# EVOEMBEDDING: EVOLVABLE REPRESENTATIONS FOR LONG-CONTEXT RETRIEVAL AND AGENTIC MEMORY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Existing embedding models are inherently static: they encode text segments in isolation, ignoring their surrounding context and temporal order. This paper introduces **EvoEmbedding**, a novel embedding model that generates *evolvable* representations for retrieval. It is tailored for long-context scenarios, where information is dynamic, sequential, and requires continuous state tracking. Our design is simple: EvoEmbedding maintains a continuously updated latent memory as it sequentially processes inputs, and uses it alongside the raw content to jointly generate evolvable embeddings. Consequently, for the same query, our model adapts its representation to retrieve distinct targets based on the evolving context, going beyond static semantic search. To equip the model with this capability, we construct **EvoTrain-180K**, a diverse dataset for the joint optimization of latent memory and retrieval. Furthermore, we introduce a memory queue to prevent representation collapse during recurrent encoding, alongside segment-batching techniques that tackle significant length variance and accelerate training by 3.8×. Extensive experiments show that our model not only outperforms larger-scale specialists (e.g., Qwen3-Embedding-8B and KaLM-Embedding-Gemma3-12B) across a range of long-context retrieval benchmarks, but also generalizes well to downstream tasks (e.g., personalization) with contexts 10× longer than its training window. Notably, EvoEmbedding seamlessly integrates into agentic workflows to boost performance. For instance, a naive RAG pipeline equipped with our model surpasses dedicated agentic memory systems. Our code is available in the supplement.

## 1 INTRODUCTION

Retrieval-Augmented Generation (RAG) has emerged as a dominant paradigm for augmenting Large Language Models (LLMs) with external knowledge and extended context (Fan et al., 2024; Zhao et al., 2026a). It is widely applied in AI agent design and context engineering (Mei et al., 2025; OpenClaw, 2026), equipping LLMs with essential long-term memory capabilities to accomplish complex, long-horizon tasks (Hu et al., 2025). However, when confronted with long-context scenarios where information is dynamic, sequential, and requires continuous state tracking, existing retrieval systems, which predominantly rely on static representations, frequently fail to retrieve the desired target information (Maharana et al., 2024; Weller et al., 2025), and often introduce noise into the generation process (Liu et al., 2024).

The root cause of this limitation lies in two key flaws of current retrieval pipelines. First, existing methods typically extract and encode text segments in isolation (Gao et al., 2023; Sarthi et al., 2024), a process that inherently disrupts the temporal continuity and contextual associations between these segments. Second, current embedding models are optimized via contrastive learning primarily on short, static samples (Chen et al., 2024; Zhang et al., 2025; Zhao et al., 2025). This restricts their capability to the mere discrimination of semantic relevance, rendering them ill-equipped for complex contextual understanding, such as coreference resolution and temporal reasoning (Guo et al., 2024; Weller et al., 2025). As illustrated in Fig. 1 (Left), a user initially schedules a meeting and later postpones it. When queried about the meeting time under varying contexts, the static embedding model fails to capture this dynamic evolution, ultimately retrieving the outdated schedule information.

To mitigate these limitations, recent advancements primarily optimize the RAG pipeline across two aspects: indexing and retrieval execution. During the indexing phase, efforts focus on transforming



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

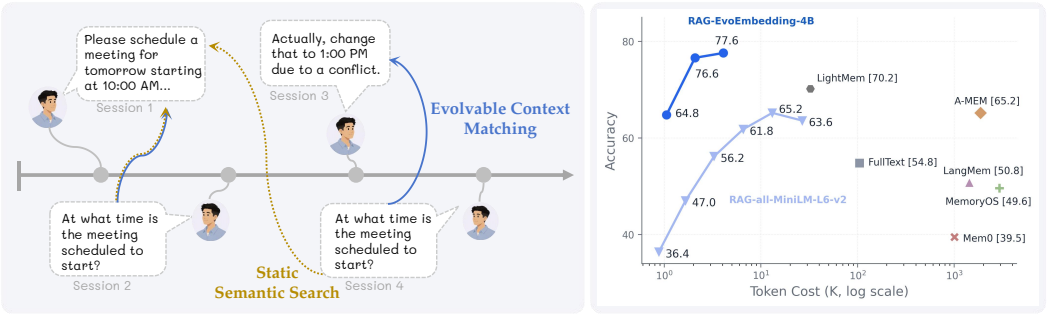



Figure 1: **(Left)** For the same query under different contexts, EvoEmbedding generates evolvable representations that avoid the outdated retrievals of static methods. **(Right)** On LongMemEval, standard RAG equipped with EvoEmbedding-4B outperforms existing memory baselines, achieving SOTA performance while minimizing costs with *zero* token overhead for explicit memory construction.

historical contexts into retrieval-friendly databases. This involves performing coreference resolution (Liu et al., 2026), constructing structured memories (Chhikara et al., 2025), and augmenting raw texts with generative metadata like captions, session summaries, and keywords (Nie et al., 2026a). During the retrieval phase, existing systems seek to bridge the semantic gap by optimizing query representations through rewriting (Ma et al., 2023), and frequently introduce additional reranker models to refine the candidate list (Li et al., 2026). Furthermore, the emerging paradigm of agentic RAG (Singh et al., 2025) integrates autonomous agents to orchestrate dynamic query routing and multi-step iterative retrieval, aiming to compensate for the inherent limitations of static embeddings through sophisticated workflow orchestration (Nie et al., 2026a; Liu et al., 2026). Although these approaches alleviate certain issues, they incur substantial computational overhead, increase inference latency, and remain suboptimal.

This paper introduces **EvoEmbedding**, which overcomes the flaws of static models at the representation level by generating contextually *evolvable* embeddings. Specifically, it maintains a fixed-capacity latent memory to preserve evolving contextual states. Upon receiving new inputs (e.g., streaming text chunks or dialogue turns), it updates this latent memory and generates context-aware embeddings in parallel. This design allows EvoEmbedding to recurrently compress ongoing contexts, injecting rich temporal dynamics and cross-segment associations into the representations. To facilitate training, we construct **EvoTrain-180K**, a diverse dataset tailored for the joint optimization of memory and retrieval capabilities. Given the wide variation in context lengths within this dataset, we propose the memory queue and segment-batching techniques. These techniques successfully prevent representation collapse and improve training efficiency by 3.8× without curriculum learning (Bulatov et al., 2024). Finally, we employ lightweight LoRA adaptation to equip general-purpose LLMs with strong representation capabilities while avoiding catastrophic forgetting. By applying this unified architecture across different base models, we develop the EvoEmbedding family, scaled at 0.8B, 2B, and 4B[1] parameters for broad applicability.

Extensive experiments across 10 benchmarks encompassing diverse tasks, domains, and context scales demonstrate that EvoEmbedding delivers: (i) **State-of-the-art retrieval performance**. As shown in Table 1, EvoEmbedding yields the highest overall accuracy across eight long-context benchmarks. It surpasses generalist KaLM-Embedding-Gemma3-12B (Zhao et al., 2025) by +7.8% and Qwen3-Embedding-8B (Zhang et al., 2025) by +11.5%. (ii) **Strong scalability and generalization**. EvoEmbedding generalizes to downstream long-context understanding and personalization tasks, effectively handling 128K contexts (10× its maximum training window, and >100× its average sample length of 1.2K). Furthermore, as depicted in Fig. 1 (Right), naive RAG with EvoEmbedding-4B achieves the highest accuracy with minimal token consumption on LongMemEval-s (Wu et al., 2026), substantially outperforming memory-based methods (77.6% vs. A-MEM's 65.2% and LightMem's 70.2%) (Fang et al., 2026; Xu et al., 2026a). (iii) **Enhancement of agentic memory systems**. EvoEmbedding seamlessly integrates into existing agentic memory systems and delivers substantial gains (e.g., +19.2% for A-MEM and +13.5% for LightMem), consistently outperforming strong baselines such as Qwen3-Reranker-4B and complex reasoning-based retrieval strategies. (iv) **Tem-**

[1]Unless otherwise specified, EvoEmbedding denotes our flagship 4B variant throughout this paper.

**poral retrieval capabilities**. We show our model is inherently suited for temporal RAG. When confronted with temporal keywords such as '*firstly*' or '*lastly*', the query-context similarities peak distinctly at the targeted historical stages, successfully decoupling temporal intents from coarse textual semantics. Overall, these compelling advantages highlight the potential of EvoEmbedding to serve as the next-generation embedding model for long-context retrieval.

## 2 Related Work

**Text Embedding Models.** Text embedding models are widely applied in natural language processing (NLP) tasks such as retrieval and reranking (Zhao et al., 2025). A representative early milestone is Sentence-BERT (Reimers & Gurevych, 2019), which encodes variable-length text into dense semantic vectors and evaluates text relevance via similarity. Recently, leveraging LLMs as embedding models has emerged as the dominant trend. By repurposing generative LLMs for representation learning, pioneering models like E5 (Wang et al., 2024), Qwen3-Embedding (Zhang et al., 2025), and KaLM-Embedding (Zhao et al., 2025) have achieved remarkable performance across diverse retrieval benchmarks (Muennighoff et al., 2023). When confronted with extended contexts, the standard practice is to segment the long input into fixed-size chunks and encode them independently. However, this static and fragmented representation severely disrupts the temporal continuity and contextual associations between segments, frequently leading to imprecision and redundancy during the subsequent retrieval process (Gao et al., 2023; Cuconasu et al., 2024).

**Agentic RAG for Long-Context.** To overcome the limitations of static embeddings, agentic RAG optimizes the retrieval pipeline during information storage and query execution (Singh et al., 2025). On the one hand, existing works strive to construct retrieval-friendly databases (Chhikara et al., 2025; Fang et al., 2026), introduce auxiliary storage structures such as knowledge graphs (Han et al., 2024; Guo et al., 2024) and various memory architectures (Hu et al., 2025), and integrate post-processing modules like rerankers (Zhang et al., 2025) to boost retrieval performance. On the other hand, frontier approaches leverage autonomous agents to orchestrate the querying process, introducing techniques such as rewriting (Ma et al., 2023), reasoning (Nie et al., 2026a), and multi-agent collaboration (Nguyen et al., 2025) to balance computational overhead and retrieval precision. Despite these advancements, such pipelines remain suboptimal due to lost global context and incur prohibitive latency in real-time scenarios.

**Latent Memory and Retrieval.** Early studies have explored maintaining long-term memory internally to bypass external retrieval altogether. For instance, the Recurrent Memory Transformer (RMT)(Bulatov et al., 2022; 2024) extends the context window to over 1M tokens, while maintaining linear computational complexity by continuously updating a latent memory in place. To scale up latent memory, M+ (Wang et al., 2025) employs a co-trained retriever to dynamically recall compressed hidden states during generation. Furthermore, LatentRAG (Zheng & Worring, 2026) and LAnR (Nguyen et al., 2026) explore a more integrated paradigm, jointly performing reasoning, retrieval, and generation within the model's latent space. While these approaches demonstrate the potential of latent representations, existing mechanisms remain deeply coupled with the autoregressive generation process of LLMs. This architectural entanglement severely limits their practical applicability, as they require full white-box access to internal hidden states, rendering them incompatible with closed-source commercial models accessed via APIs. Moreover, forcing a single model to simultaneously manage latent memory and token generation without explicit isolation can lead to unpredictable behaviors such as hallucinations (Xu et al., 2026b).

## 3 Methodology

### 3.1 EvoEmbedding

**Overview.** EvoEmbedding sequentially processes segments[2] split from a long input sequence. As illustrated in Fig. 2, given the current input segment $x_t$ and the latent memory $\mathbf{M_{t-1}}$, it performs memory evolution and representation generation tasks in parallel. This process can be formulated as:

$$\begin{aligned} \tilde{\mathbf{M}}_\mathbf{t} &= \pi_{\theta_m}(x_t, \mathbf{M_{t-1}}), \\ \mathbf{v_t} &= \pi_{\theta_r}(x_t, \mathbf{M_{t-1}}), \end{aligned} \tag{1}$$

[2]A segment refers to a text unit to be retrieved, such as a single sentence or a conversational turn.

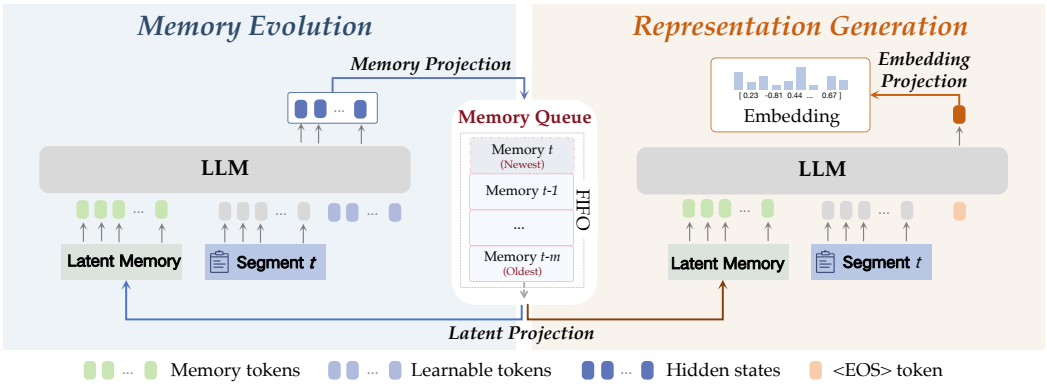



Figure 2: **Overview of the proposed EvoEmbedding.** The model performs two parallel tasks for the input segment at step $t$. (Left) **Memory Evolution**: The LLM compresses the segment and integrates the previous memory into learnable tokens, which are projected to update a FIFO latent memory queue. (Right) **Representation Generation**: The historical latent memory is combined with the current segment to generate a contextually evolvable embedding for retrieval.

where $\theta_m$ and $\theta_r$ denote the task-relevant parameters of the model $\pi$, $\tilde{\mathbf{M}}_\mathbf{t} \in \mathbb{R}^{K\times D}$ represents the newly generated $K$ latent tokens, and $\mathbf{v_t} \in \mathbb{R}^{D_{emb}}$ is the corresponding vector of $x_t$. During the query phase, only the representation generation task is executed to obtain the corresponding retrieval vector. This formulation mirrors the core design of EvoEmbedding: it dynamically maintains a latent memory as the global semantic context and jointly encodes this context with the current input to produce evolvable embeddings.

**Latent Memory Queue.** We design a queue mechanism to manage and update the latent memory of EvoEmbedding. Specifically, the updated memory $\mathbf{M_t}$ is constructed as:

$$\mathbf{M_t} = \text{Queue}(\mathbf{M_{t-1}}, f_m(\tilde{\mathbf{M}}_\mathbf{t})), \tag{2}$$

where $\mathbf{M}_t \in \mathbb{R}^{C\times D}$ is a First-In-First-Out (FIFO) queue matrix. The memory capacity is defined as $C = L \times K$, designed to store the latent representations generated from the most recent $L$ steps. $f_m(\cdot)$ is a projector to map the newly generated tokens $\tilde{\mathbf{M}}_\mathbf{t}$ into the shared memory space. The rationale behind this queue-based design is twofold: (i) **Bounded loop:** It guarantees that a single historical memory is loop-encoded at most $L$ times. This avoids the collapse caused by recurrent encoding of memory (Yu et al., 2026), allowing EvoEmbedding to be directly trained on long contexts without the need for curriculum learning (see Table 4). (ii) **Bounded Capacity:** By strictly limiting the memory size, it bounds the computational complexity and explicitly forces the model to learn to fuse new knowledge with historical states at each step. Compared to M+ (Wang et al., 2025), which uses cached layer-wise features as memory, we only store $C = 512$ latent tokens, whose memory footprint is roughly equivalent to that of an encoded image.

**Dynamic Segment-Batching.** We introduce a dynamic Segment-Batching technique to overcome the efficiency challenges of processing segment-level inputs sequentially during both training and inference. Instead of executing the forward pass segment-by-segment, EvoEmbedding processes $k$ consecutive segments in parallel. We dynamically determine $k$ to ensure that the total length of the concatenated inputs does not exceed a predefined threshold (e.g., 2048 tokens). Accordingly, the memory evolution can be formulated as the batched form of Eq. (1), calculated by $\tilde{\mathbf{M}}_{\mathbf{t:t+k}} = \pi_{\theta_m}(x_{t:t+k}, \mathbf{M_{t-1}})$. The subsequent memory queue updates and embedding generation processes remain consistent with the previous formulation. This design not only achieves a 3.8× speedup but also improves performance (see Table 4).

### 3.2 Training and Optimization

EvoEmbedding is initialized from general language models, such as the Qwen series (Yang et al., 2025; Team, 2026), allowing us to build a model family scaled at 0.8B, 2B, and 4B parameters. We employ a multi-LoRA (Hu et al., 2022) design to decouple the memory evolution and representation generation capabilities, allowing for flexible switching during inference. Given an input sample consisting of $t$ segments $\{x_i\}_{i=1}^{t}$ and a query $q$, we first construct the latent memory $\mathbf{M_t}$ and obtain

the embeddings $\{\mathbf{v_i}\}_{i=1}^{t}$ and $\mathbf{v_q}$ based on Eq. (1). We then jointly optimize the model using a combined objective:

$$\mathcal{L} = \mathcal{L}_{mem} + \mathcal{L}_{con}, \tag{3}$$

where $\mathcal{L}_{mem}$ is the memory generation loss and $\mathcal{L}_{con}$ is the contrastive representation loss. The first term $\mathcal{L}_{mem}$ is measured using standard cross-entropy. We use the generated latent memory $\mathbf{M}_t$ and the query $q$ as context to predict the target answer $y$:

$$\mathcal{L}_{mem} = -\sum_{j=1}^{|y|} \log P(y_j \mid y_{<j}, q, \mathbf{M}_t). \tag{4}$$

Crucially, during this prediction step, we keep the backbone LLM parameters frozen and deactivate all LoRA adapters. This design ensures that the loss backpropagates through the frozen backbone directly into $\mathbf{M}_t$, implicitly forcing the memory module $\theta_m$ to generate latent states that are compatible with the base LLM's native semantic space.

The second term $\mathcal{L}_{con}$ is the contrastive loss designed to align the query representation with the relevant contexts. Unlike standard retrieval settings (Zhang et al., 2025; Zhao et al., 2025), our candidate pool is dynamically partitioned from the $t$ segments of the current sample. Let $\mathcal{P} = \{\mathbf{v}_i^+\}_{i=1}^{P}$ denote the set of embeddings for the positive segments containing the supporting evidence, and $\mathcal{N} = \{\mathbf{v}_j^-\}_{j=1}^{N}$ denote the set of embeddings for the negative segments, where $P + N = t$. To optimize across multiple positive targets and balance the learning difficulty across varying context lengths, we formulate a length-weighted multi-positive contrastive loss:

$$\mathcal{L}_{con} = \frac{\log(N+1)}{P} \sum_{i=1}^{P} \left( -\log \frac{\exp(\mathbf{v}_q^\top \mathbf{v}_i^+ / \tau)}{\exp(\mathbf{v}_q^\top \mathbf{v}_i^+ / \tau) + \sum_{j=1}^{N} \exp(\mathbf{v}_q^\top \mathbf{v}_j^- / \tau)} \right), \tag{5}$$

where $\mathbf{v}_q = \pi_{\theta_r}(q, \mathbf{M}_T)$ represents the normalized embedding of the query encoded with the final memory state, and $\tau = 0.1$ is a temperature hyperparameter. The term $\log(N+1)$ serves as a length-weighting factor. Since the number of negative distractors $N$ varies significantly with the input length, this factor adaptively calibrates the loss scale, ensuring stable optimization across highly variable sequence lengths.

### 3.3 Construction of EvoTrain-180K

We construct **EvoTrain-180K** specifically for the training of EvoEmbedding. It is a large-scale, multi-domain dataset tailored for long-context retrieval. As shown in Fig. 5 (Appendix B), we design an automated three-stage data synthesis pipeline: (1) *Raw Context Construction* randomly samples multi-source texts from FineWeb, synthesized dialogues, and user memories; (2) *Dynamic QA Generation* leverages diverse LLMs guided by 40+ query templates for semantic and reasoning-based queries; and (3) *Retrieval Formulation and Verification* employs `Gemini-3.1-Pro-Preview` to annotate positive targets $v^+$ and filter out hallucinations.

Through this rigorous pipeline, we construct 184,137 high-quality samples to jointly train EvoEmbedding's memory and retrieval capabilities. To ensure training efficiency, each sample is strictly constrained to a maximum of 12K tokens and 256 segments. Although trained on less than 1% of the data volume used by contemporary embedding models (Zhao et al., 2025) and with a training context length under one-tenth of that in testing scenarios (Wu et al., 2026; Zhao et al., 2026b; Nie et al., 2026a), EvoEmbedding demonstrates exceptional scalability and robust generalization. Detailed protocols for each stage are provided in Appendix B.

## 4 Experiments

### 4.1 Experimental Setup

**Benchmarks.** We evaluate EvoEmbedding across 10 benchmarks spanning two tracks: (1) *Retrieval Tasks*: five multi-domain document datasets (ESG-Reports, MLDR, CovidQA, PeerQA, QASPER) (Zhao et al., 2026b) and three conversational memory datasets (REALTALK (Lee et al., 2025), LoCoMo (Maharana et al., 2024), LongMemEval (Wu et al., 2026)); (2) *Generation Tasks*:

Table 1: Retrieval performance across diverse long-context and conversational benchmarks. We report Recall@10 and NDCG@10 following the evaluation protocol (Zhao et al., 2026b). The **EvoEmbedding family** achieves the best overall results across all scales, outperforming strong established baselines (e.g., Qwen3-Embedding-8B and KaLM-12B) with much smaller parameter sizes and embedding dimensions. The **best** results are highlighted, and the second best are underlined.

| Model | Size | Dim | Metric | ESG-Reports | LoCoMo | LongMemEval | REALTALK | QASPER | PeerQA | CovidQA | MLDR | Overall |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Jina-v5-text-small | 0.7B | 1024 | R@10 | 53.3 | 53.1 | 85.8 | 45.9 | 73.1 | 40.9 | 90.3 | 96.0 | 67.3 |
| Multilingual-e5-large | 0.6B | 1024 | R@10 | 51.4 | 65.6 | 80.5 | 54.5 | 72.6 | 40.6 | 89.5 | 97.0 | 69.0 |
| BGE-M3 | 1.2B | 1024 | R@10 | 51.0 | 55.7 | 77.8 | 52.4 | 66.4 | 37.3 | 87.2 | 97.0 | 65.6 |
| KaLM-Embedding-Gemma3 | 12B | 3840 | R@10 | 63.6 | 58.4 | **93.0** | 54.9 | 73.8 | 44.7 | 93.0 | **100.0** | 72.7 |
| Qwen3-Embedding-0.6B | 0.6B | 1024 | R@10 | 52.8 | 46.8 | 84.7 | 46.4 | 69.7 | 37.1 | 91.6 | 95.0 | 65.5 |
| Qwen3-Embedding-4B | 4B | 2560 | R@10 | 57.9 | 41.3 | 58.7 | 42.7 | 72.9 | 36.0 | 93.2 | 96.0 | 62.3 |
| Qwen3-Embedding-8B | 8B | 4096 | R@10 | 63.6 | 49.6 | 87.9 | 50.4 | 70.5 | 41.2 | 93.9 | 95.0 | 69.0 |
| **EvoEmbedding-0.8B (Ours)** | 0.8B | 1024 | R@10 | 85.7 | 63.0 | 83.0 | 58.0 | 83.1 | 48.1 | 94.4 | 98.0 | 76.7 |
| **EvoEmbedding-2B (Ours)** | 2B | 1024 | R@10 | **86.7** | 74.1 | 90.6 | 60.7 | **87.0** | **51.8** | **95.0** | 98.0 | **80.5** |
| **EvoEmbedding-4B (Ours)** | 4B | 1024 | R@10 | 84.0 | **76.3** | 91.7 | **62.6** | 85.1 | 51.7 | 94.9 | 98.0 | **80.5** |
| Jina-v5-text-small | 0.7B | 1024 | N@10 | 36.8 | 39.4 | 74.6 | 36.9 | 50.7 | 30.3 | 73.0 | 76.5 | 52.3 |
| Multilingual-e5-large | 0.6B | 1024 | N@10 | 35.2 | 50.1 | 69.9 | 44.9 | 51.3 | 32.4 | 69.3 | 80.9 | 54.2 |
| BGE-M3 | 1.2B | 1024 | N@10 | 35.9 | 41.8 | 65.6 | 42.0 | 44.2 | 28.2 | 65.1 | 77.0 | 50.0 |
| KaLM-Embedding-Gemma3 | 12B | 3840 | N@10 | 49.5 | 42.8 | **81.4** | 44.4 | 51.5 | 32.5 | 78.1 | 76.4 | 57.1 |
| Qwen3-Embedding-0.6B | 0.6B | 1024 | N@10 | 37.4 | 34.2 | 72.8 | 37.2 | 47.3 | 27.3 | 74.2 | 76.6 | 50.9 |
| Qwen3-Embedding-4B | 4B | 2560 | N@10 | 40.8 | 29.9 | 47.1 | 34.5 | 50.2 | 27.2 | 76.7 | 77.8 | 48.0 |
| Qwen3-Embedding-8B | 8B | 4096 | N@10 | 43.2 | 36.4 | 77.6 | 41.2 | 47.4 | 30.1 | **79.4** | 77.5 | 54.1 |
| **EvoEmbedding-0.8B (Ours)** | 0.8B | 1024 | N@10 | 57.9 | 48.8 | 66.7 | 45.3 | 61.7 | 38.0 | 75.6 | 78.1 | 59.0 |
| **EvoEmbedding-2B (Ours)** | 2B | 1024 | N@10 | **69.9** | 58.3 | 76.1 | 47.8 | 66.0 | 38.6 | 79.1 | **81.6** | 64.7 |
| **EvoEmbedding-4B (Ours)** | 4B | 1024 | N@10 | 66.0 | **61.7** | 78.8 | **49.2** | **66.9** | **41.1** | 77.6 | 80.6 | **65.2** |

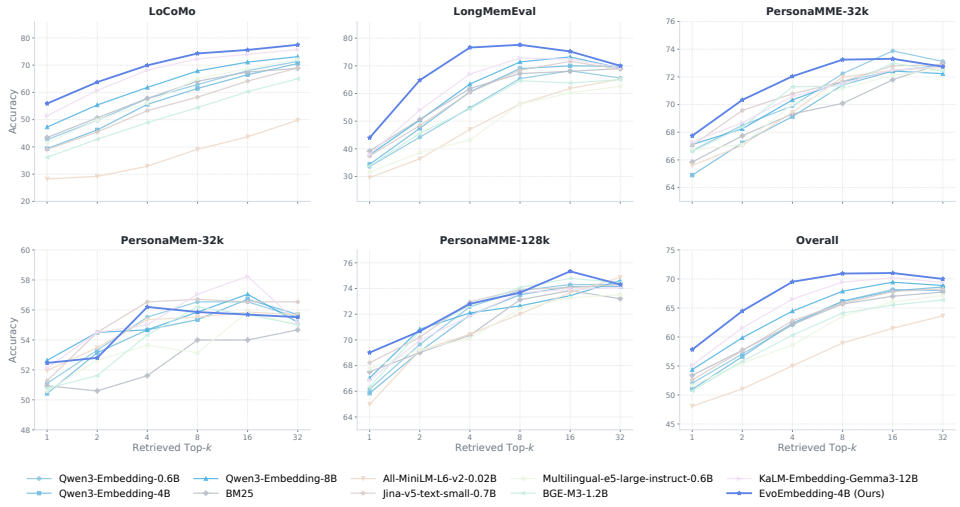



Figure 3: Generation accuracy (%) of a naive RAG pipeline using different retrieval methods. EvoEmbedding-4B achieves the best overall performance across different retrieval scales.

downstream long-context and personalization evaluation on LoCoMo, LongMemEval, PersonaMem (32K) (Jiang et al., 2025), and PersonaMME (32K/128K) (Nie et al., 2026a).

**Baselines.** We benchmark against three categories of baselines: (1) *Embedding & Lexical Retrievers*: dense models across diverse scales (Qwen3-Embedding, KaLM-Embedding, BGE-M3, Multilingual-e5, Jina-v5, All-MiniLM) (Zhang et al., 2025; Zhao et al., 2025; Chen et al., 2024; Wang et al., 2024) and BM25 (Robertson & Zaragoza, 2009); (2) *Agentic Memory Systems*: memory-augmented architectures including Mem0 (Chhikara et al., 2025), LightMem (Fang et al., 2026), A-MEM (Xu et al., 2026a), and MemoryOS (Hu et al., 2025); (3) *Retrieval Strategies*: reranking via Qwen3-Reranker-4B and iterative reasoning-based retrieval (Nie et al., 2026a).

**Implementation Details.** For retrieval-centric tasks, we report standard information retrieval metrics, including Recall@$k$ and Normalized Discounted Cumulative Gain (NDCG@$k$) at various top-$k$ cutoffs (Zhao et al., 2026b). For generation-centric tasks, we employ Qwen3-30B-A3B as the generation model and GPT-4o-mini as the judge model for automated evaluation, following the protocol in (Fang et al., 2026). By default, the capacity of the latent memory queue is set to $C = 512$ tokens, with $K = 16$ latent tokens generated per segment step. The final projected dimension for the embedding is set to $D_{emb} = 1024$. All training and evaluation procedures are conducted on a single server equipped with 8 NVIDIA H800 GPUs.

Table 2: Evaluation results on LongMemEval. A naive RAG pipeline with EvoEmbedding achieves the highest overall accuracy, surpassing agentic memory and full-context baselines. Evaluation spans six categories: Temporal Reasoning (Temp), Multi-Session Dialogue (Multi), Knowledge-Update (Knowledge), Single-User (User), Single-Assistant (Assistant), and Single-Preference (Preference).

| Model | Temp | Multi | Knowledge | User | Assistant | Preference | Overall |
|---|---|---|---|---|---|---|---|
| Full Context | 33.1 | 35.6 | 76.9 | 82.9 | 87.5 | 50.0 | 54.8 |
| *Agentic Memory Systems* | | | | | | | |
| Mem0 | 41.9 | 28.1 | 28.6 | 55.3 | 26.1 | **81.8** | 39.5 |
| MemoryOS | 28.6 | 36.8 | 61.5 | 72.9 | 92.9 | 33.3 | 49.6 |
| A-MEM | 51.9 | 51.1 | 76.9 | 90.0 | 96.4 | 40.0 | 65.2 |
| LightMem | 54.2 | 51.9 | 66.7 | 80.0 | 31.3 | 80.0 | 70.2 |
| *NaiveRAG with various embedding models* | | | | | | | |
| All-MiniLM-L6-v2 | 40.6 | 34.6 | 70.5 | 77.1 | 96.4 | 60.0 | 56.2 |
| Qwen3-Embedding-8B | 57.9 | 62.4 | 76.9 | 90.0 | 98.2 | 63.3 | 71.4 |
| KaLM-Embedding-Gemma3-12B | 60.9 | 58.6 | 80.8 | 92.9 | 98.2 | 73.3 | 72.8 |
| **EvoEmbedding-4B (Ours)** | **63.2** | **71.4** | **84.6** | **98.6** | **100.0** | 60.0 | **77.6** |

## 4.2 Main Results

**(a) EvoEmbedding achieves the best overall performance on both retrieval and downstream generation tasks, demonstrating strong scalability and generalization.** Table 1 and Fig. 3 report the retrieval and generation performance of EvoEmbedding across a series of long-context and conversational benchmarks, respectively, where the latter is evaluated using a naive RAG pipeline with top-$k$ retrieved segments as context. More results, including comparisons showing our native memory outperforms dedicated architectures (e.g., Metis (Zhang et al., 2026)), are in Appendix A.

Compared to well-established embedding baselines, our flagship EvoEmbedding-4B establishes the highest Overall Recall@10 (80.5) and NDCG@10 (65.2) on retrieval tasks, surpassing the runner-up KaLM-Embedding-Gemma3 (Zhao et al., 2025) by substantial absolute margins of 7.8% and 8.1%. Meanwhile, our other variants (0.8B and 2B) also exhibit superior performance against much larger baselines. On generation tasks, our method consistently achieves the best overall performance across varying Top-$k$ ($k \in \{1, 2, 4, 8, 16, 32\}$) retrieval budgets, as plotted in Fig. 3. Notably, while the accuracy on LoCoMo rises steadily up to $k = 32$ without saturation, other datasets generally peak around $k = 8$ or $k = 16$ and slightly decline thereafter due to the distracting noise introduced by excessively large context windows. These results demonstrate (1) the ***effectiveness*** of our approach: utilizing only 180K synthetic samples and standard SFT, EvoEmbedding outperforms peers like KaLM-Embedding-Gemma3 that rely on multi-stage training over 100M data points, highlighting a fundamental architectural advantage in long-context retrieval; and (2) its ***scalability*** and ***generalization***: despite being trained exclusively on shorter samples (maximum 12K tokens, averaging 1.2K), the model effectively generalizes to 128K-length test scenarios and out-of-domain tasks such as personalized retrieval. This confirms that the high-quality, evolvable representations generated by our model can reliably provide solid grounding for complex retrieval scenarios.

**(b) Naive RAG with EvoEmbedding surpasses specialized agentic memory systems.** Tables 2 and 9 (in Appendix C.5) detail the fine-grained performance on the LongMemEval and LoCoMo conversational benchmarks. Remarkably, a standard naive RAG pipeline powered by EvoEmbedding-4B, utilizing only the retrieved Top-8 segments, consistently surpasses agentic memory architectures (e.g., MemoryOS and Mem0) (Hu et al., 2025; Chhikara et al., 2025). For instance, on LongMemEval, EvoEmbedding establishes a new state-of-the-art accuracy of 77.6%, significantly outperforming the best agentic baseline (LightMem, 70.2%). Notably, it achieves near-perfect accuracy on the Single-User (98.6%) and Single-Assistant (100.0%) subtasks of LongMemEval. However, on the Temporal and Open-domain subtasks of LoCoMo, EvoEmbedding falls slightly behind the highly specialized LightMem, despite outperforming other static embedding baselines.

Despite this, EvoEmbedding surpasses the full-context baseline on LongMemEval (an absolute gain of +22.8%) and closely approaches the full-context upper bound on LoCoMo (with a mere 0.6% gap), which can be further improved to 77.5% as the retrieval budget increases (Appendix C). Furthermore, as illustrated in Fig. 1 (Right), our method resolves the massive token overhead issue typical of agentic systems, incurring *zero* additional token cost since it requires no separate memory construction phase[3]. This proves that EvoEmbedding can successfully filter out noisy, outdated

[3]The latent memory of EvoEmbedding is constructed during the encoding phase and does not rely on the generator model (i.e., Qwen3-30B-A3B) at test time.

Table 3: Comparison of EvoEmbedding as a plug-and-play module against reranking and reasoning strategies on LoCoMo. EvoEmbedding achieves superior overall performance.

| System | Method | GPU (G) | Single-hop | Multi-hop | Temporal | Open-domain | Overall |
|---|---|---|---|---|---|---|---|
| **A-MEM** | Original | - | 42.91 | 13.71 | 32.29 | 56.48 | 43.57 |
| | + Reasoning | - | 43.26 $_{+0.4}$ | 15.26 $_{+1.6}$ | 35.42 $_{+3.1}$ | 58.98 $_{+2.5}$ | 45.52 $_{+2.0}$ |
| | + Qwen3-Reranker-4B | 14.55 | 61.70 $_{+18.8}$ | 22.12 $_{+8.4}$ | 43.75 $_{+11.5}$ | 76.46 $_{+20.0}$ | 60.39 $_{+16.8}$ |
| | + **EvoEmbedding-4B (Ours)** | 15.27 | **67.38** $_{+24.5}$ | **24.92** $_{+11.2}$ | **45.83** $_{+13.5}$ | **77.53** $_{+21.1}$ | **62.73** $_{+19.2}$ |
| **LightMem** | Original | - | 41.13 | 23.36 | 41.67 | 46.85 | 40.58 |
| | + Reasoning | - | 48.23 $_{+7.1}$ | 24.92 $_{+1.6}$ | 39.58 $_{-2.1}$ | 49.94 $_{+3.1}$ | 43.77 $_{+3.2}$ |
| | + Qwen3-Reranker-4B | 14.75 | 55.32 $_{+14.2}$ | **29.28** $_{+5.9}$ | 44.79 $_{+3.1}$ | 61.12 $_{+14.3}$ | 52.40 $_{+11.8}$ |
| | + **EvoEmbedding-4B (Ours)** | 14.43 | **58.51** $_{+17.4}$ | 28.66 $_{+5.3}$ | **46.88** $_{+5.2}$ | **63.14** $_{+16.3}$ | **54.09** $_{+13.5}$ |
| **MemoryOS** | Original | - | 49.65 | 21.81 | 43.75 | 59.10 | 48.64 |
| | + Reasoning | - | 60.64 $_{+11.0}$ | 28.04 $_{+6.2}$ | 47.92 $_{+4.2}$ | 71.94 $_{+12.8}$ | 59.22 $_{+10.6}$ |
| | + Qwen3-Reranker-4B | 19.53 | 64.18 $_{+14.5}$ | 38.01 $_{+16.2}$ | 52.08 $_{+8.3}$ | **83.47** $_{+24.4}$ | 68.51 $_{+19.9}$ |
| | + **EvoEmbedding-4B (Ours)** | 16.27 | **68.09** $_{+18.4}$ | **38.94** $_{+17.1}$ | **57.29** $_{+13.5}$ | 82.28 $_{+23.2}$ | **69.09** $_{+20.5}$ |

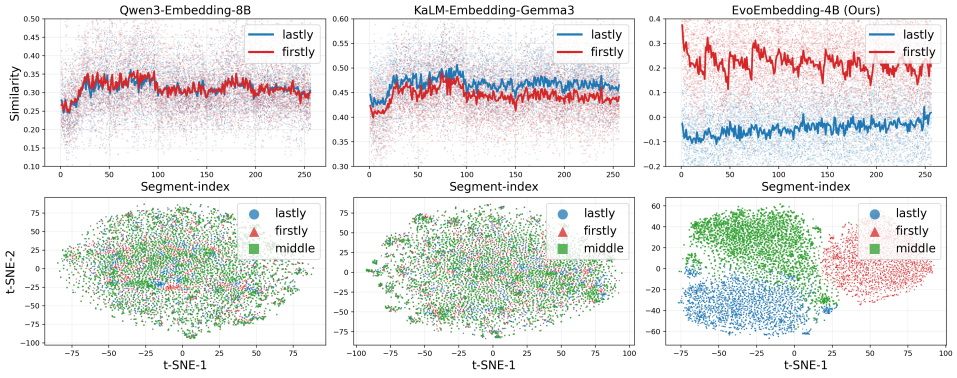



Figure 4: **Analysis of Temporal Query Sensitivity in Long-Context Retrieval.** Using 64 samples (256 segments each) and the query template *"What did I mention [keyword]...?"*, we evaluate three temporal settings: "firstly", "lastly", and "in the middle". **(Top Row):** Average similarity curves. EvoEmbedding shows a chronological increase for "lastly" and a sharp peak at initial segments for "firstly", indicating accurate retrieval from both ends. **(Bottom Row):** t-SNE visualization of query-conditioned representations (Hadamard product). Unlike entangled baselines, EvoEmbedding's latent space is highly sensitive to temporal semantics.

histories that typically distract the LLM generator. These results indicate that by embedding temporal and context awareness directly into the representations, our method achieves superior performance in long-context retrieval.

**(c) EvoEmbedding enhances existing memory systems via plug-and-play integration.** We further evaluate EvoEmbedding's compatibility as a plug-and-play enhancement to upgrade existing memory pipelines. We reproduce A-MEM, LightMem, and MemoryOS on the LoCoMo benchmark, using All-MiniLM-L6-v2 as the baseline retriever. To test the effectiveness of different context enhancement strategies, we restrict the final generator context to a highly constrained Top-20 (resulting in lower baseline scores than originally reported by (Fang et al., 2026). In this setting, both EvoEmbedding and Qwen3-Reranker-4B (Zhang et al., 2025) are integrated to rerank the initially retrieved 256 segments and select the top 20 segments to form the final context. As a comparative baseline, the reasoning strategy employs a 2-turn thought-and-retrieval context collection process, retrieving 10 non-overlapping segments in each turn. It incurs no additional GPU memory overhead, but requires two extra generation calls.

As shown in Table 3, EvoEmbedding achieves the best overall performance, providing absolute gains over the original baselines (e.g., +19.2% on A-MEM and +20.5% on MemoryOS). It consistently outperforms both the reasoning and reranking strategies across all three frameworks. Moreover, the GPU memory overheads of our model and Qwen3-Reranker-4B (evaluated with a batch size of 64) are comparable. This is because EvoEmbedding maintains only a fixed-size latent memory to avoid quadratic complexity. These results demonstrate the strong adaptability of EvoEmbedding.

**(d) EvoEmbedding captures contextual order and exhibits strong temporal retrieval capabilities.** To explicitly validate the model's sensitivity to temporal semantics, we conduct a fine-grained analysis

Table 4: Ablation study of key designs of EvoEmbedding on five benchmarks. We report the total training time (h), accuracy (%), and the relative performance drop (↓) when specific training strategies are removed. The results demonstrate that the latent memory mechanisms are fundamental to model performance, while our proposed batching strategies are critical for training efficiency.

| Strategy | Time (h) | LoCoMo | LongMemEval | PersonaMem-32K | PersonaMME-32K | PersonaMME-128K | Overall |
|---|---|---|---|---|---|---|---|
| **EvoEmbedding-4B** | 26.6 | **69.9** | **76.6** | **56.2** | **72.0** | 72.8 | **69.5** |
| w/o Memory Queue | 91.3 | 17.0$_{-52.9}$ | 10.0$_{-66.6}$ | 46.9$_{-9.3}$ | 64.8$_{-7.2}$ | 64.3$_{-8.5}$ | 40.6$_{-28.9}$ |
| w/o Memory Loss | 27.7 | 15.2$_{-54.7}$ | 11.4$_{-65.2}$ | 48.9$_{-7.3}$ | 65.5$_{-6.5}$ | 64.3$_{-8.5}$ | 41.1$_{-28.4}$ |
| w/o Length-Weighting | 26.5 | 68.4$_{-1.5}$ | 73.8$_{-2.8}$ | 54.5$_{-1.7}$ | 71.6$_{-0.4}$ | **73.2**$_{+0.4}$ | 68.3$_{-1.2}$ |
| w/o Segment-Batching | 101.4 | 66.0$_{-3.9}$ | 75.0$_{-1.6}$ | 54.3$_{-1.9}$ | 71.2$_{-0.8}$ | 71.6$_{-1.2}$ | 67.6$_{-1.9}$ |

using queries constrained by time-related keywords (e.g., "firstly", "lastly", and "in the middle"), as illustrated in Fig. 4. The top row displays the average similarity curves across segment indices. Traditional static embeddings, such as Qwen3-Embedding-8B and KaLM-Embedding-Gemma3, exhibit highly entangled and overlapping curves. This indicates that they rely entirely on coarse textual semantics, failing to differentiate the temporal intent of the queries. In sharp contrast, EvoEmbedding successfully decouples these intents: when queried with "firstly", the similarity peaks sharply at the initial segments; when queried with "lastly", the similarity exhibits a clear chronological increase, peaking towards the end of the context.

This temporal awareness is further corroborated by the t-SNE visualizations of the query-conditioned segment representations (Fig. 4, bottom row). While the latent spaces of the baseline models are mixed, EvoEmbedding structures its latent space distinctly. It cleanly separates the representations into distinct, non-overlapping clusters corresponding to their chronological positions. This visual evidence proves that our continuously updated latent memory successfully captures long-context temporal information and seamlessly integrates it into the final representations.

### 4.3 Ablation Studies

**Latent Memory and Queue Design.** Table 4 reports the ablation results across five downstream generation benchmarks. The latent memory queue serves as the core of our model. Removing either the memory queue setting or the memory loss in Eq. (3) leads to a severe representation collapse. Consequently, we observe a catastrophic performance degradation exceeding 50% on LoCoMo and LongMemEval. The performance drop on the other three benchmarks is relatively milder (around 8%) primarily because they are formulated as multiple-choice questions. Crucially, this queue mechanism enables EvoEmbedding to train directly on mixed-length samples without relying on complex curriculum learning (Bulatov et al., 2024). Furthermore, it restricts the generation overhead to only $K = 16$ tokens per step instead of updating the full memory capacity ($C = 512$), which effectively boosts the training efficiency by 3.4×.

**Segment-Batching and Length-Weighting.** First, EvoEmbedding employs a segment-batching technique to process multiple consecutive segments simultaneously, significantly accelerating both training and inference. This strategy yields an almost 3.8× training speedup (reducing time from 101.4 to 26.6 hours) while bringing an overall performance gain of 1.9% (Table 4). Second, we introduce a length-weighting technique to elegantly balance sample difficulty and context length during optimization. This regularization prevents the model from biasing towards shorter sequences, contributing an additional 1.2% improvement to the overall accuracy.

## 5 Conclusion

This paper introduces **EvoEmbedding**, a novel family of embedding models designed to overcome the limitations of static representations in long-context scenarios and agentic workflows. By seamlessly integrating a continuously updated latent memory with sequential text encoding, EvoEmbedding moves beyond static semantic search to achieve precise, context-aware matching. To resolve training inefficiency and representation collapse, we propose a memory queue and dynamic segment-batching techniques. Furthermore, we construct **EvoTrain-180K**, a large-scale dataset tailored to support the training of evolvable embeddings. Extensive evaluations confirm that EvoEmbedding not only establishes the best overall performance across diverse retrieval benchmarks but also successfully integrates into existing retrieval-augmented frameworks to boost their performance. Overall, our work paves a highly scalable and promising path for future long-context information representation.

### AI USE STATEMENT

In this work, we use generative AI tools for synthetic dataset generation, code assistance, and manuscript preparation. Specifically, various LLMs and Gemini-3.1-Pro are employed to synthesize dynamic QA pairs and annotate retrieval targets for the EvoTrain-180K dataset, as detailed in Section 3.3 and Appendix B. Codex is used to assist in developing evaluation scripts, and Gemini-3.1-Pro is employed for manuscript proofreading and polishing. AI tools are not used for scientific ideation or core algorithm design. All AI-assisted outputs were thoroughly reviewed and verified by the authors, who take full responsibility for the entire content and integrity of this paper.

### ETHICS STATEMENT

Our EvoTrain-180K dataset is synthesized from publicly available data and LLM-generated content, with precautions taken to prevent the inclusion of personally identifiable information or harmful material. Specifically, we incorporated an evaluation step in the third stage of our data construction pipeline to filter out low-quality and harmful samples (detailed in Appendix B). The dataset is intended solely for scientific research in large language models and cannot be weaponized for creating deepfakes, misinformation, or malicious propaganda.

### REPRODUCIBILITY STATEMENT

To ensure reproducibility, our complete source code and evaluation scripts are provided in the supplementary material. Detailed training configurations, hyperparameters, and runtime algorithms are documented in Section 3 and Appendix C, with the statistical results of the EvoTrain-180K dataset elaborated in Appendix B. Also, all source code, the EvoTrain-180K dataset, and pretrained model checkpoints will be publicly released.

## REFERENCES

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *NeurIPS*, 2022.

Aydar Bulatov, Yuri Kuratov, Yermek Kapushev, and Mikhail S Burtsev. Scaling transformer to 1m tokens and beyond with rmt. In *AAAI*, 2024.

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

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv:2504.19413*, 2025.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for rag systems. In *SIGIR*, 2024.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *KDD*, 2024.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, et al. Lightmem: Lightweight and efficient memory-augmented generation. In *ICLR*, 2026.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, et al. Retrieval-augmented generation for large language models: A survey. *arXiv:2312.10997*, 2023.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tian Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. *arXiv:2410.05779*, 2024.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. Retrieval-augmented generation with graphs (graphrag). *arXiv:2501.00309*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. *arXiv:2512.13564*, 2025.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 2025.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J Taylor, and Dan Roth. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. In *COLM*, 2025.

Dong-Ho Lee, Adyasha Maharana, Jay Pujara, Xiang Ren, and Francesco Barbieri. Realtalk: A 21-day real-world dataset for long-term conversation. *arXiv:2502.13270*, 2025.

Yuqing Li, Jiangnan Li, Mo Yu, Guoxuan Ding, Zheng Lin, Weiping Wang, and Jie Zhou. Query-focused and memory-aware reranker for long context processing. *arXiv:2602.12192*, 2026.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. *arXiv:2601.02553*, 2026.

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

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In *EMNLP*, 2023.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *ACL*, 2024.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, et al. A survey of context engineering for large language models. *arXiv:2507.13334*, 2025.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. In *EACL*, 2023.

Minh-Anh Nguyen, Dung D Le, et al. Latent abstraction for retrieval-augmented generation. *arXiv:2604.17866*, 2026.

Thang Nguyen, Peter Chin, and Yu-Wing Tai. Ma-rag: Multi-agent retrieval-augmented generation via collaborative chain-of-thought reasoning. *arXiv:2505.20096*, 2025.

Chang Nie, Chaoyou Fu, Yifan Zhang, Haihua Yang, and Caifeng Shan. Personavlm: Long-term personalized multimodal llms. *arXiv:2604.13074*, 2026a.

Chang Nie, Jiaju Wei, Junlan Feng, Chaoyou Fu, and Caifeng Shan. Light-omni: Reflex over reasoning in agentic video understanding with long-term memory. In *CVPR*, 2026b.

OpenClaw. Openclaw: Your own personal ai assistant. `https://github.com/openclaw/openclaw`, 2026.

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

Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc, 2009.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *ICLR*, 2024.

Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V Vasilakos. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv:2501.09136*, 2025.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual e5 text embeddings: A technical report. *arXiv:2402.05672*, 2024.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending memoryllm with scalable long-term memory. In *ICML*, 2025.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. On the theoretical limitations of embedding-based retrieval. *arXiv:2508.21038*, 2025.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *ICLR*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In *NeurIPS*, 2026a.

Zhongxing Xu, Chengzhi Liu, Qingyue Wei, Juncheng Wu, James Zou, Xin Wang, Yuyin Zhou, and Sheng Liu. More thinking, less seeing? assessing amplified hallucination in multimodal reasoning models. In *NeurIPS*, 2026b.

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

Xinlei Yu, Zhangquan Chen, Yongbo He, Tianyu Fu, Cheng Yang, Chengming Xu, Yue Ma, Xiaobin Hu, Zhe Cao, Jie Xu, et al. The latent space: Foundation, evolution, mechanism, ability, and outlook. *arXiv:2604.02029*, 2026.

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

Zeyu Zhang, Ziliang Guo, Yihang Sun, Xichong Zhang, Xixuan Hao, Zehao Lin, Yang Zhang, Xiaoyan Zhao, Tong Shen, Bo Tang, Zhi-Qin John Xu, Junchi Yan, Haofen Wang, Xu Chen, Feiyu Xiong, Zhiyu Li, and Tat-Seng Chua. Metis: Memory foundation model. Technical report, 2026.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey. *Data Science and Engineering*, 2026a.

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

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

Yijia Zheng and Marcel Worring. Latentrag: Latent reasoning and retrieval for efficient agentic rag. *arXiv:2605.06285*, 2026.

# APPENDIX

## A ANALYSIS OF NATIVE MEMORY CAPABILITIES

To evaluate the native memory capabilities of EvoEmbedding, we benchmark our model under the *No-Context* regime on LoCoMo and NextMem (Zhang et al., 2026), where the model must answer queries purely based on its internal memory state. As shown in Table 5, EvoEmbedding-4B achieves an average of **56.65%** on LoCoMo (Gold) and **52.15%** on NextMem, substantially outperforming its base model Qwen3-4B (+33.4% and +24.5%), which demonstrates that our latent memory actively assimilates and preserves factual history. Moreover, while online parameter-updating methods (Temp-LoRA and $\delta$-Mem) collapse due to catastrophic forgetting, EvoEmbedding-4B consistently surpasses recurrent memory baselines across all scales, outperforming the $7\times$ larger Metis-27B (Zhang et al., 2026) by **+29.91%** on LoCoMo (Gold) and **+1.33%** on NextMem. Notably, on the *Temporal* split, EvoEmbedding-4B achieves **39.33%** (nearly $3\times$ that of Metis-27B's 13.83%), confirming that our bounded FIFO memory queue effectively preserves chronological trajectories and avoids representation collapse.

Table 5: Evaluation of native memory capabilities under the *No-Context* regime on LoCoMo (Gold) and NextMem. Bold indicates the best result, and underline indicates the second best.

| Method | Size | LoCoMo (Gold) | | | | | NextMem | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Single | Multi | Temp. | Open | **Avg.** | SQuAD | Hotpot | L-Mem | LoCoMo | **Avg.** |
| Qwen3-4B (No Context) | 4B | 16.19 | 10.31 | 29.21 | 29.88 | 23.25 | 36.13 | 39.69 | 6.00 | 10.53 | 27.70 |
| Temp-LoRA-4B | 4B | 10.92 | 11.24 | 1.80 | 26.69 | 9.99 | 26.19 | 38.62 | 9.71 | 11.12 | 24.20 |
| Temp-LoRA-27B | 27B | 4.29 | 5.49 | 1.25 | 10.67 | 4.24 | 37.68 | 51.46 | 6.57 | 5.86 | 30.97 |
| $\delta$-Mem | 4B | 12.86 | 10.16 | 3.28 | 20.22 | 10.79 | 20.74 | 33.02 | 9.79 | 10.29 | 20.42 |
| Metis-4B | 4B | 18.90 | 15.29 | 7.03 | 28.37 | 16.31 | 29.62 | 58.13 | 39.36 | 50.48 | 41.69 |
| Metis-9B | 9B | 18.87 | 18.53 | 7.03 | 27.25 | 16.81 | 33.06 | 63.45 | 33.36 | 51.56 | 43.39 |
| Metis-27B | 27B | <u>31.01</u> | <u>27.97</u> | 13.83 | 28.93 | <u>26.74</u> | **43.42** | <u>66.54</u> | <u>39.71</u> | **60.41** | <u>50.82</u> |
| **EvoEmbedding (Ours)** | **4B** | **55.04** | **29.38** | **39.33** | **69.40** | **56.65** | <u>43.06</u> | **67.49** | **46.86** | 58.37 | **52.15** |

## B CONSTRUCTION AND STATISTICS OF EVOTRAIN-180K

To provide a comprehensive understanding of the training data used to optimize EvoEmbedding, we elaborate on the data construction pipeline, synthesis protocols, and detailed statistics of the **EvoTrain-180K** dataset. The dataset comprises a total of 184,137 training instances, meticulously constructed to encapsulate dynamic state transitions and complex temporal reasoning. The overall three-stage synthesis pipeline is illustrated in Fig. 5.

**Three-stage Synthesis Details.**

- **Stage1: Raw Context Construction.** We construct three primary types of contexts: (i) *Diverse web texts*: We randomly sample documents from the FineWeb[4] dataset and process them into sequential segments using a sliding window approach. (ii) *Dialogues*: We employ powerful LLMs to synthesize multi-turn, persona-driven dialogues based on predefined topics. (iii) *Memories*: We extract various types of memory from the raw texts or dialogues to serve as context. This stage yields a vast pool of raw contexts spanning diverse domains, types, and lengths.
- **Stage2: Dynamic QA Generation.** This stage builds QA pairs based on the given contexts and introduces two specific designs to ensure sample diversity: (i) We pre-define over 40 template types (e.g., coreference resolution, temporal understanding) to guide the QA generation. (ii) We utilize LLMs of varying types and sizes to generate questions. This ensures the inclusion of both simple questions for basic semantic matching and complex questions for deep context understanding.
- **Stage3: Retrieval Formulation and Evaluation.** This final stage employs the strong `Gemini-3.1-Pro-Preview`[5] for retrieval labeling and sample verification. The la-

[4]https://huggingface.co/datasets/HuggingFaceFW/fineweb

[5]https://deepmind.google/models/gemini/pro

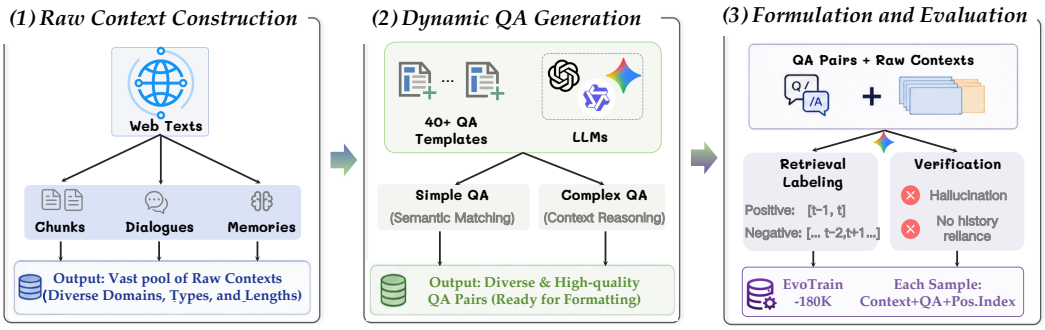



Figure 5: **Construction pipeline of the EvoTrain-180K dataset.** The process comprises three stages: (1) **Raw Context Construction** across diverse domains, formats, and lengths; (2) **Dynamic QA Generation** utilizing LLMs and 40+ templates to produce both semantic and reasoning-based queries; and (3) **Formulation and Verification** to perform positive/negative retrieval labeling and filter out noisy samples (e.g., hallucinations and context-independent queries).

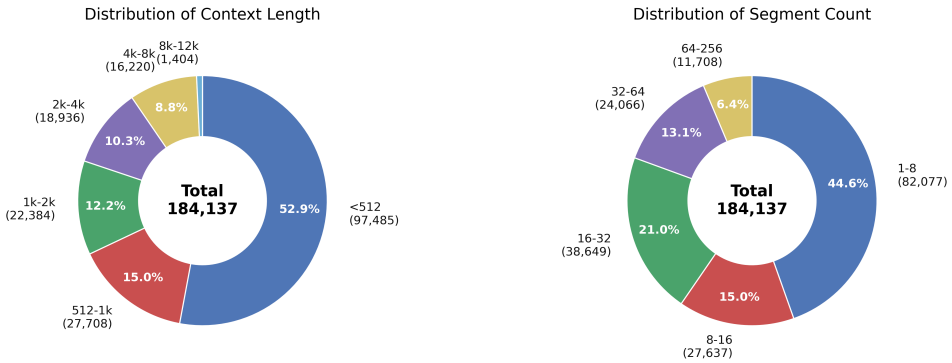



Figure 6: Data distributions of EvoTrain-180K. **(Left)** The distribution of context lengths (in tokens). **(Right)** The distribution of segment counts per sample. The highly varied distributions enable the model to learn robust evolvable representations across mixed-length scenarios.

beling process identifies the indices of query-relevant segments to serve as the positive target $v^+$. The verification process ensures valid QA pairs by ruling out hallucinations and enforcing reliance on provided history over general knowledge.

**Data Distributions.** As illustrated in Fig. 6, EvoTrain-180K features a highly skewed, mixed-length distribution. While the majority of the contexts (52.9%) are relatively short (under 512 tokens), there is a significant long-tail distribution extending up to 12K tokens. Similarly, the segment count per sample spans from as few as 2 segments to as many as 246, with 44.6% of the data falling into the 1-8 segment range. This mixed-length nature is crucial: it allows EvoEmbedding to learn basic semantic matching on shorter sequences while mastering temporal dynamics from longer contexts, ultimately enabling it to generalize to 128K testing contexts during inference.

Table 6: Detailed statistical summary of the EvoTrain-180K dataset.

| Metric | Mean | Min | Max |
|---|---|---|---|
| Context Length (tokens) | 1,289.78 | 25 | 10,270 |
| Segment Count per Sample | 20.57 | 2 | 246 |
| Question Length (words) | 15.59 | 3 | 195 |
| Contrastive Negative Samples | 19.45 | 1 | 245 |

**Algorithm 1:** Memory Evolution

**Require:** Segment $x_t$, queue $\mathbf{M}_{t-1}$
1: Activate memory LoRA adapter $\theta_m$
2: $m_{in} \leftarrow \text{LatentProjector}(\mathbf{M}_{t-1})$
3: $\tilde{x}_t \leftarrow [m_{in}\,;\, x_t\,;\, r_l]$ ▷ $r_l$ are $K$ learnable tokens
4: $\tilde{\mathbf{M}}_\mathbf{t} \leftarrow \text{LLM}(\tilde{x}_t)[-K:]$
5: $\mathbf{M}_t \leftarrow \text{Queue}(\mathbf{M}_{t-1}, f_m(\tilde{\mathbf{M}}_\mathbf{t}))$
6: **return** Updated Memory Queue $\mathbf{M}_t$

**Algorithm 2:** Embedding Generation

**Require:** Input $x_t$ (segment/query), queue $\mathbf{M}_{t-1}$
1: Activate retrieval LoRA adapter $\theta_r$
2: $m_{in} \leftarrow \text{LatentProjector}(\mathbf{M}_{t-1})$
3: $\tilde{x}_t \leftarrow [m_{in}\,;\, x_t\,;\, \texttt{<EOS>}]$
4: $\mathbf{h}_{\text{eos}} \leftarrow \text{LLM}(\tilde{x}_t)[-1]$
5: $\mathbf{v}_\mathbf{t} \leftarrow \text{EmbeddingProjector}(\mathbf{h}_{\text{eos}})$
6: **return** Representation $\mathbf{v}_\mathbf{t}$

**Statistical Summary.** Table 6 summarizes the key metrics of the dataset. The average context length and segment count are approximately 1.3K tokens and 21, respectively. Each training instance is paired with an average of 19.45 contrastive negative samples. Notably, the context lengths and segment counts in our evaluation benchmarks, such as LoCoMo (Maharana et al., 2024), LongMemEval (Wu et al., 2026), and PersonaMME-128k (Nie et al., 2026a), commonly exceed 32K tokens and 500 segments, respectively. Furthermore, the query questions are kept concise (mean length of 15.59 tokens, with 99% of queries being under 52 tokens), ensuring that the retrieval difficulty stems primarily from the complex temporal contexts rather than convoluted question phrasing.

# C More Details about EvoEmbedding

## C.1 Runtime Pipeline

To provide a clear understanding of the EvoEmbedding architecture, we detail the forward pass processes for both memory evolution and embedding generation. As illustrated in Algorithms 1 and 2, the two processes share a highly symmetrical architecture, differing primarily in their specific LoRA adapters, appended tokens, and output projection mechanisms. This design allows the model to seamlessly switch between dynamically updating its latent state and generating evolvable embeddings for retrieval.

## C.2 Training Details

For model initialization, EvoEmbedding-0.8B and -2B are derived from Qwen3.5-0.8B and Qwen3.5-2B (Team, 2026), respectively, while our flagship EvoEmbedding-4B is initialized with Qwen3-4B (Yang et al., 2025). This design ensures a fair comparison against the Qwen3-Embedding baselines, demonstrating that our performance gains stem from the evolving representations rather than from a better base model. The detailed hyper-parameter configurations used for training are summarized in Table 7. To maximize computational efficiency across highly variable sequence lengths, we enable the length-based grouping strategy (`group_by_length = True`). The entire training process finishes within 11,509 steps, yielding a total training time of 26.6 hours, as reported in Table 4.

Table 7: Hyper-parameter settings for the training of EvoEmbedding.

| **Hyper-parameter** | **Value** |
|---|---|
| Learning Rate | $5 \times 10^{-5}$ |
| Batch Size | 16 |
| Training Epochs | 1 |
| Training Steps | 11,509 |
| LR Scheduler | `cosine` |
| LR Scheduler `min_lr` | 0.1 |
| LR Scheduler `num_cycles` | 0.5 |
| Group by Length | True |

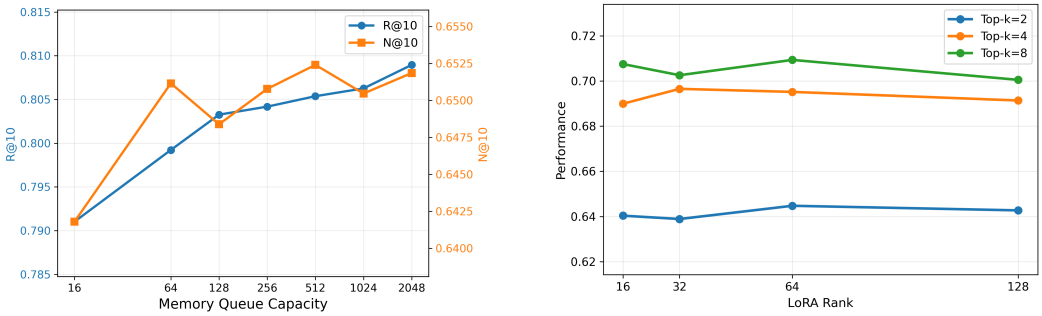



Figure 7: Ablation on memory capacity and training scale. **(Left)** Retrieval performance (Overall R@10 and N@10) across varying memory queue capacities ($C$). A larger queue effectively extends the historical horizon, with performance saturating around $C = 512$. **(Right)** Generation accuracy across varying LoRA Ranks under different Top-$k$ settings. EvoEmbedding demonstrates remarkable parameter efficiency, maintaining highly stable performance even at an extremely low rank ($r = 16$).

Table 8: Efficiency and effectiveness comparison between EvoEmbedding and static embedding models. EvoEmbedding obtains the best LongMemEval performance while requiring substantially lower peak GPU memory, though it incurs additional encoding time due to memory construction.

| **Model** | **Size** | **Encoding Time (s)** ↓ | | **Peak VRAM (GB)** ↓ | **Best Performance** | |
|---|---|---|---|---|---|---|
| | | **Context (Avg.)** | **Query (Avg.)** | | **Top-$k$** | **Accuracy (%)** ↑ |
| Qwen3-Embedding-4B | 4B | **3.80** | **0.026** | 32.3 | $k = 16$ | 70.0 |
| Qwen3-Embedding-8B | 8B | 5.52 | 0.027 | 43.1 | $k = 16$ | 73.2 |
| KaLM-Embedding-Gemma3 | 12B | 9.89 | 0.034 | 69.3 | $k = 4$ | 72.8 |
| **EvoEmbedding (Ours)** | 4B | 22.08 | 0.065 | **20.9** | $k = 8$ | **77.6** |

### C.3 Sensitivity Analysis on Memory Capacity and LoRA Rank

**Memory Capacity.** Fig. 7 (Left) illustrates the retrieval performance across varying memory queue capacities ($C$). As $C$ increases from 16 to 512, we observe a steady upward trend in both R@10 and N@10 metrics. This confirms that a larger queue effectively broadens the historical horizon, allowing the model to better capture long-range temporal dependencies. However, the performance saturates and the NDCG metric peaks when $C$ reaches 512. Beyond this threshold, expanding the queue yields diminishing returns while inevitably increasing memory consumption. Therefore, we set $C = 512$ as the default configuration, striking an elegant balance between precise context tracking and computational efficiency.

**Impact of LoRA Rank.** We further investigate the parameter efficiency by evaluating the generation performance across different LoRA adapter ranks ($r \in \{16, 32, 64, 128\}$). As shown in Fig. 7 (Right), the performance curves remain exceptionally stable across all tested ranks and Top-$k$ settings. This indicates that learning evolvable representations does not rely on massive parameter updates; a lightweight adapter is sufficient to activate these representations while fully preserving the original capabilities of the base LLM. We set the final adapter rank to $r = 64$ as this configuration achieves a near-optimal peak and ensures maximum representation stability when generalizing to complex tasks.

### C.4 Efficiency Analysis

We further analyze the inference efficiency of EvoEmbedding. We use LongMemEval as the testbed and compare EvoEmbedding with strong static embedding models, including Qwen3-Embedding (4B and 8B) and KaLM-Embedding-Gemma3-12B. For a fair comparison, we set the context encoding batch size to 16 for all methods, and report the average encoding time for both context segments and queries, as well as the peak GPU memory usage. During inference, EvoEmbedding processes the input sequentially and continuously maintains a latent memory queue that tracks the evolving user state, whereas static embedding models encode context segments in parallel.

Table 9: Evaluation results on LoCoMo. A naive RAG pipeline with EvoEmbedding outperforms agentic memory systems and other embedding baselines.

| Method | Single-hop | Multi-hop | Temporal | Open-domain | Overall |
|---|---|---|---|---|---|
| Full Context | **87.4** | 69.9 | 51.7 | 57.3 | **74.9** |
| *Agentic Memory Systems* | | | | | |
| Mem0 | 67.7 | 54.3 | 57.0 | 46.9 | 61.7 |
| A-MEM | 67.9 | 57.5 | 27.7 | 43.8 | 56.1 |
| MemoryOS | 72.3 | 62.8 | 33.0 | 51.0 | 61.0 |
| LightMem | 67.0 | 45.8 | **76.3** | **76.8** | 72.6 |
| *NaiveRAG with various embedding models* | | | | | |
| all-MiniLM-L6-v2 | 29.8 | 20.9 | 40.6 | 49.0 | 39.1 |
| Qwen3-Embedding-8B | 82.2 | 60.6 | 41.1 | 53.1 | 67.9 |
| KaLM-Embedding-Gemma3-12B | 87.2 | 62.8 | 47.7 | 52.1 | 72.3 |
| **EvoEmbedding-4B (Ours)** | 86.6 | **71.6** | 49.8 | 56.3 | 74.3 |

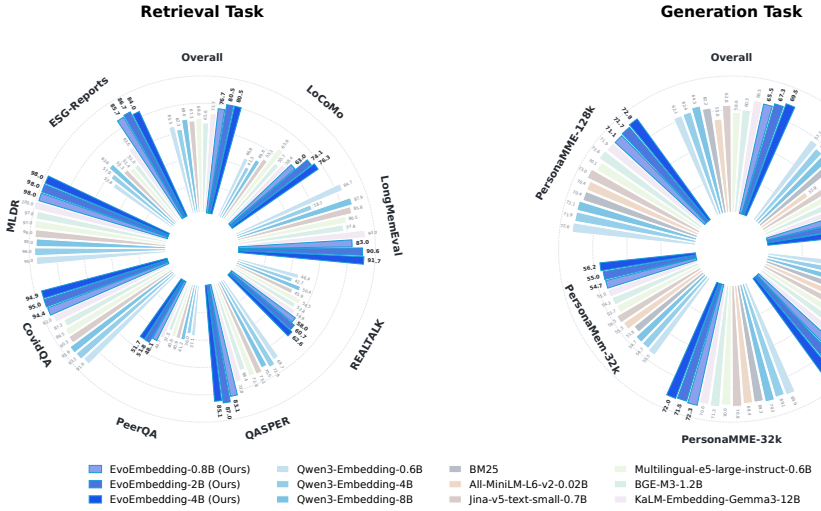



Figure 8: **Performance comparison across long-context retrieval and generation tasks.** Our **EvoEmbedding** family (0.8B, 2B, and 4B) achieves superior results across 10 benchmarks. Notably, our model excels in handling dynamic contexts (e.g., long-term personalization), outperforming both established static and large-scale models. We report Recall@10 for retrieval, and generation accuracy using a naive RAG pipeline based on the Top-4 retrieved segments.

As shown in Table 8, EvoEmbedding trades off context encoding speed for improved accuracy and reduced peak GPU memory usage. Although it requires more time to encode the context, its compact latent memory state substantially lowers memory consumption and leads to the best retrieval performance.

### C.5 More Experimental Results

Here, we present additional evaluation results. First, Fig. 8 provides a comprehensive visual comparison of the overall performance across all 10 benchmarks. Furthermore, Table 10 details the exact accuracy (%) of the naive RAG pipeline across five benchmarks under all evaluated retrieval budgets ($k \in \{1, 2, 4, 8, 16\}$). These numerical results confirm EvoEmbedding's consistent superiority across all retrieval budgets.

## D Discussion and Limitations

**Why EvoEmbedding?** Retrieval without global context is suboptimal. It is trivial to construct adversarial cases that deceive static embedding or reranking models—for instance, using shallow

Table 10: Detailed generation performance (Accuracy %) across five benchmarks and overall average under varying Top-$k$ retrieved contexts ($k \in \{1, 2, 4, 8, 16\}$). The best results are highlighted in **bold**, and the second best are underlined.

| Model | Top-$k$ | LoCoMo | LongMemEval | PersonaMem-32k | PersonaMME-32k | PersonaMME-128k | Overall |
|---|---|---|---|---|---|---|---|
| BM25 | $k=1$ | 43.44 | 39.20 | 50.93 | 65.84 | 67.51 | 53.39 |
| All-MiniLM-L6-v2 | | 28.25 | 29.60 | 51.95 | 65.59 | 64.98 | 48.07 |
| Jina-v5-text-small | | 39.09 | 37.40 | 51.27 | 67.05 | 68.22 | 52.61 |
| Multilingual-e5-large | | 41.56 | 31.60 | 50.59 | 66.73 | 67.75 | 51.65 |
| BGE-M3 | | 36.17 | 33.80 | 50.76 | 66.60 | 66.32 | 50.73 |
| KaLM-Embedding-Gemma3 | | 51.43 | 38.00 | 52.12 | 67.30 | 66.88 | 55.15 |
| Qwen3-Embedding-0.6B | | 42.66 | 33.60 | 51.10 | 66.67 | 66.17 | 52.04 |
| Qwen3-Embedding-4B | | 39.35 | 34.40 | 50.42 | 64.90 | 65.85 | 50.98 |
| Qwen3-Embedding-8B | | 47.27 | 37.80 | **52.63** | 67.11 | 67.04 | 54.37 |
| **EvoEmbedding-0.8B (Ours)** | | 42.40 | 40.20 | 51.61 | 67.49 | 67.83 | 53.91 |
| **EvoEmbedding-2B (Ours)** | | 47.73 | 41.80 | 50.59 | 67.36 | 66.48 | 54.79 |
| **EvoEmbedding-4B (Ours)** | | **55.91** | **44.00** | 52.46 | **67.74** | **69.01** | **57.82** |
| BM25 | $k=2$ | 50.65 | 50.60 | 50.59 | 67.74 | 69.01 | 57.72 |
| All-MiniLM-L6-v2 | | 29.16 | 36.40 | 53.48 | 67.05 | 69.17 | 51.05 |
| Jina-v5-text-small | | 45.45 | 48.40 | **54.50** | 69.58 | 70.20 | 57.63 |
| Multilingual-e5-large | | 49.81 | 38.60 | 52.63 | 67.11 | 69.25 | 55.48 |
| BGE-M3 | | 42.79 | 45.60 | 51.61 | 68.37 | 70.67 | 55.81 |
| KaLM-Embedding-Gemma3 | | 60.71 | 54.00 | **54.50** | 68.69 | 69.96 | 61.57 |
| Qwen3-Embedding-0.6B | | 49.74 | 44.20 | 53.31 | 68.50 | 69.64 | 57.08 |
| Qwen3-Embedding-4B | | 46.23 | 47.40 | 53.14 | 67.24 | 69.09 | 56.62 |
| Qwen3-Embedding-8B | | 55.39 | 50.40 | **54.50** | 68.25 | **70.83** | 59.87 |
| **EvoEmbedding-0.8B (Ours)** | | 56.30 | 59.60 | 52.97 | **71.28** | 69.09 | 61.85 |
| **EvoEmbedding-2B (Ours)** | | 54.61 | 62.60 | 51.27 | 69.77 | 69.25 | 61.50 |
| **EvoEmbedding-4B (Ours)** | | **63.77** | **64.80** | 52.80 | 70.34 | 70.67 | **64.47** |
| BM25 | $k=4$ | 57.79 | 61.80 | 51.61 | 69.32 | 70.36 | 62.18 |
| All-MiniLM-L6-v2 | | 32.92 | 47.00 | 55.35 | 69.45 | 70.43 | 55.03 |
| Jina-v5-text-small | | 53.31 | 60.40 | **56.54** | 70.78 | **72.96** | 62.80 |
| Multilingual-e5-large | | 56.10 | 43.20 | 53.65 | 70.02 | 70.12 | 58.62 |
| BGE-M3 | | 48.90 | 54.40 | 54.33 | 71.28 | 72.65 | 60.31 |
| KaLM-Embedding-Gemma3 | | 68.18 | 67.00 | 55.01 | 70.59 | 71.86 | 66.53 |
| Qwen3-Embedding-0.6B | | 57.73 | 54.80 | 55.52 | 69.89 | 72.57 | 62.10 |
| Qwen3-Embedding-4B | | 55.58 | 60.60 | 54.67 | 69.13 | 71.94 | 62.38 |
| Qwen3-Embedding-8B | | 61.82 | 63.40 | 54.67 | 70.34 | 72.09 | 64.46 |
| **EvoEmbedding-0.8B (Ours)** | | 58.31 | 71.40 | 54.67 | **72.30** | 71.07 | 65.55 |
| **EvoEmbedding-2B (Ours)** | | 64.48 | 74.00 | 55.01 | 71.47 | 71.70 | 67.33 |
| **EvoEmbedding-4B (Ours)** | | **69.94** | **76.60** | 56.20 | 72.04 | 72.81 | **69.52** |
| BM25 | $k=8$ | 64.09 | 67.20 | 53.99 | 70.08 | 73.12 | 65.70 |
| All-MiniLM-L6-v2 | | 39.09 | 56.20 | 55.52 | 71.98 | 72.02 | 58.96 |
| Jina-v5-text-small | | 58.25 | 68.40 | 56.71 | 71.66 | 73.91 | 65.79 |
| Multilingual-e5-large | | 65.00 | 56.20 | 53.14 | 71.16 | 72.41 | 63.58 |
| BGE-M3 | | 54.42 | 64.60 | 56.20 | 71.35 | **74.07** | 64.13 |
| KaLM-Embedding-Gemma3 | | 72.27 | 72.80 | **57.05** | 71.54 | 73.68 | 69.47 |
| Qwen3-Embedding-0.6B | | 62.86 | 65.40 | 56.54 | 72.23 | 73.83 | 66.17 |
| Qwen3-Embedding-4B | | 61.43 | 69.20 | 55.35 | 71.41 | 73.52 | 66.18 |
| Qwen3-Embedding-8B | | 67.86 | 71.40 | 55.86 | 71.66 | 72.65 | 67.89 |
| **EvoEmbedding-0.8B (Ours)** | | 55.71 | 74.80 | 55.52 | 71.92 | 73.44 | 66.28 |
| **EvoEmbedding-2B (Ours)** | | 69.94 | 76.40 | 56.71 | 72.11 | 73.60 | 69.75 |
| **EvoEmbedding-4B (Ours)** | | **74.29** | **77.60** | 55.86 | **73.24** | 73.68 | **70.93** |
| BM25 | $k=16$ | 67.47 | 68.00 | 53.99 | 71.79 | 73.83 | 67.02 |
| All-MiniLM-L6-v2 | | 43.70 | 61.80 | 55.86 | 72.49 | 73.68 | 61.50 |
| Jina-v5-text-small | | 64.16 | 71.60 | 56.54 | 72.80 | 74.07 | 67.83 |
| Multilingual-e5-large | | 68.70 | 60.20 | 55.86 | 72.11 | 73.28 | 66.03 |
| BGE-M3 | | 60.26 | 63.80 | 55.69 | 72.99 | 74.78 | 65.50 |
| KaLM-Embedding-Gemma3 | | 73.96 | 72.40 | **58.23** | 72.49 | 73.91 | 70.20 |
| Qwen3-Embedding-0.6B | | 67.99 | 68.20 | 56.54 | **73.88** | 74.31 | 68.18 |
| Qwen3-Embedding-4B | | 66.49 | 70.00 | 56.71 | 72.42 | 74.15 | 67.95 |
| Qwen3-Embedding-8B | | 71.10 | 73.20 | 57.05 | 72.42 | 73.44 | 69.44 |
| **EvoEmbedding-0.8B (Ours)** | | 59.42 | 74.20 | 57.05 | 72.68 | 74.23 | 67.51 |
| **EvoEmbedding-2B (Ours)** | | 74.48 | 73.20 | 55.69 | 73.18 | 73.83 | 70.08 |
| **EvoEmbedding-4B (Ours)** | | **75.58** | **75.20** | 55.69 | 73.31 | **75.34** | **71.02** |

keyword traps or paraphrased evidence that requires historical grounding to resolve. To address this, we design a latent memory queue to maintain a finite, rolling contextual state. This does not require the model to memorize everything, but rather to know how to query past events. This basic intuition directly inspired the design of EvoEmbedding.

**Why latent memory for retrieval instead of generation?** This choice is driven by three key considerations: ***(i) Controllability:*** Equipping LLMs with native or test-time memory is appealing, yet directly modifying their parameters, activations, or prompts often triggers unpredictable behaviors like catastrophic forgetting and hallucinations (Yu et al., 2026; Nie et al., 2026b). ***(ii) Factuality:*** Under limited memory capacity, generative recall struggles to reliably verify whether specific events occurred, whereas retrieval provides a concrete, verifiable record of historical facts (Huang et al., 2025). ***(iii) Deployability:*** State-of-the-art commercial models are typically accessible only via

black-box APIs, precluding any internal state modifications. Consequently, leveraging latent memory to construct retrieval representations offers a far more controllable and deployable paradigm.

**Why multi-LoRA design?** The multi-LoRA design unifies memory, retrieval, and generation capabilities within a single general-purpose language model, eliminating the need for specialized embedding models. Importantly, this capability decoupling isolates the training dynamics. By updating only the task-specific LoRA adapters while keeping the backbone frozen, we avoid catastrophic forgetting in the generation process. This design renders EvoEmbedding highly flexible: it can run efficiently as a standalone local model or serve seamlessly as a plug-and-play retrieval module within broader agent systems.

**Limitations.** Our current framework has certain limitations. First, constrained by limited computational resources and training data scale, EvoEmbedding may exhibit degraded performance when applied to out-of-domain scenarios. Second, the current implementation of EvoEmbedding lacks native support for multimodal retrieval. While the architecture could be extended to incorporate visual or audio modalities based on existing models, managing long-horizon memories in an omni-modal context would require substantially larger queue capacities and more sophisticated dimensional alignment, which we leave for future exploration.