DIAKV: OFFLINE KV CACHE TRAINING FOR CONTEXT-LEVEL ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time training is a learning paradigm that updates the weights of the model on a test-time input, enabling more granular adaptation than fine-tuning. However, adaptation performed after a request arrives adds online training latency, limiting its use in latency-sensitive serving. We propose DIAKV, which trains the KV cache of a reused context offline before future queries arrive. DIAKV directly optimizes the KV cache of a context while keeping the model weights and cache length fixed. We formulate this adaptation as KV-cache posterior maximization given context-specific question-answer distribution, and augment 20 seed QAs per context into a larger training set. Across four long-context QA benchmarks and three 3–4B LLMs, DIAKV improves the primary macro metric in 11 of 12 model–benchmark pairs.

1 INTRODUCTION

Large language models (LLMs) (Brown et al., 2020) are trained on vast amounts of data, making them versatile across a wide range of domains and tasks. However, users often have a specific domain, task, or context for which they want to use LLMs. This makes adapting LLMs to new contexts a critical challenge. Prompting (Brown et al., 2020) serves as the simplest way for adaptation, while fine-tuning (Gururangan et al., 2020) achieves further adaptation through weight updates but requires a large dataset and substantial cost. Test-time training (TTT) (Krause et al., 2018; 2019; Sun et al., 2020) is a learning paradigm that updates the weights of the model, as in fine-tuning, but only on a test instance (or a small set of test instances), enabling a weight-based adaptation after deployment. TTT has recently been applied to LLMs, showing effectiveness across knowledge–intensive tasks (Hu et al., 2025), reasoning tasks (Akyürek et al., 2025; Zuo et al., 2025; Yuksekgonul et al., 2026), and language modeling (Hardt & Sun, 2024; Hübotter et al., 2025).

Despite the effectiveness of TTT in LLMs, a practical gap exists between the online nature of TTT and latency-aware LLM serving. TTT inevitably incurs online latency, as training starts only after the test instance has been received. Such online latency matters less when the benefit of adaptation justifies waiting for online training, e.g., writing a superhuman CUDA kernel (Yuksekgonul et al., 2026). However, latency-sensitive LLM workloads (Zhong et al., 2024) cannot afford this latency.

This conflict can be mitigated when a reusable part of the test-time input is available early enough to move adaptation out of the online critical path. LLM inference often revolves around a shared context, e.g., a document (Rajpurkar et al., 2016), a chat history (Ouyang et al., 2022), a codebase (Jimenez et al., 2024), an agent memory (Packer et al., 2023), or a system prompt and tool list (Ouyang et al., 2022; Schick et al., 2023). These reused contexts provide information for adaptation before future queries arrive. This creates an opportunity for offline adaptation when the context remains available long enough for preparation and subsequent reuse.

Based on these observations, we propose DIAKV, a context-level adaptation method for LLMs. In DIAKV, training is applied at the context level, where the context is already KV-cached and anticipated to be reused. This places its adaptation scope between request-level TTT and model-level fine-tuning (Table 1). Every adaptation aims to perform well on the test input, and the three differ in how much of it is known at training time. TTT waits for the entire test input and trains on it. Fine-tuning sees none of it and instead speculates, under the standard assumption that training and test data share a distribution. DIAKV sits in the middle: it knows the cached part of the input, the context, and speculates only the remaining part, the query. Hence, unlike TTT, it does not wait for

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.

Table 1: Comparison of different approaches to adapting LLMs. DIAKV is an adaptation tailored to LLM workloads, which often reuse the same context (cache hits). DIAKV sits between TTT and fine-tuning in terms of granularity (§1), but is not a substitute for either.

	Test-time Training	**DIAKV**	**Fine-tuning**
Training granularity	single instance or a small batch of instances (online)	single context (offline)	a domain or task (offline)
Example application	solving a single challenging problem	working based on a document or codebase	specialized model on a niche domain
Serving granularity	single request	single KV cache	single model
Training artifact persists for	service time of the request (seconds–hours)	KV cache TTL (hours–days)	model's lifetime (months)

the next query to arrive, and compared to fine-tuning, it adapts at a finer granularity, specializing to each context rather than to an entire domain. During training, we update the KV cache of the context with its length fixed, instead of adding or updating any weights (Hu et al., 2022) or tokens (Liu et al., 2025). Under the reasonable assumption that the KV cache of the context has been saved and reused, this introduces no additional model computation at serving time beyond loading and reusing the trained KV cache. Cartridges (Eyuboglu et al., 2026), a related work to ours, also trains a KV cache offline, but to compress the context rather than to improve accuracy over it (see §2 for comparison).

We formulate DIAKV as maximizing the posterior of the KV cache given context-specific QA distribution (§3.1). This views the problem as finding a better context representation for answering the questions than the given human-readable *gold* context, which motivates the name DIAKV (short for *diamond* KV, as *diamond* is better than *gold*). To estimate the QA distribution of the context, we assume that a small set of seed QAs is available for each context, and augment it into a larger QA set in a time-efficient manner (§3.2). We then train the KV cache of the context with supervised loss on the augmented QAs using standard backpropagation (§3.3).

Our experiments on four long-context QA benchmarks (Kočiský et al., 2018; Lee et al., 2023b; Adams et al., 2025; Chen et al., 2024) with three models, Qwen3-4B-Instruct-2507 (Team, 2025), Llama-3.2-3B-Instruct (AI@Meta, 2024), and Ministral-3-3B-Instruct-2512 (Mistral AI, 2025) (§4.1), show gains in the primary macro metric for 11 of 12 model–benchmark pairs (§4.2). Regarding time, the whole preparation takes under 30 minutes on a single H100/H200 for contexts with a median length of 74k words, well below the illustrative 24-hour time-to-live (TTL) for prefix cache (§4.3).

We then evaluate three design choices of DIAKV against the alternatives: optimizing continuous KV rather than discrete text (§5.1), training KV rather than model weights (§5.2), and QA supervision rather than self-supervised loss (§5.3). In the evaluated settings, DIAKV generally outperforms ACE (Zhang et al., 2026), an LLM-based context optimizer. It is competitive in accuracy with LoRA (Hu et al., 2022) and full fine-tuning while avoiding their serving overhead, and is more robust to superficial patterns in the augmented data. With both methods training the KV cache under matched FLOPs, it also outperforms the self-supervised span loss of qTTT (Bansal et al., 2026) in most cases. Finally, we investigate the role of seed QAs. They support more consistent gains across model–benchmark pairs, as they carry information on how the context is used downstream (§6.1). This however raises a concern of degradation when test queries deviate from the seed queries. In our controlled experiments, when seed QAs are drawn only from certain parts of the context, the remaining parts still benefit from training, or at worst retain the untrained accuracy (§6.2).

Our contributions are summarized as the following.

- We introduce DIAKV, a new granularity for LLM adaptation that lies between request-level test-time training and model-level fine-tuning. It enables offline adaptation for reused contexts while avoiding online training latency.
- We formulate DIAKV as KV cache posterior maximization, yielding a principled objective for optimizing a KV cache for the QA distribution of the context.
- We evaluate DIAKV on four long-context QA benchmarks and three LLMs, demonstrating gains in most settings and advantages over textual context optimization, weight tuning, and self-supervised training loss.

2 RELATED WORK

Test-time training in LLMs Test-time training (TTT) has emerged as a promising paradigm for adapting LLMs. TTT updates the weights of an LLM on a set of given test inputs (Hu et al., 2025; Zuo et al., 2025), or on data obtained through augmentation (Akyürek et al., 2025; Zweiger et al., 2026) or retrieval (Hardt & Sun, 2024; Hübotter et al., 2025) based on inputs. However, such methods require training after receiving the test input, inevitably incurring online latency. DIAKV shifts adaptation out of the online path by training the cached context offline. Another line of TTT work internalizes the context into model weights (Chen et al., 2026; Charakorn et al., 2026), removing the need to include the full context in the prompt and potentially reducing overall generation time. However, these methods require meta-learning (Chen et al., 2026; Charakorn et al., 2026) or training over a large collection of contexts and QA pairs (Kujanpää et al., 2025; Tan et al., 2024), whereas DIAKV requires no such pre-training and thus integrates more easily into existing systems. Nevertheless, these approaches may still prove valuable when designing new systems.

Relation to KV-based compression or adaptation Training the KV of tokens with backpropagation was first suggested in prefix tuning (Li & Liang, 2021), but as a parameter-efficient fine-tuning method using the KV of pseudo-tokens. The closest prior work is Cartridges (Eyuboglu et al., 2026), which trains a compressed KV cache with distillation loss, treating the model with full context as a teacher. It trains on synthetic conversations about the context generated by LLMs. While sharing the notion of training KV cache, its goal is to compress the KV cache with minimal accuracy drop, whereas DIAKV keeps the cache size and aims to improve accuracy. Such distinct goal causes DIAKV to differ in training loss (§3.1) and training data (§3.2)—most noticeably the use of seed QAs. Seed QAs provide information for context-level adaptation, but at a risk of introducing bias, which we partially address in §6.2. Coprocessor (Liu et al., 2025) trains a module that produces additional KV to improve accuracy, but requires training over trillions of tokens and incurs additional online latency, whereas DIAKV directly adapts the KV offline; in exchange, DIAKV is intended for contexts that are cached, whereas Coprocessor can be applied to newly incoming contexts.

Context engineering Context engineering improves model behavior by modifying the context, usually at the text level. Recent works use an LLM as the optimizer, rewriting a prompt or a persistent memory from textual feedback (Lin et al., 2025; Suzgun et al., 2026; Agrawal et al., 2026; Zhang et al., 2026). DIAKV can be seen as a further extension of context optimization at the tensor level, rather than the text level, with gradient descent in place of an LLM optimizer.

Prefix caching Prefix caching, or context caching, is a critical component of LLM inference that caches KV for future reuse. Since the latency of long context prefill far exceeds that of loading the KV cache, prefix caching is increasingly being extended to lower-tier storage, such as host memory, SSDs, or even remote object storage (Liu et al., 2024; Gao et al., 2024; Cheng et al., 2025; Yao et al., 2025; Jeong & Ahn, 2025). This enables potentially much longer KV cache TTLs, which we view as an opportunity for context-level adaptation.

3 METHODOLOGY

In this section, we start with formulating DIAKV as a maximum a posteriori (MAP) problem, providing a principled view and a training objective (§3.1). Training with this objective in practice requires sufficient input-output data for a given context, where this is often unavailable. Therefore, we present a data augmentation pipeline that operates under the time constraints of DIAKV (§3.2). Lastly we introduce the details for training (§3.3).

3.1 DIAKV: BAYESIAN PERSPECTIVE

We adopt the latent formulation of QA (Chen et al., 2017; Lee et al., 2019), where the probability of an answer a given a question q is marginalized over all discrete contexts $c \in V^*$, as in equation 3.1.

$$p(a|q) = \sum_{c \in V^*} p_\theta(c|q)\, p_\theta(a|q,c) \propto \sum_{c \in V^*} p(c)\, p_\theta(q|c)\, p_\theta(a|q,c) \approx p(\hat{c})\, p_\theta(q|\hat{c})\, p_\theta(a|q,\hat{c}) \quad (3.1)$$

The second step applies Bayes' rule to the context posterior $p_\theta(c|q) \propto p(c)\, p_\theta(q|c)$. As the sum over infinite contexts is intractable, models typically keep a single context $\hat{c}$, e.g., the retrieved passages or the given document, which is the last step. A naturally arising question then is finding the optimal $\hat{c}$, the maximum a posteriori context. QA datasets often accompany *gold* contexts, which are human-labeled to be the best context (Joshi et al., 2017; Kwiatkowski et al., 2019). However, such gold contexts can be outperformed by other contexts (Yu et al., 2023; Lee et al., 2023a), implying a ceiling above a heuristic choice. While finding this optimal context in discrete space is difficult, we constrain our model p_θ to a Transformer-based autoregressive language model (Radford et al., 2019), and substitute the discrete string c with a continuous, fixed-length KV cache $h \in \mathbb{R}^m$. Given a context-specific dataset $\mathcal{D}_{qa} = \{(q_i, a_i)\}_i$, we define the *diamond* KV (DIAKV) h_{dia} as the KV that maximizes the KV posterior. This objective, as shown in equation 3.2, consists of the KV cache prior, the question likelihood, and the answer likelihood. See Appendix E for an extended discussion.

$$h_{dia}|\mathcal{D}_{qa} = \arg\max_{h \in \mathbb{R}^m} \left[\log p(h) + \sum_i (\log p_\theta(q_i|h) + \log p_\theta(a_i|q_i, h)) \right] \tag{3.2}$$

3.2 TRAINING DATA AUGMENTATION

Training with equation 3.2 requires context-specific QAs. We assume a small set of seed QAs is given for each context and use these as exemplars to generate a larger training set (Wang et al., 2023; Honovich et al., 2022). Specifically, we expand 20 seed QAs to 800 training examples. This augmentation is done for each context and must leave time for reuse before cache eviction, motivating two design choices for time-efficiency.

First, we decouple question and answer generation so that diversity and quality can be pursued separately within the time budget. Diversity is the job of question generation. Since larger models do not necessarily generate more diverse outputs (Zhang et al., 2025b; Jiang et al., 2025; Bansal et al., 2025), diversity is typically obtained through repeated sampling, either by over-generation and filtering (Jung et al., 2025; Li et al., 2024; Wang et al., 2023) or by generating diverse metadata (Lin et al., 2026; Ge et al., 2025), and its cost grows with the number of calls. We therefore use a small model, gpt-oss-20b (OpenAI, 2025), to repeatedly propose question candidates and aggressively deduplicate them, rejecting even a 4-gram overlap with an accepted question. Quality is the job of answer generation, where a larger model, GPT-5.2 (OpenAI, 2025) with medium reasoning effort, writes one answer per accepted question.

Second, we reduce generation time by exploiting QA locality (Choi et al., 2017; Karpukhin et al., 2020; Lewis et al., 2020): QA typically involves only part of the long context. We split the context into chunks and group mutually similar chunks into cliques. The question model uses one selected clique as its context, instead of the entire context. For each accepted question, the answer model receives that question and the same clique as its context. Under a uniform-chunk-length approximation, the context supplied per generation step averages just 1.1% of the full context, dramatically reducing the augmentation time.

Appendix C.2 gives the algorithm, prompts, and hyperparameters. §4.3 examines the resulting preparation time and the remaining long-tail risk from duplicate rejection during question augmentation.

3.3 TRAINING

Given the training data $\mathcal{D}_{qa}$, equation 3.2 can be optimized using standard stochastic optimization. We use AdamW (Loshchilov & Hutter, 2019) with learning rate 1e-3 and weight decay 0.1. AdamW weight decay implements the L2 regularization induced by our chosen KV prior, $h \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_m)$ (§3.1). We initialize training from the context's prefilled KV cache, thereby starting the optimization from a valid context-induced KV and biasing optimization toward a local solution; prior work similarly finds this preferable to random initialization (Eyuboglu et al., 2026). We only train the KV corresponding to the context itself, excluding the system prompt. As the system prompt lies on the beginning of the inputs, this also has the effect of freezing the attention sinks (Xiao et al., 2024; Gu et al., 2025), which is known to stabilize the training (Eyuboglu et al., 2026). The loss is calculated on all tokens of question and answer. See Appendix B.3 for more details.

Table 2: DIAKV results. ± in LLM-as-judge denotes standard deviation to account for stochasticity in LLM-as-judge. See Table 11 for comprehensive metrics.

	NarrativeQA		QASA-M		LH-M	FinText	
	ROUGE-L		*LLM-as-judge*		*Accuracy*	*LLM-as-judge*	
Model	macro	micro	macro	micro		macro	micro
Qwen3-4B-2507	0.167	0.169	**0.474**$_{\pm.003}$	0.464$_{\pm.002}$	0.717	0.325$_{\pm.015}$	0.371$_{\pm.010}$
+DIAKV	**0.301** (+80.2%)	**0.305** (+80.5%)	0.460$_{\pm.004}$ (-3.0%)	**0.467**$_{\pm.003}$ (+0.6%)	**0.728** (+1.5%)	**0.376**$_{\pm.005}$ (+15.7%)	**0.432**$_{\pm.007}$ (+16.4%)
Llama-3.2-3B	0.265	0.266	0.345$_{\pm.003}$	0.348$_{\pm.003}$	0.464	0.321$_{\pm.022}$	0.381$_{\pm.013}$
+DIAKV	**0.317** (+19.6%)	**0.320** (+20.3%)	**0.404**$_{\pm.003}$ (+17.1%)	**0.407**$_{\pm.003}$ (+17.0%)	**0.497** (+7.1%)	**0.377**$_{\pm.015}$ (+17.4%)	**0.392**$_{\pm.009}$ (+2.9%)
Ministral-3-3B	0.206	0.208	0.400$_{\pm.004}$	0.397$_{\pm.003}$	0.669	0.327$_{\pm.007}$	0.384$_{\pm.011}$
+DIAKV	**0.314** (+52.4%)	**0.318** (+52.9%)	**0.467**$_{\pm.003}$ (+16.8%)	**0.458**$_{\pm.003}$ (+15.4%)	**0.692** (+3.4%)	**0.341**$_{\pm.005}$ (+4.3%)	**0.405**$_{\pm.007}$ (+5.5%)

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

For datasets, we use NarrativeQA (NQA) (Kočiský et al., 2018), QASA (Lee et al., 2023b), LongHealth (LH) (Adams et al., 2025), and FinText (Chen et al., 2024). For QASA and LH, we create longer contexts by composing multiple *member documents* (papers and patient cases, respectively) into one context. The QAs for the composed context are the union of the QAs of all its member documents. We append the suffix "-M" for such cases. Our datasets span across different answer types and domains, while all commonly employing long context, with median word count of 74k. See Table 3 for summary statistics and Appendix B.1 for further details.

Table 3: Benchmark dataset statistics.

Name	#(ctx)	Median word count	Total QAs	Answer type	Domain
NQA	30	80k	892	Short	Fictional
QASA-M	9	39k	1207	Long	Academic
LH-M	2	55k	400	MC	Health
FinText	2	57k	115	Long	Finance
Total	**43**	**74k**	**2614**	–	–

For models, we use Qwen3-4B-Instruct-2507 (Team, 2025), Llama-3.2-3B-Instruct (AI@Meta, 2024), and Ministral-3-3B-Instruct-2512 (Mistral AI, 2025). For evaluation, we include all metrics from the respective original works plus a few additions, and base our analysis on ROUGE-L for NQA, accuracy for LH-M, and LLM-as-judge accuracy for QASA-M and FinText. For LLM-as-judge, GPT-5.1 decides the correctness of the predicted answer given the reference answer. As the LLM-as-judge verdict is non-deterministic even at temperature 0, we report the average of 5 calls with the standard deviation. See Appendix B.2 for details.

4.2 RESULTS

Table 2 reports the results of DIAKV across four benchmarks and three models. DIAKV improves over the base model in 20 of the 21 reported cases (3 models $\times$ 7 metrics), across answer types and domains; Llama-3.2-3B and Ministral-3-3B improve on every metric. The gains are largest on NQA (up to $+80\%$ ROUGE-L on Qwen3-4B-2507) and smallest on LH-M ($+1.5$–7.1%). The only drop is limited to the QASA-M macro score of Qwen3-4B-2507 ($0.474 \rightarrow 0.460$), whose micro score still improves. See Table 11 for comprehensive metrics.

4.3 OFFLINE PREPARATION TIME

Offline augmentation and training stay outside the online critical path, but must finish with time remaining for cache reuse.[1] We take 24 hours as an illustrative TTL: for reference, OpenAI supports retention up to 24 hours (OpenAI, 2026b), DeepSeek-V4.1-Flash guarantees at least 72 hours for global prefix KV (DeepSeek-AI, 2026b), and Google imposes no stated upper bound on cache expiration (Google Cloud, 2026). As explained in §3.2, the preparation pipeline of DIAKV has three

[1]Accuracy gains incur additional offline costs (Appendix C.3). Their economic justification depends on reuse and the application-specific value of improved answers; we do not claim lower total cost.

stages: question augmentation, answer augmentation, and training. Question augmentation is the only step with undetermined completion time, due to repeated proposals under deduplication. Here we report its actual time, to show the feasibility of DIAKV under realistic cache TTL.

Across 43 contexts (§4.1), question generation with gpt-oss-20b on one NVIDIA H100 takes a median of 9 minutes, which falls well below typical TTL. The risk of long tail is present though, as maximum question generation time is 16.6 hours. For this long-tail context, a 484-QA subset corresponding to 271 minutes of generation retains a score of 0.253 versus 0.263 for the full set, supporting early stopping to control the long tail. Appendix C.3 gives the timing distribution and ablation details.

The remaining stages have fixed workloads without the long-tail risk: one answer per accepted question and a fixed number of training steps. Answer generation can overlap with question generation, adding almost no overhead in overall time. Training on one NVIDIA H200 averages 10 minutes for Llama-3.2-3B and 16 minutes for Qwen3-4B-2507. Overall, §4.2 and 4.3 show that DIAKV improves base-model accuracy, with measured question generation and training times below the illustrative 24-hour TTL.

5 COMPARISON TO ALTERNATIVE DESIGN CHOICES

Three design choices are built into equation 3.2, but each must be compared against a well-established alternative. **(1) *Optimization space***: we optimize a continuous KV cache $h \in \mathbb{R}^m$ rather than the discrete text $c \in V^*$. §3.1 chose the continuous space to use gradient descent. However, LLMs are themselves strong optimizers of text (Agrawal et al., 2026; Yuksekgonul et al., 2025; Zhang et al., 2026), necessitating a comparison study. **(2) *Parametrization***: we update the KV cache of the context rather than the model weights θ, which are the standard target of fine-tuning and TTT. **(3) *Training signal***: we maximize the likelihood of the augmented QAs $\mathcal{D}_{qa}$. The alternative is a self-supervised loss on the context itself, used by TTT for long context (Bansal et al., 2026). We validate each choice against a representative method for its alternative (ACE, LoRA and full fine-tuning, and qTTT, respectively), with the remaining factors fixed. We compare accuracy and, where applicable, latency.

5.1 OPTIMIZATION SPACE: KV CACHE OVER TEXT

Text cannot be updated by gradients, so the textual alternative changes the optimizer as well: an LLM rewrites the context. We compare DIAKV to ACE (Zhang et al., 2026), which uses GPT-5.2 to iteratively update a task-specific *playbook* for a Qwen3-4B generator. We train ACE on the same data as DIAKV, treating QAs on each context as a distinct task to train for (implementation details in Appendix B.4).

Table 4: Comparison with ACE. FiNER, ACE's original task, serves as a positive control.

	FiNER*	FinText		LH-M	QASA-M	
	Acc.	*LLM-as-judge*		*Acc.*	*LLM-as-judge*	
Model		macro	micro		macro	micro
Qwen3-4B-2507	0.402	0.325	0.371	0.717	**0.474**	0.464
+DIAKV(n=800)	*n.a.*	**0.376**	**0.432**	**0.728**	0.460	**0.467**
+ACE (n=200)	0.488	0.336	0.382	0.719	0.471	0.455
+ACE (n=800)	**0.548**	0.288	0.358	—†	—†	—†

n.a.: FiNER lacks a reusable context; *ACE original-task control; †Omitted due to cost.

ACE improves accuracy on FiNER (Loukas et al., 2022), its original task, from 0.402 to 0.548, providing a positive control for our implementation (Table 4). On our context-level QA benchmarks, however, gains are at most marginal, and FinText performance declines as training increases from $n = 200$ to $n = 800$. This contrast suggests a task–method mismatch rather than a flaw in the method itself or insufficient training. Due to cost, we omit NQA and restrict the $n = 800$ QA evaluation to FinText.

Further analysis suggests that ACE's training gains generalize well in FiNER but poorly in FinText (Appendix D.6). Table 5 illustrates what textual learning captures in these tasks. In FiNER, the most helpful bullet provides an explicit task rule that can be reused across the task. In FinText, the most helpful bullet instead hints the document's structure, while another bullet overfits a particular training question. These examples suggest that reusable declarative rules are easier to extract for FiNER than for context-level QA, helping explain why textual optimization succeeds in the former but brings limited gains in the latter. Textual updates also incur serving overhead by invalidating the prefix cache or requiring additional prefill, whereas DIAKV directly reuses the trained KV cache.

Table 5: Examples of bullets from ACE. The 2nd row shows simplified bullets (unmodified ones in Table 9). The 3rd row shows simple explanations by the authors. (a) and (b) are respectively labeled by the reflector as the **most helpful** bullet from FiNER(n=800) and FinText(n=800) context #1.

(a) Helpful: FiNER	**(b) Helpful: FinText-1**	**(c) Unhelpful: FinText-1**
Revolver capacity clues: "up to $X million" → map to maximum borrowing capacity tags	*A bracketed source such as "[Reg. V, 72]" signals the FACTA framework.*	*If the prompt mentions a "segregated deposit account", predict a 100% collateralization threshold.*
Explicit plug-and-play hint	Helps identify the structure of the document	Overfitted to the training question

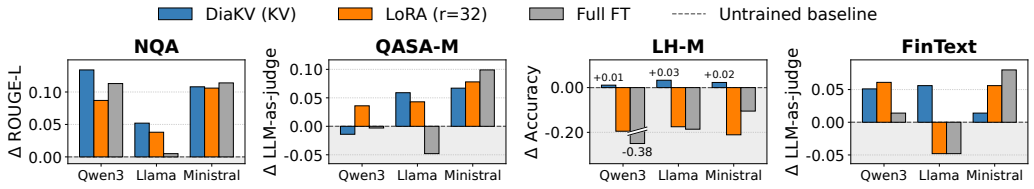


Figure 1: (Top) Gain over the untrained baseline (macro prime metric) after KV, LoRA ($r = 32$), or full fine-tuning on the same augmented QAs. See Table 11 for comprehensive results.

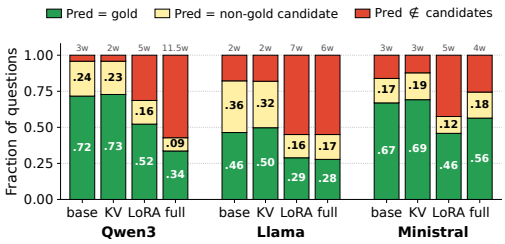


Figure 2: (Left) LH-M outcomes by parametrization. Green denotes accuracy; labels above bars give median prediction length in words.

5.2 Parametrization: KV cache over weights

Our primary motivation for training the KV cache rather than the weights, e.g., LoRA (Hu et al., 2022), is on the serving side. Serving trained KV requires the cache to be offloaded to storage and loaded on reuse, whereas LoRA adds adapter loading and computation. Loading is already the reasonable choice for contexts long enough for DiaKV: on vLLM (Kwon et al., 2023) with LMCache (Cheng et al., 2025), loading the cache from an SSD yields lower time to first token than prefill for 16k–128k contexts, and the gap widens with length, as prefill cost grows quadratically while cache size grows linearly. Multi-LoRA serving, in contrast, reduces throughput by 1.8–2.0% with as few as 3 adapters each serving 0.5% of requests, and the overhead grows with more adapters or a larger share of adapter requests. See Appendix D.4 for the setup and full results (Fig. 8).

The question is then whether training KV gives up accuracy relative to training weights. For comparison with weight tuning, we run three baselines on the same augmented QAs as DiaKV: full fine-tuning, LoRA (r=32) on attention and FFN, and LoRA (r=8) on attention. Fig. 1 shows the gain over the untrained baseline for the first two; see Appendix D.3 for the comprehensive result.

The result shows that training KV is as effective as training weights. Across 21 cases (3 models $\times$ 7 prime metrics), training KV outperforms LoRA in 14 and full fine-tuning in 13, but the outcome is model-dependent: it wins every case on Llama, whereas on Ministral weight tuning is ahead on QASA-M and FinText by up to 0.04 (LoRA) and 0.07 (full).

LH-M further shows an unintended strength: robustness to superficial patterns in the augmented data. For LH-M, all nine weight-tuning runs fall below the untrained baseline, by 5 to 49%p (Table 11), whereas training KV improves on all three models. LH-M is a multiple-choice (MC) QA whose prompt instructs the model to return exactly one of the candidates, and Fig. 2 classifies each prediction as the gold candidate (green), another candidate (yellow), or not a candidate (red), the last being a failure to follow the instruction. The red segment stays at 4–18% for the untrained models and after training KV, but grows to 26–57% after weight tuning, and accuracy falls with it. Predictions also lengthen, from a median of 2–3 words to 4–12 (numbers above the bars), toward free-form answers.

The cause lies in the augmented QAs. Although the 20 seed QAs are MC, our augmentation pipeline does not enforce their format, so nearly all augmented QAs are free-form (median answer length 13

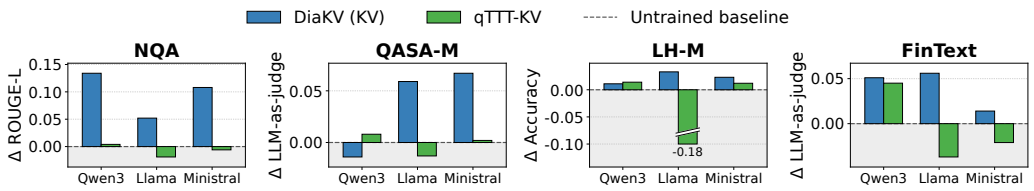


Figure 3: Gain over the untrained baseline on the prime metric (macro) from training KV on augmented QAs (DIAKV) and on the self-supervised span loss of qTTT (qTTT-KV) at matched FLOPs. The truncated bar is labeled with its value. See Table 12 for all metrics and the W_Q variant.

words). Weight tuning fits this superficial style and overrides the instruction, whereas training KV on the same data preserves instruction following and still improves by a modest margin. We attribute the difference to where the update lands: weight tuning alters the model behavior, whereas training KV alters only the representation of the context. Eyuboglu et al. (2026) reports a similar robustness of trained KV to out-of-distribution queries.

5.3 TRAINING SIGNAL: AUGMENTED QAS OVER SELF-SUPERVISED LOSS

We test whether seed-based augmented QAs are needed for DIAKV. Here we compare the supervised QA loss against a self-supervised loss; §6.1 ablates the seed QAs within supervised QA loss.

Our comparison to self-supervised loss is based on a test-time training method qTTT (Bansal et al., 2026), which originally trains the query projections of attention on short spans of the context. To compare with DIAKV, we make three changes. (1) Training is done offline, before any query arrives, rather than at test time. (2) KV cache is trained instead of W_Q (qTTT-KV), so the parametrization and serving latency are identical to DIAKV; the original W_Q variant is reported in Appendix D.5. (3) Training FLOPs is matched to DIAKV by setting the number of 128-token spans so that the total number of tokens used for training equal those of the 800 augmented QAs (Appendix B.3).

Fig. 3 shows the gain over the untrained baseline. DIAKV outperforms qTTT-KV in 10 of 12 cases. The two exceptions, QASA-M and LH-M on Qwen3, are the cases where DIAKV itself gains little or regresses (§4.2). Notably, qTTT highly depends on the model: qTTT-KV improves on all four benchmarks only for Qwen3, the only model family the original qTTT was evaluated on, and hurts on all four for Llama; the W_Q variant shows the same model dependence (Table 12). Overall, self-supervised loss from qTTT do not provide sufficient training signal compared to augmented QAs. qTTT reweights attention to retrieve what is already in the context, which pays off mainly where retrieval is the bottleneck (Bansal et al., 2026). DIAKV, in contrast, maximizes the likelihood of anticipated questions and answers, directly optimizing for the downstream use of the context.

6 SEED QA ANALYSIS

As mentioned in §3.2, DIAKV uses seed QAs to estimate the QA distribution given context, which raises the question of how much the gains depend on them. §6.1 shows that seed QAs support more consistent gains across benchmark-model pairs; in contrast, many context distillation setups (Eyuboglu et al., 2026; Chen et al., 2026) compress or consolidate the context without additional training signals. While additional training signals from seed QAs are likely beneficial for similar test queries, test queries may deviate from seed queries, risking potential degradation. §6.2 explores a setup where seed QAs are drawn only from certain parts of the context, and find that this obtains gains for seeded parts with almost no degradation on the remaining parts.

6.1 SEED QAS SUPPORT MORE CONSISTENT GAINS

We reduce the number of seed QAs used for augmentation from the default 20 to 5 and 0; Fig. 4 shows the gain over the untrained baseline. The results are mixed. Of the 11 benchmark-model pairs that DIAKV improves with 20 seed QAs, six keep the gain at both 5 and 0 seed QAs. In the remaining five, which span all three models, the gain becomes marginal or disappears. Seed QAs are thus not needed for every gain, but they make the gains more consistent.

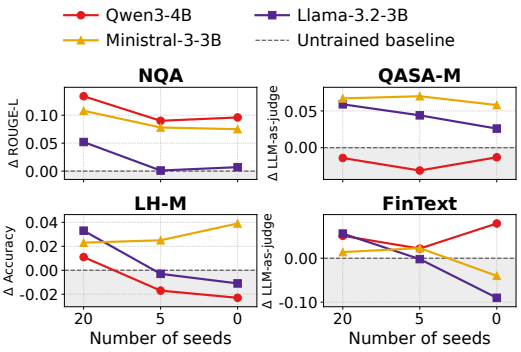


Figure 4: Ablation on the number of seed QAs (20, 5, and 0), shown as the gain over the untrained baseline.

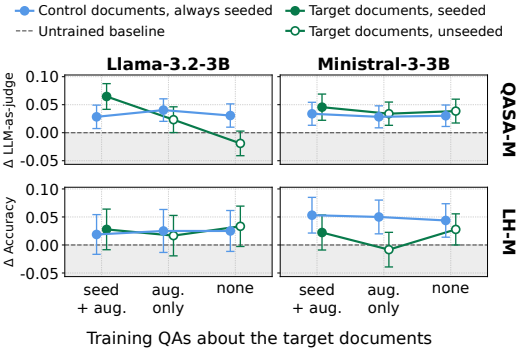


Figure 5: Gain on control and target member documents in QASA-M and LH-M. Control documents supply seed QAs in every setting; target documents are seeded only under seed + aug.

6.2 Parts without seed QAs can still benefit from DiaKV

In practice, seed QAs are likely drawn disproportionately from the context, i.e., some parts of the context do not contribute to seed at all. Can the performance for these uncovered parts retained in DiaKV? Two benchmarks allow a test for such setup, as each of their contexts concatenates *member documents* that carry their own QAs: papers in QASA-M and patient records in LH-M (§4.1).

In each context we split the member documents into a *control* group, which supplies seed QAs in every setting, and a *target* group, which may or may not supply seed QAs depending on the setting. About half of the member documents are targets in both benchmarks, and the total number of seed QAs is fixed to 20 in every setting. When target documents are seeded (*seed + aug.*, Fig. 5), control and target supply about half of seed QAs each. When target documents are unseeded, we try two different ways of augmentation. It either proceeds as usual, so target documents still receive augmented QAs, only generated with seeds from the control documents (*aug. only*, Fig. 5); or it is restricted to the control documents, so the training set says nothing about the target ones (*none*, Fig. 5). A strict control cannot rely on proprietary model like GPT-5.2, so we run the answer augmentation with DeepSeek-V4-Flash (DeepSeek-AI, 2026a) locally, including the DiaKV default *seed + aug.* setting. The absolute gains here therefore differ from Table 2. See Appendix D.2 for details.

Fig. 5 shows the results. As expected, seeded documents gain from DiaKV, both for control and target. Notably, when target documents are unseeded and receive only augmented QAs (*aug. only*), they still benefit from training, or at worst retain the untrained baseline (Ministral + LH-M, -0.008). Skipping the augmented QAs for unseeded documents altogether (*none*) gives mixed results: degradation on Llama + QASA-M and gains on the remaining three pairs.

Overall, the results show weak evidence of bias from seed locality, if any. However, this analysis alone does not resolve the seed-bias concern: member documents may be more heterogeneous, and bias may arise from factors beyond locality. Future work on more challenging workloads would benefit real-world use of DiaKV.

7 Conclusion

We introduced DiaKV, which adapts an LLM to a reused context by training its KV cache offline. This places adaptation between request-level test-time training and model-level fine-tuning, with no online training latency or serving computation beyond cache loading. Formulated as KV posterior maximization under a context-specific QA distribution, DiaKV improves 20 of 21 cases across four long-context QA benchmarks and three models, with preparation well within a realistic cache lifetime. Training the KV cache on augmented QAs outperforms textual context optimization and self-supervised training, and matches weight tuning without its serving overhead. These findings motivate studying LLM adaptation guided by context-specific supervision for other reused contexts, such as codebases or agent memories.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic datasets, implement methods, support qualitative data analysis, and interpret results. The synthetic datasets are part of our method: gpt-oss-20b and GPT-5.2 expand the seed QAs into training QAs (§3.2, Appendix C.2), with DeepSeek-V4-Flash writing the answers in the seed hold-out study (§6.2). LLMs are also components of our evaluation and baselines: GPT-5.1 is the LLM judge (§4.1), and GPT-5.2 is the reflector of ACE (Appendix B.4). We have not used generative AI tools to develop the theoretical formulation, formulate or prove mathematical claims, propose or refine hypotheses, design experiments or provide feedback on them, translate, or clean and reformat datasets. Additionally, we used generative AI tools to write the scripts that generate figures and tables, and to draft and edit parts of the paper for readability. We have reviewed all AI-assisted work: the authors reviewed and tested AI-assisted code, and checked every AI-assisted analysis and interpretation against the per-question outputs and result tables. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All four benchmarks are public, and Appendix B.1 details how we compose and filter their contexts. The three evaluated models are open-weight and decode at temperature 0 (Appendix B.2). Appendix B.3 lists the training hyperparameters and hardware of DIAKV and every baseline, and Appendix B.4 our modifications to ACE. For training data augmentation, Appendix C.2 gives the algorithm, hyperparameters, and prompts; questions are generated by the open-weight gpt-oss-20b throughout. Two components rely on proprietary models and cannot be reproduced exactly. First, the answers of the augmented QAs in the main experiments are generated by GPT-5.2, whose outputs are not deterministic and were observed to drift over time (Appendix D.2). For this reason, the control study in §6.2 generates all answers, including those of the default setting, with the open-weight DeepSeek-V4-Flash run locally. Second, the LLM-as-judge verdicts of GPT-5.1 vary across calls even at temperature 0, so we report the mean and standard deviation over 5 calls; all other metrics are deterministic. We will release our code and the augmented training QAs, so that DIAKV can be trained without regenerating them.

REFERENCES

Lisa Adams, Felix Busch, Tianyu Han, Jean-Baptiste Excoffier, Matthieu Ortala, Alexander Löser, Hugo JWL Aerts, Jakob Nikolas Kather, Daniel Truhn, and Keno Bressem. Longhealth: A question answering benchmark with long clinical documents. Journal of Healthcare Informatics Research, 9(3):280–296, 2025.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

AI@Meta. Llama 3 model card, 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), Proceedings of the 42nd International Conference on Machine Learning, volume 267 of Proceedings of Machine Learning Research, pp. 942–963. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/akyurek25a.html.

Hritik Bansal, Arian Hosseini, Rishabh Agarwal, Vinh Q. Tran, and Mehran Kazemi. Smaller, weaker, yet better: Training llm reasoners via compute-optimal sampling. In The Thirteenth International Conference on Learning Representations (ICLR), 2025. URL https://openreview.net/forum?id=3OyaXFQuDl.

Rachit Bansal, Aston Zhang, Rishabh Tiwari, Lovish Madaan, Sai Surya Duvvuri, Fnu Devvrit, David Brandfonbrener, David Alvarez-Melis, Prajjwal Bhargava, Mihir Kale, and Samy Jelassi. Let's (not) just put things in context: Test-time training for long-context LLMs. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=H0bcEdPCoc.

Steven Bird, Ewan Klein, and Edward Loper. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media, Inc., 2009.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-lora: Learning to instantly internalize contexts, 2026. URL https://arxiv.org/abs/2602.15902.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In Regina Barzilay and Min-Yen Kan (eds.), Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1870–1879, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1171. URL https://aclanthology.org/P17-1171/.

Jian Chen, Peilin Zhou, Yining Hua, Loh Xin, Kehui Chen, Ziyuan Li, Bing Zhu, and Junwei Liang. FinTextQA: A dataset for long-form financial question answering. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 6025–6047, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.328. URL https://aclanthology.org/2024.acl-long.328/.

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.

Daixuan Cheng, Yuxian Gu, Shaohan Huang, Junyu Bi, Minlie Huang, and Furu Wei. Instruction pre-training: Language models are supervised multitask learners. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pp. 2529–2550, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.148. URL https://aclanthology.org/2024.emnlp-main.148/.

Daixuan Cheng, Shaohan Huang, and Furu Wei. Adapting large language models via reading comprehension. In The Twelfth International Conference on Learning Representations, 2024b. URL https://openreview.net/forum?id=y886UXPEZ0.

Yihua Cheng, Yuhan Liu, Jiayi Yao, Yuwei An, Xiaokun Chen, Shaoting Feng, Yuyang Huang, Samuel Shen, Kuntai Du, and Junchen Jiang. Lmcache: An efficient kv cache layer for enterprise-scale llm inference. arXiv preprint arXiv:2510.09665, 2025.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. Coarse-to-fine question answering for long documents. In Regina Barzilay and Min-Yen Kan (eds.), Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 209–220, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1020. URL https://aclanthology.org/P17-1020/.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026a.

DeepSeek-AI. Deepseek-v4.1-flash: Pushing the limits of kv cache compression, 2026b.

Sabri Eyuboglu, Ryan Saul Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Ruoyu Liu, Atri Rudra, James Zou, Azalia Mirhoseini, and Christopher Re. Cartridges: Lightweight and general-purpose long context representations via self-study. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=0k5w8O0SNg.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. Cost-Efficient large language model serving for multi-turn conversations with CachedAttention. In 2024 USENIX Annual Technical Conference (USENIX ATC 24), pp. 111–126, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-41-0. URL https://www.usenix.org/conference/atc24/presentation/gao-bin-cost.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas, 2025. URL https://arxiv.org/abs/2406.20094.

Google Cloud. Context caching overview, 2026. URL https://docs.cloud.google.com/vertex-ai/generative-ai/docs/context-cache/context-cache-overview. Accessed September 17, 2026.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=78Nn4QJTEN.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL https://aclanthology.org/2020.acl-main.740/.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=CNL2bku4ra.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor, 2022. URL https://arxiv.org/abs/2212.09689.

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.

Jinwu Hu, Zitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. Test-time learning for large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), Proceedings of the 42nd International Conference on Machine Learning, volume 267 of Proceedings of Machine Learning Research, pp. 24823–24849. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/hu25z.html.

Jonas Hübotter, Sascha Bongni, Ido Hakimi, and Andreas Krause. Efficiently learning at test-time: Active fine-tuning of LLMs. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=NS1G1Uhny3.

Jinwoo Jeong and Jeongseob Ahn. Accelerating llm serving for multi-turn dialogues with efficient resource management. In Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS '25, pp. 1–15, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400710797. doi: 10.1145/3676641.3716245. URL https://doi.org/10.1145/3676641.3716245.

Liwei Jiang et al. Artificial hivemind: The open-ended homogeneity of language models (and beyond). In Advances in Neural Information Processing Systems, Datasets and Benchmarks Track, 2025. URL https://openreview.net/forum?id=saDOrrnNTz.

Zhengbao Jiang, Zhiqing Sun, Weijia Shi, Pedro Rodriguez, Chunting Zhou, Graham Neubig, Xi Victoria Lin, Wen-tau Yih, and Srinivasan Iyer. Instruction-tuned language models are better knowledge learners. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 5421–5434, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.296. URL https://aclanthology.org/2024.acl-long.296/.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Jaehun Jung, Seungju Han, Ximing Lu, Skyler Hallinan, David Acuna, Shrimai Prabhumoye, Mostafa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. Prismatic synthesis: Gradient-based data diversification boosts generalization in llm reasoning. In Advances in Neural Information Processing Systems, 2025. URL https://openreview.net/forum?id=R0dC7Xzwbk.

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 Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. Transactions of the Association for Computational Linguistics, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023/.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. In Jennifer Dy and Andreas Krause (eds.), Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pp. 2766–2775. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/krause18a.html.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of transformer language models, 2019. URL https://arxiv.org/abs/1904.08378.

Kalle Kujanpää, Pekka Marttinen, Harri Valpola, and Alexander Ilin. Efficient knowledge injection in LLMs via self-distillation. Transactions on Machine Learning Research, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=drYpdSnRJk.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In Second Conference on Language Modeling, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 6086–6096, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1612. URL https://aclanthology.org/P19-1612/.

Yejoon Lee, Philhoon Oh, and James Thorne. Knowledge corpus error in question answering. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 9183–9197, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.616. URL https://aclanthology.org/2023.findings-emnlp.616/.

Yoonjoo Lee, Kyungjae Lee, Sunghyun Park, Dasol Hwang, Jaehyeon Kim, Hong-In Lee, and Moontae Lee. QASA: Advanced question answering on scientific articles. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pp. 19036–19052. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/lee23n.html.

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.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. Self-alignment with instruction backtranslation. In The Twelfth International Conference on Learning Representations (ICLR), 2024. URL https://openreview.net/forum?id=1oijHJBRsT.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Jessy Lin, Vincent-Pierre Berges, Xilun Chen, Wen tau Yih, Gargi Ghosh, and Barlas Oguz. Learning facts at scale with active reading. In The Fourteenth International Conference on Learning Representations (ICLR), 2026. URL https://openreview.net/forum?id=mRi2cJDtIS.

Kevin Lin, Charlie Snell, Yu Wang, Charles Packer, Sarah Wooders, Ion Stoica, and Joseph E. Gonzalez. Sleep-time compute: Beyond inference scaling at test-time. arXiv preprint arXiv:2504.13171, 2025.

Luyang Liu, Jonas Pfeiffer, Jiaxing Wu, Jun Xie, and Arthur Szlam. Deliberation in latent space via differentiable cache augmentation. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), Proceedings of the 42nd International Conference on Machine Learning, volume 267 of Proceedings of Machine Learning Research, pp. 39261–39274. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/liu25bc.html.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. Cachegen: Kv cache compression and streaming for fast large language model serving. In Proceedings of the ACM SIGCOMM 2024 Conference, ACM SIGCOMM '24, pp. 38–56, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706141. doi: 10.1145/3651890.3672274. URL https://doi.org/10.1145/3651890.3672274.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Lefteris Loukas, Manos Fergadiotis, Ilias Chalkidis, Eirini Spyropoulou, Prodromos Malakasiotis, Ion Androutsopoulos, and Georgios Paliouras. FiNER: Financial numeric entity recognition for XBRL tagging. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 4419–4431, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.303. URL https://aclanthology.org/2022.acl-long.303/.

Nguyen Nhat Minh, Andrew Baker, Clement Neo, Allen G Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent LLM outputs. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=FBkpCyujtS.

Mistral AI. Ministral-3-3B-Instruct-2512 model card, 2025. URL https://huggingface.co/mistralai/Ministral-3-3B-Instruct-2512.

Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. Flipping the dialogue: Training and evaluating user language models. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=ykSmkVqzn4.

OpenAI. Reproducible outputs with the seed parameter. https://developers.openai.com/cookbook/examples/reproducible_outputs_with_the_seed_parameter. Accessed: 2026-03-24.

OpenAI. Update to GPT-5 system card: GPT-5.2, 2025. URL https://openai.com/index/gpt-5-system-card-update-gpt-5-2/.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

OpenAI. Batch API, 2026a. URL https://developers.openai.com/api/docs/guides/batch. Accessed September 17, 2026.

OpenAI. Prompt caching, 2026b. URL https://developers.openai.com/api/docs/guides/prompt-caching. Accessed September 17, 2026.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards llms as operating systems. arXiv preprint arXiv:2310.08560, 2023.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Henrique* Schechter Vera, Sahil* Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, Daniel Cer, Alice Lisak, Min Choi, Lucas Gonzalez, Omar Sanseviero, Glenn Cameron, Ian Ballantyne, Kat Black, Kaifeng Chen, Weiyi Wang, Zhe Li, Gus Martins, Jinhyuk Lee, Mark Sherwood, Juyeong Ji, Renjie Wu, Jingxiao Zheng, Jyotinder Singh, Abheesht Sharma, Divya Sreepat, Aashi Jain, Adham Elarabawy, AJ Co, Andreas Doumanoglou, Babak Samari, Ben Hora, Brian Potetz, Dahun Kim, Enrique Alfonseca, Fedor Moiseev, Feng Han, Frank Palma Gomez, Gustavo Hernández Ábrego, Hesen Zhang, Hui Hui, Jay Han, Karan Gill, Ke Chen, Koert Chen, Madhuri Shanbhogue, Michael Boratko, Paul Suganthan, Sai Meher Karthik Duddu, Sandeep Mariserla, Setareh Ariafar, Shanfeng Zhang, Shijie Zhang, Simon Baumgartner, Sonam Goenka, Steve Qiu, Tanmaya Dabral, Trevor Walker, Vikram Rao, Waleed Khawaja, Wenlei Zhou, Xiaoqi Ren, Ye Xia, Yichang Chen, Yi-Ting Chen, Zhe Dong, Zhongli Ding, Francesco Visin, Gaël Liu, Jiageng Zhang, Kathleen Kenealy, Michelle Casbon, Ravin Kumar, Thomas Mesnard, Zach Gleicher, Cormac Brick, Olivier Lacombe, Adam Roberts, Yunhsuan Sung, Raphael Hoffmann, Tris Warkentin, Armand Joulin, Tom Duerig, and Mojtaba Seyedhosseini. Embeddinggemma: Powerful and lightweight text representations, 2025. URL https://arxiv.org/abs/2509.20354.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

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 Hal Daumé III and Aarti Singh (eds.), Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pp. 9229–9248. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 7080–7106, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.333. URL https://aclanthology.org/2026.eacl-long.333/.

Sijun Tan, Xiuyu Li, Shishir G Patil, Ziyang Wu, Tianjun Zhang, Kurt Keutzer, Joseph E. Gonzalez, and Raluca Ada Popa. LLoCO: Learning long contexts offline. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pp. 17605–17621, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.975. URL https://aclanthology.org/2024.emnlp-main.975/.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu,

Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 13484–13508. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754/.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candes, and Tatsunori Hashimoto. Synthetic continued pretraining. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=07yvxWDSla.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. In Proceedings of the Twentieth European Conference on Computer Systems, EuroSys '25, pp. 94–109, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711961. doi: 10.1145/3689031.3696098. URL https://doi.org/10.1145/3689031.3696098.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. Generate rather than retrieve: Large language models are strong context generators. In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=fB0hRu9GZUS.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. Nature, 639:609–616, 2025.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time, 2026. URL https://arxiv.org/abs/2601.16175.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models.

In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=eC4ygDs02R.

Xiaoying Zhang, Baolin Peng, Ye Tian, Jingyan Zhou, Yipeng Zhang, Haitao Mi, and Helen M. Meng. Self-tuning: Instructing LLMs to effectively acquire new knowledge through self-teaching. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), Findings of the Association for Computational Linguistics: ACL 2025, pp. 5688–5724, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.297. URL https://aclanthology.org/2025.findings-acl.297/.

Yiming Zhang, Harshita Diddee, Susan Holm, Hanchen Liu, Xinyue Liu, Vinay Samuel, Barry Wang, and Daphne Ippolito. Noveltybench: Evaluating language models for humanlike diversity. In Proceedings of the Conference on Language Modeling (COLM), 2025b. URL https://openreview.net/forum?id=XZm1ekzERf.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24), pp. 193–210, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '18, pp. 1097–1100, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210080. URL https://doi.org/10.1145/3209978.3210080.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025. URL https://openreview.net/forum?id=VuVhgEiu20.

Adam Zweiger, Jyothish Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2026. URL https://openreview.net/forum?id=JsNUE84Hxi.

A LIMITATIONS

Our tested tasks are limited to QA. Context-adaptation using DIAKV has the potential to be applied for variety of reused contexts – codebase, chat history, or agent memory for examples. However, each of the applications likely require a different learning and augmentation strategy from one another, to truly improve beyond denoising the context. Furthermore, for some of the applications like codebase, composing multiple KV caches will be required (Yao et al., 2025), as the length of codebase typically exceeds the context length of LLMs. For such reasons, DIAKV focuses on long-context QA, a basic yet practical context-based task to show the benefit of DIAKV. We expect further works to expand the applications of DIAKV with application-specific approaches.

Our tested models fall short of the scale of production-level models. Our evaluations focus on 3-4B sized models (Qwen3-4B-Instruct-2507 (Team, 2025), Llama-3.2-3B-Instruct (AI@Meta, 2024), and Ministral-3-3B-Instruct-2512 (Mistral AI, 2025)). Our tested models are smaller compared to the state-of-the-art production-level models (Team et al., 2025; DeepSeek-AI, 2026a), due to compute resource constraints. However, our work provides a proof-of-concept for DIAKV, which future works can scale from.

Seed QAs are required for augmentation. As mentioned in §6.1, DIAKV shows consistent gain when seed QAs are given, potentially limiting its utility. However, we argue that without additional information about the context-specific QA distribution, substantial gains beyond denoising or preprocessing may be hard to achieve. Nevertheless, exploiting other types of distributional information to reduce the need for seed QAs presents a promising direction for future work, such as augmenting queries with verifiable rewards, e.g., unit tests (Lambert et al., 2025), or leveraging personal history or profiles for augmentation (Naous et al., 2026).

Storage cost for prefix cache has not been considered. DIAKV relies on the assumption that the KV cache of reused contexts will live long enough to be trained and then served. Despite the increasing adoption of lower-tier storage for prefix cache, storage cost still exists. However, DIAKV assumes KV cache to be evicted after an extended period of time, as implied in our efficiency-aware augmentation pipeline (§3.2) and analysis of offline preparation time (§4.3). Considering this bounded lifetime and the fact that DIAKV treats KV caches as temporary serving artifacts rather than persistent memory, we have not explicitly considered the cost of storage in our main text.

B EXPERIMENTAL SETUP

B.1 DATASET

Context composition and filtering To test the effectiveness of DIAKV , we use the datasets with minor modifications. The modification falls into three categories: (1) Composing multiple contexts into single context, (2) filtering based on number of QAs for context, and (3) filtering based on context quality. (1) and (2) were done in order to ensure that a single context gets a sufficient number of seed QAs and test QAs. Here we disclose the details for each benchmark.

NarrativeQA[2] (Kočiský et al., 2018): We filter contexts whose length is between 100k and 128k, based on Llama-3.2-3B-Instruct (AI@Meta, 2024) tokenizer. Then we exclude 3 contexts filled with gibberish HTML[3], resulting in total 30 contexts.

QASA[4] (Lee et al., 2023b): QASA originally constitutes various machine learning papers from arXiv. We cluster papers into a few research topics, and compose papers within the same topic to form one context per one topic. We end up with 9 sets of papers with the following topics: object detection & segmentation, CNN architectures & image classification, Transformers & pre-trained language models, sequence models, generative models, graph neural networks, NLP applications, 3D vision & geometry, and retrieval, ranking & recommendation.

LongHealth[5] (Adams et al., 2025): LongHealth originally constitutes 20 different synthetic patient cases, each accompanied by 20 questions. Each patient case contains between 5,090 and 6,754 words. Following prior work (Eyuboglu et al., 2026), we compose 10 patient cases into a single context, effectively using a report for a panel of 10 patients.

FinText[6] (Chen et al., 2024): When we accessed the FinText GitHub repository in Jan 2026, the authors of the dataset have made a notice that some of the documents in the original dataset were not available due to legal issues. Among 15 available documents, we filter 2 documents with more than 30 associated questions. One of the documents has a token length of 134,410 based on Llama-3.2-3B-Instruct tokenizer, exceeding the max sequence length of Llama-3.2-3B-Instruct (131,072). For this particular case, we truncate the context to length 130,560. This ends up truncating 3,850 tokens, which is only 2.9% of the entire context.

B.2 EVALUATION

Sampling parameters For evaluation, we set the sampling temperature of the models to 0.

[2] https://huggingface.co/datasets/deepmind/narrativeqa

[3] Truncated document ids for these contexts: 3ea97856, 7c963cf6, 935775d9

[4] https://github.com/lgresearch/QASA

[5] https://github.com/kbressem/LongHealth

[6] https://github.com/AlexJJJChen/FinTextQA

Evaluation metrics As mentioned in §4.1, we include all the metrics suggested by respective original paper, also adding metrics of our choice. Here we lay out comprehensive list of evaluation metrics, with the prime metric in bold. Tables 11, 10, and 12 report every metric.

NarrativeQA (NQA) (Kočiský et al., 2018) suggests BLEU-1, BLEU-4, **ROUGE-L**, and METEOR as metrics. Considering the concise answer format of NQA, we believe these metrics to be sufficient, without the need for LLM-as-judge.

QASA (Lee et al., 2023b) suggests ROUGE-1,2, and L as metrics. On top of these, we add **LLM-as-judge** to evaluate beyond surface-level match.

LongHealth (LH) (Adams et al., 2025) is originally a 5-way multiple-choice question answering, with accuracy as the metric. We prompt the model to answer the option itself, instead of outputting any option symbol like "1)" or "A)". We measure **accuracy** by exact matching the prediction to the gold answer. We also add SQuAD-like word-level F1 (Rajpurkar et al., 2016) for more soft matching.

FinText (Chen et al., 2024) suggests ROUGE-1,2,L, BLEU, and **LLM-as-judge** as metrics, which we adopt without any adjustment.

LLM-as-judge For LLM-as-judge, we use GPT-5.1 (gpt-5.1-2025-11-13) via OpenAI Responses API[7]. Reasoning effort is set to none, verbosity is set to low, and temperature is set to 0. We found that setting the temperature to 0 does not lead to deterministic output. This is a known issue in OpenAI API and has been tried to mitigate by the seed parameter (OpenAI). However, as of March 2026, we have checked that the seed parameter is marked deprecated in OpenAI Chat Completions API[8], and not supported in Responses API. Therefore, we make a best-effort attempt and report an average and standard deviation across 5 API calls. Box B.1 shows the prompt used for LLM-as-judge.

Prompt B.1: LLM-as-judge prompt

Given one question, there is a ground truth and a predicted answer. Please decide whether they are the same in semantics. Please only output True or False. Question: <query> ground truth = <gold_answer> predicted answer = <prediction>

B.3 TRAINING

Here we detail out the configuration used for training KV and weight tuning baselines. All trainings are done on NVIDIA A100 80GB or H200.

DIAKV: learning rate 1e-3, weight decay 0.1, effective batch size 2, number of epochs 3. AdamW is used with linear warmup and cosine decay, where warmup steps is 10 (15 for FinText) with start factor 0.1.

Full fine-tuning: learning rate 5e-5, weight decay 0.1, effective batch size 4, number of epochs 2. AdamW is used with linear warmup and cosine decay, where warmup steps is 10 with start factor 0.1.

LoRA(r=32): learning rate 1e-4, weight decay 0.1, effective batch size 4, number of epochs 2. LoRA rank 32, alpha 32, and target modules are attention and FFN weights, i.e., q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, and down_proj. AdamW is used with linear warmup and cosine decay, where warmup steps is 10 with start factor 0.1.

LoRA(r=8): learning rate 5e-4, weight decay 0.1, effective batch size 4, number of epochs 2. LoRA rank 8, alpha 8, and target modules are attention, i.e., q_proj, k_proj, v_proj, o_proj. AdamW is used with linear warmup and cosine decay, where warmup steps is 10 with start factor 0.1.

qTTT-KV and qTTT-WQ: Both use the span loss of qTTT (Bansal et al., 2026). After a single prefill of the context, a 128-token span of the context is appended after the cache and trained with next-token prediction while attending to the entire cache; no question, answer, or seed QA is used. qTTT-WQ trains the query projection W_Q of every layer, as in the original work, and qTTT-KV trains the same KV cache as DIAKV. To match the FLOPs of DIAKV, the number of spans N is set per

[7] https://api.openai.com/v1/responses

[8] https://api.openai.com/v1/chat/completions

dataset and tokenizer so that the span tokens over 3 epochs equal the tokens DIAKV forwards over its augmented QAs and 3 epochs: for Qwen3-4B-2507, Llama-3.2-3B, and Ministral-3-3B respectively, N is 507, 512, and 486 on NQA; 1,087, 1,087, and 1,067 on QASA-M; 858, 846, and 797 on LH-M; and 834, 812, and 817 on FinText. Span start positions are sampled uniformly from the context once per run and visited in a fresh permutation each epoch, one span per update. AdamW with a constant learning rate of 1e-3 (qTTT-KV) or 1e-5 (qTTT-WQ), weight decay 0.01, batch size 1, gradient clipping at 1.0, and 3 epochs. Evaluation is identical to DIAKV.

B.4 IMPLEMENTATION OF ACE

For implementation of ACE (Zhang et al., 2026), we use its official implementation[9]. The generator is Qwen3-4B-2507, and the reflector and curator are GPT-5.2 (gpt-5.2-2025-12-11) with medium reasoning effort (Appendix D.6). We make two modifications. (1) We simplify the evaluation prompt to simply output the answer, not the JSON with additional information, which helps evaluating the true capability of smaller generator models, aside from JSON formatting burden. (2) We hard force the playbook budget by bullet pruning. As of September 25, 2026, the official ACE implementation lacked bullet refinement to keep the playbook compact, which is mentioned in the paper. We alternatively implement a simple bullet pruning, where bullets are pruned when the playbook reaches the budget. The bullets are pruned in order of highest harmful count, lowest helpful count, longest bullet, then the oldest. The playbook budget is set to 16k tokens. Crucially, we validate that such changes do not harm the effectiveness of ACE itself, as observed in the significant gain in FiNER (Loukas et al., 2022) (see Table 4), the task used in the original paper.

C AUGMENTATION

C.1 COMPARISON TO PRIOR WORKS

There have been several works on data augmentation for adapting LLMs, where the methods largely fall into two categories: (1) continual pre-training (CPT) where the model is trained with next token prediction (self-supervised learning) on a given or augmented corpus, (2) supervised finetuning (SFT) where the model is trained with SFT loss on a given or augmented input-output pairs. We base this distinction on whether training behaves more like pre-training or SFT. The conclusions from prior works are generally mixed, while some works advocate for SFT as they resemble the test task more (Cheng et al., 2024a; Jiang et al., 2024; Cheng et al., 2024b; Zhang et al., 2025a), while others advocate for CPT based on their generality (Yang et al., 2025) or empirical advantage (Lin et al., 2026).

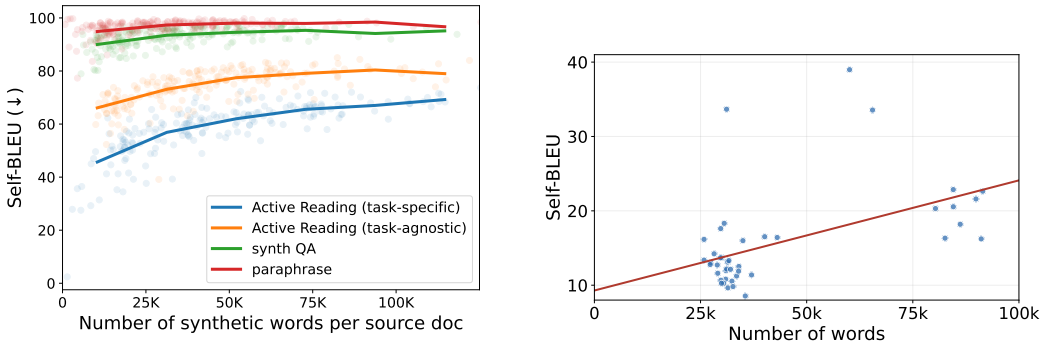
In DIAKV, we opt for SFT style given the setting that contextual information is retained in the prompt. Prior works using CPT mainly apply for internalizing the contexts into the model weights (Yang et al., 2025; Lin et al., 2026), resembling the goal of pre-training. In the case of DIAKV however, the context is retained as a part of the prompt, reducing the need for the model to learn the context itself, whereas more training signal can be provided from the QAs for the context.

The prior work of Active Reading (Lin et al., 2026) has shown the benefit of strategy-based augmentation over different augmentation methods, including the synthetic QA. The work hypothesizes that synthetic QAs underperform, due to the lack of diversity in the data (Lin et al., 2026). However, here we show that our augmentation pipeline overcomes this issue. We follow Active Reading and measure the self-BLEU (Zhu et al., 2018) of augmented dataset, where lower self-BLEU means more diversity. Fig. 6 shows that self-BLEU (Zhu et al., 2018) of our augmented QA falls well below the reported self-BLEU of either synthetic QA or Active Reading itself (Lin et al., 2026). Therefore, we conclude that while the lack of diversity is problematic for augmented data, it depends more on the specific pipeline rather than the high-level method itself.

C.2 DETAILS OF AUGMENTATION

Design rationale. The augmentation pipeline balances quality, diversity, and inference cost. Larger models can improve answer quality without necessarily improving question diversity (Zhang et al.,

[9] `https://github.com/ace-agent/ace`, commit `d241ef8`



(a) (b)

Figure 6: (a) The figure from Active Reading (Lin et al., 2026), without any modification. (b) Self-BLEU for our QA dataset. Each point refers to each context. The dataset includes both the questions and answers, and both the seed and augmented QAs. Self-BLEU (Zhu et al., 2018) (lower the more diverse) of our augmented QA falls well below the reported self-BLEU of either synthetic QA or Active Reading itself (Lin et al., 2026).

2025b; Jiang et al., 2025; Bansal et al., 2025). Diversity is often obtained through over-generation and filtering (Jung et al., 2025; Li et al., 2024; Wang et al., 2023), or repeated inference to obtain diverse metadata (Lin et al., 2026; Ge et al., 2025); both increase the number of model calls. Since DIAKV repeats augmentation for each context, we use a smaller model for diverse question proposals and reserve a stronger model for generating an answer once a question is accepted. Conditioning both stages on a small group of relevant chunks also avoids repeatedly supplying the full long context (Yang et al., 2025).

Algorithm. Algorithm 1 gives the question-generation procedure. For chunks $\{s_i\}$, we construct a graph with an edge whenever the cosine similarity between chunk embeddings is at least a threshold τ, and cycle through its cliques. Each call receives one clique, randomly sampled seed exemplars, and an instruction varying the persona (Ge et al., 2025) and a word the question must use (the opener). A proposed question enters the pool only after deduplication. Answers are subsequently generated using the clique associated with each accepted question.

Hyperparameters for question augmentation are presented in Table 7. Chunk sizes are denoted in terms of characters. We use langchain-text-splitters[10] for chunking. Chunking is purely based on characters without using additional structure, except for respecting boundaries of each original context in QASA-M and LH-M. For embedding model, we use embeddinggemma-300m (Schechter Vera et al., 2025) by using sentence-transformers (Reimers & Gurevych, 2019). We perform approximate question deduplication using MinHash over lowercased token n-grams. Questions are tokenized with NLTK (Bird et al., 2009), which separates punctuation and contractions more carefully than whitespace splitting, and n-grams are then formed over these tokens. For each accepted question, we cache MinHash signatures for token 2-, 3-, and 4-grams. A newly generated question is rejected if its estimated Jaccard similarity with any previously accepted question exceeds any configured threshold: 0.5 for 2-grams, 0.2 for 3-grams, or 0.00001 for 4-grams, so any detected 4-gram overlap is treated as a duplicate.

Box C.1 shows the prompt used for question augmentation. <X> is either one of "policy document" / "medical records" / "set of papers" / "story". Instructions such as <persona> and <opener> differ by dataset, and are also generated by the question augmentation model, gpt-oss-20b. See Table 6 for the actual values. For sampling parameters, max tokens is set to 2000, temperature is set to 1.0. We set reasoning effort as "high" via a system message. All question augmentation runs on a single NVIDIA H100 using vLLM (Kwon et al., 2023).

For answer augmentation, we use GPT-5.2 (gpt-5.2-2025-12-11) with medium reasoning effort. Only the clique used to generate the question, not the entire context, is fed as the input, together with all 20

[10] https://pypi.org/project/langchain-text-splitters/

seed QAs of the context as examples. Box C.2 shows the input prompt for the answer augmentation model. As in question augmentation, <X> is either one of "policy document" / "medical records" / "set of papers" / "story".

Algorithm 1: Question Augmentation

Input: context c, target count N, similarity threshold τ, chunking config `chunkCfg`, embedding function `embed`, human seed questions S_{human}, few-shot count K, instruction option sets $\mathcal{I}$, deduplication predicate `dedup`, language model `LLM`

Output: question set $Q = \{q_1, \ldots, q_N\}$

$s \leftarrow \texttt{SplitAndMerge}(c, \texttt{chunkCfg})$; `// text chunks` $[s_0, \ldots, s_{M-1}]$

$V \leftarrow \{s_i\}, \quad \mathbf{A}_{ij} \leftarrow \mathbb{1}[\cos(\texttt{embed}(s_i), \texttt{embed}(s_j)) \geq \tau]$

$G \leftarrow (V, \mathbf{A})$

$\texttt{cliques} \leftarrow \texttt{FindCliques}(G)$, sorted by $|\texttt{clique}|$ descending

$L \leftarrow |\texttt{cliques}|$

$Q \leftarrow S_{\text{human}}$; `// pre-populate pool with human seeds`

$i \leftarrow 0$

while $|Q| < N$ **do**

  $\ell \leftarrow i \bmod L$; `// cycle through cliques`

  $\texttt{inst} \leftarrow \texttt{NextUniqueCombo}(i, \mathcal{I})$; `// cycle through` $\prod_d |\mathcal{I}_d|$ `combos`

  $\texttt{seeds} \leftarrow \texttt{Sample}(S_{\text{human}}, K)$

  $\texttt{chunks} \leftarrow [s_j \mid j \in \texttt{cliques}[\ell]]$

  $q \leftarrow \texttt{LLM}(\texttt{chunks}, \texttt{seeds}, \texttt{inst})$

  if not $\texttt{dedup}(q, Q)$ **then**

    $Q \leftarrow Q \cup \{q\}$

  end

  $i \leftarrow i + 1$

end

return Q

Prompt C.1: Question augmentation prompt

You will be given few chunks from <X> and example questions for <X>. Your task is to generate a new question about <X>.

Chunks from <X>:

<chunk 1>
<chunk 2>
...

Example questions from the entire <X> (not just these chunks):

- <seed question 1>
- <seed question 2>
...

Now generate a grounded and creative question about <X>. You are <persona>. The question must use the word "<opener>" in the question. Only return the question, no other text.

Prompt C.2: Answer augmentation prompt

Answer the question based on the <X>.

Relevant chunks from the <X>:

<chunk 1>
<chunk 2>
...

Below are some examples from the same <X> but not just these chunks.
Use these examples to guide the style and verbosity of your answer.

Question 1: <seed question 1>
Answer 1: <seed answer 1>

Question 2: <seed question 2>
Answer 2: <seed answer 2>
...

Question: <target question>
Make sure your answer is concise and accurate. Only return the answer, no other text.

Table 6: Instruction variations used for question augmentation. All generated by the question augmentation model, gpt-oss-20b.

Dataset	**Persona**	**Opener**
QASA-M	• a professional in this field, knowledgeable about the topic and understands the background well, trying to understand the papers in depth. • a reviewer of the papers, scrutinizing and assessing the strengths and weaknesses of the papers • a student studying these papers, eager to learn and understand the topic.	what, why, how, is/are, do/does/did, can
FinText	• a compliance officer reviewing the policy document carefully. • a financial analyst extracting key details and implications from the policy document. • a customer reading the policy document to understand what is covered and what is not.	what, why, how, is/are, do/does/did, can
LH-M	• a clinician reviewing the patient's medical history. • a medical student studying the case for educational purposes. • a healthcare auditor verifying the accuracy of medical records.	what, why, how, is/are, do/does/did, which
NQA	• a literature teacher writing a comprehension question about the story. • a book club member discussing details of the story. • an avid reader trying to understand the story deeply.	where/when/who, what, why, is/are, how, which

Table 7: Hyperparameters for question augmentation.

Dataset	Chunk size	Chunk overlap	Min chunk size	Edge similarity threshold	Number of shots
NQA	1200	120	600	0.8	12
QASA-M	1600	160	1000	0.9	8
LH-M	1600	160	1000	0.9	4
FinText	800	80	200	0.8	4

C.3 Offline preparation time and cost

Question generation and its long tail. Fig. 7 shows question-generation wall time for all 43 contexts on a single NVIDIA H100 using gpt-oss-20b. The median is 9 minutes, while the slowest FinText context takes 997 minutes (16.6 hours). Compared with the fastest context (208 seconds), the slowest takes only $3\times$ longer per request but issues $111\times$ more requests, indicating that repeated proposals under deduplication drive the long tail.

We compare two ways to reduce preparation time on this slowest context: (a) increasing the sampling temperature to 2.0 with min-p sampling (Minh et al., 2025) ($p = 0.05$), and (b) reducing the training set size. Setting (a) reaches the full target in 271 minutes; the time-equivalent subset of the original generation run contains 484 QAs. With Qwen3-4B-2507, the full original set yields an LLM-as-judge accuracy of 0.263 on this context, compared with 0.158 for (a) and 0.253 for training on the reduced set in (b). The reduced-set result supports early stopping as a way to manage the tail, without changing the sampling configuration.

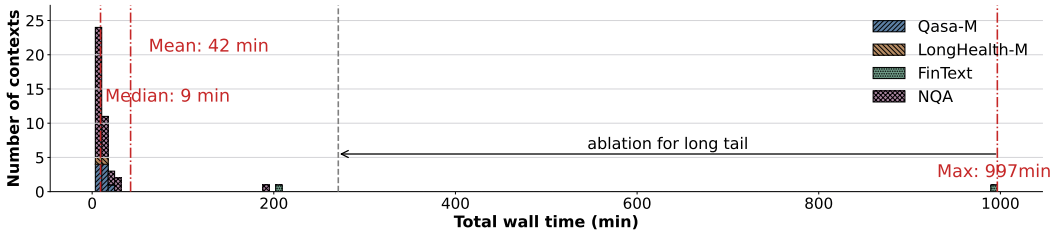


Figure 7: Question-generation wall time across 43 contexts using gpt-oss-20b on a single NVIDIA H100. The median is 9 minutes and the maximum is 997 minutes. For the slowest context, training Qwen3-4B-2507 on a 484-QA subset corresponding to 271 minutes of generation yields 0.253 LLM-as-judge accuracy, compared with 0.263 for the full set (§4.3).

Answer generation and training. Answer generation makes one call per accepted question. Because each answer depends only on its accepted question and the associated chunks, answer generation can overlap with question generation, adding almost no overhead in overall time. Training has a fixed number of steps. On a single NVIDIA H200, average training time is 10 minutes for Llama-3.2-3B and 16 minutes for Qwen3-4B-2507, with maxima of 21 and 35 minutes, respectively. We report stage-level measurements here; overall completion also depends on answer-generation scheduling.

The 24-hour TTL in §4.3 is an illustrative retention setting, not a measured deployment lifetime or a guarantee of OpenAI cache availability. OpenAI's extended retention is an upper limit; Google's explicit caching allows a chosen expiration time (OpenAI, 2026b; Google Cloud, 2026).

Monetary cost. For GPT-5.2 answer generation through the batch API (OpenAI, 2026a), each context consumes an average of 1.5M prompt tokens and 114k output tokens, costing $2.0 on average and at most $6.3. Including question generation and KV training, the estimated mean pipeline cost is approximately $3.4 per context. GPU costs use on-demand rates of $2.95 per H100-hour and $3.5 per H200-hour from Nebius on May 6, 2026. These are measurements and estimates for the reported setup, rather than a prediction of cost at larger model scales. Moving preparation offline removes it from the request's critical path but does not eliminate this expense. Our measured benefit over ordinary cache reuse is improved answer quality. Reuse spreads the extra preparation expense across requests; whether the gain justifies this expense also depends on the application-specific value of improved answers. We therefore make no claim of lower total cost.

Table 8: Seed hold-out study: the numbers behind Fig. 5. Gain of DIAKV over the untrained baseline on test questions about control / target documents under the three training settings (standard error over questions in parentheses), and the paired interaction of each hold-out setting against the default one, i.e., the drop of the target side relative to the control side. QASA-M: LLM-as-judge on 945 shared questions (515 control / 430 target); LH-M: accuracy on 340 (160 / 180), with the default and *aug. only* settings averaged over two training replicas.

		seed + aug. (default)		aug. only		none		interaction *vs* default	
Benchmark	Model	control	target	control	target	control	target	aug. only	none
QASA-M	Llama-3.2-3B	+.028 (.021)	+.065 (.023)	+.040 (.020)	+.023 (.023)	+.031 (.021)	-.019 (.022)	+.053 (.026)	+.086 (.029)
QASA-M	Ministral-3-3B	+.034 (.021)	+.046 (.023)	+.028 (.020)	+.034 (.021)	+.030 (.019)	+.039 (.021)	+.006 (.023)	+.003 (.025)
LH-M	Llama-3.2-3B	+.019 (.035)	+.028 (.036)	+.025 (.038)	+.017 (.036)	+.025 (.037)	+.033 (.036)	+.017 (.036)	+.001 (.038)
LH-M	Ministral-3-3B	+.053 (.032)	+.022 (.031)	+.050 (.030)	-.008 (.031)	+.044 (.030)	+.028 (.028)	+.027 (.025)	-.015 (.030)

D ADDITIONAL RESULTS

D.1 NUMBER OF SEEDS ABLATION

See Table 10.

D.2 SEED HOLD-OUT STUDY

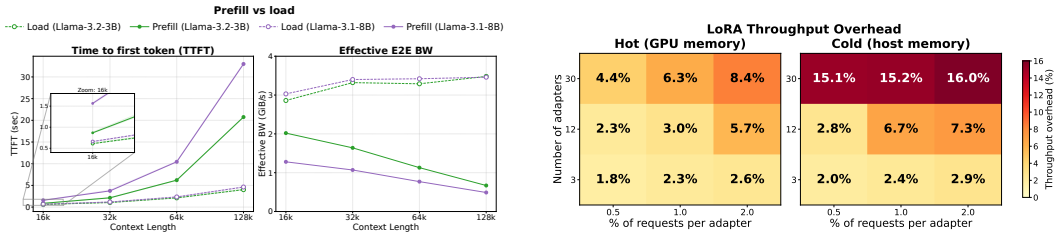
This section gives the details behind §6.2. Table 8 lists every number in Fig. 5 together with the paired interaction of each hold-out setting (*aug. only* and *none*) against the default one (*seed + aug.*).

Split. QASA-M has nine contexts of 5–13 papers. We assign 43 of the 80 papers to the control group and draw the 20 seed QAs of a context from its control papers in proportion to their QA counts, 1–9 per paper; per context, the numbers of papers and of control / target papers are 13 (7/6), 12 (6/6), 11 (6/5), 6 (3/3), 13 (7/6), 5 (3/2), 8 (4/4), 5 (3/2), and 7 (4/3). LH-M has two contexts of ten patient records; five records per context are controls, at the odd positions of the first context and the even positions of the second, with four seed QAs each, so the default seeds of a control record are among its hold-out seeds. The hold-out settings remove the hold-out seeds from the test set, whereas the default setting removes its own, so every comparison uses the questions shared by the three settings: 945 on QASA-M (515 control / 430 target) and 340 on LH-M (160 / 180).

Data. Questions are generated by gpt-oss-20b as in Appendix C.2, with 8 (QASA-M) or 4 (LH-M) seed questions as exemplars sampled from the context's 20 seeds; under the hold-out seeds every exemplar therefore comes from a control document. The default setting reuses the question set of the main experiments, while the two hold-out settings share one new generation run under the hold-out seeds. All answers, in every setting, are generated by DeepSeek-V4-Flash in thinking mode from the question's clique and the 20 seed QAs, with the same prompt and on the same day per benchmark. This re-answering is a control: a pilot showed that the answers of the GPT-5.2 endpoint used for the main experiments had drifted since they were collected (about 13% longer, with more markup), which would confound a comparison between an old and a new training set. We run DeepSeek-V4-Flash (DeepSeek-AI, 2026a) from its released checkpoint[11] locally with vLLM 0.27.1 (Kwon et al., 2023), using tensor parallelism over two NVIDIA B200 GPUs. The *none* setting keeps the seed QAs and only the augmented QAs whose clique lies entirely within control documents, i.e., 46–62% of the augmented QAs per QASA-M context and 49% per LH-M context; nothing is regenerated, so the QAs about control documents are identical between the two hold-out settings.

Training and evaluation. The training recipe is that of the main experiments (Appendix B) in every setting; the *none* setting keeps it unchanged and therefore runs about half the optimizer steps. The gain of a question is its score after the last epoch minus its score under the untrained cache, averaged per side, with the standard error taken over questions. On LH-M, the default and *aug. only* settings were trained twice with fresh training noise on the same data and per-question gains are averaged over the two runs; between runs, per-question gains correlate at 0.88–0.96 and 97–99% of the final

[11] https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash



(a) (b)

Figure 8: (a) Time to first token and effective end-to-end bandwidth for loading KV *vs* computing prefill in two models. (b) Throughput overhead from using multi-LoRA. Left shows when all adapters reside on GPU memory; right shows when all the adapters except for the active one are offloaded to host memory. The throughput overhead here excludes the prefill phase.

per-question verdicts agree, which puts training noise at about ± 0.01–0.02 on a side mean. Every QASA-M setting and the LH-M *none* setting are single runs. The interaction of a setting against the default one is the mean over control questions of its per-question gain minus the default gain, minus the same mean over target questions; its standard error combines those of the two paired differences.

Caveats. (i) With single runs on QASA-M and standard errors of about 0.03 on the LH-M side means even after two replicas, only the interaction of Llama-3.2-3B on QASA-M under *none* exceeds two standard errors; the flat cells exclude interactions above about 0.06–0.08 but not smaller ones, so the figure should be read as the absence of a large locality bias rather than proof of none. (ii) The *none* setting trains on about half the rows and steps; its control-side gains match those of *aug. only* within 0.01, so the step count does not visibly matter, but it confounds the absolute totals. (iii) The default setting's questions come from a different generation run than the hold-out settings'; the stability of the control side across settings argues against a run-level quality difference but does not exclude it. Separately, because all answers are regenerated by DeepSeek-V4-Flash, the default setting here is a rerun, not the run of Table 2. (iv) LH-M answers written by DeepSeek-V4-Flash are terse (median three words), and the absolute gains under exact match are small. (v) The cause of the harm done by augmented QAs written without a document's own seeds is open: a feature audit of those QAs against the default QAs about the same documents found no difference in question or answer length, yes/no rate, hedging rate, or lexical attribution to the right document.

D.3 *vs* WEIGHT TUNING

See Table 11.

D.4 SERVING COST: TRAINED KV *vs* LORA

This section provides the setup and full results behind the serving argument in §5.2: (a) for contexts long enough to apply DIAKV, loading KV is faster than prefill, and (b) multi-LoRA serving incurs non-negligible throughput overhead.

(a) Loading KV *vs* prefill Our experiments are based on vLLM 0.18.0 (Kwon et al., 2023) and LMCache 0.4.2 (Cheng et al., 2025). For context length ranging from 16k to 128k, we measure time to first token (TTFT) when doing prefill *vs* loading the KV cache from storage. For measuring load TTFT, we first send a request to populate the KV cache in storage, and then send the same request from a separate process. We have not used any prefix caching on GPU or CPU memory. All experiments are run on a single NVIDIA A100 and Samsung PM1733 with PCIe Gen 4.

Fig. 8a shows load yields lower TTFT than prefill across all context lengths for Llama-3.2-3B and Llama-3.1-8B. This advantage widens with larger context length as prefill computation scales quadratically with context length, whereas KV cache size scales linearly. As a result, the effective end-to-end bandwidth of prefill falls as low as 0.5GiB/s for 128k context on Llama-3.1-8B, providing an opportunity for exploiting more affordable, lower bandwidth storage at scale (Liu et al., 2024; Gao et al., 2024; Cheng et al., 2025; Yao et al., 2025; Jeong & Ahn, 2025). The dominance of load may

not sustain for shorter contexts, but we believe such short contexts are not suitable for DIAKV in the first place, as they do not provide enough training signals.

(b) Throughput overhead of multi-LoRA serving Using vLLM 0.18.0 (Kwon et al., 2023), we evaluate multi-LoRA serving with varying numbers of adapters and request shares. We exclude KV-preparation latency because both DIAKV and LoRA can use KV offloading for long contexts, making this phase common to both methods. To isolate adapter overhead, we use one shared context cached in GPU memory, run LH-M and QASA-M workloads (§4.1), repeat each context three times, and report the median on a single NVIDIA A100 80GB with PCIe Gen4 CPU–GPU connectivity.

Fig. 8b shows the median throughput overhead over 6 contexts, with differing number of adapters and portion of adapter requests. The left and right panels respectively show two extreme strategies: all adapters reside on GPU memory *vs* only the active one resides on GPU memory while all the others are offloaded to host memory. The first *hot* scenario has the benefit of eliminating any load latency, but suffers from less memory allocated to KV cache, which leads to less request concurrency. The opposite happens in the second *cold* scenario. Even when there are as few as 3 adapters, with each accounting for 0.5% of the total requests, the throughput overhead is non-negligible, ranging from 1.8% to 2.0%. The overhead is exacerbated as there are more adapters or more requests using each adapter, raising concerns on the scalability of adapter-based context-level adaptation. Overall, even with a state-of-the-art serving engine, multi-adapter inherently incurs throughput overhead.

D.5 *vs* SELF-SUPERVISED LOSS

Table 12 reports all metrics for qTTT-KV and for the original W_Q variant (qTTT-WQ), both trained at the FLOPs of DIAKV (Appendix B.3). qTTT-WQ produces a per-context weight delta and therefore carries the serving overhead of an adapter (§5.2), which is why it is excluded from the main text. On Qwen3, qTTT-WQ improves on all four benchmarks and exceeds DIAKV on QASA-M, LH-M, and FinText, but it falls below the untrained baseline on all four benchmarks for Llama and on three for Ministral, the same model dependence as qTTT-KV. One difference between the settings is where the update meets normalization: Qwen3 applies RMSNorm to queries and keys after projection (Team, 2025), so an update to W_Q is re-normalized per head before it enters the attention logits, whereas the KV cache trained by qTTT-KV and DIAKV is stored after the norm in every model. Whether this accounts for the Qwen3-only gain of qTTT-WQ is left for future work; we note that the original qTTT was evaluated only on Qwen3 models (Bansal et al., 2026).

D.6 ANALYSIS ON *vs* TEXTUAL METHODS

Following §5.1, we present an analysis of ACE on FiNER (Loukas et al., 2022) vs. FinText (Chen et al., 2024). Through this analysis, we argue that ACE produces bullets that fail to generalize to unseen test instances.

First we describe the training loop of ACE (Zhang et al., 2026). For each training sample, ACE first asks the Generator—Qwen3-4B-2507 in our experiment—to answer using the current playbook. The answer is scored using the target metric, LLM-as-judge in the case of FinText. If the answer is wrong, ACE enters up to two reflection rounds: the Reflector—GPT-5.2 (gpt-5.2-2025-12-11) with medium reasoning effort in our experiment—analyzes the generator's predicted answer, ground truth feedback, and any cited playbook bullets, then produces diagnostic feedback and tags cited bullets as helpful, harmful, or neutral. The generator then retries the same question using that reflection. If the initial answer is already correct, the reflector still runs to credit useful bullets. After every step, the Curator—GPT-5.2 (gpt-5.2-2025-12-11) with medium reasoning effort in our experiment—converts the latest reflection into playbook update operations, by adding new bullets in the playbook. We run training for only one epoch. At $n = 800$, training requires up to 2,000 reflector and curator calls ($\sim$\$50 per context). Due to this cost, we omit NQA and restrict the $n = 800$ context-level QA evaluation to FinText, alongside the FiNER positive control.

We examine whether the bullets generated from an instance's reflection actually improve performance on that same instance. For this, we examine the accuracy gain after the reflection step (Fig. 9a). The accuracy gain after the reflection step is evident both in FiNER and FinText. FiNER shows +0.26 for n=200 and +0.16 for n=800, while FinText shows on average +0.23 for n=200 and +0.15 for n=800. However in FinText, this gain in the training examples fails to generalize, as shown in the lack of

accuracy increase in the test examples (Fig. 9b; same result in Table 4). This is in contrast to FiNER where the gain from training generalizes well to the test examples.

More detailed analysis can be done using the reflector's labels on the bullets in the playbook. We analyze the *net helpful count* of the final playbook (the playbook used for evaluation on the test set), defined as the sum of all *helpful* counts minus the sum of all *harmful* counts, across all bullets. For FiNER, this value is 14 for n=200, increasing to 49 when n=800. In contrast, for FinText, it is on average -3 for n=200, and only 3.5 when n=800. These helpful/harmful labels are provided by a strong reflector (GPT-5.2), suggesting that the bullets are inherently unhelpful for the unseen examples, rather than that the generator fails to exploit them. This suggests that although newly added bullets may help on the examples that produced them, they generalize poorly to other examples.

Overall, our analysis on ACE in FiNER and FinText suggests that ACE fails to produce generalizable textual artifacts in context-level adaptation, justifying our choice of gradient-based update.

Table 9: Unmodified examples of bullets in Table 5, produced in ACE. The second row shows the raw bullets. The third row shows the helpful and harmful count labeled by the reflector (GPT-5.2). (a) and (b) are respectively the **most helpful** bullet from FiNER(n=800) and FinText(n=800) context #1. The training question associated with (c) is: "Are the collateral thresholds influenced by a segregated deposit?"

Helpful bullet in FiNER	**Helpful bullet in FinText**	**Unhelpful bullet in FinText**
Revolver capacity/commitment clues: "up to $X million", "provides for borrowings of up to", "aggregate commitments", "total revolving credit facility of $X", "facility size" → map to maximum borrowing capacity/commitment tags (not debt carrying amount/outstanding).	*If the excerpt lists "red flag indicators," "covered accounts," identity-theft patterns, and includes a bracketed source like "[Reg. V, 72 FR 63758 . . .]", treat that as a strong signal the governing framework is the FACTA/Red Flags Rule (Reg V / identity theft program), even if the surrounding document also contains "Part 223 / Regulation W" headings.*	*Indicator (Reg W collateral): if the prompt mentions a "segregated, earmarked deposit account" used to secure affiliate credit transactions, the expected collateralization threshold is 100% (same bucket as Treasuries/certain Fed-eligible paper) and the account does not modify the 110%/120%/130% thresholds for other collateral types.*
helpful=6 harmful=0	helpful=5 harmful=0	helpful=0 harmful=0

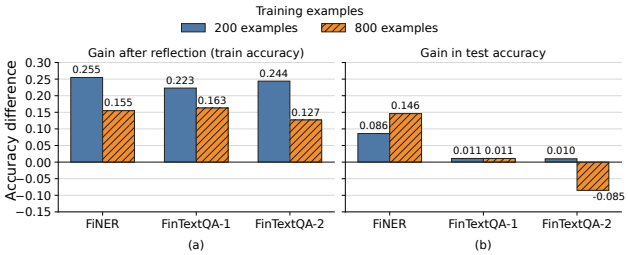


Figure 9: Comparison of training accuracy and test accuracy in ACE. ACE generalizes well in FiNER, but fails in FinText.

Table 10: DIAKV results with varying numbers of seed QAs. Our analysis focuses solely on the prime metric, indicated by **bold**.

NarrativeQA

		BLEU-1		*BLEU-4*		*METEOR*		***ROUGE-L***	
Model	Number of seeds	macro	micro	macro	micro	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.127	0.129	0.033	0.034	0.240	0.244	0.167	0.169
	20	**0.287**	**0.293**	**0.074**	**0.076**	**0.272**	**0.278**	**0.301**	**0.305**
	5	0.244	0.248	0.063	0.064	0.249	0.250	0.257	0.262
	0	0.242	0.246	0.060	0.061	0.250	0.252	0.263	0.267
Llama-3.2-3B	baseline	0.244	0.245	0.060	0.061	0.260	0.264	0.265	0.266
	20	**0.312**	**0.314**	**0.078**	**0.080**	**0.277**	**0.279**	**0.317**	**0.320**
	5	0.260	0.263	0.065	0.066	0.256	0.258	0.266	0.270
	0	0.261	0.264	0.065	0.067	0.247	0.249	0.272	0.274
Ministral-3-3B	baseline	0.184	0.185	0.043	0.043	0.255	0.259	0.206	0.208
	20	**0.315**	**0.319**	**0.084**	**0.087**	**0.287**	**0.292**	**0.314**	**0.318**
	5	0.290	0.293	0.072	0.074	0.273	0.279	0.284	0.289
	0	0.268	0.273	0.068	0.070	0.267	0.273	0.281	0.285

QASA-M

		ROUGE-1		*ROUGE-2*		*ROUGE-L*		***LLM-as-judge***	
Model	Number of seeds	macro	micro	macro	micro	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.270	0.266	**0.097**	0.096	0.239	0.235	**0.474**	0.464
	20	**0.276**	**0.276**	0.095	**0.097**	**0.243**	**0.243**	0.460	0.467
	5	0.272	0.271	0.094	0.095	0.238	0.237	0.443	0.452
	0	0.266	0.262	0.086	0.086	0.230	0.228	0.461	**0.470**
Llama-3.2-3B	baseline	**0.271**	**0.271**	**0.104**	**0.105**	**0.240**	**0.241**	0.345	0.348
	20	0.267	0.266	0.089	0.091	0.235	0.236	**0.404**	**0.407**
	5	0.258	0.257	0.086	0.087	0.229	0.227	0.389	0.379
	0	0.245	0.242	0.078	0.078	0.212	0.210	0.371	0.369
Ministral-3-3B	baseline	0.216	0.212	0.057	0.057	0.188	0.185	0.400	0.397
	20	**0.265**	**0.260**	**0.096**	**0.095**	**0.234**	**0.230**	0.467	0.458
	5	0.261	0.256	0.089	0.088	0.230	0.225	**0.470**	**0.468**
	0	0.252	0.250	0.078	0.080	0.220	0.218	0.458	0.447

LongHealth-M

		Accuracy		*Word-level F1*	
Model	Number of seeds	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.717	0.717	0.806	0.806
	20	**0.728**	**0.728**	**0.822**	**0.822**
	5	0.700	0.700	0.795	0.795
	0	0.694	0.694	0.800	0.800
Llama-3.2-3B	baseline	0.464	0.464	0.591	0.591
	20	**0.497**	**0.497**	**0.644**	**0.644**
	5	0.461	0.461	0.607	0.607
	0	0.453	0.453	0.611	0.611
Ministral-3-3B	baseline	0.669	0.669	0.792	0.792
	20	0.692	0.692	0.813	0.813
	5	0.694	0.694	0.811	0.811
	0	**0.708**	**0.708**	**0.830**	**0.830**

FinText

		ROUGE-1		*ROUGE-2*		*ROUGE-L*		*BLEU-4*		***LLM-as-judge***	
Model	Number of seeds	macro	micro	macro	micro	macro	micro	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.297	0.287	0.111	0.121	0.265	0.255	0.044	**0.047**	0.325	0.371
	20	**0.307**	**0.299**	**0.126**	**0.132**	**0.272**	**0.265**	**0.050**	**0.047**	0.376	**0.432**
	5	0.298	0.278	0.107	0.108	0.264	0.245	0.037	0.035	0.347	0.347
	0	0.291	0.275	0.105	0.105	0.261	0.246	0.038	0.036	**0.404**	0.421
Llama-3.2-3B	baseline	**0.297**	**0.299**	**0.121**	**0.131**	**0.263**	**0.268**	**0.045**	**0.050**	0.321	0.381
	20	0.288	0.283	0.102	0.108	0.259	0.252	0.041	0.040	**0.377**	**0.392**
	5	**0.297**	0.290	0.112	0.121	0.259	0.256	0.044	0.045	0.319	0.320
	0	0.280	0.274	0.099	0.107	0.250	0.241	0.037	0.036	0.231	0.267
Ministral-3-3B	baseline	0.265	0.263	0.099	0.108	0.237	0.232	0.045	0.046	0.327	0.384
	20	0.297	0.288	0.118	0.122	0.264	0.253	0.041	0.042	0.341	**0.405**
	5	**0.308**	**0.298**	**0.128**	**0.131**	**0.277**	**0.264**	**0.053**	**0.048**	**0.350**	0.397
	0	0.302	0.293	0.117	0.120	0.268	0.257	0.045	0.043	0.287	0.376

Table 11: DIAKV results across parametrizations. Our analysis focuses solely on the prime metric, indicated by **bold**.

NarrativeQA

		BLEU-1		*BLEU-4*		*METEOR*		***ROUGE-L***	
Model	Parametrization	macro	micro	macro	micro	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.127	0.129	0.033	0.034	0.240	0.244	0.167	0.169
	KV	**0.287**	**0.293**	**0.074**	**0.076**	0.272	0.278	**0.301**	**0.305**
	full	0.243	0.250	0.067	0.069	**0.274**	**0.280**	0.280	0.287
	LoRA (r=32)	0.226	0.228	0.063	0.064	0.239	0.242	0.254	0.257
	LoRA (r=8)	0.178	0.181	0.048	0.049	0.214	0.216	0.230	0.234
Llama-3.2-3B	baseline	0.244	0.245	0.060	0.061	0.260	0.264	0.265	0.266
	KV	**0.312**	**0.314**	**0.078**	**0.080**	0.277	0.279	**0.317**	**0.320**
	full	0.254	0.254	0.070	0.070	0.259	0.260	0.270	0.271
	LoRA (r=32)	0.285	0.286	0.074	0.075	**0.280**	**0.282**	0.303	0.305
	LoRA (r=8)	0.233	0.234	0.062	0.063	0.238	0.241	0.263	0.265
Ministral-3-3B	baseline	0.184	0.185	0.043	0.043	0.255	0.259	0.206	0.208
	KV	**0.315**	**0.319**	**0.084**	**0.087**	**0.287**	**0.292**	0.314	0.318
	full	0.300	0.302	0.080	0.082	**0.287**	0.291	**0.320**	**0.326**
	LoRA (r=32)	0.190	0.188	0.054	0.054	0.209	0.210	0.312	0.314
	LoRA (r=8)	0.156	0.155	0.046	0.046	0.180	0.181	0.259	0.258

QASA-M

		ROUGE-1		*ROUGE-2*		*ROUGE-L*		***LLM-as-judge***	
Model	Parametrization	macro	micro	macro	micro	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.270	0.266	**0.097**	0.096	0.239	0.235	0.474	0.464
	KV	**0.276**	**0.276**	0.095	**0.097**	**0.243**	**0.243**	0.460	0.467
	full	0.252	0.251	0.076	0.078	0.221	0.220	0.471	0.470
	LoRA (r=32)	0.270	0.266	0.090	0.088	0.238	0.233	**0.510**	**0.508**
	LoRA (r=8)	0.265	0.260	0.087	0.086	0.234	0.230	0.492	0.482
Llama-3.2-3B	baseline	**0.271**	**0.271**	**0.104**	**0.105**	**0.240**	**0.241**	0.345	0.348
	KV	0.267	0.266	0.089	0.091	0.235	0.236	**0.404**	**0.407**
	full	0.218	0.214	0.049	0.050	0.191	0.188	0.297	0.300
	LoRA (r=32)	0.251	0.249	0.072	0.072	0.222	0.220	0.388	0.378
	LoRA (r=8)	0.244	0.241	0.071	0.070	0.216	0.212	0.364	0.352
Ministral-3-3B	baseline	0.216	0.212	0.057	0.057	0.188	0.185	0.400	0.397
	KV	**0.265**	0.260	**0.096**	**0.095**	**0.234**	0.230	0.467	0.458
	full	0.257	0.255	0.083	0.083	0.223	0.222	**0.499**	**0.490**
	LoRA (r=32)	0.260	0.258	0.086	0.086	0.228	0.228	0.478	0.466
	LoRA (r=8)	0.264	**0.263**	0.085	0.088	0.233	**0.232**	0.481	0.483

LongHealth-M

		Accuracy		*Word-level F1*	
Model	Parametrization	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.717	0.717	0.806	0.806
	KV	**0.728**	**0.728**	**0.822**	**0.822**
	full	0.336	0.336	0.484	0.484
	LoRA (r=32)	0.522	0.522	0.697	0.697
	LoRA (r=8)	0.225	0.225	0.410	0.410
Llama-3.2-3B	baseline	0.464	0.464	0.591	0.591
	KV	**0.497**	**0.497**	**0.644**	**0.644**
	full	0.278	0.278	0.509	0.509
	LoRA (r=32)	0.289	0.289	0.548	0.548
	LoRA (r=8)	0.331	0.331	0.560	0.560
Ministral-3-3B	baseline	0.669	0.669	0.792	0.792
	KV	**0.692**	**0.692**	**0.813**	**0.813**
	full	0.564	0.564	0.751	0.751
	LoRA (r=32)	0.458	0.458	0.634	0.634
	LoRA (r=8)	0.617	0.617	0.747	0.747

FinText

		ROUGE-1		*ROUGE-2*		*ROUGE-L*		*BLEU-4*		***LLM-as-judge***	
Model	Parametrization	macro	micro	macro	micro	macro	micro	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.297	0.287	0.111	0.121	0.265	0.255	0.044	**0.047**	0.325	0.371
	KV	**0.307**	**0.299**	**0.126**	**0.132**	**0.272**	**0.265**	**0.050**	**0.047**	0.376	**0.432**
	full	0.291	0.279	0.095	0.097	0.253	0.241	0.033	0.030	0.339	0.392
	LoRA (r=32)	0.305	0.297	0.104	0.110	0.268	0.259	0.037	0.038	**0.386**	0.384
	LoRA (r=8)	0.305	0.293	0.113	0.114	0.270	0.259	0.042	0.037	0.364	0.419
Llama-3.2-3B	baseline	0.297	**0.299**	**0.121**	**0.131**	0.263	**0.268**	**0.045**	**0.050**	0.321	0.381
	KV	0.288	0.283	0.102	0.108	0.259	0.252	0.041	0.040	**0.377**	**0.392**
	full	0.277	0.269	0.091	0.095	0.241	0.237	0.029	0.027	0.273	0.288
	LoRA (r=32)	0.297	0.286	0.101	0.102	**0.265**	0.253	0.039	0.034	0.273	0.277
	LoRA (r=8)	**0.299**	0.288	0.108	0.109	**0.265**	0.250	0.040	0.037	0.304	0.304
Ministral-3-3B	baseline	0.265	0.263	0.099	0.108	0.237	0.232	0.045	**0.046**	0.327	0.384
	KV	0.297	0.288	**0.118**	**0.122**	0.264	0.253	0.041	0.042	0.341	0.405
	full	0.303	0.298	0.107	0.115	0.263	0.255	0.046	0.045	**0.407**	**0.427**
	LoRA (r=32)	**0.310**	**0.303**	0.111	0.117	0.265	0.257	**0.047**	0.045	0.383	0.416
	LoRA (r=8)	0.307	**0.303**	0.113	**0.122**	**0.272**	**0.267**	0.038	0.042	0.323	0.400

Table 12: Comparison with the self-supervised span loss of qTTT (Bansal et al., 2026): qTTT-KV trains the KV cache and qTTT-WQ the query projections W_Q, both at the FLOPs of DIAKV. Our analysis focuses solely on the prime metric, indicated by **bold**.

NarrativeQA

		BLEU-1		*BLEU-4*		*METEOR*		***ROUGE-L***	
Model	Parametrization	macro	micro	macro	micro	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.127	0.129	0.033	0.034	0.240	0.244	0.167	0.169
	DIAKV	**0.287**	**0.293**	**0.074**	**0.076**	**0.272**	**0.278**	**0.301**	**0.305**
	qTTT-WQ	0.129	0.133	0.034	0.035	0.233	0.237	0.169	0.173
	qTTT-KV	0.131	0.135	0.033	0.035	0.224	0.228	0.171	0.175
Llama-3.2-3B	baseline	0.244	0.245	0.060	0.061	0.260	0.264	0.265	0.266
	DIAKV	**0.312**	**0.314**	**0.078**	**0.080**	**0.277**	**0.279**	**0.317**	**0.320**
	qTTT-WQ	0.236	0.237	0.058	0.060	0.247	0.248	0.261	0.261
	qTTT-KV	0.231	0.231	0.062	0.062	0.247	0.248	0.246	0.245
Ministral-3-3B	baseline	0.184	0.185	0.043	0.043	0.255	0.259	0.206	0.208
	DIAKV	**0.315**	**0.319**	**0.084**	**0.087**	**0.287**	**0.292**	**0.314**	**0.318**
	qTTT-WQ	0.145	0.145	0.035	0.035	0.228	0.229	0.169	0.168
	qTTT-KV	0.173	0.175	0.041	0.041	0.254	0.258	0.200	0.202

QASA-M

		ROUGE-1		*ROUGE-2*		*ROUGE-L*		***LLM-as-judge***	
Model	Parametrization	macro	micro	macro	micro	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.270	0.266	0.097	0.096	0.239	0.235	0.474	0.464
	DIAKV	0.276	0.276	0.095	0.097	0.243	0.243	0.460	0.467
	qTTT-WQ	0.280	0.278	0.106	0.107	0.248	0.247	**0.492**	**0.482**
	qTTT-KV	**0.288**	**0.286**	**0.112**	**0.113**	**0.254**	**0.253**	0.482	0.475
Llama-3.2-3B	baseline	**0.271**	**0.271**	0.104	0.105	**0.240**	**0.241**	0.345	0.348
	DIAKV	0.267	0.266	0.089	0.091	0.235	0.236	**0.404**	**0.407**
	qTTT-WQ	0.263	0.262	**0.105**	0.105	0.236	0.236	0.343	0.344
	qTTT-KV	0.263	0.264	**0.105**	**0.106**	0.236	0.238	0.332	0.342
Ministral-3-3B	baseline	0.216	0.212	0.057	0.057	0.188	0.185	0.400	0.397
	DIAKV	**0.265**	**0.260**	**0.096**	**0.095**	**0.234**	**0.230**	**0.467**	**0.458**
	qTTT-WQ	0.224	0.222	0.073	0.074	0.200	0.199	0.422	0.426
	qTTT-KV	0.220	0.218	0.064	0.065	0.195	0.194	0.402	0.403

LongHealth-M

		Accuracy		*Word-level F1*	
Model	Parametrization	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.717	0.717	0.806	0.806
	DIAKV	0.728	0.728	**0.822**	**0.822**
	qTTT-WQ	**0.731**	**0.731**	0.815	0.815
	qTTT-KV	**0.731**	**0.731**	0.820	0.820
Llama-3.2-3B	baseline	0.464	0.464	0.591	0.591
	DIAKV	**0.497**	**0.497**	**0.644**	**0.644**
	qTTT-WQ	0.408	0.408	0.580	0.580
	qTTT-KV	0.281	0.281	0.506	0.506
Ministral-3-3B	baseline	0.669	0.669	0.792	0.792
	DIAKV	**0.692**	**0.692**	**0.813**	**0.813**
	qTTT-WQ	0.631	0.631	0.771	0.771
	qTTT-KV	0.681	0.681	0.788	0.788

FinText

		ROUGE-1		*ROUGE-2*		*ROUGE-L*		*BLEU-4*		***LLM-as-judge***	
Model	Parametrization	macro	micro	macro	micro	macro	micro	macro	micro	macro	micro
Qwen3-4B-2507	baseline	0.297	0.287	0.111	0.121	0.265	0.255	0.044	0.047	0.325	0.371
	DIAKV	**0.307**	**0.299**	**0.126**	**0.132**	**0.272**	**0.265**	**0.050**	0.047	0.376	0.432
	qTTT-WQ	0.298	0.291	0.121	0.128	0.267	0.259	0.049	**0.050**	**0.443**	**0.453**
	qTTT-KV	0.294	0.287	0.118	0.128	0.264	0.257	0.042	0.045	0.370	0.376
Llama-3.2-3B	baseline	**0.297**	**0.299**	**0.121**	**0.131**	**0.263**	**0.268**	**0.045**	**0.050**	0.321	0.381
	DIAKV	0.288	0.283	0.102	0.108	0.259	0.252	0.041	0.040	**0.377**	**0.392**
	qTTT-WQ	0.289	0.284	0.107	0.112	0.261	0.255	0.036	0.035	0.288	0.384
	qTTT-KV	0.268	0.266	0.095	0.103	0.241	0.238	0.029	0.031	0.284	0.347
Ministral-3-3B	baseline	0.265	0.263	0.099	0.108	0.237	0.232	**0.045**	0.046	0.327	0.384
	DIAKV	**0.297**	**0.288**	**0.118**	**0.122**	**0.264**	**0.253**	0.041	0.042	**0.341**	**0.405**
	qTTT-WQ	0.264	0.266	0.104	0.114	0.239	0.239	**0.045**	**0.052**	0.306	**0.405**
	qTTT-KV	0.269	0.274	0.103	0.114	0.242	0.244	**0.045**	0.050	0.306	**0.405**

E EXTENDED DISCUSSION FOR DIAKV : BAYESIAN PERSPECTIVE

Below, we present an extended discussion for DIAKV from a Bayesian perspective. The core ideas are well-presented in §3.1, and this section mainly aims for more formalism.

E.1 EXTENDED DISCUSSION

Here we adopt the latent formulation of QA (Chen et al., 2017; Lee et al., 2019). Let $V^* = \bigcup_{n\geq 0} V^n$ be the countably infinite set of all possible strings over tokens in vocabulary V, including the empty string ϵ. Let $(Q, A) \sim \pi_{Q,A}$ on $V^* \times V^*$. The QA objective is to learn $p(a|q)$ that approximates the true conditional $\pi_{A|Q}(a|q)$ induced by $\pi_{Q,A}$. The model is often expressed in a marginalized form over *context* $c \in V^*$.

$$\pi_{A|Q}(a|q) \approx p(a|q) \coloneqq \sum_{c\in V^*} p_\theta(a|q,c)p_\phi(c|q) \coloneqq \sum_{c\in V^*} p_\theta(a|q,c)p_\theta(c|q) \tag{E.1}$$

$p_\theta(a|q,c)$ is referred to as the *reader*, whereas $p_\phi(c|q)$ is referred as the *retriever*. In this discussion, we treat the reader and retriever as the same language model parameterized by θ. The language model p_θ naturally models two conditionals: $p_\theta(q|c)$, which models the probability of question q being generated given the context c, and $p_\theta(a|q,c)$, which models the probability of answer a being generated given the context c and the question q. Given a context prior $p(c)$, the *context posterior* $p_\theta(c|q)$ is derived as below.

$$p_\theta(c|q) = \frac{p_\theta(q|c)p(c)}{\sum_{c\in V^*} p_\theta(q|c)p(c)} \propto p_\theta(q|c)p(c) \tag{E.2}$$

After decomposing the posterior, p_θ models the conditional $\pi_{A|Q}$ as below, given the marginal π_Q.

$$\pi_{A|Q}(a|q) \approx p(a|q) \propto \sum_{c\in V^*} p(c)p_\theta(q|c)p_\theta(a|q,c), \quad q \sim \pi_Q \tag{E.3}$$

Therefore, a latent formulation of QA in theory requires summation over an infinite number of contexts. As this is intractable in practice, QA models often approximate by taking a single best context $\hat{c}$ [12]. For example, $\hat{c}$ in top-k retrieval corresponds to a concatenation of top-k retrieved chunks, whereas $\hat{c}$ in closed-book QA corresponds to an empty string. A naturally arising question then is finding the context c that best approximates $p(a|q)$. QA datasets often accompany *gold* contexts, which are human-labeled to be the best context to answer the question, among the corpus or knowledge base (Joshi et al., 2017; Kwiatkowski et al., 2019). However, such gold contexts can easily be outperformed by other contexts (Yu et al., 2023; Lee et al., 2023a), implying a ceiling above a heuristic choice. This motivates introducing a theoretical optimal context. We define a population-level *diamond* context for true $\pi_{Q,A}$ as below.

$$c^*_{dia} \coloneqq \arg\max_{c\in V^*} \mathbb{E}_{(q,a)\sim\pi_{Q,A}}[\log p(c) + \log p_\theta(q|c) + \log p_\theta(a|q,c)] \tag{E.4}$$

Accordingly, we define an empirical *diamond* context, given a set of samples $\mathcal{D}_{qa} = \{(q_i, a_i)\}_i$ drawn i.i.d. from $\pi_{Q,A}$.

$$\hat{c}_{dia}|\mathcal{D}_{qa} \coloneqq \arg\max_{c\in V^*}[\log p(c) + \sum_i (\log p_\theta(q_i|c) + \log p_\theta(a_i|q_i,c))] \tag{E.5}$$

This discrete optimization problem on V^* is difficult to solve mathematically. Now we introduce an additional constraint — language model p_θ is a Transformer-based autoregressive language model (Radford et al., 2019). This constraint allows us to substitute discrete context c with a continuous KV cache $h \in \mathbb{R}^m$, without loss of any discussion so far. We additionally impose the token length of KV cache to be constant, and arrive at the below objective, constituting three components: KV cache prior, question likelihood, and answer likelihood. We define such $\hat{h}_{dia}$ to be the empirical *diamond* KV, or simply DIAKV .

$$\hat{h}_{dia}|\mathcal{D}_{qa} = \arg\max_{h\in\mathbb{R}^m}[\log p(h) + \sum_i (\log p_\theta(q_i|h) + \log p_\theta(a_i|q_i,h))] \tag{E.6}$$

[12] Bootstrapping over multiple c is possible, but that is largely orthogonal to our discussion.

E.2 KV CACHE PRIOR

The KV cache prior is intended to play a role of constraining the optimized h_{dia} to resemble the actual prefilled context, not to drift in the much broader space of $\mathbb{R}^m$. For this purpose, we choose the prior based on the empirical distribution of prefilled context. As our distinction mainly lies on random tensor *vs.* KV cache from the tokens, we obtain prefilled KV cache from random sequence of tokens, excluding special tokens.

On Qwen3-4B-2507 (Team, 2025), we run two experiments: (1) sequence length of 4096 with 200 samples, and (2) sequence length of 1024 with 1000 samples. We consider KV cache $h \in \mathbb{R}^m$ to be flattened across every dimension including layers and attention heads, resulting in $m = 301,989,888$ in (1) and $m = 75,497,472$ in (2). We find that the coordinate-wise means, i.e., $\mathbb{E}[h_i]$, are mostly near-zero in both of the cases. The mean of coordinate-wise means, i.e., $\mathbb{E}_i[\mathbb{E}[h_i]]$, is -0.0000012 for (1) and 0.0010 for (2). Moreover, the 75% quartile of $|\mathbb{E}[h_i]|$ is 0.961 for (1) and 0.859 for (2), indicating that most coordinates have a near-zero mean. This empirical distribution motivates our choice of zero-mean Gaussian prior. Our further assumption on shared variance is less based on the empirical distribution, and rather serves as a simple regularizing baseline with efficient implementation. We believe future works can improve on the choice of KV cache prior, potentially with further analysis on the empirical distribution of prefilled KV cache.