# EmbeddingRWKV: State-Centric Retrieval with Reusable States

**Anonymous ACL submission**


## Abstract

Current Retrieval-Augmented Generation (RAG) systems typically employ a traditional two-stage pipeline: an embedding model for initial retrieval followed by a reranker for refinement. However, this paradigm suffers from significant inefficiency due to the lack of shared information between stages, leading to substantial redundant computation. To address this limitation, we propose **State-Centric Retrieval**, a unified retrieval paradigm that utilizes "states" as a bridge to connect embedding models and rerankers. First, we perform state representation learning by fine-tuning an RWKV-based LLM, transforming it into **EmbeddingRWKV**, a unified model that serves as both an embedding model and a state backbone for extracting compact, reusable states. Building upon these reusable states, we further design a state-based reranker to fully leverage precomputed information. During reranking, the model processes only query tokens, decoupling inference cost from document length and yielding a 5.4×–44.8× speedup. Furthermore, we observe that retaining all intermediate layer states is unnecessary; with a uniform layer selection strategy, our model maintains 98.62% of full-model performance using only 25% of the layers. Extensive experiments demonstrate that State-Centric Retrieval achieves high-quality retrieval and reranking results while significantly enhancing overall system efficiency.

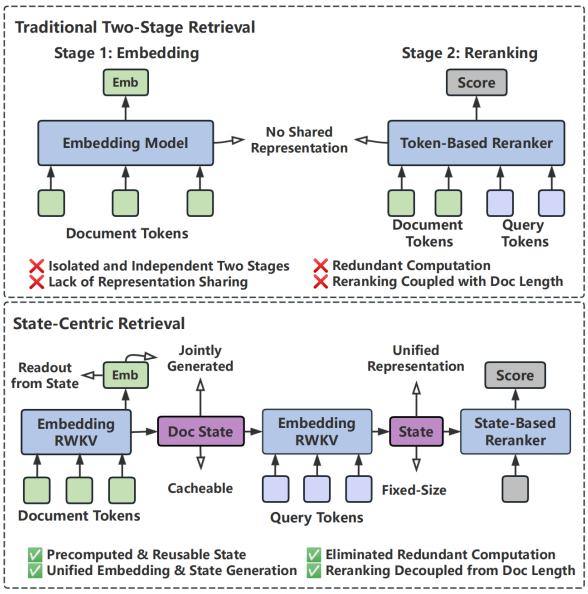




Figure 1: **Traditional vs. State-Centric Retrieval.** (Top) The traditional two-stage retrieval is fundamentally disjointed, suffering from redundant computation as the reranker re-encodes full document tokens. (Bottom) Our **State-Centric Retrieval** unifies the two stages into an efficient system via a shared, reusable state. By jointly generating embeddings and compact states, it enables offline state-based reranking that decouples inference cost from document length, **yielding 5.4×–44.8× speedup**.

## 1 Introduction

Retrieval-Augmented Generation (RAG) has emerged as a common paradigm in modern Large Language Models (LLMs) (Brown et al., 2020; Wu et al., 2025), enabling models to incorporate external knowledge beyond parametric memory (Lewis et al., 2020). Most RAG systems adopt a two-stage retrieval pipeline (Glass et al., 2022): an embedding model is first used to efficiently recall candidate documents, followed by a reranker that performs refinement (Li et al., 2025). Within this pipeline, rerankers play a critical role in improving the quality of retrieved contexts, thereby directly impacting the downstream generation performance of LLMs (Zhao et al., 2025).

Despite its effectiveness, the prevailing two-stage retrieval paradigm suffers from fundamental efficiency limitations arising from both model architecture and pipeline design. *First*, most state-of-the-art embedding models and rerankers are predominantly built on Transformer architectures (Zhang et al., 2025; Günther et al., 2023). While powerful, Transformers incur quadratic com-

putational complexity with respect to sequence length and exhibit rapidly growing memory consumption due to the linearly expanding key–value (KV) cache, making them increasingly inefficient in both computation and storage as context length grow. *Second*, the two-stage pipeline itself introduces additional inefficiency due to the lack of shared representations across stages as shown in Figure 1. As embedding models and rerankers operate as isolated modules, they independently encode the same documents without any information reuse, resulting in substantial redundant computation.

In this work, we propose **State-Centric Retrieval**, a novel retrieval paradigm that improves efficiency from both the architectural and pipeline perspectives. *At the model architecture level*, we adopt RWKV (Peng et al., 2023, 2024, 2025), a linear RNN with matrix-valued hidden states that achieves linear computational complexity with respect to sequence length and constant space complexity. Compared to Transformer-based models, RWKV offers substantially improved efficiency in both computation and space. *At the pipeline level*, we introduce a unified retrieval framework (Fig. 1) that leverages reusable states as a shared representation across the retrieval and reranking stages, enabling information reuse and eliminating redundant computation.

We instantiate this paradigm through two synergistic contributions: *learning reusable states* and *efficiently utilizing these states for reranking*.

First, we observe that directly utilizing RWKV-based LLMs to extract state representations results in significant information loss. To address this, we conduct state representation learning by fine-tuning RWKV-based LLMs, transforming them into unified models that serve as both embedding models and state backbones for extracting effective state representations. Specifically, we employ a single-stage domain-aware curriculum strategy to enhance data efficiency. This approach achieves competitive performance on MTEB benchmarks while requiring only $5\%$ of the training data typically consumed by conventional multi-stage pipelines. This approach outperforms conventional multi-stage training pipelines on MTEB benchmarks while using only $5\%$ of the training data. Importantly, for a sequence of length $T$, the resulting states require only $32/T$ of the memory footprint of a full Transformer's KV cache, making large-scale state caching feasible in practice.

Second, building upon cached states, we design a state-based reranker that fully exploits the precomputed information stored in states. During inference, the reranker processes only query tokens together with cached document states, completely decoupling inference cost from document length. Furthermore, we observe that retaining states from all intermediate layers is unnecessary. With a uniform layer selection strategy, our model preserves $98.62\%$ of full-model performance while using only $25\%$ of the layers, enabling further reductions in memory footprint and inference latency.

In summary, our contributions are as follows:

- We propose **State-Centric Retrieval**, a unified framework that synergizes embedding-based retrieval and reranking through reusable states, effectively eliminating redundant computation.
- We introduce a domain-aware curriculum strategy for state representation learning, achieving competitive performance with only $5\%$ of the standard training data.
- We design a state-based reranker that significantly reduces computational overhead and memory footprint while maintaining high reranking effectiveness.

## 2 Related Work

**Two-Stage Retrieval for Text Embeddings.** Most retrieval-augmented generation systems adopt a two-stage retrieval paradigm, where dense embedding models are used for initial retrieval and rerankers are applied for relevance refinement (Reimers and Gurevych, 2019). Text embedding models have evolved from early BERT-based encoders (Devlin et al., 2019; Hou et al., 2024a) to large language model backbones, with recent systems such as NV-Emb (Lee et al., 2024), E5-Mistral (Wang et al., 2024c), and GTE (Zhang et al., 2024) achieving strong performance on benchmarks like MTEB (Muennighoff et al., 2023a; Xiao et al., 2023). Despite these advances, embedding models and rerankers are typically trained and deployed as independent components. Each stage encodes the same documents separately, without sharing representations or intermediate information, resulting in duplicated encoding and redundant computation across the retrieval pipeline. This design contrasts with our state-centric retrieval framework, which enables representation reuse across retrieval stages.

**Multi-Stage Training Recipes.** State-of-the-art embedding models commonly rely on multi-stage training pipelines that combine large-scale pre-training with successive fine-tuning stages. These pipelines often include supervised fine-tuning and hard-negative mining to progressively refine retrieval-specific representations. For instance, BGE (Chen et al., 2024a) employs RetroMAE pre-training followed by contrastive learning and instruction tuning with hard negatives. Jina Embeddings (Sturua et al., 2024) adopts a two-stage data-centric approach that combines large-scale data filtering with triplet-based training. Arctic-Embed (Yu et al., 2024) follows a coarse-to-fine strategy, transitioning from billion-scale pre-training to high-precision fine-tuning on curated datasets. While effective, existing multi-stage pipelines depend on massive datasets and complex training workflows, which limit data efficiency. By contrast, our domain-aware curriculum strategy consolidates training into a single stage with reduced data usage, enabling a simpler and more data-efficient training process.

**From Transformers to Linear RNNs.** The majority of pioneering embedding models and rerankers are built on Transformer architectures (Zhang et al., 2025; Vera et al., 2025). However, Transformers face inherent scalability issues: *quadratic computational complexity* and a *linearly growing KV cache* memory footprint (Vaswani et al., 2017). These limitations make long-context retrieval and large-scale reranking prohibitively expensive. To address this, our work leverages RWKV, a member of the *Linear RNNs with Matrix-valued States* family. Unlike Transformers, RWKV achieves *linear computational complexity* and *constant spatial complexity*, offering a more efficient architectural backbone for dense retrieval without sacrificing the expressive power required for complex semantic matching.

**Linear RNNs with Matrix-valued States.** While linear RNNs like Mamba (Gu and Dao, 2023) and RWKV-4 (Peng et al., 2023) reduce complexity to $O(N)$, they often suffer from a *state capacity gap* due to compressing context into fixed-size vectors. To bridge this gap, recent advancements have focused on scaling hidden states into matrices. Since the explicit transition to matrix-valued states in RWKV-5 and RWKV-6 (Peng et al., 2024), the architecture has demonstrated an enhanced ability to maintain rich historical context. Most recently, RWKV-7 (Peng et al., 2025) further refined this with *dynamic state evolution* mechanisms. By adopting this architecture, our framework enables the caching of high-capacity states that are both compact and highly representative across the retrieval pipeline.

## 3 State-Centric Retrieval

State-Centric Retrieval takes states as the primary representation. A precise definition of the state is therefore necessary, as it forms the basis of the entire framework and directly affects model architecture and pipeline design.

### 3.1 Background: Matrix-valued States

In recurrent models, a *state* summarizes past inputs and serves as the carrier of contextual information. Conventional RNNs (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) represent the state as a fixed-length vector, which limits the amount of information that can be preserved over long sequences.

*Matrix-valued states* generalize this formulation by representing the recurrent state as a matrix rather than a vector, allowing the model to maintain a richer and more structured memory. Formally, given a matrix-valued state $S_t$ and the current input $x_t$, the recurrent update takes the form (Hou et al., 2025):

$$S_t = S_{t-1}W_t + v_t k_t^\top, \tag{1}$$

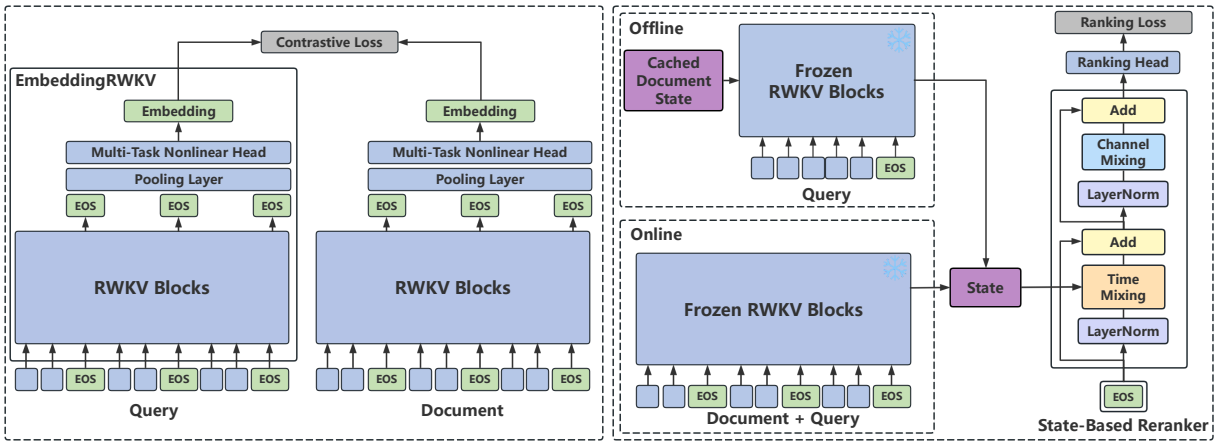
where $S_t \in \mathbb{R}^{d\times d}$, $W_t \in \mathbb{R}^{d\times d}$, and $v_t, k_t \in \mathbb{R}^d$ are column vectors computed from $x_t$. Under this formulation, $S_t$ acts as a dynamic associative memory that incrementally accumulates outer products of keys and values, while selectively retaining or discarding past information through $W_t$.

In the case of RWKV-7 (Peng et al., 2025), the transition matrix $W_t$ is further refined through dynamic evolution, typically expressed by diagonalizing a vector $w_t$ and incorporating a low-rank update:

$$W_t = \mathrm{diag}(w_t) - \hat{\kappa}_t\big(a_t \odot \hat{\kappa}_t\big)^\top \tag{2}$$

where $\hat{\kappa}_t$ and $a_t$ are fast weights (Schlag et al., 2021) that enhance the model's ability to selectively update its internal memory.

Importantly, the matrix-valued state can be updated incrementally in constant memory, without storing the full token history. This property allows the model to process arbitrarily long sequences with linear-time computation and constant space




(a) State Representation Learning (b) State-Based Reranking

Figure 2: (a) **State Representation Learning**: illustrates the process of "Learning the State." (b) **State-Based Reranking**: demonstrates the paradigm of "Utilizing the State" by directly reusing the cached state for inference, thereby avoiding redundant re-computation.

complexity, while retaining strong contextual modeling capacity.

### 3.2 State Representation Learning

State representation learning aims to adapt RWKV-based language models into unified backbones that can simultaneously produce high-quality embeddings and reusable states.

**Model Architecture.** To realize state representation learning, we introduce *EmbeddingRWKV*, whose architecture is illustrated in Fig. 2(a). EmbeddingRWKV consists of RWKV blocks, a pooling layer, and a nonlinear head. Multiple end-of-sequence (EOS) tokens are inserted into the input sequence, and the corresponding output representations are extracted. These representations are averaged by the pooling layer and projected by the nonlinear head to produce the final embedding.

**Training and Data Recipe.** We introduce a *Domain-Aware Curriculum Strategy* for efficient state representation learning. Data samples from the same domain share similar semantic subspaces and thus naturally form hard negatives for contrastive learning. In contrast, randomly mixed batches across heterogeneous domains often produce easy negatives that contribute limited training signal.

To exploit this property, we organize training batches by domain in a distributed manner. Specifically, we first partition the training dataset $\mathcal{D}$ into domain-specific subsets $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$. During distributed training on $N$ GPUs, each GPU processes a micro-batch sampled from a single domain. Different GPUs are assigned different domains at the same training step.

As a result, samples within the same domain naturally form hard negatives within each GPU's local batch, This simple scheduling strategy enables effective hard-negative training without explicit mining.

**Learning Objective.** We train state representations using a contrastive objective, referred to as the loss $L_{\text{state}}$. we adopt a standard InfoNCE loss (Oord et al., 2018) with in-batch negatives. Given a batch of $B$ training instances, the loss is defined as:

$$L_{\text{state}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{e^{s(q_i, d_i^+)/\tau}}{Z_i}, \qquad (3)$$

where $s(\cdot,\cdot)$ denotes the cosine similarity between the embeddings produced by the pooling layer and nonlinear head for the query and document, and $\tau$ is the temperature parameter. The normalization term $Z_i = \sum_{j=1}^{B} e^{s(q_i, d_j)/\tau}$ aggregates the scores across the positive pair and all in-batch negatives.

### 3.3 State-Based Reranking

State-based reranking aims to efficiently utilize learned states for reranking.

**Model Architecture.** The state-based RWKV-Reranker consists of RWKV blocks followed by a ranking head, as illustrated in Fig. 2(b). It takes the matrix-valued state as input and outputs a relevance score. There are two modes for using the state-based RWKV-Reranker: offline mode and online mode.

In offline mode, document states $S_d$ are precomputed and cached. We initialize the EmbeddingRWKV with the cached document state $S_d$ and then feed only the query tokens $\{x_t^q\}_{t=1}^{T_q}$ into the **frozen** EmbeddingRWKV to obtain the updated state $S'$. The updated state $S'$ is fed into the state-based reranker to produce the relevance score.

In online mode, the document and query are processed jointly by the same **frozen** EmbeddingRWKV to produce the state on the fly for reranking. Unless otherwise specified, we freeze all RWKV blocks in EmbeddingRWKV and train only the parameters of the state-based RWKV-Reranker.

**Reranking Training Objective.** For the reranking task, we formulate the relevance prediction as a binary classification problem and optimize the model using the Binary Cross-Entropy (BCE) loss:

$$L_{\text{reranking}} = -\big[y \log s + (1-y)\log(1-s)\big], \quad (4)$$

where $y \in \{0, 1\}$ denotes the ground truth label. $s = s(q, d)$ represents the predicted relevance probability for the query-document pair, derived from the state-based reranker's output.

**Computational Cost Analysis.** Transformer-based rerankers rely on full attention between query and document tokens, resulting in a quadratic computational complexity. This cost grows rapidly with document length and becomes a major bottleneck for long-document reranking.

In contrast, our state-based reranker caches document states and processes only query tokens at inference time. This design decouples inference latency from document length and achieves linear-time computation with respect to the query length.

**Memory Footprint Analysis.** We compare the memory cost of caching representations using Transformer's KV caches and our matrix-valued states. The ratio between their memory footprints is given by

$$\frac{\text{Bytes}_{\text{token}}}{\text{Bytes}_{\text{state}}} = \frac{2LHSTb}{LHS^2b} = \frac{2T}{S}.$$

Here, $L$ denotes the number of layers, $H$ the number of heads, $S$ the per-head state dimension, $T$ the sequence length, and $b$ the number of bytes per element. For RWKV-7 with $S = 64$, this ratio becomes $T/32$. In other words, the KV cache of a Transformer requires $T/32$ times more memory than a single state. While the Transformer memory footprint grows linearly with sequence length $T$, the RWKV state remains constant, making it significantly more memory-efficient for long-document reranking.

# 4 Experiments

## 4.1 Retrieval Experiments and Results

We trained our embedding models (EmbeddingRWKV) using contrastive learning on our self-curated open-source dataset (Details are in Appendix C). We report the performance of our models on the MTEB English benchmark (Muennighoff et al., 2023b).

**EmbeddingRWKV is competitive.** Table 1 shows that EmbeddingRWKV delivers strong retrieval quality across three model scales (0.1B, 0.4B, and 1.4B parameters), with performance improving monotonically as model capacity increases. In the base-size regime, EmbeddingRWKV-0.1B achieves competitive mean scores on MTEB English, matching or surpassing widely adopted open-source baselines of similar scale. At the medium scale, EmbeddingRWKV-0.4B further advances performance and remains competitive with strong Transformer-based embedding models in the same parameter range. When scaled to 1.4B parameters, EmbeddingRWKV-1.4B reaches near-parity with top-performing large embedding models and commercial APIs on the overall MTEB mean, while exhibiting particularly strong results on classification-oriented subsets, suggesting that the learned state representations capture discriminative semantic structure effectively. Overall, these results indicate that RWKV-based backbones can serve as a competitive alternative to Transformer models for dense retrieval, without sacrificing embedding quality.

**Curriculum training is data-efficient.** Table 2 compares our single-stage domain-aware curriculum training with a conventional multi-stage pipeline (pretraining followed by SFT). The multi-stage baseline uses 132.1M training samples in total, while our curriculum uses only 6.7M samples (about 5%). Despite this large reduction in data and a simpler training recipe, the curriculum-trained EmbeddingRWKV achieves higher MTEB English performance than the multi-stage baseline, i.e., Curriculum (6.7M) $>$ Multi-stage (132.1M). These results indicate that curriculum training provides a stronger and more data-efficient way to learn

Table 1: Comparison of embedding models on MTEB (English, v2). $^{\gamma}$Taken from (Lee et al., 2025). For other compared models, the scores are retrieved from MTEB online leaderboard.

| Model | Size | Mean | Classification | Clustering | Reranking | Retrieval | STS | Summ. |
|---|---|---|---|---|---|---|---|---|
| e5-base-v2 (Wang et al., 2024a) | 109M | 62.67 | 75.48 | 45.20 | 45.13 | 49.67 | 80.64 | 34.26 |
| e5-base (Wang et al., 2023) | 109M | 61.79 | 75.20 | 43.77 | 44.86 | 47.70 | 80.70 | 33.38 |
| gte-base (Li et al., 2023a) | 109M | 63.90 | 75.04 | 47.74 | 47.17 | 51.90 | 82.17 | 30.90 |
| jina-embedding-b-en-v1 (Günther et al., 2023) | 110M | 59.76 | 72.06 | 39.95 | 46.98 | 46.38 | 80.00 | 27.15 |
| granite-embedding-english-r2 (Awasthy et al., 2025b) | 149M | 62.84 | 70.71 | 47.20 | 49.10 | 56.43 | 78.12 | 29.31 |
| granite-embedding-125m-english (Awasthy et al., 2025a) | 125M | 62.08 | 68.29 | 47.18 | 49.35 | 55.65 | 77.56 | 29.34 |
| nomic-embed-text-v1.5 (Nussbaum et al., 2024) | 137M | 62.20 | 75.71 | 47.55 | 46.01 | 47.97 | 78.70 | 28.56 |
| snowflake-arctic-embed-m-v1.5 (Merrick, 2024a) | 109M | 61.51 | 70.71 | 44.65 | 45.90 | 58.05 | 72.96 | 29.89 |
| mmlw-roberta-base (Dadas et al., 2023a) | 124M | 61.15 | 77.67 | 46.26 | 46.69 | 40.15 | 81.81 | 30.83 |
| EmbeddingRWKV-0.1B | 144M | 63.06 | 81.01 | 46.21 | 46.07 | 48.06 | 80.30 | 25.13 |
| SearchMap-Preview (Team, 2025) | 435M | 64.08 | 74.97 | 48.47 | 47.90 | 52.21 | 81.56 | 33.25 |
| bilingual-embedding-large (Thakur et al., 2024b) | 559M | 63.77 | 77.17 | 46.53 | 46.25 | 46.86 | 86.00 | 32.95 |
| mmlw-e5-large (Dadas et al., 2024) | 559M | 62.32 | 79.63 | 48.30 | 47.63 | 41.37 | 81.35 | 34.07 |
| gte-large (Li et al., 2023b) | 335M | 64.77 | 75.47 | 48.20 | 47.84 | 53.29 | 83.27 | 32.90 |
| snowflake-arctic-embed-l (Merrick, 2024b) | 335M | 62.04 | 69.59 | 45.70 | 44.71 | 59.04 | 75.42 | 20.38 |
| mmlw-roberta-large (Dadas et al., 2023b) | 434M | 61.80 | 79.66 | 47.89 | 47.56 | 39.69 | 81.20 | 34.97 |
| e5-large-v2 (Wang et al., 2024b) | 335M | 62.79 | 76.44 | 45.23 | 45.72 | 49.31 | 80.67 | 32.34 |
| EmbeddingRWKV-0.4B | 389M | 64.86 | 86.21 | 47.78 | 46.92 | 49.42 | 79.58 | 31.37 |
| sentence-croissant-alpha-v0.4 (Reimers and Gurevych, 2020) | 1B | 57.71 | 70.34 | 43.22 | 44.65 | 43.18 | 75.34 | 28.46 |
| gte-Qwen2-1.5B-instruct (Li et al., 2023c) | 1.5B | 67.20 | 85.84 | 53.54 | 49.25 | 50.25 | 82.51 | 33.94 |
| $^{\gamma}$OpenAI Commercial APIs: text-embedding-3-large | - | 66.43 | - | - | - | - | - | - |
| $^{\gamma}$Cohere Commercial APIs: cohere-embed-multilingual-v3.0 | - | 66.01 | - | - | - | - | - | - |
| EmbeddingRWKV-1.4B | 1.4B | 66.41 | 87.52 | 48.16 | 48.22 | 52.43 | 81.04 | 32.73 |

reusable state representations, which not only improves retrieval but also supplies high-quality states for the subsequent state-based reranking stage.

Table 2: Comparison of retrieval performance between standard Multi-stage training and Single-stage Domain-Aware Curriculum training. The curriculum method achieves higher scores using only ∼5% of the baseline's total training data.

| Model | Stage | Size | Sample Size | MTEB-E | MTEB-C |
|---|---|---|---|---|---|
| *Multi-stage Training* | | | | | |
| EmbeddingRWKV | Pretrain | 144M | 123.2M | 53.85 | 51.47 |
| EmbeddingRWKV | SFT | 144M | 8.9M | 59.60 | 54.37 |
| EmbeddingRWKV | Pretrain | 389M | 123.2M | 54.88 | 53.69 |
| EmbeddingRWKV | SFT | 389M | 8.9M | 60.85 | 55.17 |
| *Single-stage Curriculum Training* | | | | | |
| EmbeddingRWKV | SFT | 144M | 6.7M | **63.06** | **57.12** |
| EmbeddingRWKV | SFT | 389M | 6.7M | **64.86** | **58.68** |

## 4.2 Reranking Experiments and Results

We freeze EmbeddingRWKV and train the state-based RWKV-Reranker with supervised learning. We report reranking results on NanoBEIR, MTEB (English, v2) Retrieval, and MTEB (Chinese, v1) Retrieval.

**State representation learning is necessary.** A natural question is whether we can directly reuse states from a base RWKV language model for reranking, without retrieval-oriented state representation learning. Table 3 shows that this is not the case: initializing the reranker with states from the base LM consistently underperforms using states from EmbeddingRWKV across all evaluated metrics and benchmarks. We attribute this gap to the mismatch in training objectives. Reranking requires fine-grained relevance matching and a retrieval-specific representation geometry, whereas base LMs primarily learn generative language modeling signals that are not aligned with retrieval supervision. Therefore, explicitly learning retrieval-aligned state representations is crucial for effective state-based reranking.

Table 3: Comparison of state-based reranking performance using states generated by RWKV-7 LM and EmbeddingRWKV on NanoBEIR.

| State Backbone | Size | MAP | MRR@10 | NDCG@10 |
|---|---|---|---|---|
| RWKV-7 LM | 190M | 48.58 | 59.37 | 54.78 |
| EmbeddingRWKV | 144M | 58.29 | 69.50 | 63.41 |

**State-based reranking is effective.** A central question in our design is whether the state-based mechanism compromises ranking quality compared to traditional architectures. To investigate this, we trained a standard RWKV-based single-tower reranker under an identical configuration (0.1B parameters) to our proposed model. As shown in Table 4, the state-based reranker demonstrates that our model achieves performance parity with single-tower baseline on NanoBEIR, MTEB-E (eng, v2), and MTEB-C (cmn, v1). Furthermore, scaling the model from 90M to 1.3B parameters yields monotonic quality improvements, validating the favorable scaling behavior of the state-based reranker.

We additionally compare against three

Transformer-based rerankers, including ModernBERT (Warner et al., 2024), EuroBERT (Boizard et al., 2025), and the Jina reranker (Wang et al., 2025). Overall, our state-based reranker exhibits only a small gap to strong single-tower Transformer rerankers, while maintaining performance parity with these Transformer baselines on the NanoBEIR benchmark.

Table 4: State-Based Reranker performance on NanoBEIR (NDCG@10) and MTEB-E (English, v2) Retrieval, MTEB-C (Chinese, v1) Retrieval. Scores are scaled by 100 (higher is better).

| Model | Size | NanoBEIR | MTEB-E-Retr | MTEB-C-Retr |
|---|---|---|---|---|
| EmbeddingRWKV | 0.1B | 59.10 | 48.06 | 57.26 |
| *Traditional Single-Tower Reranker* | | | | |
| RWKV-Reranker | 90M | 64.88 | 49.31 | 58.16 |
| EuroBERT | 210M | 47.38 | - | - |
| ModernBERT | 395M | 64.18 | - | - |
| Jina Reranker | 278M | 70.68 | - | - |
| *State-Based Reranker* | | | | |
| RWKV-Reranker | 90M | 63.41 | 47.78 | 58.42 |
| RWKV-Reranker | 317M | 68.60 | 53.22 | 63.60 |
| RWKV-Reranker | 1.3B | 71.58 | 56.44 | 66.30 |

**State-based reranking is efficient.** As shown in Figure 3, our offline state-based reranker delivers substantial acceleration over a strong Transformer reranker with FlashAttention-2 (Dao, 2023; Li et al., 2025), with speedup increasing from 5.4× at a document length of 512 to 44.8× at 4096, while also using markedly less GPU memory. The key mechanism is that offline reranking processes only query tokens together with cached document states; thus inference cost is independent of document length. This decoupling is reflected by the scaling curves: Transformer-based rerankers exhibit rapidly degrading throughput as document length increases, whereas the RWKV offline reranker remains nearly flat across lengths. Similarly, peak VRAM usage for attention-based baselines grows sharply with longer documents, while our approach stays stable due to constant-size state. Moreover, even in the online setting, the state-based reranker is more memory-efficient than Transformer-based rerankers: its peak VRAM scales more slowly with sequence length, consistent with RWKV's linear memory scaling. Additional efficiency breakdowns across model sizes are provided in Appendix B.1 and Table 8.

**State is redundant.** An important practical question is how many states to cache and which layers to keep, i.e., where the compression boundary lies. Table 5 shows substantial redundancy across layers:

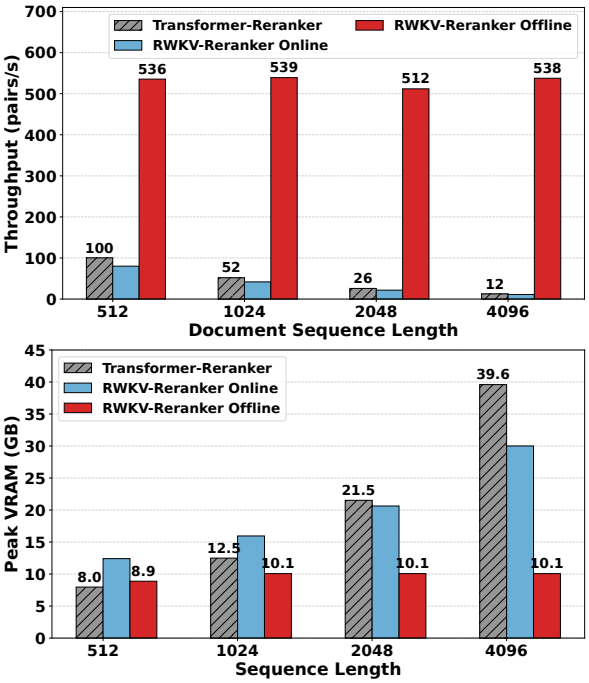



Figure 3: Performance benchmarking across various sequence lengths. Baseline: mxbai-rerank-large-v2 (1.5B) (Li et al., 2025) with FlashAttention-2 vs. Ours: EmbeddingRWKV-1.4B + RWKV-Reranker-1.3B. Top: Throughput comparison (pair/s), where higher is better. Bottom: Peak VRAM usage (GB), where lower is better.

keeping only a small subset of layer states preserves most of the full-depth performance. For example, using 50% of the layers already retains 99.59% of the performance, and even more aggressive configurations with only a few uniformly selected layers can still achieve 97.70%–98.62% relative performance. These results suggest that reranking signals are highly compressible, with overlapping information distributed across model depth rather than concentrated in a single layer.

Importantly, we find that layer choice matters more than layer count. With the same three-layer budget, a top-heavy selection leads to a sharp drop (85.99%), whereas a uniformly distributed selection recovers performance to 98.62%. This indicates that effective reranking relies on combining multi-level signals spanning early to late layers. Based on this observation, we adopt uniform stride layer selection as the default strategy in our state caching design.

**Sharing state fails catastrophically.** Although layer pruning suggests substantial redundancy, compression is not arbitrary. As a counterexample, Table 6 shows that forcing all layers to share a

Table 5: Ablation on state-based reranker layer selection strategies. **Mean** is the average of NanoBEIR, MTEB-E-Retr, and MTEB-C-Retr scores. **Ratio** indicates performance retention relative to the best configuration in the same model group. **Uniform** selection strategies (e.g., 1, 6, 10) significantly outperform contiguous top-layer strategies.

| Strategy | Layers | Layer Index | Size | NanoBEIR | MTEB-E-Retr | MTEB-C-Retr | Mean | Ratio |
|---|---|---|---|---|---|---|---|---|
| *RWKV-Reranker-0.1B (Full Depth: 12 Layers)* | | | | | | | | |
| Top-heavy | 3 | 9-11 | 23.1M | 56.94 | 38.86 | 50.07 | 48.62 | 85.99% |
| Uniform | 3 | 0, 5, 11 | 23.1M | 61.85 | 44.82 | 55.71 | 54.13 | 95.74% |
| Uniform | 3 | 1, 6, 10 | 23.1M | 62.35 | 48.05 | 56.88 | 55.76 | 98.62% |
| Top-heavy | 6 | 6-11 | 45.8M | 62.01 | 46.99 | 57.65 | 55.55 | 98.25% |
| Uniform | 6 | 0, 3, 5, 7, 9, 11 | 45.8M | 64.08 | 47.52 | 57.32 | 56.31 | 99.59% |
| Full | 12 | 0-11 | 90.0M | 63.41 | 47.78 | 58.42 | 56.54 | 100.0% |
| *RWKV-Reranker-1.3B (Full Depth: 24 Layers)* | | | | | | | | |
| Top-heavy | 1 | 23 | 54M | 50.02 | 35.55 | 45.10 | 43.56 | 67.25% |
| Uniform | 3 | 1,11,22 | 161M | 70.33 | 54.92 | 64.58 | 63.28 | 97.70% |
| Top-heavy | 6 | 18-23 | 318M | 69.01 | 51.43 | 64.39 | 61.61 | 95.12% |
| Uniform | 6 | 1,6,11,15,19,22 | 318M | 71.19 | 55.10 | 65.51 | 63.93 | 98.70% |
| Full | 24 | 0-23 | 1.3B | 71.58 | 56.44 | 66.30 | 64.77 | 100.0% |

single state from the final layer leads to a marked performance degradation. This result clarifies that redundancy does not imply that all useful signals can be collapsed into one state.

We attribute this failure to the functional specialization across depth. Intermediate-layer states tend to preserve fine-grained matching cues and local evidence, while the final-layer state is more abstract and task-agnostic. Hard sharing therefore removes the multi-scale signals required for accurate relevance estimation, resulting in catastrophic degradation in reranking quality.

Table 6: Ablation on sharing a single state across layers in RWKV-Reranker 0.1B.

| States | Layers | NanoBEIR | MTEB-E-Retr | MTEB-C-Retr |
|---|---|---|---|---|
| All states | 12 | 63.41 | 47.78 | 58.42 |
| One shared state | 12 | 51.23 | 11.00 | 5.62 |

**Why not cache Transformer KV for reranking?** A natural question is whether a similar idea can be applied to Transformers: can we cache document KV caches to accelerate reranking? The answer is no, for a simple reason—the KV cache is prohibitively large. For a document of 2000 tokens, a Transformer reranker with comparable parameter scale would require $62.5\times$ more storage to cache its KV states than our matrix-valued state. This makes KV-based caching impractical at scale, especially when reranking requires caching states for large corpora. In contrast, RWKV-style models maintain a constant-size matrix-valued state, which is exactly the architectural property we aim to exploit: we design the reranking pipeline around this compact, reusable representation to achieve efficient inference. A detailed discussion are provided in Appendix B.3.

## 5 Conclusion

We propose a state-centric retrieval paradigm, leveraging states as a bridge to achieve architectural synergy between embedding and reranking stages. This approach enables rerankers to reuse cached states, achieving high-quality results with improved efficiency. Beyond retrieval, our results suggest that direct modeling over states is both feasible and promising. This paradigm opens up opportunities for extending state-centric modeling to a broader range of scenarios, including state-based value models, verifiers, and reward models. We believe that state-centric modeling has significant potential to generalize to diverse application scenarios, particularly in the development of intelligent agents, where it may be naturally integrated into traditional reinforcement learning frameworks.

## Limitations

Despite its strong performance and efficiency, our method has several limitations. First, although state-based reranking reduces inference-time dependence on document length, the pipeline still requires maintaining both an embedder and a reranker, introducing additional system overhead. Second, cached matrix-valued states provide only compressed document representations, which may lose fine-grained evidence in extremely long or detail-rich documents and thus affect reranking accuracy.

## References


Parul Awasthy, Aashka Trivedi, Yulong Li, Mihaela Bornea, David Cox, Abraham Daniels, Martin Franz, Gabe Goodhart, Bhavani Iyer, Vishwajeet Kumar, Luis Lastras, Scott McCarley, Rudra Murthy, Vignesh P, Sara Rosenthal, Salim Roukos, Jaydeep Sen, Sukriti Sharma, Avirup Sil, and 3 others. 2025a. Granite embedding models. Hugging Face. Accessed: 2025-11-07.

Parul Awasthy, Aashka Trivedi, Yulong Li, Meet Doshi, Riyaz Bhat, Vignesh P, Vishwajeet Kumar, Yushu Yang, Bhavani Iyer, Abraham Daniels, Rudra Murthy, Ken Barker, Martin Franz, Madison Lee, Todd Ward, Salim Roukos, David Cox, Luis Lastras, Jaydeep Sen, and Radu Florian. 2025b. Granite embedding r2 models. Hugging Face. Accessed: 2025-11-07.

Nicolas Boizard, Hippolyte Gisserot-Boukhlef, Duarte M. Alves, André Martins, Ayoub Hammal, Caio Corro, C'eline Hudelot, Emmanuel Malherbe, Etienne Malaboeuf, Fanny Jourdan, Gabriel Hautreux, João Alves, Kevin El-Haddad, Manuel Faysse, Maxime Peyrard, Nuno M. Guerreiro, Patrick Fernandes, Ricardo Rei, and Pierre Colombo. 2025. Eurobert: Scaling multilingual encoders for european languages. *ArXiv*, abs/2503.05500.

Vera Boteva, Demian Gholipour Ghalandari, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings*, volume 9626 of *Lecture Notes in Computer Science*, pages 716–722. Springer.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642. The Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*. Key mapped to 'li2025' as requested for BGE. Actual paper is Chen et al. (2024).

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024b. BGE m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *CoRR*, abs/2402.03216.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *SSST@EMNLP*.

Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Y. Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loïc Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, and 19 others. 2022. No language left behind: Scaling human-centered machine translation. *CoRR*, abs/2207.04672.

Sławomir Dadas, Michał Perełkiewicz, and Rafał Poświata. 2023a. PIRB: A comprehensive benchmark of polish dense and hybrid text retrieval methods. Hugging Face. Accessed: 2025-11-07.

Sławomir Dadas, Michał Perełkiewicz, and Rafał Poświata. 2023b. PIRB: A comprehensive benchmark of polish dense and hybrid text retrieval methods. Hugging Face. Accessed: 2025-11-07.

Sławomir Dadas, Michał Perełkiewicz, and Rafał Poświata. 2024. PIRB: A comprehensive benchmark of polish dense and hybrid text retrieval methods. Hugging Face. Accessed: 2025-11-07.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

DataCanary, hilfialkaff, Lili Jiang, Meg Risdal, Nikhil Dandekar, and tomtung. 2017. Quora question pairs.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

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

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 1156–1165. ACM.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: long form question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3558–3567. Association for Computational Linguistics.

Wikimedia Foundation. 2024. Wikimedia downloads. Accessed: 2024-05-01.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Michael Günther, Louis Milliken, Jonathan Geuter, Georgios Mastrapas, Bo Wang, and Han Xiao. 2023. Jina embeddings: A novel set of high-performance sentence embedding models. Hugging Face. Accessed: 2025-11-07.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. 2017. news-please - A generic news crawler and extractor. In *Everything Changes, Everything Stays the Same? Understanding Information Spaces. Proceedings of the 15th International Symposium of Information Science, ISI 2017, Berlin, Germany, March 13-15, 2017*, volume 70 of *Schriften zur Informationswissenschaft*, pages 218–223. Verlag Werner Hülsbusch.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Samin Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. Xl-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 4693–4703. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Doris Hoogeveen, Karin M. Verspoor, and Timothy Baldwin. 2015. Cqadupstack: A benchmark data set for community question-answering research. In *ADCS*, pages 3:1–3:8. ACM.

Haowen Hou, Fei Ma, Zihang Li, and Fei Richard Yu. 2025. Visualrwkv-hm: Enhancing linear visual-language models via hybrid mixing. *Inf. Fusion*, 124:103336.

Haowen Hou, Xiaopeng Yan, and Yigeng Zhang. 2024a. Bagformer: Better cross-modal retrieval via bag-wise interaction. *Engineering Applications of Artificial Intelligence*, 136:108912.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian J. McAuley. 2024b. Bridging language and items for retrieval and recommendation. *CoRR*, abs/2403.03952.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *CoRR*, abs/1909.09436.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. 2019. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 2567–2577. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. 2021. Gooaq: Open question answering with diverse answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 421–433. Association for Computational Linguistics.

Yuta Koreeda and Christopher D. Manning. 2021. Contractnli: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 1907–1919. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*. NV-Emb Model.

Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, and 1 others. 2025. Gemini embedding: Generalizable embeddings from gemini. *arXiv preprint arXiv:2503.07891*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Patrick S. H. Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 million probably-asked questions and what you can do with them. *Trans. Assoc. Comput. Linguistics*, 9:1098–1115.

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

Yudong Li, Yuqing Zhang, Zhe Zhao, Linlin Shen, Weijie Liu, Weiquan Mao, and Hui Zhang. 2022. CSL: A large-scale chinese scientific literature dataset. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 3917–3923. International Committee on Computational Linguistics.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023a. Towards general text embeddings with multi-stage contrastive learning. Hugging Face. Accessed: 2025-11-07.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023b. Towards general text embeddings with multi-stage contrastive learning. Hugging Face. Accessed: 2025-11-07.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023c. Towards general text embeddings with multi-stage contrastive learning. Hugging Face. Accessed: 2025-11-07.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: Financial opinion mining and question answering. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, pages 1941–1942. ACM.

Chaitanya Malaviya, Subin Lee, Sihao Chen, Elizabeth Sieber, Mark Yatskar, and Dan Roth. 2024. 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), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3025–3045. Association for Computational Linguistics.

Luke Merrick. 2024a. Embedding and clustering your data can improve contrastive pretraining. Hugging Face. Accessed: 2025-11-07.

Luke Merrick. 2024b. Embedding and clustering your data can improve contrastive pretraining. Hugging Face. Accessed: 2025-11-07.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. *CoRR*, abs/2402.09906.

Niklas Muennighoff, Nouamane Tathi, Marjan Ebrahimi, and Kexin Luo. 2023a. Mteb: Massive text embedding benchmark. In *Proceedings of EACL*.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023b. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M. Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2023c. Crosslingual generalization through multitask finetuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 15991–16111. Association for Computational Linguistics.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of GPT-4. *CoRR*, abs/2306.02707.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located*

*with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic embed: Training a reproducible long context text embedder. Hugging Face. Accessed: 2025-11-07.

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

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, and 1 others. 2023. Rwkv: Reinventing rnns for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, and 1 others. 2024. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. 2025. Rwkv-7 "goose" with expressive dynamic state evolution. *Preprint*, arXiv:2503.14456.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of EMNLP-IJCNLP*.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. Hugging Face. Accessed: 2025-11-07.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. 2021. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*.

Shivalika Singh, Freddie Vargus, Daniel D'souza, Börje Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura O'Mahony, Mike Zhang, Ramith Hettiarachchi, Joseph Wilson, Marina Machado, Luisa Souza Moura, Dominik Krzeminski, Hakimeh Fadaei, Irem Ergün, Ifeoma Okoh, and 14 others. 2024. 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), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 11521–11567. Association for Computational Linguistics.

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

Maosong Sun, Jingyang Li, Zhipeng Guo, Yu Zhao, Yabin Zheng, Xiance Si, and Zhiyuan Liu. 2016. Thuctc: An efficient chinese text classifier. Accessed: 2024-05-01.

VectorPath Research Team. 2025. Searchmap: Conversational e-commerce search embedding model. Hugging Face. Accessed: 2025-11-07.

Nandan Thakur, Jianmo Ni, Gustavo Hernández Ábrego, John Wieting, Jimmy Lin, and Daniel Cer. 2024a. Leveraging llms for synthesizing training data across many languages in multilingual dense retrieval. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 7699–7724. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2024b. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. Hugging Face. Accessed: 2025-11-07.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. 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, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, and 1 others. 2025. Embeddinggemma: Powerful and lightweight text representations. *arXiv preprint arXiv:2509.20354*.

Ellen M. Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2020. TREC-COVID: constructing a pandemic information retrieval test collection. *SIGIR Forum*, 54(1):1:1–1:12.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7534–7550. Association for Computational Linguistics.

Feng Wang, Yuqing Li, and Hang Xiao. 2025. jina-reranker-v3: Last but not late interaction for listwise document reranking.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023. Text embeddings by weakly-supervised contrastive pre-training. Hugging Face. Accessed: 2025-11-07.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024a. Text embeddings by weakly-supervised contrastive pre-training. Hugging Face. Accessed: 2025-11-07.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024b. Text embeddings by weakly-supervised contrastive pre-training. Hugging Face. Accessed: 2025-11-07.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, and Furu Wei. 2024c. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*. E5-Mistral Model.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *ArXiv*, abs/2412.13663.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Xinwei Wu, Haojie Li, Hongyu Liu, Xinyu Ji, Ruohan Li, Yule Chen, and Yigeng Zhang. 2025. Uncovering the fragility of trustworthy llms through chinese textual ambiguity. *arXiv preprint arXiv:2507.23121*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*. Corresponds to the C-MTEB/C-Pack citation.

Xiaohui Xie, Qian Dong, Bingning Wang, Feiyang Lv, Ting Yao, Weinan Gan, Zhijing Wu, Xiangsheng Li, Haitao Li, Yiqun Liu, and Jin Ma. 2023. 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, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 2681–2690. ACM.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3685–3690. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel Campos. 2024. Arctic-embed 2.0: Multilingual retrieval without compromise. *arXiv preprint arXiv:2412.04506*. Key mapped to 'hu2025' for Arctic-Embed. Actual paper is Yu et al. (2024).

Sha Yuan, Hanyu Zhao, Zhengxiao Du, Ming Ding, Xiao Liu, Yukuo Cen, Xu Zou, Zhilin Yang, and Jie Tang. 2021. Wudaocorpora: A super large-scale chinese corpora for pre-training language models. *AI Open*, 2:65–68.

Xin Zhang, Zehan Li, and 1 others. 2024. Gte-multilingual: A multilingual text embedding model. *arXiv preprint*. Placeholder for updated GTE work.

Xinyu Zhang, Xueguang Ma, Peng Shi, and Jimmy Lin. 2021. Mr. tydi: A multi-lingual benchmark for dense retrieval. *CoRR*, abs/2108.08787.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. 2023. MIRACL: A multilingual retrieval dataset covering 18 diverse languages. *Trans. Assoc. Comput. Linguistics*, 11:1114–1131.

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

Xinping Zhao, Xinshuo Hu, Zifei Shan, Shouzheng Huang, Yao Zhou, Xin Zhang, Zetian Sun, Zhenyu Liu, Dongfang Li, Xinyuan Wei, and 1 others. 2025. Kalm-embedding-v2: Superior training techniques and data inspire a versatile embedding model. *arXiv preprint arXiv:2506.20923*.

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

# A Implementation Details

**Evaluation Setup** We evaluate our method on MTEB (eng, v2) for embeddings, and on NanoBEIR, MTEB (eng, v2), and MTEB (cmn, v1) for reranking tasks. We adopt a standard two-stage retrieval pipeline, utilizing an initial retriever to generate candidates followed by a reranker for fine-grained sorting, ensuring an optimal trade-off between retrieval latency and precision. Specifically, we employ a 0.1B EmbeddingRWKV model to retrieve the top-100 document candidates, which are subsequently re-ranked by the reranker.

**Hyperparameters** We show the training hyperparameters for both EmbeddingRWKV models and RWKV-Reranker models in Table 7.

| Hyperparameter | EmbeddingRWKV-0.1B | EmbeddingRWKV-0.4B | EmbeddingRWKV-1.4B | RWKV-Reranker-0.1B | RWKV-Reranker-0.3B | RWKV-Reranker-1.3B |
|---|---|---|---|---|---|---|
| batch size | 1024 | 1024 | 512 | 512 | 512 | 512 |
| lr init | 1e-5 | 1e-5 | 1e-5 | 1e-5 | 1e-5 | 1e-5 |
| lr end | 1e-5 | 1e-5 | 1e-5 | 5e-6 | 5e-6 | 5e-6 |
| lr schedule | constant | constant | constant | cosine decay | cosine decay | cosine decay |
| lr warmup ratio | 0 | 0 | 0 | 0 | 0 | 0 |
| weight decay | 0 | 0 | 0 | 0 | 0 | 0 |
| epoch | 1 | 1 | 1 | 1 | 1 | 1 |
| optimizer | AdamW | AdamW | AdamW | AdamW | AdamW | AdamW |
| DeepSpeed stage | 1 | 1 | 1 | 1 | 1 | 1 |

Table 7: **Hyperparameters** of EmbeddingRWKV and RWKV-Reranker.

# B Additional Results and Discussions

## B.1 Efficiency Analysis of State-Based Reranking

We analyze Table 8, comparing RWKV-Reranker (Online/Offline) with attention-based baselines across Large ($\approx$1.5B), Base ($\approx$0.4B), and Small ($\approx$0.1B) model regimes.

**Throughput and sequence-length scaling** The key advantage appears in RWKV Offline, where document states/embeddings are pre-computed and reused.

- **Large models (1.5B):** The attention baseline throughput degrades sharply with length (97.3 pairs/s at 512 $\rightarrow$ 11.9 pairs/s at 4096), while RWKV Offline stays nearly constant at $\sim$**500+ pairs/s** across lengths.

- **Small models (0.1B):** FlashAttention remains fast at short contexts but still slows down markedly at long contexts. In contrast, RWKV Offline sustains $\sim$**2440 pairs/s** up to 4096 tokens, reaching a **75.6**$\times$ speedup at the longest setting.

**Memory efficiency (VRAM)** RWKV's linear-state computation avoids the quadratic activation/memory growth of attention. At 4096 tokens, Standard Attention peaks at **39.59 GB** VRAM, whereas RWKV Offline requires only **10.07 GB**, i.e., an $\approx$**4**$\times$ reduction. This substantially lowers deployment barriers and enables larger inference batch sizes.

**Online vs. Offline modes** RWKV Online processes the full sequence on-the-fly and is therefore bounded by serial state updates; it primarily improves memory scaling rather than throughput. The intended operating point of our design is **split processing (Offline)**, which decouples document encoding from query interaction and reduces reranking to the lightweight query-stage computation (e.g., $\sim$**38 ms** for large models).

Table 8: Performance comparison of Attention mechanisms vs RWKV-Reranker (Online/Offline) across various sequence lengths (Query Len = 64, Batch Size = 100).

| Mode | Doc Len | Total (ms) | Embed (ms) | Rerank (ms) | Peak VRAM | Throughput (pair/s) | Speedup |
|---|---|---|---|---|---|---|---|
| *Baseline: mxbai-rerank-large-v2 (1.5B) vs. Ours: EmbeddingRWKV-1.4B + RWKV-Reranker-1.3B* | | | | | | | |
| Standard Attention | 512 | 1027.48 | - | - | 7.97 GB | 97.3 | 1.0x |
| Flash Attention | 512 | 997.66 | - | - | 7.97 GB | 100.2 | 1.0x |
| RWKV-Reranker Online | 512 | 1247.90 | 1209.66 | 38.21 | 12.41 GB | 80.1 | 0.8x |
| RWKV-Reranker Offline | 512 | 186.70 | 148.67 | 38.02 | 8.88 GB | 535.6 | 5.5x |
| Standard Attention | 1024 | 1988.10 | - | - | 12.49 GB | 50.3 | 1.0x |
| Flash Attention | 1024 | 1926.90 | - | - | 12.49 GB | 51.9 | 1.0x |
| RWKV-Reranker Online | 1024 | 2394.71 | 2356.14 | 38.55 | 15.95 GB | 41.8 | 0.8x |
| RWKV-Reranker Offline | 1024 | 185.46 | 148.50 | 36.94 | 10.07 GB | 539.2 | 10.7x |
| Standard Attention | 2048 | 4026.59 | - | - | 21.52 GB | 24.8 | 1.0x |
| Flash Attention | 2048 | 3860.74 | - | - | 21.52 GB | 25.9 | 1.0x |
| RWKV-Reranker Online | 2048 | 4598.01 | 4560.24 | 37.74 | 20.63 GB | 21.7 | 0.9x |
| RWKV-Reranker Offline | 2048 | 195.35 | 147.95 | 47.37 | 10.07 GB | 511.9 | 20.6x |
| Standard Attention | 4096 | 8410.25 | - | - | 39.59 GB | 11.9 | 1.0x |
| Flash Attention | 4096 | 8024.71 | - | - | 39.59 GB | 12.5 | 1.0x |
| RWKV-Reranker Online | 4096 | 9003.55 | 8965.33 | 38.20 | 30.01 GB | 11.1 | 0.9x |
| RWKV-Reranker Offline | 4096 | 185.99 | 147.52 | 38.45 | 10.07 GB | 537.7 | 45.2x |
| *Baseline: mxbai-rerank-base-v2 (0.5B) vs. Ours: EmbeddingRWKV-0.4B + RWKV-Reranker-0.3B* | | | | | | | |
| Standard Attention | 512 | 381.15 | - | - | 3.55 GB | 262.4 | 1.0x |
| Flash Attention | 512 | 360.84 | - | - | 3.55 GB | 277.1 | 1.1x |
| RWKV-Reranker Online | 512 | 479.16 | 444.06 | 35.08 | 5.02 GB | 208.7 | 0.8x |
| RWKV-Reranker Offline | 512 | 96.07 | 59.55 | 36.50 | 3.26 GB | 1040.9 | 4.0x |
| Standard Attention | 1024 | 723.18 | - | - | 5.87 GB | 138.3 | 1.0x |
| Flash Attention | 1024 | 687.43 | - | - | 5.87 GB | 145.5 | 1.1x |
| RWKV-Reranker Online | 1024 | 913.71 | 878.26 | 35.43 | 6.79 GB | 109.4 | 0.8x |
| RWKV-Reranker Offline | 1024 | 94.10 | 59.66 | 34.43 | 3.85 GB | 1062.7 | 7.7x |
| Standard Attention | 2048 | 1461.67 | - | - | 10.51 GB | 68.4 | 1.0x |
| Flash Attention | 2048 | 1385.56 | - | - | 10.51 GB | 72.2 | 1.1x |
| RWKV-Reranker Online | 2048 | 1747.95 | 1712.42 | 35.51 | 9.14 GB | 57.2 | 0.8x |
| RWKV-Reranker Offline | 2048 | 94.53 | 59.88 | 34.64 | 3.85 GB | 1057.8 | 15.5x |
| Standard Attention | 4096 | 3098.93 | - | - | 19.79 GB | 32.3 | 1.0x |
| Flash Attention | 4096 | 2946.56 | - | - | 19.79 GB | 33.9 | 1.1x |
| RWKV-Reranker Online | 4096 | 3424.99 | 3388.77 | 36.19 | 13.82 GB | 29.2 | 0.9x |
| RWKV-Reranker Offline | 4096 | 95.42 | 60.05 | 35.34 | 3.85 GB | 1048.0 | 32.5x |
| *Baseline: gte-reranker-modernbert-base (0.1B) vs. Ours: EmbeddingRWKV-0.1B + RWKV-Reranker-0.1B* | | | | | | | |
| Standard Attention | 512 | 171.85 | - | - | 1.16 GB | 581.9 | 1.0x |
| Flash Attention | 512 | 110.02 | - | - | 1.00 GB | 908.9 | 1.6x |
| RWKV-Reranker Online | 512 | 178.52 | 160.77 | 17.74 | 2.34 GB | 560.2 | 1.0x |
| RWKV-Reranker Offline | 512 | 41.00 | 23.34 | 17.65 | 1.24 GB | 2438.9 | 4.2x |
| Standard Attention | 1024 | 387.41 | - | - | 2.14 GB | 258.1 | 1.0x |
| Flash Attention | 1024 | 212.34 | - | - | 1.62 GB | 470.9 | 1.8x |
| RWKV-Reranker Online | 1024 | 333.24 | 315.05 | 18.18 | 3.45 GB | 300.1 | 1.2x |
| RWKV-Reranker Offline | 1024 | 40.98 | 23.41 | 17.56 | 1.46 GB | 2440.0 | 9.5x |
| Standard Attention | 2048 | 1009.89 | - | - | 4.68 GB | 99.0 | 1.0x |
| Flash Attention | 2048 | 443.68 | - | - | 2.87 GB | 225.4 | 2.3x |
| RWKV-Reranker Online | 2048 | 620.39 | 602.03 | 18.33 | 5.21 GB | 161.2 | 1.6x |
| RWKV-Reranker Offline | 2048 | 40.83 | 23.37 | 17.45 | 1.46 GB | 2449.3 | 24.7x |
| Standard Attention | 4096 | 3098.45 | - | - | 12.11 GB | 32.3 | 1.0x |
| Flash Attention | 4096 | 991.55 | - | - | 5.38 GB | 100.9 | 3.1x |
| RWKV-Reranker Online | 4096 | 1214.92 | 1196.54 | 18.36 | 8.73 GB | 82.3 | 2.6x |
| RWKV-Reranker Offline | 4096 | 40.99 | 23.42 | 17.55 | 1.46 GB | 2439.8 | 75.6x |

### B.2 Computational Complexity of Backbone

Unlike Transformer-based embedding models (e.g., BERT, GPT) which suffer from quadratic time complexity $O(L^2)$ due to the global self-attention mechanism, EmbeddingRWKV adopts RWKV as its backbone, which employs a linear attention formulation. This results in a time complexity of $O(L)$ and a

constant memory footprint $O(1)$ regarding the sequence state during inference, irrespective of the input sequence length.

As shown in Table 9, this property allows our model to scale efficiently to long-context tasks where traditional transformers typically encounter memory bottlenecks.

Table 9: Computational complexity comparison between Transformer-based backbones and RWKV during inference. $L$ denotes the sequence length, and $d$ denotes the hidden dimension.

| Architecture | Attention Mechanism | Time Complexity | Space Complexity | Max Context |
|---|---|---|---|---|
| BERT / GPT | $O(L^2)$ (Global) | $O(L^2)$ | $O(L^2 + Ld)$ | Finite |
| FlashAttention* | $O(L^2)$ (Tiled) | $O(L^2)$ | $O(L)$ | Finite |
| **RWKV** | $O(L)$ **(Linear)** | $\mathbf{O(L)}$ | $\mathbf{O(1)}^{\dagger}$ | **Infinite** |

† *Note: RWKV maintains a fixed-size hidden state (RNN-mode) regardless of history length L, whereas Transformers must store the Attention Matrix or KV Cache.*

### B.3 Memory Footprint Usage

**Analytical Memory Cost** We compare the per-document cache footprint of RWKV document states against Transformer KV caches under the same precision setting (FP16). For RWKV, each layer stores a document state with shape $(H, S, S)$ (one state per head), hence the cache size is $\text{Bytes}_{\text{RWKV}} = L \times H \times S^2 \times 2$, where $L$ is the number of layers, $H$ the number of heads, $S$ the state size, and 2 accounts for FP16 bytes.

For Transformers, the document-side cache is the concatenated Key/Value tensors. Per layer it stores $(K, V)$ with total width $2d_{\text{model}}$ over $T$ cached tokens, giving $\text{Bytes}_{\text{KV}} = L \times (2d_{\text{model}}) \times T \times 2$. To make the comparison interpretable, we use the common RWKV state–KV equivalence $T_{\text{eq}} = S/2$. With $S = 64$, this yields $T_{\text{eq}} = 32$, meaning one RWKV document state per layer is roughly comparable to caching $\sim 32$ Transformer tokens (KV) per layer.

**Per-document Cache Usage Comparison** Table 10 reports the per-document cache size for RWKV states (with $S = 64$) and Transformer KV caches at $T = 2000$. A key observation is that the ratio stays constant at $2000/32 = 62.5\times$ across model sizes: when $S$ and $T$ are fixed, both RWKV and KV scale linearly with $L$, while the dominant difference comes from RWKV being *token-length independent* (state cache does not grow with $T$), whereas Transformer KV grows linearly with $T$. Concretely, RWKV requires only 1–16 MB per document in the configurations shown, while Transformer KV reaches 70–1000 MB at $T = 2000$. This gap translates directly into higher cache density (more documents stored per GPU/CPU memory budget) and substantially lower retrieval-system memory pressure when large-scale document caching is required.

Table 10: Per-document cache usage for RWKV states vs. Transformer KV cache at $T = 2000$ (FP16). RWKV uses $S = 64$ and $T_{\text{eq}} = S/2 = 32$, leading to a constant ratio $T/T_{\text{eq}} = 62.5\times$ under fixed $S$ and $T$.

| Model Size | L | $\mathbf{d_{model}}$ | H | S | RWKV State (MB) | Transformer KV @ T=2000 (MB) | Ratio (Trans/RWKV) |
|---|---|---|---|---|---|---|---|
| 0.1B | 12 | 768 | 12 | 64 | 1.12 | 70.31 | 62.5× |
| 0.4B | 24 | 1024 | 16 | 64 | 3.00 | 187.50 | 62.5× |
| 1.5B | 24 | 2048 | 32 | 64 | 6.00 | 375.00 | 62.5× |
| ∼3B | 32 | 2560 | 40 | 64 | 10.00 | 625.00 | 62.5× |
| ∼7B | 32 | 4096 | 64 | 64 | 16.00 | 1000.00 | 62.5× |

# C Data Details

## C.1 Pre-training Data Construction

To build a robust and versatile embedding model, we curated a large-scale pre-training mixture of approximately 470M text pairs. The collection strategy targets (i) linguistic diversity, (ii) broad domain coverage, and (iii) multiple levels of text granularity. As summarized in Table 11, the pre-training sources fall into three coarse groups: multilingual corpora, Chinese-focused corpora, and English (including code) resources.

Multilingual corpora provide the backbone for cross-lingual semantic alignment. We include large unstructured resources such as *CC-News* (100M pairs) and *Wikipedia* (100M pairs) to capture general knowledge and multilingual semantics. To strengthen instruction following and IR-oriented matching, we incorporate curated query–document style datasets including *xP3* (Muennighoff et al., 2023c) and the *SWIM-IR* suite (Thakur et al., 2024a). We further leverage parallel corpora such as *NLLB* to anchor representations across distant language families.

Given the unique challenges of Chinese semantic modeling, we explicitly up-weight high-quality Chinese sources. *Wudao* (Yuan et al., 2021) serves as a primary component (44M pairs) for high-coverage web text, complemented by vertical-domain corpora such as *CSL* (scientific literature), *THUCNews* (news), and *Zhihu-KOL* (community Q&A). This combination improves robustness across both formal registers and colloquial discourse.

English and code data are included to reinforce general semantic reasoning and technical coverage. We use *Reddit* (100M pairs) and *StackExchange* (14M pairs) for conversational and QA-style semantics, together with *S2ORC* (41M pairs) and *CodeSearchNet* for scientific and programming-oriented content. We additionally include *PAQ* (Lewis et al., 2021) to improve retrieval behavior via probable questions derived from Wikipedia.

Overall, this mixture is designed to be complementary along three axes. First, it spans diverse *granularity*, from short texts (e.g., reviews, tweets) to long documents (e.g., Wudao, S2ORC), enabling the model to handle varying lengths and information densities. Second, it improves *domain generalization* by combining formal sources (papers, news) with informal sources (forums, Q&A), reducing style overfitting. Third, the large multilingual portion acts as an *alignment anchor*, allowing English- and Chinese-specific knowledge to transfer more effectively to lower-resource languages.

Table 11: List of pre-training data sources. The list now includes SkyPile and Falcon datasets.

| Source | Language | Pairs |
|---|---|---|
| Amazon-Reviews (Hou et al., 2024b) | Multilingual | 23M |
| CC-News (Hamborg et al., 2017) | Multilingual | 100M |
| NLLB (Costa-jussà et al., 2022) | Multilingual | 2M |
| Wikipedia (Foundation, 2024) | Multilingual | 100M |
| xP3 (Muennighoff et al., 2023c) | Multilingual | 19M |
| XL-Sum (Hasan et al., 2021) | Multilingual | 1M |
| SWIM-IR (Mono) (Thakur et al., 2024a) | Multilingual | 3M |
| SWIM-IR (Cross) (Thakur et al., 2024a) | Multilingual | 15M |
| CSL (Li et al., 2022) | Chinese | 0.4M |
| Wudao (Yuan et al., 2021) | Chinese | 44M |
| SkyPile | Chinese | 81M |
| THUCNews (Sun et al., 2016) | Chinese | 0.8M |
| Zhihu-KOL | Chinese | 0.8M |
| CodeSearchNet (Husain et al., 2019) | English | 1M |
| Falcon | English | 128M |
| PAQ (Lewis et al., 2021) | English | 9M |
| Reddit | English | 100M |
| StackExchange | English | 14M |
| S2ORC | English | 41M |

### C.2 Fine-tuning Data Curation

To align the pre-trained model with instruction-following behavior and downstream objectives, we construct a fine-tuning mixture spanning over 70 high-quality corpora. Table 12 summarizes the datasets, which cover three primary task families: retrieval, semantic textual similarity (STS), and classification. The curation balances English, Chinese, and multilingual coverage while preserving task heterogeneity.

Retrieval data constitutes the core of the fine-tuning stage, focusing on query–passage matching (s2p). We include established English benchmarks such as *MSMARCO* (Nguyen et al., 2016) and *Natural Questions*, together with large-scale Chinese retrieval corpora such as *T2Ranking* (Xie et al., 2023) and *mMARCO*. For cross-lingual retrieval, we incorporate *MIRACL* and *Mr. TyDi*, which provide high-quality query–passage pairs across diverse languages.

To improve sensitivity to fine-grained semantic nuances, we add STS-style datasets including *STS Benchmark*, *Quora Question Pairs*, and *AFQMC*. These typically follow an s2s format and encourage the model to separate semantically equivalent from non-equivalent pairs.

We further include classification and clustering corpora (e.g., *AmazonPolarity*, *TNews*, *Reddit-Clustering*) to encourage linearly separable semantic structure in the embedding space, improving representation geometry beyond pure pair matching.

Quality control is critical at this stage. As indicated by the "Pairs (filtered)" column in Table 12, we apply a filtering pipeline that removes duplicates, filters low-overlap positives, and discards samples that are too short or noisy. For example, in *Reddit-P2P*, aggressive filtering reduces the data from ∼12M to ∼42k pairs, mitigating noise-induced degeneration and improving training efficiency.

Table 12: List of fine-tuning datasets. The datasets are grouped by language and categorized by task type (Retrieval, STS, Classification). Columns show the original pair count and the count after filtering.

| Source | Type | Categ. | Pairs | Pairs (Filt.) |
|---|---|---|---|---|
| ***English Resources*** | | | | |
| CodeFeedback (Zheng et al., 2024) | Retrieval | s2p | 50,000 | 49,090 |
| ELI5 (Fan et al., 2019) | Retrieval | s2p | 100,000 | 76,408 |
| ExpertQA (Malaviya et al., 2024) | Retrieval | s2p | 1,261 | 1,252 |
| GooAQ (Khashabi et al., 2021) | Retrieval | s2p | 50,000 | 49,833 |
| MEDI2BGE (Muennighoff et al., 2024) | Retrieval | s2p | 100,000 | 71,790 |
| OpenOrca (Mukherjee et al., 2023) | Retrieval | s2p | 40,000 | 38,623 |
| PAQ (Lewis et al., 2021) | Retrieval | s2p | 50,000 | 49,849 |
| PubMedQA (Jin et al., 2019) | Retrieval | s2p | 80,000 | 79,954 |
| SearchQA (Dunn et al., 2017) | Retrieval | s2p | 10,000 | 9,988 |
| arxiv_qa | Retrieval | s2p | 23,397 | 17,927 |
| CC-News (Hamborg et al., 2017) | Retrieval | s2p | 30,000 | 28,246 |
| TREC-COVID (Voorhees et al., 2020) | Retrieval | s2p | 50,000 | 48,517 |
| DBpedia-Entity (Thakur et al., 2021) | Retrieval | s2p | 100,000 | 96,792 |
| ESCI (Reddy et al., 2022) | Retrieval | s2p | 30,000 | 26,043 |
| FEVER (Thorne et al., 2018) | Retrieval | s2p | 87,855 | 87,216 |
| FiQA (Maia et al., 2018) | Retrieval | s2p | 5,490 | 4,689 |
| HotpotQA (Yang et al., 2018) | Retrieval | s2p | 184,057 | 150,153 |
| MLDR (Chen et al., 2024b) | Retrieval | s2p | 41,434 | 31,097 |
| MSMARCO (Nguyen et al., 2016) | Retrieval | s2p | 175,133 | 174,190 |
| MSMARCO-v2 (Nguyen et al., 2016) | Retrieval | s2p | 277,144 | 258,617 |
| NFCorpus (Boteva et al., 2016) | Retrieval | s2p | 10,824 | 10,471 |
| rag-dataset-12k | Retrieval | s2p | 9,590 | 9,272 |

*Continued on next page...*

Table 12: Fine-tuning data list (continued)

| Source | Type | Categ. | Pairs | Pairs (Filt.) |
|---|---|---|---|---|
| SciFact (Wadden et al., 2020) | Retrieval | s2p | 809 | 794 |
| SQuAD 2.0 (Rajpurkar et al., 2018) | Retrieval | s2p | 130,217 | 125,816 |
| TriviaQA (Joshi et al., 2017) | Retrieval | s2p | 52,886 | 44,442 |
| WebGPT Comp. (Nakano et al., 2021) | Retrieval | s2p | 19,242 | 18,924 |
| Natural Questions (Kwiatkowski et al., 2019) | Retrieval | s2p | 58,622 | 56,377 |
| Yahoo Answers | Retrieval | s2p | 30,000 | 21,724 |
| CQADupStack (Hoogeveen et al., 2015) | Retrieval | s2p | 24,045 | 7,356 |
| ContractNLI (Koreeda and Manning, 2021) | STS | s2s | 3,195 | 628 |
| MultiNLI (Williams et al., 2018) | STS | s2s | 64,674 | 63,701 |
| NLLB (Costa-jussà et al., 2022) | STS | s2s | 36,000 | 26,504 |
| Quora (DataCanary et al., 2017) | STS | s2s | 92,674 | 89,558 |
| WikiAnswers (Fader et al., 2014) | STS | s2s | 50,000 | 47,686 |
| SimCSE NLI (Gao et al., 2021) | STS | s2s | 252,397 | 217,099 |
| SNLI (Bowman et al., 2015) | STS | s2s | 24,686 | 16,480 |
| arXiv | Classification | s2s, p2s | 15,000 | 14,529 |
| Biorxiv | Classification | s2s, p2s | 6,862 | 6,787 |
| Medrxiv | Classification | s2s, p2s | 2,012 | 1,999 |
| Reddit-Clustering | Classification | s2s | 128,000 | 25,600 |
| Reddit-P2P | Classification | p2s | 12,704,958 | 42,480 |
| Stack-Clustering | Classification | s2s | 1,014,826 | 50,530 |
| Stack-P2P | Classification | p2s | 25,333,327 | 48,800 |
| 20Newsgroups | Classification | s2s | 11,314 | 6,233 |
| AmazonPolarity | Classification | s2s | 10,000 | 9,007 |
| IMDB | Classification | s2s | 10,000 | 8,575 |
| Banking77 | Classification | s2s | 10,000 | 9,937 |
| Emotion | Classification | s2s | 10,000 | 10,000 |
| TweetSentiment | Classification | s2s | 10,000 | 10,000 |
| ToxicConversations | Classification | s2s | 7,916 | 7,800 |
| ***Chinese Resources*** | | | | |
| AdvertiseGen | Retrieval | s2p | 20,000 | 17,526 |
| CHEF | Retrieval | s2p | 4,952 | 4,824 |
| ChatMed | Retrieval | s2p | 20,000 | 18,608 |
| CMRC 2018 | Retrieval | s2p | 10,000 | 9,753 |
| DRCD | Retrieval | s2p | 5,000 | 4,714 |
| LCSTS | Retrieval | s2p | 20,000 | 19,535 |
| LIMA | Retrieval | s2p | 2,058 | 1,991 |
| Multi-CPR | Retrieval | s2p | 287,881 | 234,587 |
| PAWS-X (zh) | Retrieval | s2p | 49,401 | 19,289 |
| RefGPT | Retrieval | s2p | 50,000 | 49,896 |
| T2Ranking | Retrieval | s2p | 199,412 | 188,606 |
| THUCNews | Retrieval | s2p | 20,000 | 19,288 |
| UMETRIP-QA | Retrieval | s2p | 2,647 | 2,537 |
| WebCPM | Retrieval | s2p | 1,605 | 1,602 |
| cCOVID-News | Retrieval | s2p | 5,000 | 4,727 |
| cMedQA-V2.0 | Retrieval | s2p | 223,851 | 88,109 |
| CSL | Retrieval | s2p | 20,000 | 19,945 |
| DuReader | Retrieval | s2p | 80,416 | 79,229 |

*Continued on next page...*

Table 12: Fine-tuning data list (continued)

| Source | Type | Categ. | Pairs | Pairs (Filt.) |
|---|---|---|---|---|
| DuReader$_{\text{checklist}}$ | Retrieval | s2p | 99,992 | 97,764 |
| law-gpt | Retrieval | s2p | 500 | 500 |
| lawzhidao | Retrieval | s2p | 8,000 | 6,784 |
| mMARCO (zh) | Retrieval | s2p | 400,000 | 379,870 |
| retrieval_data_llm | Retrieval | s2p | 32,768 | 32,551 |
| webqa | Retrieval | s2p | 5,000 | 4,988 |
| AFQMC | STS | s2s | 4,041 | 3,876 |
| ATEC | STS | s2s | 62,477 | 11,387 |
| BQ | STS | s2s | 100,000 | 10,000 |
| CAIL2019-SCM | STS | s2s | 5,102 | 648 |
| CINLID | STS | s2s | 5,000 | 2,883 |
| ChineseSTS | STS | s2s | 2,500 | 2,497 |
| CMNLI | STS | s2s | 125,356 | 119,029 |
| nli_zh | STS | s2s | 218,887 | 185,787 |
| OCNLI | STS | s2s | 13,464 | 11,937 |
| QBQTC | STS | s2s | 51,620 | 47,223 |
| SimCLUE | STS | s2s | 344,038 | 290,699 |
| XNLI (zh) | STS | s2s | 80,000 | 74,252 |
| CSL | Classification | s2s, p2s | 15,000 | 12,249 |
| THUCNews | Classification | s2s | 10,000 | 9,690 |
| TNews | Classification | s2s | 10,000 | 6,762 |
| JDReview | Classification | s2s | 1,232 | 1,232 |
| IFlyTek | Classification | s2s | 10,000 | 8,221 |
| OnlineShopping | Classification | s2s | 7,852 | 7,600 |
| Waimai | Classification | s2s | 7,384 | 7,376 |
| ***Multilingual Resources*** | | | | |
| Aya Dataset (Singh et al., 2024) | Retrieval | s2p | 30,000 | 26,292 |
| MIRACL (Zhang et al., 2023) | Retrieval | s2p | 40,151 | 39,946 |
| Mr. TyDi (Zhang et al., 2021) | Retrieval | s2p | 48,729 | 46,997 |
| PAWS-X (Yang et al., 2019) | STS | s2s | 128,435 | 128,398 |
| AmazonReviews | Classification | s2s | 10,000 | 7,721 |
| AmazonCounterfactual | Classification | s2s | 10,000 | 8,323 |
| MultiSentiment | Classification | s2s | 10,000 | 9,804 |
| Amazon Massive | Classification | s2s | 10,000 | 7,832 |
| Amazon Scenario | Classification | s2s | 10,000 | 7,078 |
| MTOPDomain | Classification | s2s | 10,000 | 9,610 |
| MTOPIntent | Classification | s2s | 10,000 | 7,952 |

### C.3 Unified Instruction Templates for MTEB Evaluation

Table 13 summarizes the instruction templates used across MTEB (eng, v2). Although MTEB covers heterogeneous capabilities—classification, clustering, semantic similarity, retrieval, reranking, and summarization—a large fraction of tasks can be expressed using a small family of instruction patterns. This enables a unified instruction-driven evaluation without per-dataset prompt engineering.

For label-centric tasks such as *Classification* and *Clustering*, we use task-specific natural language descriptions that encode dataset semantics (e.g., emotion, intent, topic). This formulation maps the original objective into a consistent "Instruct/Query" interface.

For semantic matching tasks—including *Retrieval*, *Reranking*, *Pair Classification*, and *STS*—we use

shared templates due to their closely aligned objectives. As indicated by the asterisks in Table 13, the same instruction is applied to all remaining tasks of each type. Only a small number of datasets require overrides to better reflect dataset intent (e.g., duplicate question detection in QuoraRetrieval or StackExchange-style tasks).

This unified design offers two practical benefits: it reduces the mismatch between instruction-tuned behavior and embedding-style evaluation, and it provides a principled way to inject task intent into continuous representations, yielding a more standardized and interpretable evaluation pipeline.

# D Responsible NLP Checklist

## D.1 Potential Risks

Our method introduces several potential risks. First, cached document states may retain compressed information from the source text, which could raise privacy concerns when the system is applied to sensitive corpora. Second, because states are compressed representations, they may discard fine-grained evidence needed for safety-critical decision making, especially for long or highly detailed documents. Third, biases or domain imbalance in the training data may be inherited by the learned state representations and affect downstream retrieval and reranking behavior. Finally, in dynamic corpora, stale cached states may become inconsistent with the latest document contents, potentially reducing retrieval reliability. These risks suggest that practical deployment should be accompanied by privacy safeguards, cache refreshing mechanisms, and careful evaluation in domain-specific settings.

## D.2 Use of AI Assistants

We used AI assistants for limited writing support, such as language polishing and phrasing suggestions. No AI assistant was used to generate scientific claims, experimental results, or conclusions. All content was reviewed and verified by the authors, who take full responsibility for the paper.

Table 13: Detailed task instruction list for MTEB evaluation. Pair Classification*, Reranking*, Retrieval*, and STS* indicate we use the same instructions for all the respective remaining tasks.

| Task Name | Instruction |
|---|---|
| | ***Classification*** |
| AmazonCounterfactual | Instruct: Given an Amazon review, judge whether it is counterfactual. \n Query: {query} |
| AmazonPolarity | Instruct: Classify Amazon reviews into positive or negative sentiment. \n Query: {query} |
| AmazonReviews | Instruct: Classify the given Amazon review into its appropriate rating category. \n Query: {query} |
| Banking77 | Instruct: Given an online banking query, find the corresponding intents. \n Query: {query} |
| Emotion | Instruct: Classify the emotion expressed in the given Twitter message into one of the six emotions: anger, fear, joy, love, sadness, and surprise. \n Query: {query} |
| Imdb | Instruct: Classify the sentiment expressed in the given movie review text from the IMDB dataset. \n Query: {query} |
| MassiveIntent | Instruct: Given a user utterance as query, find the user intents. \n Query: {query} |
| MassiveScenario | Instruct: Given a user utterance as query, find the user scenarios. \n Query: {query} |
| MTOPDomain | Instruct: Classify the intent domain of the given utterance in task-oriented conversation. \n Query: {query} |
| MTOPIntent | Instruct: Classify the intent of the given utterance in task-oriented conversation. \n Query: {query} |
| ToxicConversations | Instruct: Classify the given comments as either toxic or not toxic. \n Query: {query} |
| TweetSentiment | Instruct: Classify the sentiment of a given tweet as either positive, negative, or neutral. \n Query: {query} |
| TNews | Instruct: Categorize the given news title. \n Query: {query} |
| IFlyTek | Instruct: Given an App description text, find the appropriate fine-grained category. \n Query: {query} |
| MultilingualSentiment | Instruct: Classify sentiment of the customer review into positive, neutral, or negative. \n Query: {query} |
| JDReview | Instruct: Classify sentiment of the customer review for iPhone into positive or negative. \n Query: {query} |
| OnlineShopping | Instruct: Classify sentiment of the customer review into positive or negative. \n Query: {query} |
| Waimai | Instruct: Classify the customer review from a food takeaway platform into positive or negative. \n Query: {query} |
| | ***Clustering*** |
| ArxivClusteringP2P | Instruct: Identify the main and secondary category of Arxiv papers based on the titles and abstracts. \n Query: {query} |
| ArxivClusteringS2S | Instruct: Identify the main and secondary category of Arxiv papers based on the titles. \n Query: {query} |
| BiorxivClusteringP2P | Instruct: Identify the main category of Biorxiv papers based on the titles and abstracts. \n Query: {query} |
| BiorxivClusteringS2S | Instruct: Identify the main category of Biorxiv papers based on the titles. \n Query: {query} |
| MedrxivClusteringP2P | Instruct: Identify the main category of Medrxiv papers based on the titles and abstracts. \n Query: {query} |
| MedrxivClusteringS2S | Instruct: Identify the main category of Medrxiv papers based on the titles. \n Query: {query} |
| RedditClustering | Instruct: Identify the topic or theme of Reddit posts based on the titles. \n Query: {query} |
| RedditClusteringP2P | Instruct: Identify the topic or theme of Reddit posts based on the titles and posts. \n Query: {query} |
| StackExchangeClustering | Instruct: Identify the topic or theme of StackExchange posts based on the titles. \n Query: {query} |
| StackExchangeP2P | Instruct: Identify the topic or theme of StackExchange posts based on the given paragraphs. \n Query: {query} |
| TwentyNewsgroups | Instruct: Identify the topic or theme of the given news articles. \n Query: {query} |
| CLSClusteringS2S | Instruct: Identify the main category of scholar papers based on the titles. \n Query: {query} |
| CLSClusteringP2P | Instruct: Identify the main category of scholar papers based on the titles and abstracts. \n Query: {query} |
| ThuNewsClusteringS2S | Instruct: Identify the topic or theme of the given news articles based on the titles. \n Query: {query} |
| ThuNewsClusteringP2P | Instruct: Identify the topic or theme of the given news articles based on the titles and contents. \n Query: {query} |
| | ***Pair Classification*** |
| Pair Classification* | Instruct: Retrieve semantically similar text. \n Query: {query} |
| SprintDuplicateQuestions | Instruct: Retrieve semantically similar questions. \n Query: {query} |
| | ***Reranking*** |
| Reranking* | Instruct: Given a query, retrieve documents that answer the query. \n Query: {query} |
| AskUbuntuDupQuestions | Instruct: Retrieve semantically similar questions. \n Query: {query} |
| StackOverflowDupQuestions | Instruct: Retrieve semantically similar questions. \n Query: {query} |
| SciDocsRR | Instruct: Retrieve relevant paper titles. \n Query: {query} |
| | ***Retrieval*** |
| Retrieval* | Instruct: Given a query, retrieve documents that answer the query. \n Query: {query} |
| QuoraRetrieval | Instruct: Retrieve semantically similar questions. \n Query: {query} |
| CQADupstack | Instruct: Given a question, retrieve detailed question descriptions from StackExchange that are duplicates to the given question. \n Query: {query} |
| | ***STS*** |
| STS* | Instruct: Retrieve semantically similar text. \n Query: {query} |
| | ***Summarization*** |
| SummEval | Instruct: Retrieve semantically similar summaries. \n Query: {query} |