TEST-TIME TRAINING OF CONTEXTUAL EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Dense retrievers typically use fixed, context-independent embedding functions, limiting their ability to adapt to user- or corpus-specific semantics. We introduce **TTT-Retrieval**, a framework for *test-time training of contextual embeddings* that adapts a retriever directly to an unlabeled target corpus after deployment. TTT-Retrieval augments a frozen pretrained embedding model with compact, meta-trained neural memory modules. During a *write phase*, these memories are updated through self-supervision on the target corpus; during a *read phase*, they condition query embeddings on the learned corpus context. Because the adapted memories are retained across queries, adaptation cost is amortized rather than repeatedly incurred at inference time. Across personalized retrieval on LoCoMo, domain-specialized retrieval on QASPER, and open-domain retrieval on HotpotQA, TTT-Retrieval consistently improves over frozen embedding baselines and is competitive with or outperforms reranking approaches while requiring substantially less inference-time computation.

1 INTRODUCTION

Retrieval-augmented generation (RAG) has become a fundamental mechanism for grounding large language models (LLMs) in knowledge external to their parameters (Lewis et al., 2020). This is especially important for personalized assistants, which must retrieve relevant information from conversations, messages, notes, and private documents accumulated over time. Such retrieval is challenging because queries often depend on user-specific entities, relationships, terminology, and past events whose meaning is difficult to interpret without the surrounding personal context, as illustrated in Fig. 1. More broadly, the same challenge arises whenever retrieval depends on the characteristics of a particular target corpus.

Standard dense retrievers are largely fixed across corpora. A pretrained encoder maps the same query to the same representation regardless of the corpus being searched (Vemuri et al., 2023; Zhao et al., 2024). Yet the same query can reflect different intents across users or collections (Teevan et al., 2008), and retrieval performance can degrade under domain and distribution shifts (Thakur et al., 2021). This raises a natural question: **How can an embedding model adapt to the corpus it is asked to search?**

One approach is supervised fine-tuning. Using labeled query–document pairs or synthetic supervision, an embedding model can learn representations better suited to the target retrieval setting (Karpukhin et al., 2020; Wang et al., 2022). A strong retrieval pipeline can further apply a cross-encoder reranker to refine retrieved candidates through direct query–document interaction (Nogueira & Cho, 2019; Zhang et al., 2025). However, supervised fine-tuning requires target-specific training data, which may be difficult to obtain for individual users or specialized corpora. Cross-encoder reranking also adds substantial query-time computation.

Contextual embedding methods instead use the unlabeled target corpus to make embeddings corpus-aware (Morris & Rush, 2025). A separate contextualizer converts selected corpus documents into context-token embeddings, which are then used to condition query and document embeddings. This avoids the need for target-specific supervision. However, only a fixed number of context documents can be used. For a large corpus, only a subset of documents contributes to the context, and each selected document is compressed into a single context-token embedding.

This motivates a different approach. Instead of selecting a subset of the corpus and representing it through a fixed set of context tokens, the embedding model could learn directly from the unlabeled

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.

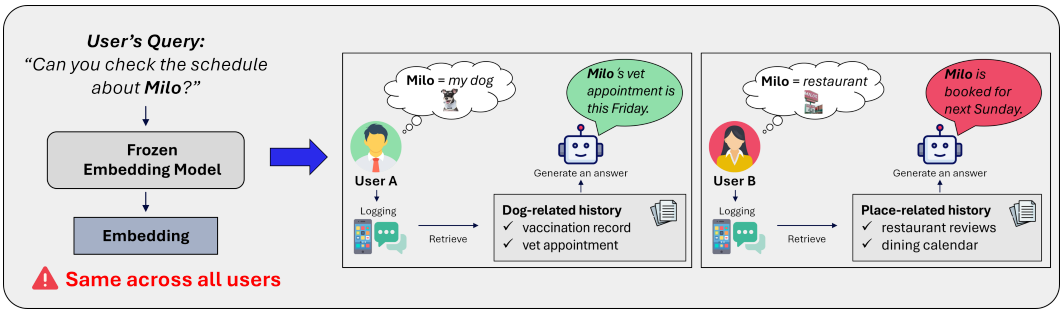


Figure 1: **Motivation.** The same query can reflect different intents across users depending on their personal histories. For example, "Milo" refers to a dog for User A but to a restaurant for User B. A frozen embedding model, however, maps the same query to the same representation regardless of the user-specific corpus.

target corpus at test time. This allows the full corpus to contribute to adaptation while avoiding target-specific supervision.

To this end, we introduce **TTT-Retrieval**, a framework for *test-time contextualization of embeddings* through self-supervised test-time training (Sun et al., 2020; Chen et al., 2026; Cho et al., 2026). TTT-Retrieval augments intermediate layers of a frozen pretrained embedding model with meta-trained neural memory modules. At test time, the target corpus is used to update these memories through self-supervised learning. The adapted memories are then used to embed queries and documents without providing the target corpus again as input.

We evaluate TTT-Retrieval on personalized conversational retrieval with LoCoMo (Maharana et al., 2024), domain-specialized retrieval with QASPER (Dasigi et al., 2021), and open-domain retrieval with HotpotQA (Yang et al., 2018). Across these settings, TTT-Retrieval consistently improves over frozen embedding baselines and achieves strong performance relative to supervised fine-tuning, fine-tuning with reranking, and contextual embedding methods. On LoCoMo, TTT-Retrieval reaches 84.7 Recall@10, compared with 83.9 for Fine-Tuning + Reranking. After the target corpus has been written to memory, TTT-Retrieval requires only 46.1G FLOPs per query at read time, compared with 15,181G FLOPs for the reranking-based supervised pipeline.

Our contributions are:

- We formulate *test-time contextualization of embeddings*, where an embedding model learns from an unlabeled target corpus after deployment and reuses the resulting adaptation across future queries.
- We propose **TTT-Retrieval**, which equips a frozen pretrained embedding model with meta-trained neural memories that are updated through self-supervised test-time training on the target corpus.
- We demonstrate consistent gains across personalized, domain-specialized, and open-domain retrieval, with strong performance relative to supervised adaptation, reranking-based pipelines, and contextual embedding methods while requiring substantially less per-query computation than cross-encoder reranking.

2 Background

Let $\mathcal{C} = \{d_i\}_{i=1}^{N}$ denote a target corpus consisting of N documents, and let q denote a query. A dense retriever uses a pretrained embedding model f_θ to encode queries and documents and ranks documents according to their embedding similarity:

$$\mathbf{e}_q = f_\theta(q), \qquad \mathbf{e}_{d_i} = f_\theta(d_i). \tag{1}$$

Under the standard train-then-deploy paradigm, the model parameters θ remain fixed after deployment. Consequently, the embedding function does not adapt to the particular corpus being searched. Our goal is instead to adapt retrieval to an unlabeled target corpus at deployment while keeping the pretrained embedding backbone frozen.

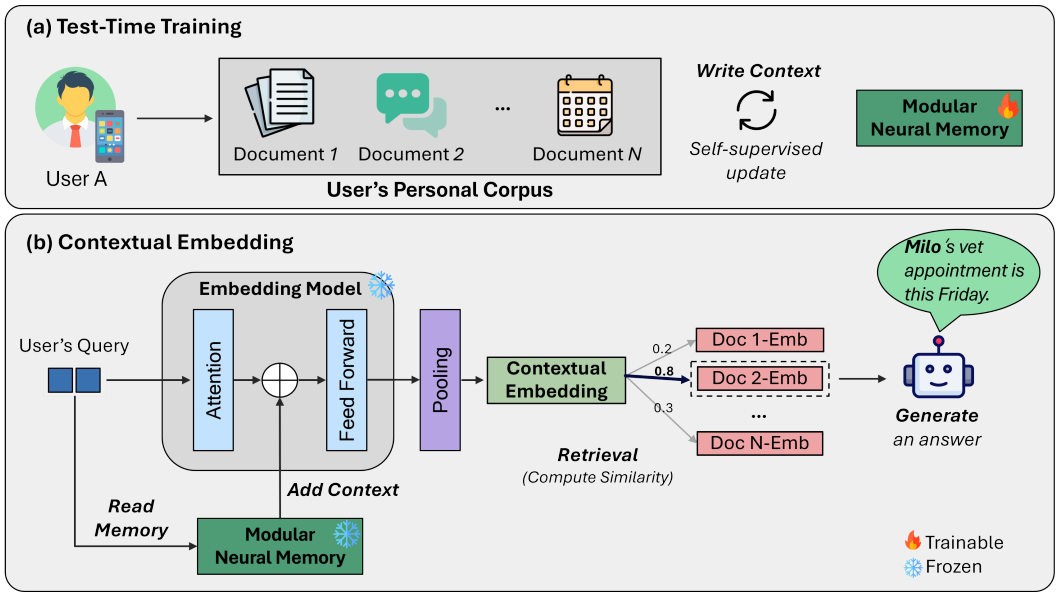


Figure 2: **Overview of TTT-Retrieval.** At test time, TTT-Retrieval writes the target corpus into lightweight neural memories through self-supervised adaptation. The corpus-specific state is reused to produce corpus-conditioned embeddings for subsequent retrieval queries.

3 METHOD

3.1 OVERVIEW

We introduce **TTT-Retrieval**, which adapts a pretrained embedding model to an unlabeled target corpus at deployment through lightweight neural memory. Unlike a conventional retriever whose embedding function remains fixed across corpora, TTT-Retrieval uses the target corpus to adapt memory parameters while keeping the pretrained backbone frozen.

The model consists of a frozen embedding backbone f_θ and adaptable memory parameters ϕ. Before deployment, we meta-learn an initialization ϕ_0. Given a target corpus $\mathcal{C}$, the *write phase* adapts the memory to the corpus:

$$\phi_{\mathcal{C}} = \mathrm{Write}(\phi_0, \mathcal{C}). \tag{2}$$

The resulting corpus-specific memory state conditions subsequent query and document representations:

$$\mathbf{e}_{x|\mathcal{C}} = f_{\theta,\phi_{\mathcal{C}}}(x). \tag{3}$$

Thus, the same input can obtain a different embedding depending on the corpus written into memory.

The adapted state is shared across subsequent retrieval operations, allowing the cost of corpus adaptation to be amortized over multiple queries. Figure 2 summarizes the framework.

3.2 MEMORY-AUGMENTED EMBEDDING MODEL

We augment intermediate layers of the pretrained embedding model with lightweight fast-weight memory modules while keeping the backbone parameters θ frozen. Let $W^{(\ell)}$ denote the adaptable memory parameters associated with layer ℓ, and let

$$\phi = \{W^{(\ell)}\}_{\ell=1}^{L} \tag{4}$$

collect the memory parameters across the L augmented layers.

Given hidden states $H^{(\ell)}$ at layer ℓ, the corresponding memory module produces an additional residual $\Delta^{(\ell)}$:

$$H^{(\ell+1)} = \mathrm{Block}^{(\ell)}(H^{(\ell)}) + \Delta^{(\ell)}. \tag{5}$$

This allows information stored in the memory to influence intermediate representations throughout the embedding model. Architectural details of the memory modules are provided in Appendix A. We next describe how these memories are written from a target corpus and read during retrieval.

3.3 Write Phase: Learning Corpus Context

At deployment, the target corpus is written into the memory modules without using query–document relevance labels. For each augmented layer, token-level representations are converted into memory keys and values $(\mathbf{k}_t, \mathbf{v}_t)$. The fast weights are then updated through a self-supervised compatibility objective:

$$W \leftarrow W - \eta \sum_{t \in \mathcal{B}} \nabla_W \mathcal{L}_{\text{write}} \left(\mathcal{M}_W(\mathbf{k}_t), \mathbf{v}_t \right), \tag{6}$$

where $\mathcal{B}$ denotes a batch of corpus tokens and η is the write learning rate.

We use the compatibility loss

$$\mathcal{L}_{\text{write}} \left(\mathcal{M}_W(\mathbf{k}), \mathbf{v} \right) = - \left\langle \mathcal{M}_W(\mathbf{k}), \mathbf{v} \right\rangle. \tag{7}$$

All write updates are local to the memory parameters, while the pretrained backbone remains frozen. After processing the corpus, the resulting adapted memory state is

$$\phi_{\mathcal{C}} = \{ W_{\mathcal{C}}^{(\ell)} \}_{\ell=1}^{L}. \tag{8}$$

The write phase therefore adapts the model to the target corpus using only its unlabeled documents, without requiring deployment-time relevance supervision.

3.4 Read Phase: Corpus-Conditioned Embeddings

After the corpus has been written, the resulting memory state is fixed and reused across retrieval operations. At each augmented layer, a token-level memory query $\mathbf{q}_t^{(\ell)}$ reads from the corresponding adapted memory:

$$\Delta_t^{(\ell)} = \mathcal{M}_{W_{\mathcal{C}}^{(\ell)}} \left(\mathbf{q}_t^{(\ell)} \right). \tag{9}$$

The resulting memory output is injected through Eq. 5, allowing corpus information stored in the adapted memory to influence hidden representations throughout the encoder. The backbone's original pooling and normalization procedure then produces the final corpus-conditioned embedding

$$\mathbf{e}_{x|\mathcal{C}} = f_{\theta, \phi_{\mathcal{C}}}(x). \tag{10}$$

The same adapted memory state is used to encode retrieval inputs, providing a corpus-conditioned representation space without modifying the pretrained backbone.

3.5 Meta-Learning Corpus Writes for Retrieval

The self-supervised write objective determines how corpus information is stored in memory, but does not by itself ensure that the resulting state improves retrieval. We therefore meta-learn the memory initialization ϕ_0 such that self-supervised writes on a target corpus produce representations that are useful for retrieval. The inner loop writes corpus information without relevance labels, while the outer objective learns how the memory should be initialized so that these writes benefit subsequent retrieval.

Corpus-level meta-training. We organize meta-training into corpus-level episodes $(\mathcal{C}_j, \mathcal{Q}_j)$, where

$$\mathcal{Q}_j = \{ (q_i, G_i) \}_{i=1}^{B}, \tag{11}$$

and $G_i \subseteq \mathcal{C}_j$ denotes the set of relevant documents for query q_i. Each episode mirrors the deployment setting. A write set from the corpus is first written into memory:

$$\phi_j = \text{Write} \left(\phi_0, \mathcal{C}_j^{\text{write}} \right), \tag{12}$$

and the resulting state ϕ_j is shared across all queries in the episode. The inner write operation uses only document representations and does not directly optimize a query–document relevance objective.

Retrieval outer objective. After writing the corpus, we evaluate the adapted memory using a contrastive retrieval objective commonly used in dense retrieval (van den Oord et al., 2019; Karpukhin et al., 2020). For query q_i and document d, we define

$$s(q_i, d \mid \phi_j) = \mathrm{sim}\left(f_{\theta,\phi_j}(q_i), f_{\theta,\phi_j}(d)\right), \tag{13}$$

and optimize

$$\mathcal{L}_{\mathrm{ret}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\sum_{d \in G_i} \exp(s(q_i, d \mid \phi_j)/\tau)}{\sum_{d \in \mathcal{C}_j} \exp(s(q_i, d \mid \phi_j)/\tau)}, \tag{14}$$

where τ is a temperature parameter. Using candidates from the same corpus trains the adapted model to distinguish documents within the retrieval environment in which the memory will be used.

The initialization is optimized through the write operation:

$$\min_{\phi_0} \mathbb{E}_{(\mathcal{C}_j, \mathcal{Q}_j) \sim \mathcal{D}_{\mathrm{meta}}} \left[\mathcal{L}_{\mathrm{ret}}(\phi_j)\right], \qquad \phi_j = \mathrm{Write}\left(\phi_0, \mathcal{C}_j^{\mathrm{write}}\right). \tag{15}$$

Retrieval supervision is therefore applied only through the outer objective, while deployment-style corpus adaptation remains self-supervised. The resulting initialization is trained so that writing an unlabeled corpus produces a memory state that improves retrieval across multiple queries.

Training-time write-set construction. The formulation above conceptually allows the full corpus to be written within each meta-training episode. In practice, differentiating through writes over an entire corpus can be prohibitively expensive. We therefore construct a bounded write set $\mathcal{C}_j^{\mathrm{write}} \subseteq \mathcal{C}_j$ that preserves the retrieval structure of each episode. Relevant documents and difficult same-corpus negatives are included, and the remaining budget is filled with randomly sampled corpus documents.

We further adopt a curriculum over the write budget, starting with smaller write sets and gradually increasing the amount of corpus context during meta-training. This exposes the memory to progressively larger and more interfering contexts while keeping early training tractable and stable.

At deployment, this training-time construction is removed, and the target corpus is written without access to query–document relevance labels. Further details on write-set construction and the curriculum schedule are provided in Appendix B.

4 Experiments

We evaluate TTT-Retrieval in three settings: personalized conversational retrieval, domain-specific document retrieval, and multi-hop retrieval. We first describe the experimental setup and then report retrieval performance and computational efficiency.

4.1 Experimental Setup

Datasets. We evaluate on three retrieval benchmarks, each organized as a corpus-level episode containing one or more retrieval queries:

- **LoCoMo** (Maharana et al., 2024) evaluates retrieval from long conversational histories containing user-specific facts, events, preferences, and temporal information. Each conversation forms one target corpus with multiple questions.
- **QASPER** (Dasigi et al., 2021) evaluates evidence retrieval from scientific papers. Each paper forms one target corpus with multiple associated questions.
- **HotpotQA** (Yang et al., 2018) evaluates multi-hop retrieval over Wikipedia. For each episode, we construct a corpus of 200 documents containing the supporting evidence and distractors, and evaluate on 100 sampled test questions.

Additional dataset details are provided in Appendix C.

Models and Metrics. Our primary backbone is Qwen3-Embedding-0.6B (Zhang et al., 2025), using last-token pooling. We use the same backbone for the frozen baseline, supervised fine-tuning, and

Table 1: Retrieval performance comparison on LoCoMo. Results are reported as Recall@k (R@k), where higher is better. Baselines are grouped by whether they require training-time optimization and, if so, whether that optimization requires labeled query–document pairs. Best results for each top-k are shown in **bold**.

Method	top-k	Single Hop	Multi Hop	Temporal	Open Domain	Overall
Off-the-Shelf Retrieval						
Base	10	64.9	21.7	62.2	27.7	54.3
	20	71.8	30.3	69.1	32.9	61.3
Base + Reranking	10	54.6	33.5	69.0	28.6	52.2
	20	64.5	41.1	73.3	34.5	60.3
Supervised Fine-Tuning						
Fine-Tuning	10	83.5	39.7	75.4	34.6	64.5
	20	92.4	49.1	78.8	40.1	68.6
Fine-Tuning + Reranking	10	90.6	64.5	90.2	58.7	83.9
	20	92.4	75.6	**93.5**	**68.3**	88.1
Test-Time Adaptation						
TTT-Retrieval (Ours)	10	**90.7**	**67.8**	**90.5**	**60.7**	**84.7**
	20	**93.9**	**79.1**	92.6	66.9	**89.3**

Table 2: Retrieval performance comparison on QASPER and HotpotQA. Results are reported as Recall@k (R@k), where higher is better. Best results are shown in **bold**.

Method	QASPER		HotpotQA	
	Recall@10	Recall@20	Recall@10	Recall@20
Off-the-Shelf Retrieval				
Base	65.1	82.5	72.5	79.8
Base + Reranking	72.1	86.0	84.0	84.8
Supervised Fine-Tuning				
Fine-Tuning	76.7	89.3	82.3	87.3
Fine-Tuning + Reranking	72.6	86.4	87.8	89.5
Test-Time Adaptation				
TTT-Retrieval (Ours)	**81.4**	**91.7**	**88.3**	**90.3**

TTT-Retrieval. For reranking, we use Qwen3-Reranker-0.6B (Zhang et al., 2025). For comparison with CDE, we also evaluate TTT-Retrieval using the same nomic-embed-text-v1 backbone (Nussbaum et al., 2024). We report Recall@k as the primary retrieval metric.

Baselines. We compare against the following methods:

- *Base.* The frozen embedding model without target-corpus adaptation.
- *Fine-Tuning.* The embedding model fine-tuned using labeled query–document pairs.
- *Reranking.* The top-50 candidates retrieved by either Base or Fine-Tuning are reranked using a cross-encoder.
- *CDE* (Morris & Rush, 2025). Contextual Document Embeddings condition query and document representations on additional inputs derived from the target corpus. We report the publicly released checkpoints and include a controlled comparison using the same backbone.

Additional baseline and implementation details are provided in Appendix D.

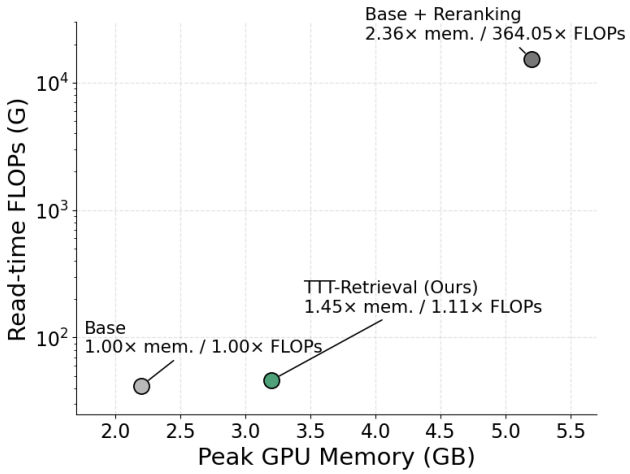


Figure 3: **Computational efficiency.** Peak GPU memory and per-query read-time FLOPs after the corpus memory has been written. FLOPs are shown on a logarithmic scale.

4.2 Evaluation Results

Tables 1 and 2 report retrieval performance across the three evaluation settings. TTT-Retrieval consistently improves over the frozen embedding baseline and achieves strong performance relative to supervised fine-tuning and reranking, without repeated query–document scoring at inference time.

Personalized retrieval. On LoCoMo, TTT-Retrieval substantially improves over the frozen retriever (*Base*), increasing overall Recall@10 from 54.3 to 84.7 and Recall@20 from 61.3 to 89.3. The largest gain occurs on multi-hop questions, where Recall@10 increases from 21.7 to 67.8 (+46.1 points). Improvements are also substantial for open-domain (+33.0), temporal (+28.3), and single-hop (+25.8) questions, suggesting that corpus adaptation is particularly useful when relevant information is distributed across a user's conversational history.

TTT-Retrieval also slightly exceeds *Fine-Tuning + Reranking* in overall Recall@10 (84.7 vs. 83.9) and Recall@20 (89.3 vs. 88.1), without target-corpus relevance labels or cross-encoder scoring at query time. We further analyze where the two approaches differ in Section 4.5.

Domain-specific and multi-hop retrieval. The gains extend beyond personalized retrieval. On QASPER, TTT-Retrieval achieves 81.4 Recall@10 and 91.7 Recall@20, compared with 65.1 and 82.5 for *Base*, and 76.7 and 89.3 for *Fine-Tuning*. It also outperforms both reranking variants.

On HotpotQA, TTT-Retrieval reaches 88.3 Recall@10 and 90.3 Recall@20, substantially improving over *Base* (72.5 and 79.8) and slightly exceeding *Fine-Tuning + Reranking* (87.8 and 89.5). These results show that test-time corpus adaptation remains effective when retrieval spans scientific documents or multiple Wikipedia documents.

4.3 Computational Efficiency

Figure 3 compares peak GPU memory and per-query read-time FLOPs after the corpus memory has been written. TTT-Retrieval incurs only modest overhead over the frozen *Base* retriever, increasing peak GPU memory from 2.2 GB to 3.2 GB ($1.45\times$) and read-time computation from 41.7G to 46.1G FLOPs ($1.11\times$).

In contrast, *Base + Reranking* requires 5.2 GB of peak GPU memory and 15,181G FLOPs per query. TTT-Retrieval therefore uses approximately $329\times$ fewer read-time FLOPs than cross-encoder reranking, while keeping inference-time computation close to standard bi-encoder retrieval. This efficiency comes from writing the corpus once and reusing the adapted memory state across subsequent queries.

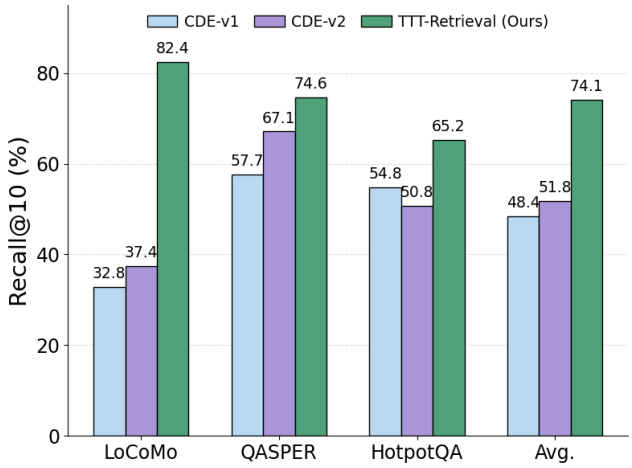


Figure 4: **Comparison with Contextual Document Embeddings (CDE).** Recall@10 on LoCoMo, QASPER, and HotpotQA using the nomic-embed-text-v1 backbone.

Table 3: Representative aggregative queries for which TTT-Retrieval retrieves complete evidence in the top 10 while Fine-Tuning + Reranking does not.

Solved by TTT-Retrieval
"What has Melanie painted?"
"What items does John collect?"
"What movies have both Joanna and Nate seen?"
"Which country has Tim visited most frequently in his travels?"

4.4 Comparison with Corpus-Contextualized Retrieval

We further compare TTT-Retrieval with Contextual Document Embeddings (CDE) under a controlled backbone setting. Specifically, TTT-Retrieval uses the same nomic-embed-text-v1 backbone as CDE. As shown in Fig. 4, TTT-Retrieval outperforms both CDE variants across all three benchmarks. The largest gap appears on LoCoMo, where TTT-Retrieval reaches 82.4 Recall@10 compared with 37.4 for CDE-v2. The gains are also consistent on QASPER (74.6 vs. 67.1) and HotpotQA (65.2 vs. 54.8, compared with the stronger CDE variant on each dataset).

4.5 Analysis: When Does Corpus Adaptation Help Beyond Reranking?

TTT-Retrieval and Fine-Tuning + Reranking achieve similar overall recall on LoCoMo, but differ in how query–corpus interaction is incorporated into retrieval. Reranking refines a fixed set of first-stage candidates using the query, whereas TTT-Retrieval first adapts its neural memory to the test corpus and then retrieves evidence through query-dependent access to this corpus-adapted memory.

We therefore analyze queries for which the two methods differ in whether their top-10 results contain all evidence required to answer the query. We define *aggregative queries* as those requiring information to be collected or combined across multiple events or occurrences.[1] Such queries account for 28.6% of cases solved by TTT-Retrieval but not Fine-Tuning + Reranking, compared with 10.3% in the reverse direction.

These results suggest that corpus adaptation may be particularly useful for improving first-stage evidence coverage when relevant information is distributed across multiple occurrences. Table 3 shows representative examples, such as "What has Melanie painted?" and "What items does John collect?", which require retrieving related facts from different parts of the conversation. TTT-Retrieval can use query-dependent access to its corpus-adapted memory to determine which documents enter the candidate set, whereas reranking can only reorder candidates that have already been retrieved.

[1]Detailed annotation criteria are provided in Appendix E.

5 Related Work

5.1 Personalized and Contextual Retrieval

Dense retrievers typically encode queries and documents independently, which can limit adaptation to domain-specific or personalized corpora (Karpukhin et al., 2020; Thakur et al., 2021; Teevan et al., 2008). Long-term conversational memory benchmarks. For example, LoCoMo further highlights the difficulty of accessing relevant information distributed across extended interaction histories (Maharana et al., 2024). Most closely related to our work, Contextual Document Embeddings (CDE) condition query and document representations on information derived from the target corpus (Morris & Rush, 2025). TTT-Retrieval instead incorporates corpus information through test-time adaptation of lightweight neural memory, producing a reusable corpus-specific model state.

5.2 Retriever Adaptation and Reranking

Retriever adaptation is commonly achieved either by fine-tuning the retriever or by applying a more expressive model after initial retrieval. Supervised dense retrieval methods such as DPR learn from labeled query–document pairs (Karpukhin et al., 2020), while domain-adaptation approaches reduce annotation requirements by generating synthetic supervision. GPL, for example, generates target-domain queries and obtains pseudo relevance labels using a cross-encoder (Wang et al., 2022), and Promptagator uses a small number of task demonstrations to generate synthetic training data for retriever adaptation (Dai et al., 2023).

Reranking provides an alternative by applying a more expressive model to candidates produced by a first-stage retriever (Nogueira & Cho, 2019). Recent approaches also rank multiple candidates jointly using setwise or listwise formulations, reducing the need for independent query–document scoring (Zhuang et al., 2024; Adeyemi et al., 2024; Pradeep et al., 2023). Nevertheless, reranking remains query-dependent post-retrieval computation and is restricted to candidates retrieved in the first stage. In contrast, TTT-Retrieval adapts directly from the unlabeled target corpus and amortizes this adaptation across subsequent queries through a reusable memory state.

5.3 Test-Time Training and Neural Memory

Test-time training adapts model parameters using unlabeled data encountered after deployment, with early work demonstrating that self-supervised test-time optimization can improve robustness under distribution shift (Sun et al., 2020). More recent work has extended this idea to long-context modeling and neural memory. TTT layers represent recurrent hidden states as learnable models updated on the test sequence (Sun et al., 2025), while Titans introduces a neural long-term memory that learns to memorize historical context at test time (Behrouz et al., 2025). LaCT improves the scalability of test-time training by using larger chunk-wise updates, and PERK meta-learns parameter-efficient test-time adaptation by encoding long contexts into lightweight LoRA-based memory modules (Zhang et al., 2026; Chen et al., 2026).

Most directly related to our architecture, Cho et al. (2026) introduce *MoNe*, a modular neural memory that attaches layer-wise fast-weight memory networks to a frozen pretrained Transformer. MoNe processes context incrementally and updates these memories through layer-localized gradients, allowing information from long contexts to be stored without modifying the pretrained backbone.

6 Conclusion

We introduced **Test-Time Training of Contextual Embeddings**, a framework that adapts embedding representations to a target retrieval corpus through a lightweight neural memory while keeping the embedding backbone frozen. By separating adaptation into a write phase, performed when the corpus is created or updated, and a read phase, reused across subsequent queries, the proposed approach learns corpus-specific semantics without labeled query–document pairs or full-model fine-tuning. This design enables efficient, privacy-preserving, and on-device personalization while preserving retrieval efficiency. More broadly, our work demonstrates that test-time learning can be used to adapt the representation space of embedding models, providing a practical foundation for personalized retrieval systems and AI agents that continuously learn from evolving local knowledge.

AI USE STATEMENT

We used generative AI tools to assist with language editing, proofreading, and improving the clarity and readability of the manuscript. Generative AI was not used to generate research ideas, formulate scientific claims, design experiments, analyze results, or draw conclusions. All AI-assisted content was reviewed by the authors for technical correctness and consistency with the reported methodology and findings. The authors take full responsibility for the final content of the paper, including any text or artifacts produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

We provide the methodological, implementation, data, and evaluation details necessary to reproduce our experiments in the main paper and appendices. In particular, we describe the neural-memory architecture and write/read procedure in Appendix A, dataset construction and splits in Appendix C, and meta-training and test-time write configurations in Appendix B. We additionally report the training-time write-set construction, write-budget curriculum, and dataset-specific test-time configurations used in our experiments. All experiments use a fixed random seed of 42, and all hyperparameters used for test-time corpus writing are selected on the corresponding validation set.

REFERENCES

Mofetoluwa Adeyemi, Akintunde Oladipo, Ronak Pradeep, and Jimmy Lin. Zero-shot cross-lingual reranking with large language models for low-resource languages. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 650–656, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-short.59. URL https://aclanthology.org/2024.acl-short.59/.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 113506–113543. Curran Associates, Inc., 2025. doi: 10.52202/085713-3786. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4ca07aa108036f80cbb5b82285fd4b1-Paper-Conference.pdf.

Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. PERK: Long-context reasoning as parameter-efficient test-time learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qxDTe8fIyA.

Wonguk Cho, Kyubyung Chae, Tribhuvanesh Orekondy, Sunghyun Park, Hyoungwoo Park, Jeongho Kim, Arash Behboodi, Kyuwoong Hwang, and Sungrack Yun. Mone: Modular neural memory for efficient long context inference. In *AdaptFM: Resource-Adaptive Foundation Model Inference*, 2026.

Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith Hall, and Ming-Wei Chang. Promptagator: Few-shot dense retrieval from 8 examples. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=gmL46YMpu2J.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.365. URL https://aclanthology.org/2021.naacl-main.365/.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

John Xavier Morris and Alexander M Rush. Contextual document embeddings. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Wqsk3FbD6D.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*, 2019. URL https://arxiv.org/abs/1901.04085.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024. URL https://arxiv.org/abs/2402.01613.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankvicuna: Zero-shot listwise document reranking with open-source large language models, 2023. URL https://arxiv.org/abs/2309.15088.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57503–57522. PMLR, 2025.

Jaime Teevan, Susan T. Dumais, and Daniel J. Liebling. To personalize or not to personalize: Modeling queries with variation in user intent. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 163–170. ACM, 2008. doi: 10.1145/1390334.1390364. URL https://doi.org/10.1145/1390334.1390364.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. URL https://arxiv.org/abs/1807.03748.

Hemanth Vemuri, Sheshansh Agrawal, Shivam Mittal, Deepak Saini, Akshay Soni, Abhinav V. Sambasivan, Wenhao Lu, Yajun Wang, Mehul Parsana, Purushottam Kar, and Manik Varma. Personalized retrieval over millions of items. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, pp. 1014–1022, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086. doi: 10.1145/3539618.3591749. URL https://doi.org/10.1145/3539618.3591749.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2345–2360. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.168. URL https://aclanthology.org/2022.naacl-main.168/.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Tb9qAxT3xv.

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

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey. *ACM Trans. Inf. Syst.*, 42(4), February 2024. ISSN 1046-8188. doi: 10.1145/3637870. URL https://doi.org/10.1145/3637870.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 38–47, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657813. URL https://doi.org/10.1145/3626772.3657813.

A Architecture and Algorithm Details

This section provides implementation details of the neural memory modules omitted from Section 3. Our implementation follows the modular fast-weight memory architecture of Cho et al. (2026), adapted to the embedding backbone used in TTT-Retrieval.

Memory feature maps. At each augmented layer ℓ, token representations are projected into memory queries, keys, and values through layer-specific feature maps. These projections are initialized from the corresponding pretrained attention projections and remain anchored to the frozen backbone, while lightweight LoRA adapters are optimized during meta-training. The same feature maps are used for corpus writing and subsequent memory reads.

Specifically, for $u \in \{q, k, v\}$, we denote the projected representation by

$$f_u^{(\ell)}(\mathbf{x}) = \mathrm{Concat}_{h=1}^{H} \mathbf{z}_{u,h}^{(\ell)}, \tag{16}$$

where H is the number of memory heads. We use $H = 8$ throughout our experiments.

Multi-head fast-weight memory. Each memory head contains an independent SwiGLU fast-weight network. Suppressing the token index for clarity, the layer-wise memory output is

$$\mathcal{M}_{W^{(\ell)}}(\mathbf{z}) = \mathrm{Concat}_{h=1}^{H} \mathcal{M}_{W^{(\ell,h)}}(\mathbf{z}_h), \tag{17}$$

where $W^{(\ell,h)}$ denotes the fast weights of head h at layer ℓ.

During reading, each head processes its corresponding memory query. The resulting head outputs are normalized and modulated by learned head-wise gates before being concatenated to form the memory residual $\Delta^{(\ell)}$ used in Eq. 5. Thus, the memory output has the same dimensionality as the backbone representation and can be inserted through a residual connection without modifying the pretrained Transformer block.

Fast-weight updates. Equation 6 presents the write operation as a standard gradient update for clarity. In the implementation, updates are applied independently to each memory head using the associative objective in Eq. 7. Following Cho et al. (2026), the update additionally maintains a momentum state across successive write blocks and allows the effective step size to depend on the token representation.

We write this implementation-level update abstractly as

$$\left(W_s^{(\ell)}, m_s^{(\ell)}\right) = \mathcal{U}^{(\ell)}\left(W_{s-1}^{(\ell)}, m_{s-1}^{(\ell)}, \nabla_{W^{(\ell)}} \mathcal{L}_{\mathrm{write}}\right), \tag{18}$$

where s indexes write blocks and $m_s^{(\ell)}$ denotes the corresponding momentum state. The fast weights are additionally rescaled after each update to preserve their initialized channel-wise norms.

Streaming write and read. For corpora processed in multiple blocks, block s is encoded using the memory state $W_{s-1}^{(\ell)}$ and subsequently updates it to $W_s^{(\ell)}$. Both the fast weights and their update state are carried forward across blocks, allowing corpus information to accumulate throughout the write phase.

After the final corpus block, the resulting state $W_{\mathcal{C}}^{(\ell)}$ is fixed. Queries and documents then read from this final state through Eq. 9 without further memory updates. Consequently, all retrieval inputs associated with the same corpus are encoded using the same adapted memory state $\phi_{\mathcal{C}}$.

The construction of the training-time write set, including relevant documents, hard negatives, random corpus documents, and the write-budget curriculum, is described separately in Appendix B.

Pseudo-code Overview. Algorithm 1 summarizes the deployment procedure of TTT-Retrieval. The target corpus is first written into the meta-trained memory state, after which the adapted state is fixed and reused to encode both documents and subsequent queries for retrieval.

Algorithm 1 TTT-Retrieval: Write Once, Read Many

Input: frozen backbone θ; meta-trained memory initialization ϕ_0; target corpus $\mathcal{C}$
Write phase
$\phi \leftarrow \phi_0$
partition $\mathcal{C}$ into write blocks $\mathcal{B}_1, \ldots, \mathcal{B}_K$
for $b = 1$ **to** K **do**
  encode $\mathcal{B}_b$ with the frozen backbone
  for each augmented layer ℓ **do**
    construct token-level memory keys and values $\{(\mathbf{k}_t^{(\ell)}, \mathbf{v}_t^{(\ell)})\}_t$
    update $W^{(\ell)}$ using the associative write objective in Eqs. 6–7
  end for
end for
$\phi_{\mathcal{C}} \leftarrow \phi$
encode each $d \in \mathcal{C}$ as $\mathbf{e}_{d|\mathcal{C}} = f_{\theta,\phi_{\mathcal{C}}}(d)$ and build the retrieval index
Read phase
for each incoming query q **do**
  encode $\mathbf{e}_{q|\mathcal{C}} = f_{\theta,\phi_{\mathcal{C}}}(q)$ using the fixed adapted memory state
  retrieve the top-k documents by embedding similarity
end for

B Training Details

We use `Qwen3-Embedding-0.6B`[2] as the frozen embedding backbone. TTT-Retrieval augments the backbone with layer-wise fast-weight memory modules and meta-trains their initialization for retrieval.

Meta-training. We use 8 memory heads and LoRA-parameterized query/key/value write projections with rank 128. Meta-training uses a contrastive retrieval objective with temperature $\tau = 0.05$ and two hard negatives per query. We optimize the trainable parameters with AdamW using an outer learning rate of 1×10^{-3}, weight decay 0.01, gradient clipping at 1.0, and 20 warmup steps, for a total of 3,380 steps. All runs use seed 42.

During meta-training, we use a four-stage curriculum over the write budget, increasing it from 1K to 2K, 4K, and finally 8K tokens.

C Dataset Details

Table 4 summarizes the corpus-level episodes used for training, validation, and evaluation. An episode consists of a target corpus and its associated retrieval queries.

Table 4: Dataset statistics. We report the number of corpus-level episodes, the total number of queries, and the average number of queries per episode for each split.

Dataset	Split	Episodes	Queries	Queries / Episode
LoCoMo	Train	2,449	49,163	20.1
	Validation	100	2,026	20.3
	Test	10	1,540	154.0
QASPER	Train	845	2,233	2.6
	Validation	277	851	3.1
	Test	405	1,195	3.0
HotpotQA	Train	4,000	32,000	8.0
	Validation	787	787	1.0
	Test	100	100	1.0

[2] `https://huggingface.co/Qwen/Qwen3-Embedding-0.6B`

Dataset construction. For LoCoMo, since no training split is provided, we generate meta-training conversations using the official data-generation pipeline,[3] with Claude Haiku. The resulting training conversations are substantially shorter than the held-out test conversations, allowing us to evaluate generalization to longer target corpora at deployment.

For QASPER, we exclude unanswerable questions and treat each paper as a corpus, splitting it into paragraph-level retrieval units and grouping its questions into a single episode.

For HotpotQA, we use the distractor setting from the Hugging Face release.[4] For each test query, we construct one retrieval episode containing 200 Wikipedia documents.

D Baseline Details

Fine-Tuning details. Given the limited amount of supervised training data, we use LoRA instead of full fine-tuning to provide a more stable parameter-efficient adaptation. Fine-Tuning uses the same training datasets and data splits as TTT-Retrieval. We fine-tune `Qwen3-Embedding-0.6B` using LoRA (Hu et al., 2022) on the query, key, and value projections ($r = 128$, $\alpha = 128$), while keeping the backbone frozen. Training uses an InfoNCE objective ($\tau = 0.05$) over query–gold-document pairs, with two same-conversation hard negatives per query and additional in-batch negatives. We train for 3 epochs with AdamW using a learning rate of 5×10^{-5} and batch size 64, and select the checkpoint with the best validation Recall@10.

CDE evaluation protocol. We evaluate the publicly released CDE checkpoints: cde-small-v1[5] and cde-small-v2[6]. CDE operates in two stages (Morris & Rush, 2025). In the *first stage*, the model produces uncontextualized query and document embeddings. For each episode, we construct a "minicorpus" by randomly sampling 512 documents from the corresponding corpus and encoding them with the document prefix `search_document:`; queries use the prefix `search_query:`. In the *second stage*, queries and documents are re-encoded by the contextual encoder conditioned on the minicorpus representation. We report the second-stage results as the CDE performance, as these reflect the model's intended contextual conditioning.

E Analysis of Aggregative Queries

We manually annotate the queries analyzed in Section 4.5 based only on their query text. A query is labeled *aggregative* if answering it requires collecting or combining information across multiple events or occurrences in the conversation. We further group aggregative queries into four types: *Set/List*, *Intersection*, *Frequency/Ranking*, and *Repeated-event Summary*. Queries targeting a single localized fact or event are labeled non-aggregative.

Table 5: Breakdown of manually annotated aggregative-query types among LoCoMo queries where TTT-Retrieval and Fine-Tuning + Reranking differ in top-10 complete-evidence retrieval. Percentages are computed within each outcome group.

Aggregation Type	Solved by TTT-Retrieval	Solved by FT + Reranking
Set / List	14 / 63 (22.2%)	6 / 68 (8.8%)
Intersection	3 / 63 (4.8%)	0 / 68 (0.0%)
Frequency / Ranking	2 / 63 (3.2%)	4 / 68 (5.9%)
Repeated-event Summary	5 / 63 (7.9%)	1 / 68 (1.5%)
All Aggregative	24 / 63 (38.1%)	11 / 68 (16.2%)
Non-aggregative	39 / 63 (61.9%)	57 / 68 (83.8%)

[3] https://github.com/snap-research/locomo

[4] https://huggingface.co/datasets/hotpotqa/hotpot_qa

[5] https://huggingface.co/jxm/cde-small-v1

[6] https://huggingface.co/jxm/cde-small-v2