-left corner indicate better performance–cost trade–offs.

where the encoder is dedicated to producing passage representations that can be precomputed offline and reused across queries. To further reduce storage and improve efficiency, `Lychee-Reranker` is trained using Matryoshka embedding pooling (MEP), which produces compact passage representations while preserving their effectiveness for reranking. By varying the MEP compression ratio, `Lychee-Reranker` supports flexible test-time compute, allowing additional inference-time computation to progressively improve reranking quality. The decoder takes the system instruction, user instruction, and query as inputs. The decoder then models relevance through merged self-attention and cross-attention over the concatenated query context and pre-encoded passage representations, enabling fine-grained query-conditioned interactions without fully re-encoding each passage for every user request. Overall, this FBNL design offers four key advantages in practice:

- **Efficiency**: Passage representations can be precomputed offline, reducing online computation costs.
- **Expressiveness**: The decoder supports more expressive relevance modeling than late interaction.
- **Compactness**: MEP compresses passage representations, significantly reducing storage overhead.
- **Test-time compute**: The MEP compression ratio enables flexible allocation of inference-time computation, allowing the model to achieve better reranking quality with more computation.

We instantiate `Lychee-Reranker` in three sizes: *Nano*, *Small*, and *Large*, with 0.27B, 1B, and 4B activated parameters, respectively, to accommodate diverse performance and serving requirements. Experiments on BEIR (Thakur et al., 2021), MIRACL (Zhang et al., 2023), and LMEB (Zhao et al., 2026b) show that `Lychee-Reranker` achieves leading reranking performance with substantially higher efficiency across multi-domain, multilingual, and long-horizon memory retrieval settings. Specifically, as shown in Figure 1, `Lychee-Reranker-Nano` substantially outperforms gte-reranker-base on BEIR and slightly outperforms it on MIRACL, while being about 8x more efficient. Across MEP compression ratios from 2x to 128x, performance degrades gradually, with only a very small drop between 2x and 8x. On LMEB-Dialogue, a compact embedding model paired with the 0.27B Nano reranker remains competitive with 7–12B embedding models. Last but not least, our models support test-time compute: they cheaply screen many candidates (*e.g.,* 64x compression), then thoroughly examine the most promising ones (*e.g.,* 2x compression). Results show that this cascaded multi-stage reranking strategy substantially improves efficiency with negligible performance loss.

2 Related Work

Embedding Models. Text embedding models map text into continuous vectors for efficient similarity search (Muennighoff et al., 2023; Enevoldsen et al., 2025; Zhang et al., 2025b). From GloVe

word vectors (Pennington et al., 2014) to transformer-based sentence embedders such as Sentence-BERT (Reimers & Gurevych, 2019), dense representations have been widely adopted for semantic similarity search. DPR (Karpukhin et al., 2020) further demonstrated that dual-encoder passage retrieval can serve as an effective and scalable first-stage retriever for open-domain QA. Subsequent models such as Contriever (Izacard et al., 2022), GTR (Ni et al., 2022), and E5 (Wang et al., 2022) improved generalization through contrastive pretraining, large-scale supervision, and stronger backbones. Recent LLM-driven embedding models have become dominant, including GTE (Li et al., 2023; Zhang et al., 2024), Qwen3-Embedding (Zhang et al., 2025c), KaLM (Hu et al., 2025; Zhao et al., 2026a), BGE (Chen et al., 2024), Jina (Sturua et al., 2024; Akram et al., 2026), NV-Embed (Lee et al., 2025), and F2LLM (Zhang et al., 2026b). Despite the strong performance of LLM-based embeddings, these models typically adopt a bi-encoder architecture, where queries and documents are encoded separately and matched only through vector similarity. This design limits fine-grained query–document interaction, making rerankers necessary to further refine the retrieved candidates.

Reranking Models. Reranking addresses the limited interaction in embedding retrieval by performing deeper query–document relevance modeling. Early neural rerankers such as BERT-based rerankers concatenate the query and passage and apply full self-attention for relevance modeling (Nogueira & Cho, 2019). Generative and LLM-based rerankers, including RankGPT (Sun et al., 2023), RankVicuna (Pradeep et al., 2023), BGE (Chen et al., 2024), Qwen3-Reranker (Zhang et al., 2025c), Jina-Reranker (Wang et al., 2025), and GMR (Zhang et al., 2026a), further strengthen this paradigm through generative scoring, instruction tuning, and stronger backbones. Despite their effectiveness, these methods tightly couple query and passage computation, requiring each candidate passage to be processed jointly with each query. In parallel, late-interaction methods such as ColBERT (Khattab & Zaharia, 2020) and ColBERTv2 (Santhanam et al., 2022) decouple query and passage encoding while preserving token-level matching. They improve efficiency, but their interaction is largely limited to similarity aggregation between independently contextualized query and passage token representations. Overall, reranking methods have developed along multiple directions, including full interaction, generative scoring, and efficient late interaction. However, prevailing models often face a trade-off between efficiency and effectiveness (Zhang et al., 2025c; Wang et al., 2025). Our models introduce a new reranking paradigm, fast but not late interaction (FBNL), that offers a more favorable trade-off between efficiency and effectiveness: the encoder produces reusable passage representations, while the decoder performs fine-grained query–passage relevance modeling.

3 Model Architecture

3.1 Architecture

`Lychee-Reranker` is built on the T5Gemma2 model family (Zhang et al., 2025a) and instantiated at three encoder–decoder scales: 270M–270M, 1B–1B, and 4B–4B. We initialize the models from T5Gemma2 to inherit its pretrained language modeling capabilities while adopting its encoder–decoder architecture for efficient document reranking. Table 1 summarizes the model specifications.

Figure 2 illustrates our model architecture, which is designed to address key limitations of existing rerankers. Concretely, existing rerankers (Chen et al., 2024; Wang et al., 2025; Zhang et al., 2025c) tightly couple query and document computation, requiring joint processing of each query–document pair and resulting in high serving cost at scale. Late-interaction methods such as ColBERT (Khattab & Zaharia, 2020) improve efficiency by encoding queries and documents separately and then performing multi-vector interaction, but cannot capture fine-grained query–document interactions during scoring.

Our *fast but not late-interaction* (FBNL) paradigm differs from both conventional encoder- and decoder-based rerankers and late-interaction models. Unlike the former, it reuses document representations to improve serving efficiency; unlike the latter, it preserves rich query–document interaction through decoder cross-attention rather than restricting interaction to post-encoding scoring. These designs, together with Matryoshka embedding pooling in §4, enable efficient and expressive reranking at scale without sacrificing rich query–document interaction. Formally, given an instruction I, a query q, and a candidate document p, the encoder maps p to a reusable representation:

$$\mathbf{H}_p = \mathrm{Enc}(p) \in \mathbb{R}^{n \times d}, \tag{1}$$

where n denotes the document sequence length and d is the hidden dimension. The decoder takes the system instruction (omitted from the equation for brevity), the task-specific instruction I, and the

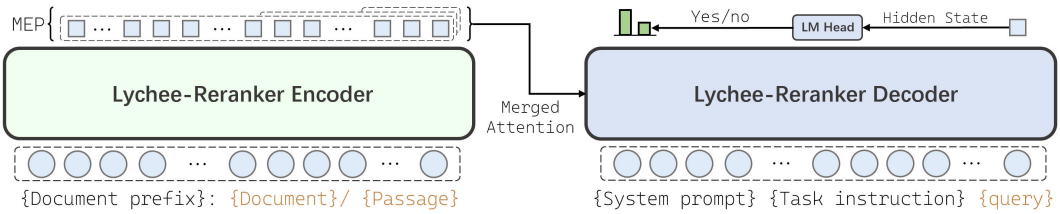


Figure 2: Overview of `Lychee-Reranker`. The encoder produces reusable passage representations compressed by MEP, while the decoder computes accurate relevance scores via merged attention.

Table 1: The `Lychee-Reranker` series. Activated parameters denote the number of parameters activated during online inference, excluding the offline encoder. "#Layers" is the number of decoder layers; values in parentheses give the total encoder–decoder layer count used as L in Eq. 12. Document token dimension is the size of each encoded document token. MEP denotes Matryoshka embedding pooling, with supported compression ratios. "Instruction Aware" indicates support for task-specific instructions. "Test-Time Compute" is support for on-demand allocation of compute during inference.

Models	Activated Params.	Non-Embedding Params.	Embedding Params.	#Layers	Sequence Length	Document Token Dim.	MEP Support	Instruction Aware	Test-Time Compute
`Lychee-Reranker-Nano`	0.27B	100M	168M	18 (36)	128K	640	1x—128x	Yes	Yes
`Lychee-Reranker-Small`	1B	698M	302M	26 (52)	128K	1152	1x—128x	Yes	Yes
`Lychee-Reranker-Large`	4B	3209M	675M	34 (68)	128K	2560	1x—128x	Yes	Yes

user query q as input. It then attends to $\mathbf{H}_p$ through merged attention to produce vocabulary logits:

$$\mathbf{Z} = \mathrm{Dec}(I, q; \mathbf{H}_p) \in \mathbb{R}^{|\mathcal{V}|}, \tag{2}$$

where $\mathcal{V}$ denotes the token vocabulary. During training and inference, we use `logits_to_keep` to retain only the logits of the final token, substantially reducing GPU memory usage. More specifically, let $\mathbf{X} \in \mathbb{R}^{m \times d}$ denote the decoder input. The decoder projects $\mathbf{X}$ into queries and projects the concatenation of the decoder input $\mathbf{X}$ and the encoder output $\mathbf{H}_p$ into keys and values, as follows:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = [\mathbf{X}; \mathbf{H}_p]\mathbf{W}_K, \quad \mathbf{V} = [\mathbf{X}; \mathbf{H}_p]\mathbf{W}_V, \tag{3}$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ denote the projection matrices for queries, keys, and values, respectively. The merged self-attention and cross-attention output is then computed as:

$$\mathbf{O} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}} + \mathbf{M}\right)\mathbf{V}\mathbf{W}_O, \tag{4}$$

where d_h is the attention head dimension, $\mathbf{M} \in \mathbb{R}^{m \times (m+n)}$ is the attention mask, and $\mathbf{W}_O$ denotes the output projection matrix. The language-model head projects the decoder hidden state at the first prediction position into a vector of vocabulary logits $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}|}$. To compute the relevance score, we compare the next-token likelihoods of `yes` and `no`. We use only the two entries in $\mathbf{Z}$ corresponding to these label tokens, denoted by z_{yes} and z_{no}, respectively. We define the relevance score as follows:

$$\mathrm{score}(q, p) = \frac{\exp(z_{\mathrm{yes}})}{\exp(z_{\mathrm{yes}}) + \exp(z_{\mathrm{no}})}. \tag{5}$$

Here, $\mathrm{score}(q, p)$ denotes the relevance score of document p with respect to query q. Appendix A presents a comprehensive time- and space-complexity analysis comparing our encoder–decoder reranker, `Lychee-Reranker`, with conventional encoder-based and decoder-based rerankers.

3.2 Prompt Templates

Table 2 summarizes the encoder and decoder prompt templates used by `Lychee-Reranker`. The encoder receives only the **document** and produces reusable passage representations that can be stored and shared across queries. Because T5Gemma2 does not support a system role (Google, 2025), we include the system prompt (*i.e.,* "`Judge whether ...`") in the **user** message, together with the **instruction** and **query**. The **model** response then predicts whether the encoded document satisfies the specified requirements. This separation follows the FBNL design: document representations remain reusable, while the decoder performs instruction- and query-aware interaction through merged self- and cross-attention during scoring. The decoder output is restricted to the binary label tokens "`yes`" and "`no`", whose logits are used to compute the relevance score. Instructions are provided in §E.

Table 2: Prompt templates for `Lychee-Reranker`. The encoder takes a document and produces reusable representations; the decoder takes an instruction and query to compute the reranking score.

Prompt Templates for `Lychee-Reranker`

Prompt Template for Encoder

```
<Document>:  {Document}
```

Prompt Template for Decoder

```
<bos><start_of_turn>user
Judge whether the Document meets the requirements based on the Query
and the Instruct provided.  Note that the answer can only be "yes" or
"no".

<Instruct>:  {Instruction}
<Query>:  {Query}<end_of_turn>
<start_of_turn>model\n\n\n\n
```

4 Model Training

Appendix B provides details on the training data, including data sources and processing procedures.

4.1 Training Objective

Supervised Fine-Tuning (SFT). Given an instruction I, a query q, and a document p, `Lychee-Reranker` predicts a binary relevance label $l \in \{\texttt{yes}, \texttt{no}\}$. Following the scoring method in §3, we define the SFT objective as the negative log-likelihood of the ground-truth label:

$$\mathcal{L}_{\text{sft}}(I, q, p, l) = -\log \frac{\exp(z_l)}{\exp(z_{\text{yes}}) + \exp(z_{\text{no}})}, \tag{6}$$

where z_l is the logit of the ground-truth label token l, with $l = \texttt{yes}$ for relevant documents and $l = \texttt{no}$ for irrelevant ones. Though reusable passage representations improve serving efficiency, storing the full representation $\mathbf{H}_p \in \mathbb{R}^{n \times d}$ for every passage in a large corpus can incur substantial storage overhead. To reduce this overhead, we introduce **Matryoshka Embedding Pooling** (MEP), which compresses passage representations along the sequence dimension while preserving reranking effectiveness. For a compression ratio r, MEP partitions the passage tokens into consecutive groups of at most r tokens and applies mean pooling to each group, yielding a compact token representation:

$$\mathbf{H}_p^{(r)}[j] = \text{MeanPool}\left(\mathbf{H}_p[(j-1)r + 1 : \min(jr, n)]\right), \quad j = 1, \ldots, \left\lceil \frac{n}{r} \right\rceil, \tag{7}$$

where $\mathbf{H}_p^{(r)} \in \mathbb{R}^{\lceil n/r \rceil \times d}$ denotes the compressed passage representation and $\lceil \cdot \rceil$ is the ceiling function. Having established $\mathbf{H}_p^{(r)}$, the decoder computes the relevance score at compression ratio r as follows:

$$\text{score}^{(r)}(q, p) = \frac{\exp(z_{\text{yes}}^{(r)})}{\exp(z_{\text{yes}}^{(r)}) + \exp(z_{\text{no}}^{(r)})}. \tag{8}$$

During training, we jointly optimize the SFT loss over a set of compression ratios, *e.g.*, $\mathcal{R} = \{2, 8, 32, 128\}$, where $r = 1$ corresponds to the uncompressed representation. The final objective is:

$$\mathcal{L}_{\text{sft}}(I, q, p, l, \mathcal{R}) = \sum_{r \in \mathcal{R}} \lambda_r \left(-\log \frac{\exp(z_l^{(r)})}{\exp(z_{\text{yes}}^{(r)}) + \exp(z_{\text{no}}^{(r)})} \right), \tag{9}$$

where λ_r controls the loss weight and is set to 1 by default. This training objective encourages the model to learn robust passage representations that remain effective under different compression ratios.

Knowledge Distillation (KD). We use a stronger teacher reranker to provide soft supervision (Hinton et al., 2015; Rao et al., 2024). Concretely, for each query–document pair (q, p), the teacher produces a soft relevance label $y \in [0, 1]$, while the student predicts a score $\hat{y} = \text{score}(q, p)$. To this end, we minimize the binary cross-entropy between the teacher's soft label and the student's prediction:

$$\mathcal{L}_{\text{kd}} = \sum\nolimits_{r \in \mathcal{R}} \lambda_r (-y \log \hat{y}^{(r)} - (1 - y) \log(1 - \hat{y}^{(r)})), \tag{10}$$

where y is the teacher's soft label and $\hat{y}^{(r)}$ is the student's prediction at compression ratio r. Teacher label construction is defined in §B. Compared with the hard binary labels used in Equation 9, soft labels preserve fine-grained relevance signals and mitigate the impact of potential false hard negatives.

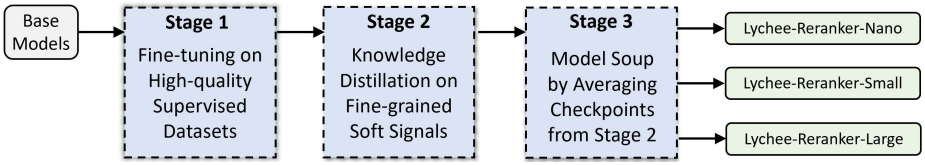


Figure 3: Multi-stage training pipeline of the `Lychee-Reranker` series.

4.2 Multi-stage Training

Multi-stage training is widely used to improve the generalization and robustness of retrieval models across diverse data distributions (Zhang et al., 2025c; Akram et al., 2026; Zhao et al., 2026a; Zhang et al., 2026b). To progressively enhance the model's reranking capabilities, we employ the three-stage training pipeline shown in Figure 3: **(1)** fine-tuning, **(2)** knowledge distillation, and **(3)** model soup.

Fine-tuning on High-Quality Supervised Datasets. First, we fine-tune the model on high-quality supervised reranking datasets. Using the SFT objective in Eq. 9, the model learns from binary hard labels across multiple compression ratios, establishing a robust foundation for document reranking.

Knowledge Distillation with Fine-Grained Soft Signals. Next, rather than continuing to rely on coarse-grained hard labels, the model distills fine-grained soft knowledge from a strong teacher reranker (*i.e.,* Qwen3-Reranker-8B (Zhang et al., 2025c)). The KD objective in Eq. 10 enables the student model to capture nuanced semantic differences between positive passages and hard negatives.

Model Soup Using Stage-2 Checkpoints. Finally, we merge the LoRA adapters into their respective backbones and average the resulting Stage 2 model checkpoints trained with different LoRA configurations. The averaged checkpoint serves as the final model. More details are provided in §D.

5 Evaluation

5.1 Experimental Settings

Experimental settings, training and implementation details, and task-specific evaluation instructions for `Lychee-Reranker` are provided in Appendices C, D, and E, respectively. Unless otherwise specified, `Lychee-Reranker` uses a default compression ratio of $r = 4$ in all experiments.

5.2 Main Results

Tables 3 and 4 compare the reranking performance and efficiency of the `Lychee-Reranker` series with competitive baselines on BEIR and MIRACL. We make the following key observations:

Lychee-Reranker achieves strong reranking performance with superior efficiency. The `Lychee-Reranker` series matches or outperforms strong industrial rerankers from the Qwen, BGE, Jina, and mxbai families. For example, on BEIR, `Lychee-Reranker-Nano`, `-Small`, and `-Large` achieve the best or second-best results on 13/13, 12/13, and 13/13 tasks, respectively, within their respective size groups. Beyond its strong reranking performance, `Lychee-Reranker` is also highly efficient. For example, `Lychee-Reranker-Small` outperforms Qwen3-Reranker-0.6B at a substantially lower online computation cost (4.6x vs. 28.2x).

Lychee-Reranker demonstrates strong multilingual reranking capabilities. Despite limited multilingual training, `Lychee-Reranker` achieves competitive performance on MIRACL. For example, `Lychee-Reranker-Large` outperforms Qwen3-Reranker-8B on MIRACL with a substantially lower online computation cost (29.1x vs. 359.8x). Within its parameter group, `Lychee-Reranker-L` ranks first in 11 of the 18 languages (*i.e.,* ar, bn, de, en, es, fi, fr, ja, ko, ru, and sw); six of these languages (*i.e.,* en, es, de, ja, fr, and ru) have reported website usage rates totaling approximately 74.2% among websites with known content languages (W3Techs, 2026).

The FBNL reranking paradigm combines efficiency with expressive relevance modeling. Built on the FBNL paradigm and trained on 3.9M instances (Table 16), the `Lychee-Reranker` series achieves performance on BEIR comparable to that of the Qwen3-Reranker series. In contrast, the

Table 3: BEIR (Thakur et al., 2021) reranking results measured by nDCG@10. The abbreviations for the 13 tasks are defined in Table 11. "N", "S", and "L" are abbreviations for Nano, Small, and Large, respectively. Avg. denotes the average performance across the 13 tasks. Cost denotes the relative online computation cost estimated using the time-complexity analysis in Appendix A.1. The estimate assumes $|q| = 32$, $n = 1024$, $K = 1$, and $r = 4$, with model-specific L and d taken from Tables 1 and 9. Costs are normalized relative to `Lychee-Reranker-Nano` (1.0x). Within each parameter group, the best results are **boldfaced**, and the second-best results are underlined.

Models	Size	Cost	Avg.	AA	CF	CQA	DB	FV	FQA	HQA	MS	NF	NQ	QR	SD	SF
First-stage Retriever																
KaLM-Embedding-V2.5	0.5B	–	53.78	59.60	32.53	45.54	42.62	83.72	46.99	70.74	40.15	36.95	57.53	88.88	19.88	73.98
Second-stage Reranker																
Models with more than 4B parameters																
Qwen3-Reranker-8B	8B	359.8x	65.11	82.92	49.63	52.49	56.07	94.96	59.72	84.24	46.84	41.71	77.13	89.88	28.97	81.85
Models with 1B–4B parameters																
Qwen3-Reranker-4B	4B	157.8x	63.50	**79.85**	**49.92**	49.94	**54.59**	**94.46**	56.69	83.68	44.47	41.57	74.03	88.40	**27.14**	80.73
`Lychee-Reranker-L`	4B	**29.1x**	**63.53**	72.90	47.94	**52.36**	52.73	94.22	**59.15**	**84.23**	47.38	**42.35**	**75.61**	**90.72**	24.89	**81.42**
bge-reranker-v2-gemma	2.5B	54.2x	54.49	69.34	29.60	37.28	43.02	82.87	47.21	82.09	45.38	28.59	68.80	86.64	18.86	68.66
mxbai-rerank-large-v2	1.5B	52.8x	60.32	71.43	46.54	45.90	49.74	93.74	50.75	82.66	**47.65**	37.59	70.77	89.56	16.99	80.86
Models with 0.5B–1B parameters																
`Lychee-Reranker-S`	1B	**4.6x**	**61.07**	69.43	45.63	47.06	**49.77**	93.36	**53.62**	**83.55**	46.90	**39.89**	**73.14**	**90.77**	**22.15**	78.72
bge-reranker-large	0.6B	24.2x	51.86	28.47	41.51	38.89	48.76	93.01	41.41	83.12	46.96	34.14	52.46	73.84	17.23	74.41
bge-reranker-v2-m3	0.6B	24.2x	55.15	41.73	37.41	39.09	47.77	90.15	45.07	82.62	**47.79**	34.54	69.25	89.14	18.04	74.35
Qwen3-Reranker-0.6B	0.6B	28.2x	59.36	**73.13**	**48.84**	**47.21**	49.43	**93.45**	47.99	81.19	42.35	39.32	64.04	84.12	**22.15**	78.46
mxbai-rerank-base-v2	0.5B	19.9x	58.18	63.41	43.38	45.79	48.97	92.71	46.43	81.45	46.79	38.30	66.83	88.14	14.93	**79.16**
Models with fewer than 0.5B parameters																
gte-reranker-base	0.3B	8.0x	56.77	58.44	**47.04**	39.21	47.31	**94.25**	45.35	81.48	45.44	36.61	65.58	81.83	17.74	77.71
jina-reranker-v2	0.3B	8.0x	56.26	51.59	34.40	**40.85**	**49.03**	92.35	46.10	79.73	**47.71**	37.12	67.16	87.80	20.01	77.50
`Lychee-Reranker-N`	0.27B	**1.0x**	**58.54**	**63.97**	43.25	40.84	47.93	92.95	**47.09**	**82.58**	45.64	**38.72**	**69.30**	**89.92**	**20.80**	**78.03**

Table 4: MIRACL (Zhang et al., 2023) reranking results measured by nDCG@10 across all 18 languages. The language abbreviations are defined in Table 12. Other settings follow those in Table 3.

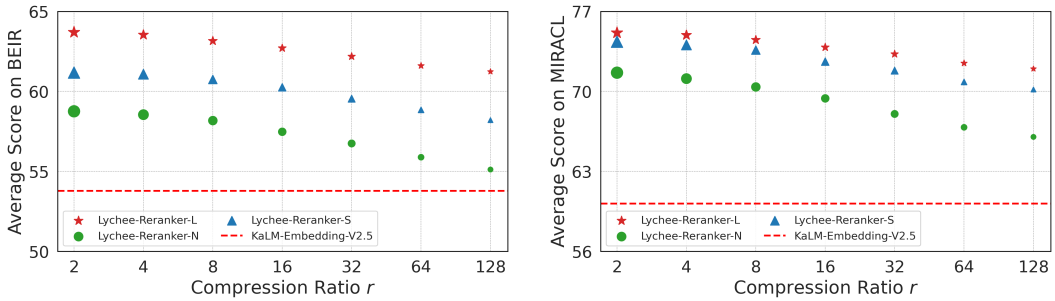
Models	Size	Cost	Avg.	ar	bn	de	en	es	fa	fi	fr	hi	id	ja	ko	ru	sw	te	th	yo	zh
First-stage Retriever																					
KaLM-Embedding-V2.5	0.5B	–	60.19	69.32	63.65	53.05	53.40	51.32	49.55	70.98	52.14	44.90	49.89	61.37	58.03	57.87	66.54	74.53	70.19	79.20	57.48
Second-stage Reranker																					
Models with more than 4B parameters																					
Qwen3-Reranker-8B	8B	359.8x	74.13	81.94	83.78	64.87	67.21	61.21	63.84	82.50	63.30	66.00	59.17	80.01	75.05	78.87	78.95	86.20	86.66	87.47	67.30
Models with 1B–4B parameters																					
Qwen3-Reranker-4B	4B	157.8x	74.28	81.35	84.94	65.45	66.19	61.25	**65.37**	82.67	63.39	**66.47**	**60.16**	79.94	73.66	77.89	78.59	**87.56**	**86.58**	**87.42**	**68.20**
`Lychee-Reranker-L`	4B	**29.1x**	**74.92**	**82.56**	**85.80**	**65.75**	**67.65**	**63.76**	65.09	**83.38**	**65.99**	66.30	59.17	**80.27**	**74.45**	**79.51**	**80.56**	87.53	85.96	87.13	67.71
bge-reranker-v2-gemma	2.5B	54.2x	69.82	75.43	80.28	62.27	63.53	60.42	60.30	77.45	61.91	62.32	57.26	72.34	70.42	71.51	74.70	83.10	79.37	80.85	63.36
mxbai-rerank-large-v2	1.5B	52.8x	63.26	74.50	65.53	50.08	61.42	55.82	52.33	72.42	56.15	45.44	53.50	71.92	66.96	69.97	63.75	57.90	79.35	79.06	62.50
Models with 0.5B–1B parameters																					
`Lychee-Reranker-S`	1B	**4.6x**	**74.06**	80.92	**85.69**	**64.22**	**66.45**	63.53	62.95	**82.09**	**65.29**	65.65	59.48	**78.91**	72.84	**78.13**	80.12	87.39	**85.53**	86.34	67.56
bge-reranker-large	0.6B	24.2x	52.52	55.24	62.45	44.94	50.45	42.47	42.96	68.89	46.47	38.00	37.96	42.19	46.41	44.29	57.98	70.03	64.18	75.16	55.26
bge-reranker-v2-m3	0.6B	24.2x	73.92	**81.45**	84.02	63.77	63.52	64.52	64.24	82.04	64.02	66.26	**61.87**	78.51	73.19	76.68	**80.41**	**88.30**	84.53	87.34	65.83
Qwen3-Reranker-0.6B	0.6B	28.2x	73.14	79.84	80.54	63.81	64.75	**64.79**	**65.28**	80.16	62.98	**67.27**	61.24	76.23	**76.58**	74.50	70.25	85.81	84.78	**87.35**	**70.30**
mxbai-rerank-base-v2	0.5B	19.9x	59.58	73.65	57.16	29.12	60.86	56.97	47.74	69.56	55.81	42.68	54.26	71.21	64.36	66.54	60.51	49.32	76.98	75.34	60.43
Models with fewer than 0.5B parameters																					
gte-reranker-base	0.3B	8.0x	69.99	77.48	79.34	57.21	63.16	59.80	59.41	77.77	59.19	60.67	57.61	73.67	73.45	72.05	74.54	**85.78**	78.93	**83.88**	**65.92**
jina-reranker-v2	0.3B	8.0x	69.05	73.20	81.45	**58.43**	58.52	58.32	59.64	77.44	56.40	59.96	59.01	71.47	**73.88**	66.47	**76.75**	85.46	80.71	82.10	63.73
`Lychee-Reranker-N`	0.27B	**1.0x**	**71.11**	**78.76**	**83.14**	57.45	**65.00**	**61.54**	**60.81**	**79.30**	**61.03**	**61.95**	**59.04**	**74.25**	72.76	**73.77**	75.74	**85.78**	**82.47**	83.00	64.14

Qwen3-Reranker series follows the conventional reranking paradigm and is trained on 19M high-quality instances (Zhang et al., 2025c). Meanwhile, `Lychee-Reranker` is substantially more efficient, demonstrating that the FBNL reranking paradigm supports expressive relevance modeling while reducing serving costs. Results and analysis on LMEB (Zhao et al., 2026b) are provided in §G.

5.3 In-depth Analysis

5.3.1 Effectiveness of Matryoshka Reranking

Figure 4 and Tables 18 and 19 report performance across MEP compression ratios on BEIR and MIRACL. They reveal the compactness–effectiveness trade-off. We make the following observations:



(a) BEIR. (b) MIRACL.

Figure 4: Reranking performance of `Lychee-Reranker` on BEIR and MIRACL under different Matryoshka embedding pooling compression ratios. Per-task results are reported in Tables 18 and 19.

Reranking performance decreases as the compression ratio increases. Overall, performance decreases as the compression ratio increases because aggressive pooling discards some useful information. However, performance degrades only modestly as r increases from 2 to 16, indicating that our models retain most of their reranking effectiveness under moderate compression. Even at $r = 128$, all `Lychee-Reranker` variants retain at least 92% of their average nDCG@10 at $r = 2$ on both BEIR and MIRACL. For practical deployment, we recommend compression ratios from $r = 4$ to $r = 16$, which offer a favorable trade-off between effectiveness and efficiency.

Larger models are more robust to compression. As r increases, passage representations retain less information, which can reduce reranking performance. Larger models show smaller performance drops at the same compression ratio. For example, at $r = 128$ on BEIR, `Lychee-Reranker-Large` retains 96.1% of its average nDCG@10 at $r = 2$, compared with 95.1% for `Lychee-Reranker-Small` and 93.8% for `Lychee-Reranker-Nano`. Overall, larger rerankers may be better suited to higher compression ratios, partly because their larger hidden dimensions (Table 1) may preserve more information per compressed passage token.

Performance gains show diminishing returns as the compression ratio decreases. Lower compression ratios preserve more information and generally improve performance, but diminishing gains suggest redundancy in passage representations. Moderate compression ratios, ranging from $r = 2$ to $r = 8$, therefore provide substantial efficiency gains while incurring little performance loss. These findings support moderate compression in practice and confirm the effectiveness of the FBNL design.

5.3.2 Potential of Test-time Compute Scaling

Tables 5 and 17 examine whether selectively allocating more test-time compute to promising candidates can achieve a favorable trade-off between reranking effectiveness and efficiency when the same model is used throughout. Specifically, we first rerank 100 candidates at a high compression ratio ($r = 32$ or $r = 64$), then rescore the top-20 candidates with the same model at $r = 2$ and return these 20 candidates in descending score order as the final ranked list. Results on BEIR and MIRACL show that this strategy largely preserves reranking effectiveness while reducing the estimated online computation cost severalfold. For example, on BEIR, `Lychee-Reranker-Small` achieves an average nDCG@10 of 61.11 under setting (c), compared with 61.17 under setting (a), while reducing the estimated relative serving cost from 8.65x to 2.75x. Interestingly, settings (c) and (d) outperform setting (a) on some tasks, suggesting that reranking more candidates at a low compression ratio does not necessarily improve reranking quality. The preliminary reranking stage ($r = 32$ or $r = 64$) may filter out misleading candidates before the subsequent post-ranking stage ($r = 2$). Overall, high- and low-compression reranking can complement each other, highlighting their practical value for deployment. §F provides results on performance scaling with test-time compute.

5.3.3 Ablation Study on Multi-stage Training

In §4.2, we introduce a multi-stage training pipeline to progressively improve reranking performance. Table 6 presents an ablation study of the three stages across diverse compression ratios $r \in \{2, 4, 8, 16, 32, 64, 128\}$. For Stage 2, we train models using two LoRA configurations: a rank

Table 5: Test-time compute scaling results on BEIR, measured by nDCG@10. The reranking and post-ranking columns specify the number of documents (d) and compression ratios (r). Cost denotes the estimated relative online computation cost, with the cost of `Lychee-Reranker-Nano` at $r = 4$ normalized to 1.0x. Note that all other evaluation settings follow those described in Table 3.

Models	Setting	Cost	Avg.	Test-Time Compute		BEIR											
				Reranking	Post-ranking	AA	CF	CQA	DB	FV	FQA	HQA	MS	NF	NQ	QR	SD	SF
Lychee-Reranker-N	(a)	1.89x	**58.75**	100d $\xrightarrow{r=2}$ 100d	N/A	63.87	43.23	41.08	**48.92**	93.13	**47.25**	**82.80**	**46.17**	38.42	**69.57**	**90.11**	**21.04**	**78.17**
	(b)	1.0x	58.54	100d $\xrightarrow{r=4}$ 100d	N/A	**63.97**	**43.25**	40.84	47.93	92.95	47.09	82.58	45.64	**38.72**	69.30	89.92	20.80	78.03
	(c)	0.6x	58.51	100d $\xrightarrow{r=32}$ 100d	20d $\xrightarrow{r=2}$ 20d	63.96	43.02	**41.16**	48.53	**93.18**	46.26	82.55	45.51	38.11	69.56	90.08	20.81	77.89
	(d)	0.54x	58.46	100d $\xrightarrow{r=64}$ 100d	20d $\xrightarrow{r=2}$ 20d	63.89	43.01	41.05	48.81	**93.18**	46.17	82.34	45.50	38.09	69.29	90.06	20.85	77.76
Lychee-Reranker-S	(a)	8.65x	**61.17**	100d $\xrightarrow{r=2}$ 100d	N/A	68.79	**45.98**	**47.35**	**50.30**	**93.50**	54.08	**83.74**	**47.33**	39.95	73.09	90.85	22.05	78.26
	(b)	4.58x	61.07	100d $\xrightarrow{r=4}$ 100d	N/A	**69.43**	45.63	47.06	49.77	93.36	53.62	83.55	46.90	39.89	**73.14**	90.77	22.15	**78.72**
	(c)	2.75x	61.11	100d $\xrightarrow{r=32}$ 100d	20d $\xrightarrow{r=2}$ 20d	69.28	45.77	**47.35**	50.10	93.47	**54.42**	83.66	46.79	**40.14**	72.95	**90.86**	22.15	77.48
	(d)	2.49x	61.06	100d $\xrightarrow{r=64}$ 100d	20d $\xrightarrow{r=2}$ 20d	69.29	45.82	47.21	49.90	93.47	54.20	83.48	46.67	40.08	72.83	90.85	**22.22**	77.80
Lychee-Reranker-L	(a)	55.05x	**63.69**	100d $\xrightarrow{r=2}$ 100d	N/A	72.97	47.80	**52.76**	**53.29**	**94.26**	**59.31**	**84.39**	**47.72**	42.26	**76.06**	90.76	24.91	**81.49**
	(b)	29.14x	63.53	100d $\xrightarrow{r=4}$ 100d	N/A	72.90	**47.94**	52.36	52.73	94.22	59.15	84.23	47.38	**42.35**	75.61	90.72	24.89	81.42
	(c)	17.49x	63.62	100d $\xrightarrow{r=32}$ 100d	20d $\xrightarrow{r=2}$ 20d	73.31	47.76	52.68	53.09	**94.26**	59.22	84.35	47.39	42.33	75.84	**90.79**	**25.10**	80.95
	(d)	15.87x	63.60	100d $\xrightarrow{r=64}$ 100d	20d $\xrightarrow{r=2}$ 20d	**73.42**	47.79	52.56	52.99	94.23	59.10	84.23	47.42	42.24	75.84	**90.79**	25.06	81.17

Table 6: Ablation study on the progressive multi-stage training pipeline (§4.2). "Nano", "Small", and "Large" denote `Lychee-Reranker-Nano`, `Lychee-Reranker-Small`, and `Lychee-Reranker-Large`, respectively. We report the average nDCG@10 on BEIR and MIRACL after each training stage under a range of compression ratios $r \in \{2, 4, 8, 16, 32, 64, 128\}$. "Retention" denotes the percentage of average nDCG@10 retained at $r = 128$ relative to $r = 2$. Within each model group, the best results are **boldfaced**, and the second-best results are underlined.

Models	Training Stage	BEIR								MIRACL							
		$r = 2$	$r = 4$	$r = 8$	$r = 16$	$r = 32$	$r = 64$	$r = 128$	Retention	$r = 2$	$r = 4$	$r = 8$	$r = 16$	$r = 32$	$r = 64$	$r = 128$	Retention
Nano	Stage 1	57.72	57.44	56.84	56.18	55.29	54.26	53.61	92.88%	68.93	68.33	67.48	66.16	65.03	63.39	62.68	90.93%
	Stage 2	58.51	58.28	57.82	57.20	56.34	55.47	54.62	93.35%	71.04	70.53	69.77	68.55	67.38	66.21	65.36	92.00%
	Stage 3	**58.75**	**58.54**	**58.17**	**57.47**	**56.74**	**55.88**	**55.11**	**93.80%**	**71.63**	**71.11**	**70.38**	**69.38**	**68.02**	**66.85**	**66.01**	**92.15%**
Small	Stage 1	59.57	59.49	58.96	58.41	57.60	56.91	56.11	94.19%	72.90	72.61	72.03	70.91	69.78	68.75	67.97	93.24%
	Stage 2	61.07	60.92	60.53	59.96	59.28	58.52	57.76	94.58%	74.10	73.63	73.08	72.15	71.28	70.40	69.62	93.95%
	Stage 3	**61.17**	**61.07**	**60.74**	**60.25**	**59.54**	**58.84**	**58.20**	**95.14%**	**74.34**	**74.06**	**73.61**	**72.60**	**71.81**	**70.82**	**70.16**	**94.38%**
Large	Stage 1	62.30	62.04	61.61	61.14	60.51	59.84	59.29	95.17%	74.63	74.24	73.54	72.52	71.54	70.67	69.96	93.74%
	Stage 2	63.56	63.37	63.01	62.53	62.06	61.46	61.07	96.08%	75.03	74.83	74.40	73.74	73.02	72.31	71.89	**95.82%**
	Stage 3	**63.69**	**63.53**	**63.15**	**62.70**	**62.18**	**61.60**	**61.23**	**96.14%**	**75.12**	**74.92**	**74.50**	**73.85**	**73.26**	**72.46**	**71.97**	95.81%

of 64 with an alpha of 32 and a rank of 96 with an alpha of 48. Since the two configurations perform similarly, Table 6 reports results for the configuration with a rank of 64 and an alpha of 32. Across all three model sizes and all seven compression ratios, performance on BEIR and MIRACL improves consistently from Stage 1 to Stage 3, demonstrating the effectiveness of our multi-stage training pipeline. Concretely, Stage 1 establishes a robust foundation for document reranking, distillation in Stage 2 substantially improves performance, and Stage 3 yields further modest gains. More importantly, robustness to compression generally improves across the three training stages. For example, from Stage 1 to Stage 3, the performance retention of `Lychee-Reranker-Nano` at $r = 128$ relative to $r = 2$ increases from 92.88% to 93.80% on BEIR and from 90.93% to 92.15% on MIRACL. Overall, these results show the effectiveness of the multi-stage training pipeline and provide a reference for future research on the FBNL paradigm. §H examines the effect of compression on the model's discriminative ability. §I presents an ablation study on MEP. §J presents case studies.

6 Conclusion

We present `Lychee-Reranker`, a family of efficient rerankers built on the *fast but not late-interaction* (FBNL) paradigm. Under the FBNL paradigm, the encoder produces reusable passage representations offline, while the decoder processes instructions and queries, interacting with these representations through merged self-attention and cross-attention to perform expressive relevance modeling. We introduce Matryoshka embedding pooling (MEP) to compress passage representations along the sequence dimension, reducing storage and serving costs while enabling flexible trade-offs between efficiency and effectiveness. Together, these designs offer four key advantages: (i) *efficient online serving*, (ii) *expressive relevance modeling*, (iii) *compact passage representations*, and (iv) *adjustable test-time compute*. Extensive experiments on BEIR, MIRACL, and LMEB show that the `Lychee-Reranker` series is competitive with strong industrial rerankers while significantly reducing online computation costs. In-depth analyses further demonstrate the models' strong robustness to compression, the effectiveness of our multi-stage training pipeline, and the value of test-time compute scaling. Overall, we hope that `Lychee-Reranker` will inspire further innovation within the FBNL paradigm and facilitate the development of scalable retrieval systems.

AI USE STATEMENT

AI tools were used to polish the writing and provide feedback on experimental design. The authors reviewed and verified all suggestions and take full responsibility for the final paper.

ETHICS STATEMENT

This work uses only publicly available datasets and pretrained models; it involves neither human-subjects research nor the collection of new personal data. We comply with the licenses of the resources employed in this study and use them for their intended purposes. Like other retrieval systems, our models may inherit biases from their training data; downstream users should therefore assess fairness, privacy, and potential misuse before deployment.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of our method and experimental setup to support reproducibility. Sections 3, 4, and 5 present the model architecture, model training, and experimental results, respectively. Appendices B, C, D, and E present the training data and processing procedures, experimental setup, implementation details, and task-specific instructions, respectively. Tables 8, 9, and 10 summarize the evaluation hyperparameters, public checkpoints, and training hyperparameters used in our experiments, respectively. We will release our code, training data, and model checkpoints from all training stages to facilitate reproducibility.

REFERENCES

Mohammad Kalim Akram, Saba Sturua, Nastia Havriushenko, Quentin Herreros, Michael Günther, Maximilian Werk, and Han Xiao. jina-embeddings-v5-text: Task-targeted embedding distillation. *arXiv preprint arXiv:2602.15547*, 2026.

Anthropic. Claude 3.7 sonnet and Claude Code, February 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet.

Yauhen Babakhin, Ronay Ak, Jiarui Cai, Vinay Raman, Radek Osmulski, Jakub Zakrzewski, Anmol Gupta, Oliver Holworthy, Sahel Sharifymoghaddam, Khang Pham, James Rong, Steve Han, Sean Sodha, Isabel Hulseman, and Bo Liu. NVIDIA Nemotron 3 Embed ranks #1 overall on RTEB, advancing agentic retrieval. HuggingFace Blog, 2026. URL https://huggingface.co/blog/nvidia/nemotron-3-embed-wins-rteb. Published July 16, 2026.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

BDBC-KG-NLP. UMETRIP-QA. https://github.com/BDBC-KG-NLP/CCL2022-DQAB, 2022.

Luiz Bonifacio, Vitor Jeronymo, Hugo Queiroz Abonizio, Israel Campiotti, Marzieh Fadaee, Roberto Lotufo, and Rodrigo Nogueira. mmarco: A multilingual version of the ms marco passage ranking dataset. *arXiv preprint arXiv:2108.13897*, 2021.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. A full-text learning to rank dataset for medical information retrieval. In *European Conference on Information Retrieval*, pp. 716–722. Springer, 2016.

Huiyao Chen, Yi Yang, Yinghui Li, Meishan Zhang, Baotian Hu, and Min Zhang. Beyond chunking: Discourse-aware hierarchical retrieval for long document question answering. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18176–18198, 2026.

Jianlyu 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*, pp. 2318–2335, 2024.

Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. A span-extraction dataset for chinese machine reading comprehension. 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. 5883–5889, 2019.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Winata, et al. Mmteb: Massive multilingual text embedding benchmark. In *International Conference on Learning Representations*, volume 2025, pp. 101715–101771, 2025.

Google. Gemma Formatting and System Instructions. `https://ai.google.dev/gemma/docs/core/prompt-structure`, 2025.

Zhipeng Guo, Yu Zhao, Yabin Zheng, Xiance Si, Zhiyuan Liu, and Maosong Sun. THUCTC: An efficient chinese text classifier. `https://github.com/thunlp/THUCTC`, 2016.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pp. 218–223, March 2017. doi: 10.5281/zenodo.4120316.

Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. Dbpedia-entity v2: A test collection for entity search. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pp. 1265–1268, 2017.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, et al. Dureader: a chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the workshop on machine reading for question answering*, pp. 37–46, 2018.

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

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. Cqadupstack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian document computing symposium*, pp. 1–8, 2015.

Baotian Hu, Qingcai Chen, and Fangze Zhu. Lcsts: A large scale chinese short text summarization dataset. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1967–1972, 2015.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Xinshuo Hu, Zifei Shan, Xinping Zhao, Zetian Sun, Zhenyu Liu, Dongfang Li, Shaolin Ye, Xinyuan Wei, Qian Chen, Baotian Hu, Haofen Wang, Jun Yu, and Min Zhang. Kalm-embedding: Superior training data brings a stronger embedding model, 2025. URL `https://arxiv.org/abs/2501.01028`.

Xuming Hu, Zhijiang Guo, GuanYu Wu, Aiwei Liu, Lijie Wen, and Philip S Yu. Chef: A pilot chinese dataset for evidence-based fact-checking. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3362–3376, 2022.

infgrad. retrieval_data_llm. https://huggingface.co/datasets/infgrad/retrieval_data_llm, 2024.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. First quora dataset release: Question pairs. https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs, 2017.

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.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. 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. 2567–2577, 2019.

Jina AI. jina-reranker-v2-base-multilingual. https://huggingface.co/jinaai/jina-reranker-v2-base-multilingual, 2024.

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

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. Gooaq: Open question answering with diverse answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 421–433, 2021.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 39–48, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

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 *ICLR*. OpenReview.net, 2025.

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

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. Paq: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115, 2021.

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *arXiv preprint arXiv:1607.06275*, 2016.

Xianming Li, Aamir Shakir, Rui Huang, Tsz-fung Andrew Lee, Julius Lipp, Benjamin Clavié, and Jing Li. Prorank: Prompt warmup via reinforcement learning for small language models reranking. *arXiv preprint arXiv:2506.03487*, 2025.

Yudong Li, Yuqing Zhang, Zhe Zhao, Linlin Shen, Weijie Liu, Weiquan Mao, and Hui Zhang. Csl: A large-scale chinese scientific literature dataset. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 3917–3923, 2022.

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

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and Teknium. Openorca: An open dataset of gpt augmented flan reasoning traces. https://huggingface.co/datasets/Open-Orca/OpenOrca, 2023.

Hongcheng Liu, Yusheng Liao, Yutong Meng, and Yuhao Wang. Xiezhi: Chinese law large language model. https://github.com/LiuHC0428/LAW_GPT, 2023.

Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. Neural re-ranking in multi-stage recommender systems: A review. In *IJCAI*, pp. 5512–5520. ijcai.org, 2022.

Zheng Liu, Chaofan Li, Shitao Xiao, Yingxia Shao, and Defu Lian. Llama2vec: Unsupervised adaptation of large language models for dense retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3490–3500, 2024.

Dingkun Long, Qiong Gao, Kuan Zou, Guangwei Xu, Pengjun Xie, Ruijie Guo, Jian Xu, Guanjun Jiang, Luxi Xing, and Ping Yang. Multi-cpr: A multi domain chinese dataset for passage retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3046–3056, 2022.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. Www'18 open challenge: financial opinion mining and question answering. In *Companion Proceedings of the The Web Conference 2018*, pp. 1941–1942, 2018.

Chaitanya Malaviya, Subin Lee, Sihao Chen, Elizabeth Sieber, Mark Yatskar, and Dan Roth. Expertqa: Expert-curated questions and attributed answers. 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. 3025–3045, 2024.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. SFR-Embedding-2: Advanced text embedding with multi-stage training, 2024a. URL https://huggingface.co/Salesforce/SFR-Embedding-2_R.

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, 2024b. URL https://www.salesforce.com/blog/sfr-embedding/.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 2014–2037, 2023.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

neural-bridge. Retrieval-Augmented Generation (RAG) Dataset 12000. https://huggingface.co/datasets/neural-bridge/rag-dataset-12000, 2023.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. Large dual encoders are generalizable retrievers. In *EMNLP*, pp. 9844–9855. Association for Computational Linguistics, 2022.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

OpenAI. Introducing Codex, May 2025. URL https://openai.com/index/introducing-codex/.

OpenClaw Contributors. The ai that really does things. any os. any platform. the lobster way. https://github.com/openclaw/openclaw, 2026. GitHub repository, accessed: 2026-09-04.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*, 2023.

Yujia Qin, Zihan Cai, Dian Jin, Lan Yan, Shihao Liang, Kunlun Zhu, Yankai Lin, Xu Han, Ning Ding, Huadong Wang, et al. Webcpm: Interactive web search for chinese long-form question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8968–8988, 2023.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 2383–2392, 2016.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, 2018.

Jun Rao, Xv Meng, Liang Ding, Shuhan Qi, Xuebo Liu, Min Zhang, and Dacheng Tao. Parameter-efficient and student-friendly knowledge distillation. *IEEE Trans. Multim.*, 26:4230–4241, 2024.

Chandan K Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. Shopping queries dataset: A large-scale esci benchmark for improving product search. *arXiv preprint arXiv:2206.06588*, 2022.

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.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3715–3734, 2022.

Sentence Transformers. Yahoo Answers. https://huggingface.co/datasets/sentence-transformers/yahoo-answers, 2024.

Chih Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. Drcd: a chinese machine reading comprehension dataset. *arXiv preprint arXiv:1806.00920*, 2018.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. Long and diverse text generation with planning-based hierarchical variational model. 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. 3257–3268, 2019.

Shivalika Singh, Freddie Vargus, Daniel D'souza, Börje F Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura O'Mahony, et al. Aya dataset: An open-access collection for multilingual instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11521–11567, 2024.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *ICLR*. OpenReview.net, 2025.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, et al. jina-embeddings-v3: Multilingual embeddings with task lora. *arXiv preprint arXiv:2409.10173*, 2024.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. *arXiv preprint arXiv:2304.09542*, 2023.

Hongxuan Tang, Hongyu Li, Jing Liu, Yu Hong, Hua Wu, and Haifeng Wang. Dureader_robust: A chinese dataset towards evaluating robustness and generalization of machine reading comprehension in real-world applications. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 955–963, 2021.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819, 2018.

TitanMLData. arxiv_qa. `https://huggingface.co/datasets/TitanMLData/arxiv_qa`, 2023.

Ustinian. Law question-answering dataset. `https://www.heywhale.com/mw/dataset/5e953ca8e7ec38002d02fca7`, 2020.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. Trec-covid: constructing a pandemic information retrieval test collection. *ACM SIGIR Forum*, 54:1–12, 2021.

W3Techs. Usage statistics of content languages for websites. `https://w3techs.com/technologies/overview/content_language`, 2026. Accessed: 2026-09-07.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 241–251, 2018.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7534–7550, 2020.

Feng Wang, Yuqing Li, and Han Xiao. Jina-reranker-v3: Last but not late interaction for listwise document reranking. *arXiv preprint arXiv:2509.25085*, 2025.

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 62nd annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 11897–11916, 2024.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, et al. Cord-19: The covid-19 open research dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, 2020.

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*, pp. 641–649, 2024a.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Xingrun Xing. Lm-cocktail: Resilient tuning of language models via model merging. In *ACL (Findings)*, volume ACL 2024 of *Findings of ACL*, pp. 2474–2488. Association for Computational Linguistics, 2024b.

Xiaohui Xie, Qian Dong, Bingning Wang, Feiyang Lv, Ting Yao, Weinan Gan, Zhijing Wu, Xiangsheng Li, Haitao Li, Yiqun Liu, et al. T2ranking: A large-scale chinese benchmark for passage ranking. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pp. 2681–2690, 2023.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: improving retrieval-augmented lms with context compression and selective augmentation. In *ICLR*. OpenReview.net, 2024.

Dongjie Yang, Ruifeng Yuan, Yuantao Fan, Yifei Yang, Zili Wang, Shusen Wang, and Hai Zhao. Refgpt: Dialogue generation of gpt, by gpt, and for gpt. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2511–2535, 2023.

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

Biao Zhang, Paul Suganthan, Gaël Liu, Ilya Philippov, Sahil Dua, Ben Hora, Kat Black, Gus Martins, Omar Sanseviero, Shreya Pathak, Cassidy Hardin, Francesco Visin, Jiageng Zhang, Kathleen Kenealy, Qin Yin, Xiaodan Song, Olivier Lacombe, Armand Joulin, Tris Warkentin, and Adam Roberts. T5gemma 2: Seeing, reading, and understanding longer. *CoRR*, abs/2512.14856, 2025a.

Meishan Zhang, Xin Zhang, Xinping Zhao, Shouzheng Huang, Baotian Hu, and Min Zhang. On the role of pretrained language models in general-purpose text embeddings: A survey. *arXiv preprint arXiv:2507.20783*, 2025b.

Sheng Zhang, Xin Zhang, Hui Wang, Lixiang Guo, and Shanshan Liu. Multi-scale attentive interaction networks for chinese medical question answer selection. *IEEE access*, 6:74061–74071, 2018.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, et al. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1393–1412, 2024.

Xin Zhang, Ziqi Dai, Mingxin Li, Yanzhao Zhang, Dingkun Long, Pengjun Xie, Meishan Zhang, Wenjie Li, and Min Zhang. Supervised fine-tuning or contrastive learning? towards better multimodal LLM reranking. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Xinyu Zhang, Xueguang Ma, Peng Shi, and Jimmy Lin. Mr. tydi: A multi-lingual benchmark for dense retrieval. In *Proceedings of the 1st workshop on multilingual representation learning*, pp. 127–137, 2021.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. Miracl: A multilingual retrieval dataset covering 18 diverse languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131, 2023.

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

Ziyin Zhang, Zihan Liao, Hang Yu, Peng Di, and Rui Wang. F2llm-v2: Inclusive, performant, and efficient embeddings for a multilingual world. *arXiv preprint arXiv:2603.19223*, 2026b.

Xinping Zhao, Baotian Hu, Yan Zhong, Shouzheng Huang, Zihao Zheng, Meng Wang, Haofen Wang, and Min Zhang. Raserec: Retrieval-augmented sequential recommendation. *CoRR*, abs/2412.18378, 2024a.

Xinping Zhao, Dongfang Li, Yan Zhong, Boren Hu, Yibin Chen, Baotian Hu, and Min Zhang. SEER: self-aligned evidence extraction for retrieval-augmented generation. In *EMNLP*, pp. 3027–3041. Association for Computational Linguistics, 2024b.

Xinping Zhao, Xinshuo Hu, Zifei Shan, Shouzheng Huang, Yao Zhou, Xin Zhang, Zetian Sun, Zhenyu Liu, Dongfang Li, Xinyuan Wei, Youcheng Pan, Yang Xiang, Meishan Zhang, Haofen Wang, Jun Yu, Baotian Hu, and Min Zhang. KaLM-embedding-v2: Superior training techniques and data inspire a versatile embedding model. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Xinping Zhao, Xinshuo Hu, Jiaxin Xu, Danyu Tang, Xin Zhang, Mengjia Zhou, Yan Zhong, Yao Zhou, Zifei Shan, Meishan Zhang, Baotian Hu, and Min Zhang. LMEB: long-horizon memory embedding benchmark. *CoRR*, abs/2603.12572, 2026b.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12834–12859, 2024.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

A Complexity Analysis

A.1 Time Complexity

Table 7: Time complexity of reranking K passages. Enc- and dec-based rerankers process each query–passage pair online. By contrast, our enc–dec-based reranker reuses cached passage representations and applies merged self- and cross-attn to the query context and compressed passage representations.

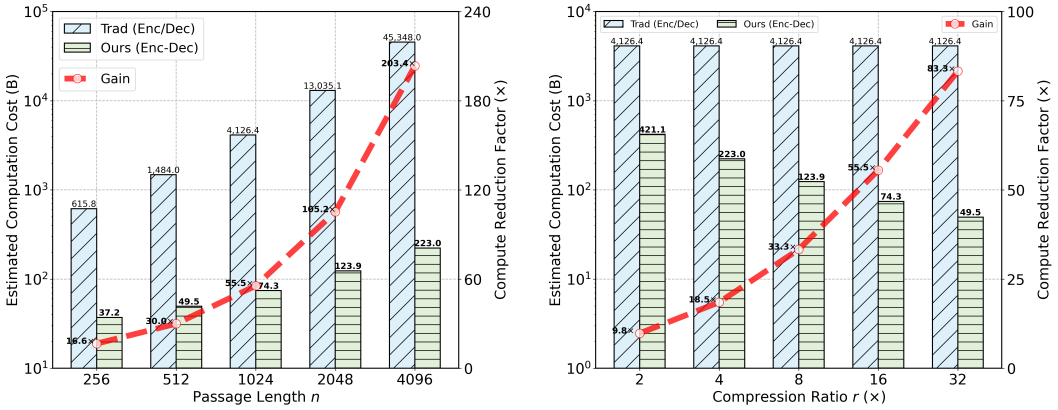
Model Type	Interaction Pattern	Time Complexity
Encoder-based reranker	Bidirectional attention	$\mathcal{O}\left(KL\left((\lvert q\rvert+n)^2d+(\lvert q\rvert+n)d^2\right)\right)$
Decoder-based reranker	Causal attention	$\mathcal{O}\left(KL\left((\lvert q\rvert+n)^2d+(\lvert q\rvert+n)d^2\right)\right)$
Enc-Dec-based reranker **(ours)**	Merged self-attn. + cross-attn.	$\mathcal{O}\left(\frac{L}{2}K\left(\lvert q\rvert\left(\lvert q\rvert+\lceil\frac{n}{r}\rceil\right)d+\left(\lvert q\rvert+\lceil\frac{n}{r}\rceil\right)d^2\right)\right)$

Table 7 compares the online time complexity of our encoder–decoder reranker with that of conventional encoder- and decoder-based rerankers. Let K denote the number of candidate passages to rerank, $|q|$ the query length, n the average passage length, and d the hidden dimension. Let L denote the total number of Transformer layers in a reranker. For an encoder–decoder reranker, L includes both the encoder and decoder layers. Let r denote the MEP compression ratio. For a Transformer layer processing a sequence of length n, self-attention has a complexity of $\mathcal{O}(n^2d)$, while feed-forward and projection operations have a complexity of $\mathcal{O}(nd^2)$. For a fair comparison, we assume that conventional encoder- and decoder-based rerankers have the same total parameter count as `Lychee-Reranker`. Since `Lychee-Reranker` splits its parameters evenly between the encoder and decoder, the decoder used during online reranking has approximately $L/2$ layers. The encoder is used only to pre-encode passages offline and therefore incurs no online reranking cost. Conventional rerankers jointly process each query–passage pair during online inference to estimate relevance. For each candidate, the resulting input sequence has a length of $|q|+n$[1]. Therefore, reranking K candidates requires:

$$\mathcal{O}\left(KL\left((|q|+n)^2d+(|q|+n)d^2\right)\right). \tag{11}$$

By contrast, `Lychee-Reranker` performs only decoder computation online and reuses pre-encoded passage representations. Through merged self- and cross-attention, the query tokens attend to

[1]We ignore prompt-template and instruction tokens, as their length is negligible compared with $|q|+n$.



(a) Varying the passage length n while fixing $|q| = 32$, $r = 16$, $K = 100$, $d = 640$, and $L = 36$.

(b) Varying the compression ratio r while fixing $|q| = 32$, $n = 1024$, $K = 100$, $d = 640$, and $L = 36$.

Figure 5: Comparison of online serving time complexity for varying passage lengths n and compression ratios r. Bars show the estimated computational cost, and the line shows the factor by which our reranking paradigm reduces computation relative to traditional rerankers. Here, "B" denotes billion, "Trad" denotes traditional, "Enc/Dec" denotes encoder- or decoder-based rerankers, and "Enc-Dec" represents our encoder–decoder reranker.

both the query context and the cached passage representations. With MEP, each passage is represented by only $\lceil n/r \rceil$ cached tokens, which remain fixed throughout relevance modeling. Therefore, the online serving complexity of `Lychee-Reranker` is:

$$\mathcal{O}\left(\frac{L}{2} K \left(|q| \left(|q| + \left\lceil \frac{n}{r} \right\rceil\right) d + \left(|q| + \left\lceil \frac{n}{r} \right\rceil\right) d^2\right)\right), \tag{12}$$

where **the first term** accounts for merged self- and cross-attention between the query and the combined query–passage context. **The second term** accounts for query-side projections and feed-forward computation, together with key/value projections of the cached passage representations. Encoding K passages offline incurs a computational cost of $\mathcal{O}\left(K \frac{L}{2}(n^2 d + n d^2)\right)$. Since each passage is encoded only once and its resulting representation is reused across queries, this cost is excluded from the online serving complexity.

Efficiency Gains. Figure 5 quantifies serving efficiency under representative settings. In Figure 5a, we fix $|q|$, r, K, d, and L and vary the passage length over $n \in \{256, 512, 1024, 2048, 4096\}$. Even at $n = 256$, our method reduces the estimated computational cost by a factor of 16.6. The efficiency gain increases with passage length, reaching $203.4\times$ at $n = 4096$. In Figure 5b, we fix $|q|$, n, K, d, and L and vary the compression ratio over $r \in \{2, 4, 8, 16, 32\}$. Even at the low compression ratio of $r = 2$, our method reduces the estimated computational cost by a factor of nearly 10; at $r = 4$ and $r = 8$, the reduction factor increases to 18.5 and 33.3, respectively. Overall, these results demonstrate that offline passage encoding and MEP substantially reduce online serving overhead, with particularly large benefits for longer passages and higher compression ratios.

A.2 Space Complexity

We analyze the storage overhead of caching passage representations in `Lychee-Reranker` at scale. Conventional encoder- and decoder-based rerankers process each query–passage pair online and therefore require no caching of passage representations.

By contrast, our model caches pre-encoded passage representations to avoid recomputing them for each query, thereby introducing additional storage overhead. Suppose the corpus contains N passages, each with an average length of n tokens, and that the passage representations have a hidden dimension of d. Caching the full passage representations requires:

$$\mathcal{O}(Nnd), \tag{13}$$

where each passage has a representation $\mathbf{H}_p \in \mathbb{R}^{n \times d}$. When reranking a candidate list of size K, the online buffer for passage representations requires $\mathcal{O}(Knd)$ space before compression. This buffer

constitutes the main additional space overhead of our model relative to encoder- and decoder-based rerankers. To reduce this overhead, `Lychee-Reranker` introduces **Matryoshka Embedding Pooling** to compress passage representations along the sequence dimension. With a compression ratio of r, each cached passage representation is compressed to $\mathbf{H}_p^{(r)} \in \mathbb{R}^{\lceil n/r \rceil \times d}$, reducing the total cache size to

$$\mathcal{O}\left(N \left\lceil \frac{n}{r} \right\rceil d\right). \tag{14}$$

The space required by the online candidate buffer is likewise reduced to $\mathcal{O}(K\lceil n/r \rceil d)$. Thus, MEP balances storage cost and reranking quality: increasing r reduces the cache size nearly linearly while retaining information for fine-grained relevance modeling by the decoder.

B Training Data

We fine-tune `Lychee-Reranker` on retrieval-specific datasets to develop its reranking capabilities. To improve robustness and generalization, we collect and process large-scale, multilingual, and multi-domain training data spanning diverse retrieval scenarios, including web search, question answering, duplicate-question retrieval, and fact verification. Our training data is drawn primarily from two public sources: the retrieval subset of the KaLM embedding fine-tuning dataset (Zhao et al., 2026a) and a selected subset of the BGE-M3 training dataset (Chen et al., 2024). For queries with fewer than 16 hard negatives, we use KaLM-Embedding-V2.5 (Zhao et al., 2026a) to retrieve the top-100 candidate passages from the corresponding corpus and sample 16 hard negatives from ranks 10–50. Otherwise, we retain the existing hard negatives. This sampling strategy encourages the model to distinguish relevant passages from hard negatives while avoiding overly easy low-ranked negatives and reducing the risk of selecting false negatives from the highest-ranked candidates. Each resulting training instance contains 1 query, 1 positive passage, and at least 16 hard negatives. To provide fine-grained supervision, we use Qwen3-Reranker-8B (Zhang et al., 2025c) to assign a teacher score to each query–passage pair. The score is defined as the difference between the logits of the `yes` and `no` labels, *i.e.,* $z_{\text{yes}} - z_{\text{no}}$. Applying the sigmoid function to this score yields the soft teacher label y used in Equation 10. After preprocessing, the final training set contains approximately 3.9 million instances. Detailed statistics are provided in Table 16.

Table 8: Evaluation hyperparameters used in our experiments. Because bge-reranker-large supports a maximum sequence length of only 512 tokens, we set the maximum combined query and passage length to 512 tokens for this model.

Hyperparameter	**Value**
Evaluation metric	nDCG@10
First-stage retriever	KaLM-Embedding-V2.5
Number of candidates retrieved per query, K	100
Default compression ratio r	$r = 4$
Precision	bf16
Embedder query max length $\lvert q \rvert$	512
Embedder passage max length n	512
Reranker query max length $\lvert q \rvert$	512
Reranker passage max length n	1024

C Experimental Setup

Benchmarks. We evaluate reranking performance on retrieval benchmarks covering diverse aspects of reranking capability. We use BEIR (Thakur et al., 2021) to assess reranking performance across 13 heterogeneous English retrieval tasks, ranging from question answering on HotpotQA (Yang et al., 2018) and duplicate-question retrieval on CQADupStack (Hoogeveen et al., 2015) to entity retrieval on DBPedia (Hasibi et al., 2017) and fact verification on FEVER (Thorne et al., 2018). We exclude Touche2020 and TRECCOVID because their test sets contain only 49 and 50 queries, respectively, making the evaluation results sensitive to individual queries. We use MIRACL (Zhang et al., 2023) to

Table 9: Model specifications and public checkpoints for the embedding and reranking models evaluated in this work.

Model	Params.	#Layers	Hidden Dim.	Public Model Checkpoints (Link)
KaLM-Embedding-V2.5	0.5B	24	896	https://huggingface.co/KaLM-Embedding/KaLM-embedding-multilingual-mini-instruct-v2.5
bge-reranker-large	0.6B	24	1024	https://huggingface.co/BAAI/bge-reranker-large
bge-reranker-v2-m3	0.6B	24	1024	https://huggingface.co/BAAI/bge-reranker-v2-m3
bge-reranker-v2-gemma	2.5B	18	2048	https://huggingface.co/BAAI/bge-reranker-v2-gemma
gte-multilingual-reranker-base	0.3B	12	768	https://huggingface.co/Alibaba-NLP/gte-multilingual-reranker-base
Qwen3-Reranker-0.6B	0.6B	28	1024	https://huggingface.co/Qwen/Qwen3-Reranker-0.6B
Qwen3-Reranker-4B	4B	36	2560	https://huggingface.co/Qwen/Qwen3-Reranker-4B
Qwen3-Reranker-8B	8B	36	4096	https://huggingface.co/Qwen/Qwen3-Reranker-8B
jina-reranker-v2-base-multilingual	0.3B	12	768	https://huggingface.co/jinaai/jina-reranker-v2-base-multilingual
mxbai-rerank-base-v2	0.5B	24	896	https://huggingface.co/mixedbread-ai/mxbai-rerank-base-v2
mxbai-rerank-large-v2	1.5B	28	1536	https://huggingface.co/mixedbread-ai/mxbai-rerank-large-v2

evaluate multilingual reranking performance across 18 languages from diverse linguistic families, including Arabic, Bengali, Hindi, and Swahili. We further evaluate on the six dialogue memory retrieval tasks from LMEB (Zhao et al., 2026b), which assess retrieval and reranking capabilities in long-horizon dialogue memory scenarios. We adopt a two-stage retrieval-then-reranking pipeline for all experiments. In the first stage, we use KaLM-Embedding-V2.5 (Zhao et al., 2026a) as the dense retriever to obtain the top-100 candidate passages for each query. In the second stage, all rerankers rescore the same top-100 candidates to ensure a fair comparison. We use nDCG@10 as the main evaluation metric. Task-specific evaluation instructions are provided in Appendix E. Overall, these benchmarks provide a comprehensive evaluation across diverse retrieval tasks and domains, multilingual settings, and long-horizon memory scenarios.

Baselines. We compare the `Lychee-Reranker` series with representative open-source rerankers whose parameter counts range from 0.3B to 8B. These include bge-reranker-large, bge-reranker-v2-m3, and bge-reranker-v2-gemma (Liu et al., 2024; Xiao et al., 2024a; Chen et al., 2024); gte-multilingual-reranker-base (gte-reranker-base), Qwen3-Reranker-0.6B, Qwen3-Reranker-4B, and Qwen3-Reranker-8B (Zhang et al., 2025c; 2024); jina-reranker-v2-base-multilingual (jina-reranker-v2) (Jina AI, 2024); and mxbai-rerank-base-v2 and mxbai-rerank-large-v2 (Li et al., 2025). These baselines cover widely used encoder- and decoder-based rerankers, as well as recent multilingual and LLM-based rerankers, enabling comprehensive comparisons across architectures and model scales. Unless otherwise specified, `Lychee-Reranker` uses the default compression ratio of $r = 4$ in all experiments. Detailed evaluation hyperparameter are provided in Table 8. Table 9 summarizes the public checkpoints and key specifications of the embedding and reranking models used in our evaluation.

D Implementation Details

The `Lychee-Reranker` series is initialized from T5Gemma2 encoder–decoder checkpoints (Zhang et al., 2025a) and fine-tuned using LoRA (Hu et al., 2021), with LoRA adapters applied to both the encoder and decoder. For `Lychee-Reranker-Nano` and `Lychee-Reranker-Small`, LoRA targets all attention and feed-forward projection modules: `q_proj`, `k_proj`, `v_proj`, `out_proj`, `gate_proj`, `up_proj`, and `down_proj`. For `Lychee-Reranker-Large`, LoRA targets only the attention projection modules: `q_proj`, `k_proj`, `v_proj`, and `out_proj`. Specifically, `Lychee-Reranker-Nano`, `Lychee-Reranker-Small`, and `Lychee-Reranker-Large` are initialized from the `t5gemma-2-270m-270m`, `t5gemma-2-1b-1b`, and `t5gemma-2-4b-4b` checkpoints, respectively. All models are trained using bf16 precision, with maximum query and passage lengths of 128 and 512 tokens, respectively. The training group size is 16, with each training group containing 1 positive passage and 15 negative passages sampled from the negative pool. For Matryoshka Embedding Pooling (MEP), we use compression ratios $\mathcal{R} = \{2, 8, 32, 128\}$ and assign an equal loss weight of $\lambda_r = 1$ to each ratio. We optimize the models using Adam (Kingma & Ba, 2014) ($\beta_1 = 0.9$, $\beta_2 = 0.999$), with a weight decay of 1×10^{-5} and a cosine learning-rate schedule. We clip gradients to a maximum gradient norm of 1.0.

We follow the progressive multi-stage training pipeline described in §4.2. All models undergo all three training stages. We set the learning rate to 2×10^{-4} for the first two stages and train each model for

one epoch in each stage. The third stage applies model soup and involves no additional optimization. In the distillation stage, Qwen3-Reranker-8B (Zhang et al., 2025c) serves as the teacher for all models. For each model, we train two variants in the second stage: one with a LoRA rank of 96 and an alpha of 48, and the other with a LoRA rank of 64 and an alpha of 32. In the third stage, we first merge each LoRA adapter into its backbone and then construct a model soup (Xiao et al., 2024b) by averaging the two resulting model checkpoints with equal weights of 0.5. The per-GPU batch sizes are 2, 2, and 1, and the numbers of gradient accumulation steps are 2, 2, and 4 for `Lychee-Reranker-Nano`, `Lychee-Reranker-Small`, and `Lychee-Reranker-Large`, respectively. All models are trained on 32 NVIDIA RTX 5090 GPUs, with an effective global batch size of 128. Training takes approximately 30k steps per epoch. Table 10 summarizes the detailed training hyperparameters.

Table 10: Training hyperparameters for the `Lychee-Reranker` series.

Hyperparameter	**Nano**	**Small**	**Large**
Number of stages	3	3	3
Precision	bf16	bf16	bf16
Query max length $\|q\|$	128	128	128
Passage max length n	512	512	512
MEP compression ratios $\mathcal{R}$	$\{2, 8, 32, 128\}$	$\{2, 8, 32, 128\}$	$\{2, 8, 32, 128\}$
Loss weight λ_r for each $r \in \mathcal{R}$	1	1	1
Learning rate (Stages 1–2)	2×10^{-4}	2×10^{-4}	2×10^{-4}
Per-GPU Batch size	2	2	1
Gradient Accumulation steps	2	2	4
Number of GPUs	32	32	32
Effective Global batch size	128	128	128
GPU hours per epoch	$\sim$600	$\sim$1400	$\sim$2550
LoRA configuration	Stage 1: (rank 96, alpha 48); Stage 2: (rank 96, alpha 48) and (rank 64, alpha 32)		
LoRA target modules	`q_proj`, `k_proj`, `v_proj`, `out_proj`, `gate_proj`, `up_proj`, `down_proj`		`q_proj`, `k_proj`, `v_proj`, `out_proj`
Model soup weights (Stage 3)	$0.5, 0.5$	$0.5, 0.5$	$0.5, 0.5$
Number of epochs (Stages 1–2)	1 epoch per stage	1 epoch per stage	1 epoch per stage
Warmup ratio	0.1	0.1	0.1
Backbone	t5gemma-2-270m-270m	t5gemma-2-1b-1b	t5gemma-2-4b-4b
Teacher model	Qwen3-Reranker-8B	Qwen3-Reranker-8B	Qwen3-Reranker-8B

Table 11: Task instructions used for evaluation on the BEIR (Thakur et al., 2021) benchmark.

Task Name	**Instruction Template**
ArguAna (**AA**)	Given a claim, find documents that refute the claim.
Climate-FEVER (**CF**)	Given a claim about climate change, retrieve documents that support or refute the claim.
CQADupStack (**CQA**)	Given a question, retrieve detailed question descriptions from Stackexchange that are duplicates to the given question.
DBPedia (**DB**)	Given a query, retrieve relevant entity descriptions from DBPedia.
FEVER (**FV**)	Given a claim, retrieve documents that support or refute the claim.
FiQA2018 (**FQA**)	Given a financial question, retrieve user replies that best answer the question.
HotpotQA (**HQA**)	Given a multi-hop question, retrieve documents that can help answer the question.
MSMARCO (**MS**)	Given a query, retrieve documents that answer the query.
NFCorpus (**NF**)	Given a question, retrieve relevant documents that best answer the question.
NQ (**NQ**)	Given a question, retrieve Wikipedia passages that answer the question.
Quora (**QR**)	Given a question, retrieve questions that are semantically equivalent to the given question.
SCIDOCS (**SD**)	Given a scientific paper title, retrieve paper abstracts that are cited by the given paper.
SciFact (**SF**)	Given a scientific claim, retrieve documents that support or refute the claim.

E TASK INSTRUCTIONS

Tables 11, 12, and 13 summarize the task-specific instructions used to evaluate models on BEIR, MIRACL, and LMEB, respectively.

Table 12: Task instructions used for evaluation on the MIRACL (Zhang et al., 2023) benchmark.

Task Name	Instruction Template
Arabic (**ar**), Bengali (**bn**), German (**de**), English (**en**), Spanish (**es**), Persian (**fa**), Finnish (**fi**), French (**fr**), Hindi (**hi**), Indonesian (**id**), Japanese (**ja**), Korean (**ko**), Russian (**ru**), Swahili (**sw**), Telugu (**te**), Thai (**th**), Yoruba (**yo**), Chinese (**zh**)	Given a query, retrieve documents that answer the query.

Table 13: Task instructions used for evaluation on the LMEB (Zhao et al., 2026b) benchmark.

Task Name	Instruction Template
ConvoMem, LoCoMo, LongMemEval, MemBench, REALTALK, TMD	We use the task instructions provided by LMEB (Zhao et al., 2026b) without modification.

F Performance Scaling with Test-time Compute

We further analyze how performance scales with test-time compute across model sizes and compression ratios. Figure 6 summarizes the trends on BEIR and MIRACL. We draw the following main conclusions:

Whether larger models provide a free lunch at similar cost depends on the application scenario. At comparable computational cost, larger models with stronger compression can improve performance on BEIR, but such gains are not consistently observed on MIRACL. For example, `Lychee-Reranker-Large` at $r = 32$ incurs a lower computational cost than `Lychee-Reranker-Small` at $r = 2$ (46.6x vs. 62.3x) and achieves higher average nDCG@10 on BEIR (62.18 vs. 61.17), but lower performance on MIRACL (73.26 vs. 74.34). This suggests that model size and compression ratio should be chosen jointly according to the application scenario, performance requirements, and available compute budget.

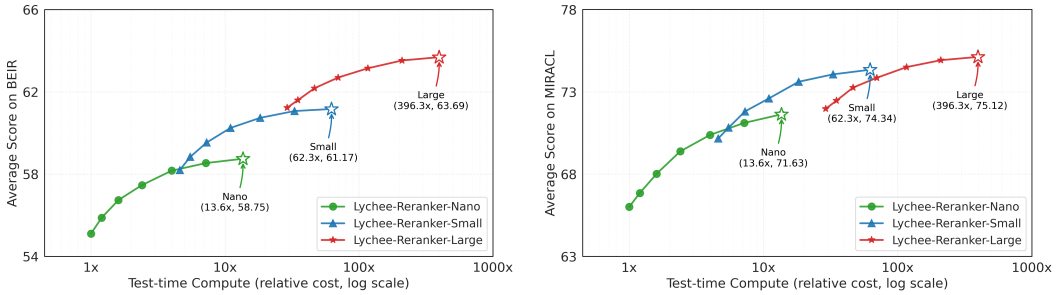
Performance exhibits diminishing returns as computation increases. For a fixed model size, increasing test-time compute by reducing the compression ratio generally improves reranking performance, although the marginal gains diminish. Across both BEIR and MIRACL, using $r = 4$ or $r = 8$ often provides a favorable trade-off, achieving performance close to that at $r = 2$ at substantially lower cost. As the performance curve flattens, switching to a larger model can offer greater gains than allocating additional compute to the same model.

Smaller models are more sensitive to compression. Smaller models exhibit larger performance drops as the compression ratio increases. For example, as r increases from 2 to 128 on BEIR, the average nDCG@10 of `Lychee-Reranker-Nano` drops by 3.64 points, from 58.75 to 55.11, compared with a 2.46 points drop for `Lychee-Reranker-Large`, from 63.69 to 61.23. These results suggest that smaller models rely more heavily on retaining passage information, while larger models are more robust to compression.

G Results on LMEB

Agentic memory is increasingly important for LLM-based agents, as systems such as OpenClaw (OpenClaw Contributors, 2026), OpenAI Codex (OpenAI, 2025), and Claude Code (Anthropic, 2025) retrieve and use stored information to maintain contextual continuity and support long-horizon tasks. Motivated by this, we further evaluate `Lychee-Reranker` on the six dialogue memory retrieval tasks in LMEB (Zhao et al., 2026b), a benchmark for long-horizon memory retrieval. The results in Tables 14 and 15 reveal the following key findings:

Reranking is particularly important for memory retrieval. `Lychee-Reranker-Nano`, `Lychee-Reranker-Small`, and `Lychee-Reranker-Large` improve over the first-stage retriever by 10.27, 12.85, and 14.91 nDCG@10 points on LMEB, respectively, compared with gains



(a) BEIR. (b) MIRACL.

Figure 6: Performance scaling of the `Lychee-Reranker` series with test-time compute. Each curve shows how reranking performance changes as computational cost increases. Results are taken from Tables 18 and 19.

Table 14: Reranking results measured by nDCG@10 on the six dialogue memory retrieval tasks in LMEB (Zhao et al., 2026b). "N", "S", and "L" denote Nano, Small, and Large, respectively. Avg. denotes the average performance across the six tasks. All `Lychee-Reranker` variants use a compression ratio of $r = 4$. Within each parameter group, the best results are **boldfaced**, and the second-best results are underlined. Other settings follow those in Table 3.

Models	Size	Cost	Avg.	ConvoMem	LoCoMo	LongMemEval	MemBench	REALTALK	TMD
First-stage Retriever									
KaLM-Embedding-V2.5	0.5B	–	50.80	62.74	41.88	75.18	69.59	38.60	16.82
Second-stage Reranker									
Models with more than 4B parameters									
Qwen3-Reranker-8B	8B	359.8x	66.42	64.64	65.91	78.93	76.31	57.13	55.58
Models with 1B–4B parameters									
Qwen3-Reranker-4B	4B	157.8x	62.86	66.68	63.43	53.72	78.65	55.91	**58.79**
`Lychee-Reranker-L`	4B	**29.1x**	65.71	70.34	**64.80**	**89.54**	77.99	**56.68**	34.94
bge-reranker-v2-gemma	2.5B	54.2x	63.50	67.12	61.30	87.23	76.50	54.74	34.13
mxbai-rerank-large-v2	1.5B	52.8x	**66.08**	**70.41**	61.33	87.16	**81.18**	54.43	41.96
Models with 0.5B–1B parameters									
`Lychee-Reranker-S`	1B	**4.6x**	63.65	68.41	**62.51**	**86.66**	75.41	**54.05**	34.87
bge-reranker-large	0.6B	24.2x	60.64	62.55	58.94	84.10	71.48	51.50	35.30
bge-reranker-v2-m3	0.6B	24.2x	62.38	69.15	60.20	82.39	76.82	53.90	31.84
Qwen3-Reranker-0.6B	0.6B	28.2x	**63.69**	67.36	60.32	84.91	75.67	53.26	**40.63**
mxbai-rerank-base-v2	0.5B	19.9x	62.47	**69.41**	55.76	86.29	**79.97**	49.75	33.61
Models with fewer than 0.5B parameters									
gte-reranker-base	0.3B	8.0x	56.08	54.78	51.01	78.94	68.76	47.97	35.02
jina-reranker-v2	0.3B	8.0x	59.60	54.70	56.22	86.55	72.66	**51.53**	**35.97**
`Lychee-Reranker-N`	0.27B	**1.0x**	**61.07**	**64.89**	**58.44**	**86.66**	**74.11**	50.89	31.44

of 4.76, 7.29, and 9.75 points on BEIR. These consistently larger gains on LMEB highlight the value of reranking for long-horizon memory retrieval.

Memory reranking remains challenging. Across the six dialogue memory reranking tasks, no reranker achieves the best result on all tasks. Each model shows weaknesses on some tasks. For example, Qwen3-Reranker-8B achieves an nDCG@10 of 78.93 on LongMemEval, trailing the smaller `Lychee-Reranker-Large` (4B), which scores 89.54. This indicates the need for further improvements in memory retrieval.

Reranking is especially effective for complex queries. Specifically, TMD contains many temporal queries involving relative or absolute time expressions. Such queries are challenging for embedding models, but rerankers handle them more effectively. For example, Qwen3-Reranker-4B achieves an nDCG@10 of 58.79 on TMD, compared with 16.82 for the first-stage retriever.

The retrieve-then-rerank pipeline can be more effective than increasing embedding model size. As shown in Table 15, bge-multilingual-gemma (9B) and Nemotron-3-Embed-8B-BF16 (8B) achieve average nDCG@10 scores of 59.30 and 59.86, respectively. In contrast, pairing KaLM-Embedding-

Table 15: Embedding-based retrieval results (nDCG@10) on the six dialogue memory retrieval tasks in LMEB (Zhao et al., 2026b). Results are taken from the online LMEB leaderboard, accessed September 8, 2026.

Models	Size	Avg.	ConvoMem	LoCoMo	LongMemEval	MemBench	REALTALK	TMD
KaLM-Embedding-Gemma3 (Zhao et al., 2026a)	12B	56.66	64.09	42.61	80.97	73.82	44.51	33.95
bge-multilingual-gemma (Chen et al., 2024)	9B	59.30	66.80	56.02	81.66	72.47	48.15	30.69
Nemotron-3-Embed-8B-BF16 (Babakhin et al., 2026)	8B	59.86	69.04	56.51	85.70	76.45	45.70	25.74
Qwen3-Embedding-8B (Zhang et al., 2025c)	8B	51.67	55.60	36.00	77.66	71.10	40.71	28.96
F2LLM-v2-8B (Zhang et al., 2026b)	8B	55.26	66.48	42.29	81.46	70.97	40.56	29.79
SFR-Embedding-Mistral (Meng et al., 2024b)	7B	56.38	67.84	50.05	79.75	70.19	45.27	25.17
gte-Qwen2-7B-instruct (Li et al., 2023)	7B	56.48	64.42	51.55	80.45	72.36	46.25	23.86
NV-Embed-v2 (Lee et al., 2025)	7B	56.02	71.05	51.24	75.76	70.83	48.09	19.13
SFR-Embedding-2_R (Meng et al., 2024a)	7B	55.10	69.42	52.99	67.96	69.99	46.12	24.10
e5-mistral-7b-instruct (Wang et al., 2022; 2024)	7B	55.31	68.63	46.82	78.21	68.99	42.87	26.36

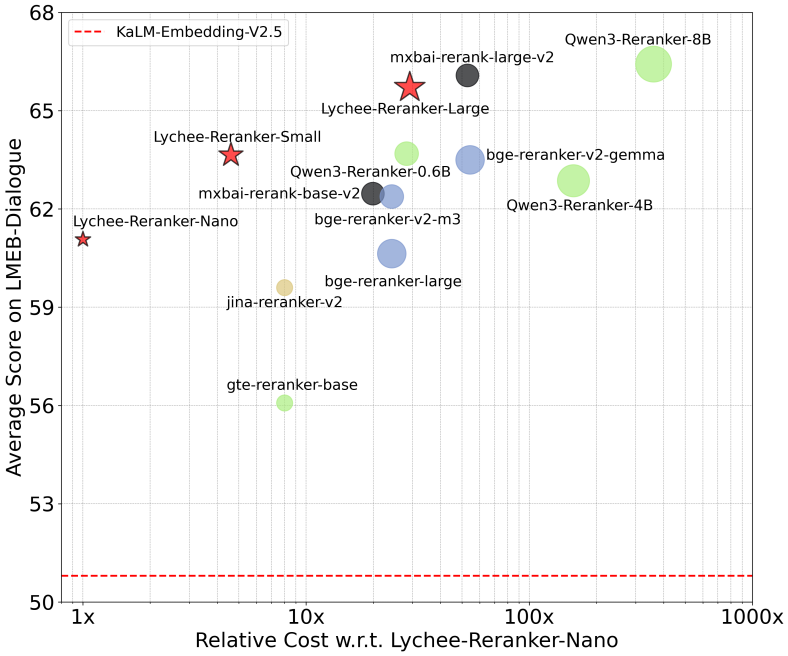
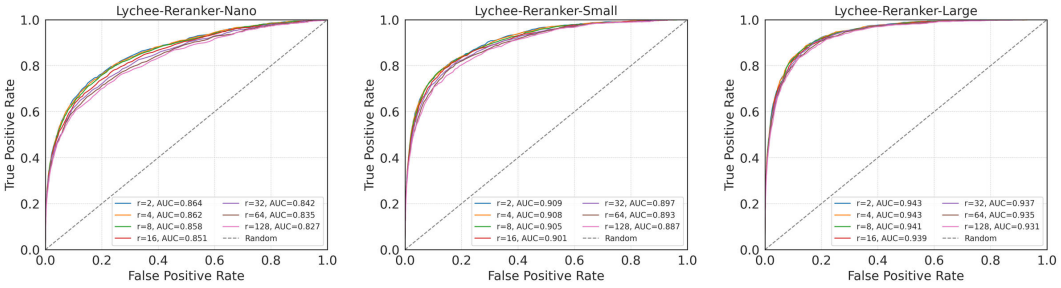


Figure 7: Comparison between the `Lychee-Reranker` series and other reranking models on LMEB-Dialogue (Zhao et al., 2026b) in terms of reranking performance and relative online computation cost. The cost is estimated following the analysis in Appendix A.1; the x-axis is plotted on a logarithmic scale. Marker sizes are proportional to model sizes. The results are taken from Table 14.

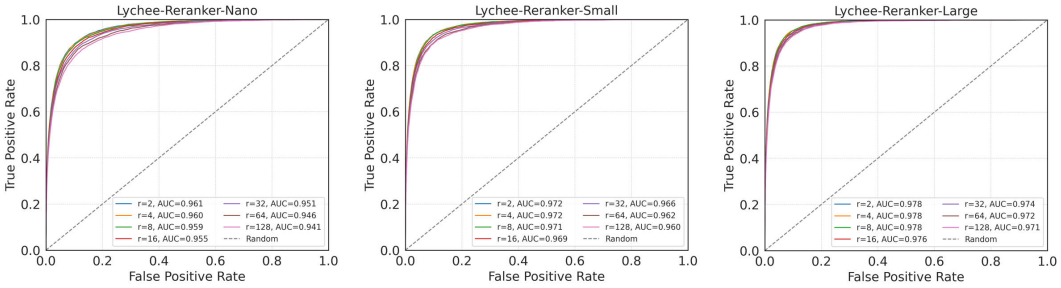
V2.5 (0.5B) with `Lychee-Reranker-Nano` (0.27B activated parameters) achieves an average nDCG@10 of 61.07, outperforming both of these larger embedding models. These results highlight the advantage of the retrieve-then-rerank pipeline over simply increasing embedding model size and illustrate the complementary roles of retrieval and reranking.

H Understanding Reranking Performance Degradation under Compression

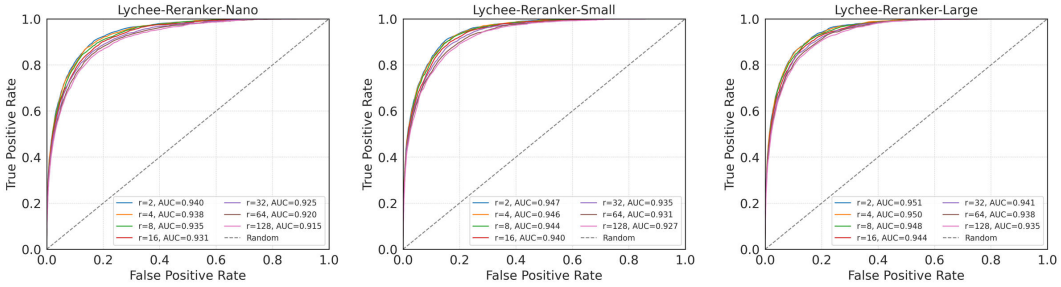
To better understand how compression affects reranking performance, we plot ROC curves on representative tasks using the scores assigned by each reranker to the top-100 candidates for each query (Figure 8). ROC-AUC measures how well a reranker's relevance scores distinguish relevant passages from irrelevant ones without relying on a specific ranking cutoff. As r increases, AUC values generally decrease, with more noticeable declines for smaller models. For example, on FQA, as r increases from 2 to 128, the AUC drops from 0.864 to 0.827 for `Lychee-Reranker-Nano`, from 0.909 to 0.887 for `Lychee-Reranker-Small`, and from 0.943 to 0.931 for `Lychee-Reranker-Large`. This indicates that higher compression ratios cause a greater decline in smaller models' ability to dis-



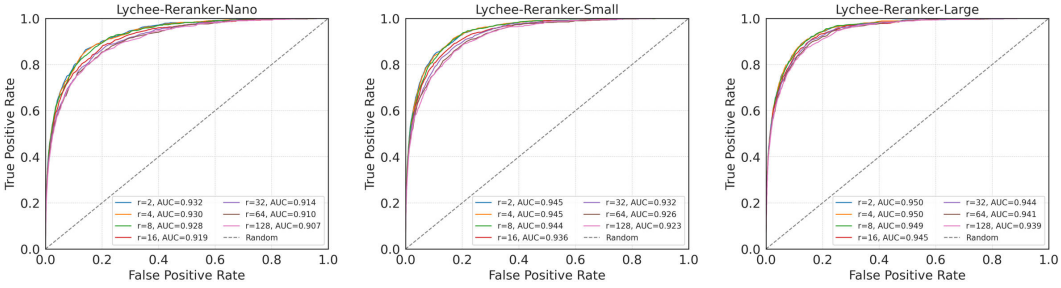
(a) FQA on BEIR.



(b) NQ on BEIR.



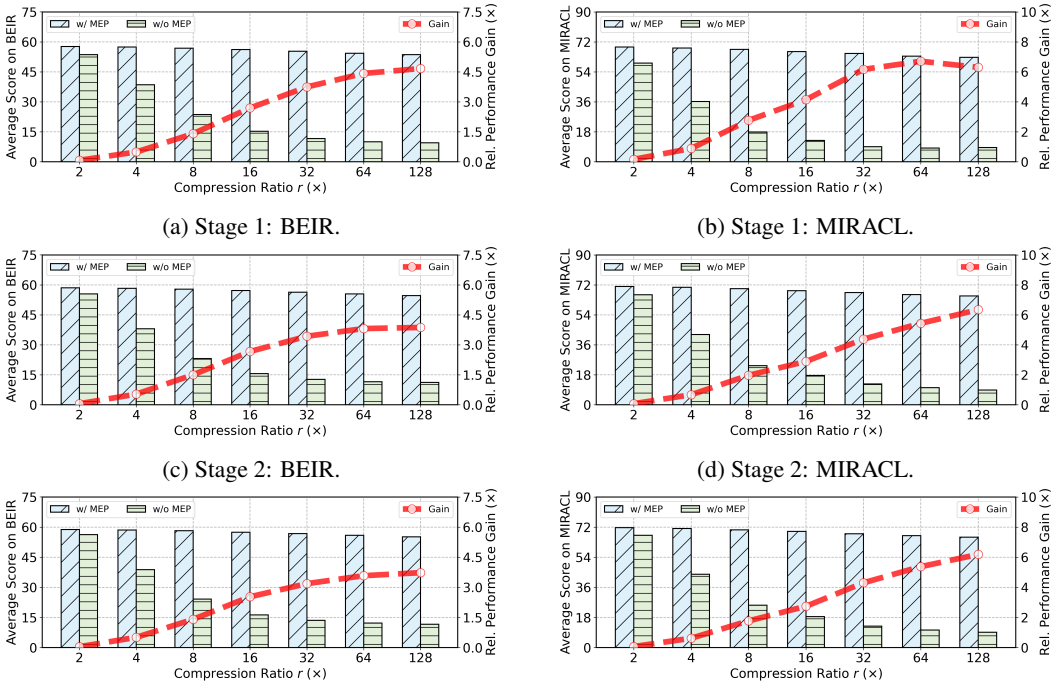
(c) English (en) on MIRACL.



(d) Chinese (zh) on MIRACL.

Figure 8: ROC curves for the `Lychee-Reranker` series on representative tasks from BEIR (FQA and NQ) and MIRACL (English and Chinese) under Matryoshka embedding pooling compression ratios $r \in \{2, 4, 8, 16, 32, 64, 128\}$. Curves are computed over the top-100 candidates scored by each reranker. Higher AUC values indicate better discrimination between relevant and irrelevant passages.

tinguish relevant passages from irrelevant ones. The AUC trends suggest two broad groups of compression ratios: AUC values remain relatively stable from $r = 2$ to $r = 32$, but decline more noticeably at $r = 64$ and $r = 128$. These results are consistent with the findings in §5.3.1: `Lychee-Reranker` largely preserves its discriminative ability under moderate compression, whereas overly aggressive compression leads to noticeable information loss and performance degradation. These findings motivate the development of more effective compression methods that better preserve information for relevance modeling at high compression ratios.



(a) Stage 1: BEIR.

(b) Stage 1: MIRACL.

(c) Stage 2: BEIR.

(d) Stage 2: MIRACL.

(e) Stage 3: BEIR.

(f) Stage 3: MIRACL.

Figure 9: Ablation study on Matryoshka embedding pooling using `Lychee-Reranker-Nano` on BEIR (left) and MIRACL (right). From top to bottom, the rows show results after Stages 1, 2, and 3, respectively. "w/ MEP" denotes training with MEP using compression ratios $\mathcal{R} = \{2, 8, 32, 128\}$, whereas "w/o MEP" denotes training only at $r = 1$ (no compression). Results for "w/ MEP" are taken from Table 6. "Rel." denotes the relative performance gain, computed as $(S_{\text{w/ MEP}} - S_{\text{w/o MEP}})/S_{\text{w/o MEP}}$ at the same compression ratio, where S denotes the average nDCG@10 score.

I Ablation Study on Matryoshka Embedding Pooling

Figure 9 compares the performance of `Lychee-Reranker-Nano` trained with and without MEP on BEIR and MIRACL across different compression ratios. Results are reported after each stage of the multi-stage training pipeline described in §4.2. For Stage 2, both settings use checkpoints trained with a LoRA rank of 64 and an alpha of 32. Except for whether MEP is used during training, the two settings use identical model initialization, training data, number of training steps, and all other hyperparameters. The relative performance gains from training with MEP generally increase with the compression ratio, reaching severalfold improvements over training without MEP at high compression ratios. Specifically, the model trained without MEP maintains reasonable performance at the low compression ratio $r = 2$, but still substantially underperforms the model trained with MEP when both are evaluated at $r = 2$. The performance of the model trained without MEP plummets when the compression ratio increases to $r = 4$. For example, on BEIR after Stage 1, increasing the compression ratio from $r = 2$ to $r = 4$ reduces the average nDCG@10 of the model trained without MEP from 53.58 to 38.57, a decrease of 15.01 points. In contrast, the performance of the model trained with MEP declines very slowly as the compression ratio increases. Overall, these results indicate that training with MEP not only substantially improves the model's robustness to compression but also yields higher peak performance at low compression ratios.

Table 16: `Lychee-Reranker` training datasets. "en", "zh", and "ml" denote English, Chinese, and multilingual data, respectively. For BGE-M3-data, we primarily use samples with lengths ranging from 0 to 500.

Source	Language	Size	URL
KaLM embedding fine-tuning data (retrieval subset)			
AdvertiseGen (Shao et al., 2019)	zh	17,526	https://huggingface.co/datasets/shibing624/AdvertiseGen
CHEF (Hu et al., 2022)	zh	4,824	https://github.com/THU-BPM/CHEF
CodeFeedback (Zheng et al., 2024)	en	49,090	https://huggingface.co/datasets/m-a-p/CodeFeedback-Filtered-Instruction
DRCD (Shao et al., 2018)	zh	4,714	https://huggingface.co/datasets/voidful/DRCD
Expertqa (Malaviya et al., 2024)	en	1,252	https://github.com/chaitanyamalaviya/ExpertQA
GooAQ (Khashabi et al., 2021)	en	49,833	https://github.com/allenai/gooaq
LCSTS (Hu et al., 2015)	zh	19,535	https://huggingface.co/datasets/hugcyp/LCSTS
MEDI2BGE (Xiao et al., 2024a)	en	71,790	https://hf.co/datasets/GritLM/MEDI2BGE
Multi-CPR (Long et al., 2022)	zh	234,587	https://github.com/Alibaba-NLP/Multi-CPR
OpenOrca (Lian et al., 2023)	en	38,623	https://huggingface.co/datasets/Open-Orca/OpenOrca
PAQ (Lewis et al., 2021)	en	49,849	https://huggingface.co/datasets/sentence-transformers/paq
PubMedQA (Jin et al., 2019)	en	79,954	https://huggingface.co/datasets/qiaojin/PubMedQA
RefGPT (Yang et al., 2023)	zh	49,896	https://github.com/sufengniu/RefGPT
SearchQA (Dunn et al., 2017)	en	9,988	https://huggingface.co/datasets/kyunghyuncho/search_qa
T2Ranking (Xie et al., 2023)	zh	188,606	https://huggingface.co/datasets/THUIR/T2Ranking
THUCNews (Guo et al., 2016)	zh	19,288	https://huggingface.co/datasets/SirlyDreamer/THUCNews
UMETRIP-QA (BDBC-KG-NLP, 2022)	zh	2,537	https://aistudio.baidu.com/datasetdetail/149933
WebCPM (Qin et al., 2023)	zh	1,602	https://github.com/thunlp/WebCPM
arxiv_qa (TitanMLData, 2023)	en	17,927	https://huggingface.co/datasets/TitanMLData/arxiv_qa
aya_dataset (Singh et al., 2024)	ml	26,292	https://huggingface.co/datasets/CohereLabs/aya_dataset
cCOVID-News	zh	4,727	https://www.datafountain.cn/competitions/424/datasets
cMedQA-V2.0 (Zhang et al., 2018)	zh	88,109	https://huggingface.co/datasets/wangrongsheng/cMedQA-V2.0
ccnews (Hamborg et al., 2017)	en	28,246	https://edoc.hu-berlin.de/items/ad915f2d-bb2c-4abd-887f-3d50bd3f2516
CMRC 2018 (Cui et al., 2019)	zh	9,753	https://huggingface.co/datasets/erhwenkuo/squad-cmrc2018-zhtw
cord19_trec-covid (Voorhees et al., 2021; Wang et al., 2020)	en	48,517	https://huggingface.co/datasets/irds/cord19_trec-covid
CQADupstack (Hoogeveen et al., 2015)	en	7,355	http://nlp.cis.unimelb.edu.au/resources/cqadupstack/
csl (Li et al., 2022)	zh	19,945	https://huggingface.co/datasets/neuclir/csl
dbpedia-entity (Hasibi et al., 2017)	en	96,792	https://github.com/iai-group/DBpedia-Entity/
dureader (He et al., 2018)	zh	79,229	https://huggingface.co/datasets/sentence-transformers/dureader
dureader-checklist (Tang et al., 2021)	zh	97,764	https://huggingface.co/datasets/luozhouyang/dureader
esci (Reddy et al., 2022)	en	26,043	https://huggingface.co/datasets/tasksource/esci
fever (Thorne et al., 2018)	en	87,216	https://huggingface.co/datasets/maxzoech/fever
fiqa (Maia et al., 2018)	en	4,689	https://huggingface.co/datasets/irds/beir_fiqa_train
hotpot_qa (Yang et al., 2018)	en	150,153	https://huggingface.co/datasets/hotpotqa/hotpot_qa
law-gpt (Liu et al., 2023)	zh	500	https://huggingface.co/datasets/sentence-transformers/law-gpt
lawzhidao (Ustinian, 2020)	zh	6,784	https://www.heywhale.com/mw/dataset/5e953ca8e7ec38002d02fca7/content
lima-chinese (Zhou et al., 2023)	zh	1,991	https://huggingface.co/datasets/paralym/lima-chinese
miracl (Zhang et al., 2023)	ml	128,174	https://huggingface.co/datasets/sentence-transformers/miracl
mmarco-chinese (Bonifacio et al., 2021)	zh	379,870	https://huggingface.co/datasets/unicamp-dl/mmarco
mr-tydi (Zhang et al., 2021)	ml	96,124	https://huggingface.co/datasets/castorini/mr-tydi
msmarco-passage (Bajaj et al., 2016)	en	174,190	https://huggingface.co/datasets/Tevatron/msmarco-passage
msmarco-v2 (Bajaj et al., 2016)	en	258,617	https://huggingface.co/datasets/mteb/msmarco-v2
nfcorpus (Boteva et al., 2016)	en	10,471	https://huggingface.co/datasets/BeIR/nfcorpus-generated-queries
rag-dataset-12000 (neural-bridge, 2023)	en	9,272	https://huggingface.co/datasets/neural-bridge/rag-dataset-12000
retrieval_data_llm_infgrad (infgrad, 2024)	zh	32,551	https://huggingface.co/datasets/infgrad/retrieval_data_llm
scifact (Wadden et al., 2020)	en	794	https://huggingface.co/datasets/Tevatron/scifact
squad_v2 (Rajpurkar et al., 2018; 2016)	en	125,816	https://huggingface.co/datasets/rajpurkar/squad_v2
triviaqa (Joshi et al., 2017)	en	44,442	https://huggingface.co/datasets/multi-train/emb-triviaqa-train
webgpt_comparisons (Nakano et al., 2021)	en	18,924	https://huggingface.co/datasets/openai/webgpt_comparisons
webqa (Li et al., 2016)	zh	4,988	https://huggingface.co/datasets/suolyer/webqa
wikipedia-nq (Kwiatkowski et al., 2019)	en	56,377	https://huggingface.co/datasets/Tevatron/wikipedia-nq
yahoo-answers (Sentence Transformers, 2024)	en	21,724	https://huggingface.co/datasets/sentence-transformers/yahoo-answers
quora-question-pairs (Iyer et al., 2017)	en	83,098	https://huggingface.co/datasets/AlekseyKorshuk/quora-question-pairs
arguana (Wachsmuth et al., 2018)	en	4,065	https://zenodo.org/records/3973258
mldr (Chen et al., 2024)	ml	41,434	https://huggingface.co/datasets/sentence-transformers/mldr
BGE-M3-data			
MSMARCO (Bajaj et al., 2016)	en	476,968	https://huggingface.co/datasets/Shitao/bge-m3-data
NQ (Kwiatkowski et al., 2019)	en	58,554	https://huggingface.co/datasets/Shitao/bge-m3-data
HotpotQA (Yang et al., 2018)	en	84,228	https://huggingface.co/datasets/Shitao/bge-m3-data
Trivia (Joshi et al., 2017)	en	60,283	https://huggingface.co/datasets/Shitao/bge-m3-data

Table 17: Test-time compute scaling results on MIRACL (Zhang et al., 2023), measured by nDCG@10 across all 18 languages. "Nano", "Small", and "Large" denote `Lychee-Reranker-Nano`, `Lychee-Reranker-Small`, and `Lychee-Reranker-Large`, respectively. The reranking and post-ranking columns specify the number of documents (d) and compression ratios (r) used at each stage. Cost denotes the estimated relative online computation cost, with `Lychee-Reranker-Nano` at $r = 4$ normalized to 1.0x. All other evaluation settings follow those in Table 3.

Models	Setting	Cost	Avg.	Test-Time Compute		MIRACL																	
				Reranking	Post-ranking	ar	bn	de	en	es	fa	fi	fr	hi	id	ja	ko	ru	sw	te	th	yo	zh
Nano	(a)	1.89x	**71.63**	100d $\xrightarrow{r=2}$ 100d	N/A	**79.01**	83.20	58.56	**65.74**	**62.47**	**61.58**	**79.82**	**62.32**	**62.47**	**59.32**	**74.32**	**73.99**	**74.04**	**76.10**	**86.06**	82.41	82.73	**65.24**
	(b)	1.0x	71.11	100d $\xrightarrow{r=4}$ 100d	N/A	78.76	83.14	57.45	65.00	61.54	60.81	79.30	61.03	61.95	59.04	74.25	72.76	73.77	75.74	85.78	82.47	**83.00**	64.14
	(c)	0.6x	71.35	100d $\xrightarrow{r=32}$ 100d	20d $\xrightarrow{r=2}$ 20d	78.83	83.39	**58.59**	65.25	61.90	61.31	79.68	61.97	62.37	59.11	74.31	73.64	73.80	75.78	86.02	**82.55**	81.81	64.02
	(d)	0.54x	71.18	100d $\xrightarrow{r=64}$ 100d	20d $\xrightarrow{r=2}$ 20d	78.82	**83.45**	57.46	65.14	61.43	61.29	79.44	62.23	62.21	58.82	73.97	73.17	73.75	75.85	85.92	**82.55**	81.64	64.17
Small	(a)	8.65x	**74.34**	100d $\xrightarrow{r=2}$ 100d	N/A	**81.29**	85.66	64.79	**66.85**	**64.09**	63.55	82.26	**65.47**	66.02	59.46	**79.28**	**73.40**	**78.19**	80.05	87.48	85.50	**86.44**	**68.37**
	(b)	4.58x	74.06	100d $\xrightarrow{r=4}$ 100d	N/A	80.92	**85.69**	64.22	66.45	63.53	62.95	82.09	65.29	65.65	**59.48**	78.91	72.84	78.13	80.12	87.39	85.53	86.34	67.56
	(c)	2.75x	74.18	100d $\xrightarrow{r=32}$ 100d	20d $\xrightarrow{r=2}$ 20d	81.22	85.48	**65.21**	66.59	63.54	63.71	**82.27**	64.81	65.94	59.40	79.05	73.04	78.08	**80.16**	**87.59**	**85.61**	86.03	67.43
	(d)	2.49x	74.08	100d $\xrightarrow{r=64}$ 100d	20d $\xrightarrow{r=2}$ 20d	81.16	85.38	65.20	66.35	63.46	**63.79**	82.06	64.90	**66.32**	59.17	78.99	72.80	77.99	79.84	87.54	85.46	86.03	67.00
Large	(a)	55.05x	**75.12**	100d $\xrightarrow{r=2}$ 100d	N/A	82.56	85.96	**66.70**	67.64	63.97	**65.24**	83.37	65.90	66.45	**59.47**	80.37	**75.11**	**79.71**	**81.12**	**87.96**	85.89	**87.26**	67.45
	(b)	29.14x	74.92	100d $\xrightarrow{r=4}$ 100d	N/A	82.56	85.80	65.75	**67.65**	63.76	65.09	**83.38**	**65.99**	66.30	59.17	80.27	74.45	79.51	80.56	87.53	**85.96**	87.13	**67.71**
	(c)	17.49x	75.05	100d $\xrightarrow{r=32}$ 100d	20d $\xrightarrow{r=2}$ 20d	82.53	**86.10**	66.54	67.45	**63.98**	65.09	83.30	65.88	66.57	59.45	**80.48**	74.64	79.66	80.94	87.91	85.94	86.89	67.63
	(d)	15.87x	75.01	100d $\xrightarrow{r=64}$ 100d	20d $\xrightarrow{r=2}$ 20d	**82.59**	85.93	66.54	67.31	63.90	65.15	83.22	65.90	**66.60**	59.31	**80.48**	74.70	79.47	80.85	87.91	85.89	86.88	67.65

Table 18: BEIR (Thakur et al., 2021) reranking results under different Matryoshka embedding pooling compression ratios r, measured by nDCG@10. The 13 task abbreviations follow Table 11. "CR" denotes the compression ratio, where a larger value indicates more compressed passage representations. Cost denotes the estimated relative serving computation cost derived from the time complexity analysis in Appendix A.1. We use $|q| = 32$, $n = 1024$, $K = 1$, and the corresponding compression ratio r; obtain L and d from Table 1; and normalize the cost of `Nano` at $r = 128$ to 1.0x. Avg. denotes the average performance across the 13 tasks.

Model	CR	Cost	Avg.	AA	CF	CQA	DB	FV	FQA	HQA	MS	NF	NQ	QR	SD	SF
Lychee-Reranker-Nano	2	13.6x	58.75	63.87	43.23	41.08	48.92	93.13	47.25	82.80	46.17	38.42	69.57	90.11	21.04	78.17
	4	7.2x	58.54	63.97	43.25	40.84	47.93	92.95	47.09	82.58	45.64	38.72	69.30	89.92	20.80	78.03
	8	4.0x	58.17	63.46	43.17	40.52	47.51	92.90	46.60	82.13	45.42	38.13	68.66	89.77	20.64	77.30
	16	2.4x	57.47	62.63	42.35	40.21	46.40	92.85	45.27	81.29	44.36	37.72	67.61	89.68	20.30	76.50
	32	1.6x	56.74	61.31	42.16	39.36	44.80	92.45	44.44	80.17	43.82	37.28	66.70	89.64	20.12	75.35
	64	1.2x	55.88	60.13	41.38	38.73	44.07	91.74	43.05	78.63	43.07	36.91	64.88	89.62	19.71	74.53
	128	1.0x	55.11	58.42	40.96	38.28	43.11	91.23	42.22	77.26	42.90	36.23	63.73	89.62	19.32	73.16
Lychee-Reranker-Small	2	62.3x	61.17	68.79	45.98	47.35	50.30	93.50	54.08	83.74	47.33	39.95	73.09	90.85	22.05	78.26
	4	33.0x	61.07	69.43	45.63	47.06	49.77	93.36	53.62	83.55	46.90	39.89	73.14	90.77	22.15	78.72
	8	18.3x	60.74	69.08	45.20	46.72	49.25	93.28	53.20	83.21	46.82	39.62	72.59	90.55	21.95	78.16
	16	11.0x	60.25	68.52	44.78	46.07	48.55	93.08	52.63	82.73	46.13	39.31	71.76	90.49	21.67	77.49
	32	7.3x	59.54	67.26	44.58	45.15	46.60	92.80	51.73	81.92	45.42	39.01	70.50	90.39	21.34	77.29
	64	5.5x	58.84	66.76	44.15	44.13	45.82	92.30	50.73	80.71	44.54	38.71	69.46	90.39	21.01	76.20
	128	4.6x	58.20	65.88	44.07	43.37	44.88	91.65	50.00	79.70	44.53	37.86	68.54	90.39	20.78	74.98
Lychee-Reranker-Large	2	396.3x	63.69	72.97	47.80	52.76	53.29	94.26	59.31	84.39	47.72	42.26	76.06	90.76	24.91	81.49
	4	209.8x	63.53	72.90	47.94	52.36	52.73	94.22	59.15	84.23	47.38	42.35	75.61	90.72	24.89	81.42
	8	116.6x	63.15	71.88	47.70	52.02	52.56	94.16	58.71	83.98	47.34	42.09	75.31	90.55	24.89	79.78
	16	69.9x	62.70	71.48	47.63	51.43	51.74	93.92	58.57	83.51	46.73	41.50	74.16	90.51	24.61	79.37
	32	46.6x	62.18	69.97	47.59	50.64	50.95	93.56	57.31	82.94	46.18	41.39	73.28	90.45	24.54	79.49
	64	35.0x	61.60	69.81	47.22	49.76	50.17	93.22	56.92	82.04	45.70	40.81	72.52	90.46	24.17	77.95
	128	29.1x	61.23	68.76	46.96	49.37	49.64	92.82	56.34	81.33	45.72	40.69	71.47	90.46	24.28	78.23

Table 19: MIRACL (Zhang et al., 2023) reranking results under different Matryoshka embedding pooling compression ratios r, measured by nDCG@10. All 18 language abbreviations follow Table 12. "CR" denotes the compression ratio, where a larger value indicates more compressed passage representations. Cost denotes the estimated relative serving computation cost derived from the time complexity analysis in Appendix A.1. We use $|q| = 32$, $n = 1024$, $K = 1$, and the corresponding compression ratio r; obtain L and d from Table 1; and normalize the cost of `Nano` at $r = 128$ to 1.0x. Avg. denotes the average performance across the 18 languages.

Model	CR	Cost	Avg.	ar	bn	de	en	es	fa	fi	fr	hi	id	ja	ko	ru	sw	te	th	yo	zh
	2	13.6x	71.63	79.01	83.20	58.56	65.74	62.47	61.58	79.82	62.32	62.47	59.32	74.32	73.99	74.04	76.10	86.06	82.41	82.73	65.24
	4	7.2x	71.11	78.76	83.14	57.45	65.00	61.54	60.81	79.30	61.03	61.95	59.04	74.25	72.76	73.77	75.74	85.78	82.47	83.00	64.14
	8	4.0x	70.38	78.23	82.94	57.16	63.88	60.98	59.52	78.61	60.92	61.33	58.29	74.09	71.80	72.50	74.78	85.40	81.82	80.78	63.79
`Lychee-Reranker-N`	16	2.4x	69.38	77.56	81.76	56.04	62.48	59.21	58.54	77.65	60.01	60.77	57.13	72.82	71.62	71.11	73.81	84.80	81.58	79.86	62.07
	32	1.6x	68.02	76.56	80.08	52.80	61.04	57.67	56.88	76.04	59.22	59.78	56.40	71.81	69.69	70.07	72.68	84.72	80.98	77.50	60.53
	64	1.2x	66.85	75.71	77.93	52.47	59.60	56.36	55.60	74.94	59.14	58.30	55.85	70.10	68.17	68.37	71.63	82.99	79.92	76.88	59.40
	128	1.0x	66.01	74.63	76.67	51.65	59.31	55.75	54.98	73.22	59.07	57.41	54.78	68.44	66.38	67.47	71.29	83.05	78.70	75.65	59.75
	2	62.3x	74.34	81.29	85.66	64.79	66.85	64.09	63.55	82.26	65.47	66.02	59.46	79.28	73.40	78.19	80.05	87.48	85.50	86.44	68.37
	4	33.0x	74.06	80.92	85.69	64.22	66.45	63.53	62.95	82.09	65.29	65.65	59.48	78.91	72.84	78.13	80.12	87.39	85.53	86.34	67.56
	8	18.3x	73.61	80.46	85.48	63.64	65.86	62.78	63.17	81.62	64.39	64.75	59.06	78.96	72.08	77.90	79.21	87.27	85.43	85.93	66.94
`Lychee-Reranker-S`	16	11.0x	72.60	80.20	85.02	62.55	64.69	61.45	62.06	81.46	64.00	63.45	58.30	77.79	71.88	76.36	77.82	87.05	85.22	82.56	65.03
	32	7.3x	71.81	79.70	83.90	62.24	62.90	60.76	61.54	81.05	63.01	62.92	57.44	76.90	71.27	75.55	77.74	86.59	84.39	80.80	63.92
	64	5.5x	70.82	79.05	82.59	61.91	61.82	58.77	61.13	79.51	61.86	61.92	56.61	76.00	70.21	74.59	77.27	86.23	83.46	79.78	62.04
	128	4.6x	70.16	78.32	81.26	60.87	61.19	58.50	60.87	78.96	61.81	60.53	56.86	75.64	68.75	73.62	76.97	84.92	82.83	79.09	61.84
	2	396.3x	75.12	82.56	85.96	66.70	67.64	63.97	65.24	83.37	65.90	66.45	59.47	80.37	75.11	79.71	81.12	87.96	85.89	87.26	67.45
	4	209.8x	74.92	82.56	85.80	65.75	67.65	63.76	65.09	83.38	65.99	66.30	59.17	80.27	74.45	79.51	80.56	87.53	85.96	87.13	67.71
	8	116.6x	74.50	82.34	85.37	65.45	67.04	63.07	64.18	82.95	65.12	65.50	58.79	79.87	73.70	78.93	80.41	87.72	86.01	87.34	67.23
`Lychee-Reranker-L`	16	69.9x	73.85	82.04	85.29	64.42	66.12	62.27	63.86	82.68	63.72	64.96	58.32	79.53	72.59	78.24	79.60	87.26	85.79	85.68	66.94
	32	46.6x	73.26	81.83	84.31	63.14	64.97	61.02	63.17	82.18	64.53	63.73	58.06	78.62	71.97	77.88	79.71	86.70	85.15	85.63	66.10
	64	35.0x	72.46	81.55	82.56	63.11	63.83	60.14	62.81	81.78	62.42	62.77	56.77	77.60	70.83	76.51	79.43	86.61	84.83	85.68	65.09
	128	29.1x	71.97	80.90	82.70	62.57	63.37	59.54	63.02	81.14	62.32	61.69	56.37	76.48	70.59	76.01	78.87	86.43	84.02	84.75	64.73

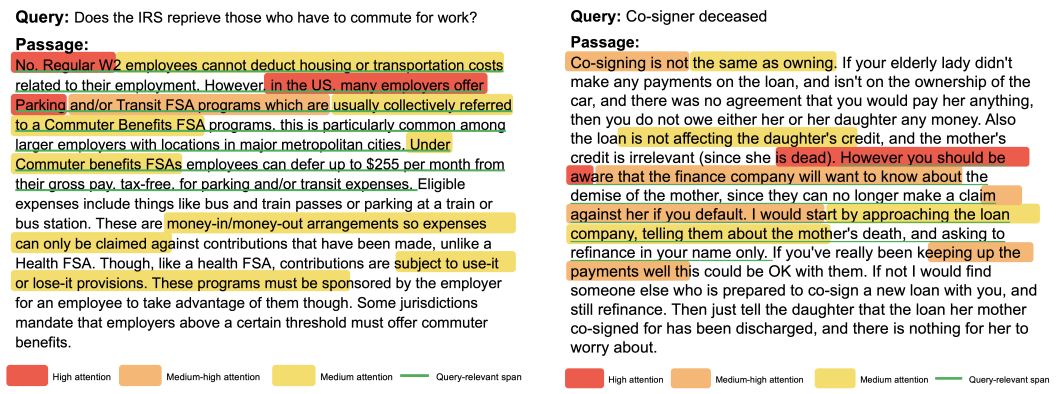
J CASE STUDY

To examine whether the **compact passage representations** produced by Matryoshka embedding pooling retain sufficient information for **expressive relevance modeling** (§1), we conduct a qualitative analysis of two query–positive-passage pairs from the FiQA test set in BEIR (Thakur et al., 2021). We use `Lychee-Reranker-Small` with a compression ratio of $r = 8$, such that each position in the compressed passage representation summarizes up to eight consecutive passage tokens. This allows us to assess both whether query-relevant information is preserved after MEP and whether the decoder can recover and use it for relevance modeling.

Visualization setup. We extract decoder-to-encoder attention at the first prediction position, which corresponds to the last non-padding token in the decoder input. The hidden state at this position produces the logits for the `yes` and `no` labels, which determine the relevance score, as illustrated in Figure 2. We first average attention weights across heads within each decoder layer. For each compressed passage position, we then select and average the six largest layer-wise attention weights. The resulting score is mapped back to the corresponding chunk in the original passage. This aggregation highlights strong attention signals while reducing layer-specific variation.

Qualitative observations. Figure 10 uses color intensity to indicate attention strength and green underlines to mark query-relevant spans. In the first example (Figure 10a), the query concerns tax relief for commuting expenses. The strongest attention falls on chunks stating that regular W-2 employees cannot deduct transportation costs, as well as those describing employer-sponsored parking and transit FSA programs as a tax-advantaged alternative. In the second example (Figure 10b), which concerns a deceased co-signer, the strongest attention falls on chunks describing the co-signer's death and the need to notify the finance company. The highlighted evidence also includes the recommendation to refinance the loan. Together, these regions capture the key facts and practical guidance needed to establish passage relevance.

Key insights. Qualitatively, these examples illustrate the two advantages highlighted in §1: *compactness* and *expressiveness*. MEP compresses the precomputed passage representations by a factor of eight, substantially reducing storage requirements while retaining query-relevant evidence. The decoder can then selectively attend to and integrate this evidence through merged self-attention and cross-attention, supporting expressive relevance modeling. Overall, these visualizations provide qualitative evidence that compact passage representations retain information accessible to the decoder.



(a) IRS Commuter Benefit (b) Co-signer Liability After Death

Figure 10: Decoder-to-encoder attention in `Lychee-Reranker-Small` at a compression ratio of $r = 8$ for two query–positive-passage pairs from the FiQA test set in BEIR (Thakur et al., 2021). Attention weights are extracted from the first prediction position (*i.e.,* the last non-padding token in the decoder input). We first average attention weights across heads within each decoder layer. For each compressed passage position, we then average the six largest layer-wise attention weights and map the resulting score to the corresponding chunk in the original passage. Colors indicate attention strength, and green underlines mark query-relevant spans. The resulting high-attention chunks overlap with the query-relevant spans.