DKL: Decoupled Knowledge Learning for Instruction-Tuned Language Models

Anonymous ACL submission

Abstract

RAG has become the de facto method for incorporating new, corpus-specific knowledge into an instruction following LLM (Instruct LLM). Although RAG-based prompting improves factual grounding, it fails when retrieval is incorrect or incomplete, leading to hallucinations. Finetuning methods such as RAFT (Zhang et al., 2024b) and PA-RAG (Bhushan et al., 2025) enhance RAG by injecting new knowledge into the model's parameters, but require generating a massive amount of synthetic QA that covers the entire corpus. Extended Pre-Training (EPT) on the text corpus avoids the need for comprehensive synthetic data generation but compromises an Instruct LLM's instruction-following capabilities, necessitating instruction fine-tuning (IFT) after pre-training. However, IFT is costly and may be infeasible due to the unavailability of an instruction-tuning corpus. In this work, we propose DKL-Decoupled Knowledge Learning for Instruction-Tuned Language Models. Instead of doing EPT on the Instruct LLM, DKL performs EPT on its corresponding base LLM to infuse new knowledge. These knowledge-infused weights are then merged with the Instruct LLM, imparting new knowledge without affecting their instruction-following capabilities. DKL is a lightweight method that avoids expensive instruction fine-tuning and relies on model merging to infuse the new knowledge into the Instruct LLM without destroying its instruction following capabilities. Empirical results show that DKL improves RAG accuracy from 54.17% to 79.26% on retrieval failure cases, while outperforming prior approaches with substantially less training data.

1 Introduction

'*Base LLMs*' trained on enormous amounts of textual data possess immense knowledge but lack instruction-following capabilities. This necessitates post-training, which typically involves massive Instruction Fine-Tuning (IFT) (Ouyang et al., 2022; Shengyu et al., 2023) followed by RLHF (Schulman et al., 2017; Rafailov et al., 2023; Pandey et al., 2024). '*Instruct LLMs*' (model obtained after post–training) have achieved remarkable success across general-purpose tasks (Brown et al., 2020; Wei et al., 2022). However, in specialized applications such as question answering over technical or confidential policy documents, success depends less on general reasoning and more on producing highly accurate, document-grounded responses. Often, these specialized documents are either too scarce or proprietary and, therefore, unavailable during the pre-training stage. Hence, even the state-of-the-art LLMs struggle to answer queries that require access to these documents.

A common solution is Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Karpukhin et al., 2020), which conditions LLM's responses on relevant passages retrieved from the target documents. While effective, RAG is highly sensitive to retrieval quality, and retriever failures often lead to hallucinations or incomplete answers ((Ji et al., 2023; Nandwani et al., 2023)). Injecting the new knowledge from the specialized documents into the parameters of the model can potentially alleviate the issues caused by retriever failures, as the model can fall back on its parametric knowledge.

Extended pre-training (EPT) via unsupervised next-token prediction on new documents (Ke et al., 2023) is an effective way to ingest new knowledge. However, doing so on Instruct LLMs results in catastrophic forgetting of the skills acquired during IFT (Ke et al., 2025). As a result, most of the prior works (Ma et al., 2023; Yang et al., 2024; Lu et al., 2025) apply extended pre-training on the 'base LLM' but have to redo IFT to re-acquire the skills present in the instruct LLM. This may not always be feasible due to the lack of the IFT dataset used to create the instruct LLM.

Another way of knowledge ingestion involves direct finetuning of the instruct LLM using IFT-

style training data, such as question-answers (QAs) from the new documents. However, QAs from the new documents are often not readily available and hence works such as Zhang et al. (2024b); Bhushan et al. (2025) resort to QAs generated synthetically by prompting a stronger LLM. A major advantage of such techniques is that they don't require expensive IFT after knowledge ingestion. However, there are two main issues: (1) we need to generate a massive amount of synthetic data, which may become prohibitively expensive (Yang et al., 2025), and (2) unlike EPT, it is difficult to guarantee coverage of the entire knowledge via QAs.

In this work, we ask whether one can obtain the knowledge-ingestion benefits of extended pre-training without sacrificing the instruction-following ability of an existing instruct LLM. Our key intuition is that these two capabilities arise from different stages of training: new corpus knowledge is most effectively acquired during pre-training, while instruction-following behavior is introduced later through instruction fine-tuning (IFT). Repeating IFT after each round of knowledge ingestion, however, is often prohibitively expensive or impossible when the instruction-tuning data is unavailable. We therefore draw on task arithmetic (Ilharco et al., 2023), which treats the effect of fine-tuning as vector addition in parameter space.

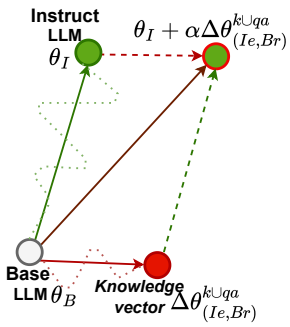


Figure 1: Exploiting task-arithmetic to combine new knowledge adapter with instruction following capabilities of Instruct LLM.

In particular, we compute an '*instruction following vector*' of the instruct LLM by subtracting the publicly available instruct and base LLM's weights. This vector captures the transformation induced by instruction-tuning. To obtain a '*knowledge vector*' that captures the new knowledge, we use the new documents to perform extended pretraining via unsupervised next token prediction on top of the base LLM. Adding the '*instruction following vector*' and the '*knowledge vector*' to the base LLM results in a model that has the new knowledge from the documents, as well as the instruction following skills of the Instruct LLM. (see fig. 1). We call our method DKL: Decoupled Knowledge Learning for Instruction-Tuned Language Models. Optionally, to improve the model's ability to recall the newly acquired knowledge at query time, we augment this training with a small amount of synthetic QA supervision (Allen-Zhu and Li, 2024).

Since the knowledge vector is trained on the base LLM but applied to the instruct LLM, its effectiveness can be limited by representation mismatch between the two models. This mismatch may arise from differences in token embeddings, including additional chat-formatting tokens in instruct LLMs such as '[INST]' for Mistral (Jiang et al., 2023; Mistral AI, 2024). To improve transferability, we train the knowledge vector using the instruct model's token embeddings, which makes the learned update better aligned with the target instruct model. Our ablations show that this consistently improves performance.

To the best of our knowledge, DKL is the first work that proposes a lightweight and efficient way of imparting new knowledge to an existing Instruct LLM. To summarize, our contributions are:

1. We propose DKL (Decoupled Knowledge Learning for Instruction-Tuned Language Models), an efficient and lightweight method for injecting knowledge from new documents without the costly IFT phase, usually required after EPT.
2. To make DKL work, we devise a novel method that uses token embeddings from the instruct LLM during extended pretraining on the base LLM. This significantly improves the adaptability of the knowledge vector.
3. We empirically demonstrate that our method outperforms state-of-the-art SFT based knowledge infusion methods (Zhang et al., 2024b; Bhushan et al., 2025) as well as a closely related baseline, Chat-Vector (Huang et al., 2024), while requiring substantially less synthetic data.

2 Related Work

Knowledge Ingestion: Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020) ingests external knowledge in the context at inference time to ground responses on retrieved passages. Recent progress has showcased its effectiveness across diverse domains (Asai et al., 2024; Qiu et al., 2023; Kim et al., 2024; Tang

et al.; Yan et al., 2024). However, RAG-based approaches remain vulnerable to retrieval failures, leading to hallucinations (Nandwani et al., 2023; Ji et al., 2023; Setty et al., 2024).

Another research direction has been static knowledge injection, which infuses domain knowledge into the model's parameters via fine-tuning or extended pretraining, enabling closed-book inference without access to external documents (Ke et al., 2023; Lu et al., 2025; Ovadia et al., 2025). Various works have established the utility of extended pretraining across multiple fields, such as medical (Wu et al., 2024; Christophe et al., 2024), materials (Zhang et al., 2024a), and finance (Wu et al., 2023; Xie et al., 2023). However, EPT on new text causes instruction-tuned models to regress on general instruction-following abilities, necessitating expensive re-instruction tuning.

Recent works like RAFT(Zhang et al., 2024b) and PA-RAG(Bhushan et al., 2025) bridge static and dynamic paradigms by finetuning instruct LLMs to absorb document knowledge while using retrieved passages during inference. However, they still rely heavily on large synthetic QA corpora.

Model Merging via Task Vectors: Inspired by model merging (Wortsman et al., 2022), Ilharco et al. (2023) introduce task vectors (difference between the weights of two trained models) as an effective mechanism for imparting different skills to an LLM. Task vectors provide a general model-merging framework. Unlike DKL it does not prescribe training on the base model, and it does not address the vocabulary distribution mismatch between the base and Instruct LLMs. Various works have extended the basic idea of task vectors in different ways. For instance, Daheim et al. (2024) leverage task vectors to mitigate hallucinations, while Zhang et al. (2023) combine multiple PEFT modules (Hu et al., 2022; Liu et al., 2022) to achieve distribution generalization, multitask adaptation, detoxification, and domain transfer.

Most closely related to our work, Huang et al. (2024) propose Chat-Vector, which transfers conversational abilities from a chat-tuned Llama model to a Chinese-adapted Llama base model. Our approach differs from Chat-Vector in two important ways: (1) unlike Chat-Vector, DKL uses the embedding layer of the instruction-tuned model while training the knowledge vector, thereby addressing vocabulary distribution mismatch; (2) DKL combines the knowledge and chat vector using an optimized interpolation ratio. We use Chat-Vector as a baseline and show that both of these design choices are critical for effective knowledge injection.

3 Methodology

Let θ represent the parameters of an LLM that assigns a probability $\mathbf{Pr}(\mathbf{p};\theta)$ to a sequence of tokens $\mathbf{p} = (t_1 \cdots t_m)$. Let θ_B and θ_I be the weights of the corresponding base and instruct LLMs, respectively. Further, let $\mathcal{D}_k = \{\mathbf{p}^i\}_{i=1}^N$ represent the new knowledge that we wish to ingest on top of the instruct LLM.

We propose to ingest knowledge from $\mathcal{D}_k$ into a knowledge adapter, $\Delta\theta^k$. A naïve way to train such an adapter would be to start with the most optimal weights θ_I and minimize the negative log likelihood over $\mathcal{D}_k$:

$$\Delta\theta_I^k = \arg\min_{\Delta\theta} \sum_{\mathbf{p}\in\mathcal{D}_k} -\log \mathbf{Pr}(\mathbf{p};(\theta_I + \Delta\theta)) \tag{1}$$

However, Ke et al., 2025 observe that such an adapter results in deterioration of instruction following abilities of the instruct LLM θ_I. This can be attributed to the fact that the instruct LLM is obtained via supervised finetuning of the base LLM, whereas the training objective in eq. (1) is unsupervised next token prediction. Next, we observe that this objective is the same as the training objective of the base LLMs. Therefore, θ_B could be more amenable to extended pretraining and thus may provide an ideal starting point for ingesting new knowledge. Motivated by this observation, we propose to train the knowledge adapter on top of the base LLM -

$$\Delta\theta_B^k = \arg\min_{\Delta\theta} \sum_{\mathbf{p}\in\mathcal{D}_k} -\log \mathbf{Pr}(\mathbf{p};(\theta_B + \Delta\theta)) \tag{2}$$

In our notation, the superscript captures the training data and the subscript captures the starting point, *i.e.*, model initialisation. Note that the final knowledge-infused parameters returned by eq. (2) are $\theta_B + \Delta\theta_B^k$. However, such a model lacks the instruction following ability of the instruct LLM. But notice that the new knowledge from $\mathcal{D}_k$ is mainly captured in $\Delta\theta_B^k$, which may be combined with θ_I that already has instruction following abilities. Therefore, instead of using $\theta_B + \Delta\theta_B^k$ as our final parameters, we propose to use $\theta_I + \alpha\Delta\theta_B^k$, where $\alpha \in (0, 1]$ is a hyperparameter -

$$\theta^* = \theta_I + \alpha\Delta\theta_B^k \tag{3}$$

Task-arithmetic inspired interpretation: Ilharco et al., 2023 show that if θ_1 and θ_2 are two different models finetuned from the same base model θ_B, then the corresponding task vectors, $\Delta\theta_B^1 = \theta_1 - \theta_B$ and $\Delta\theta_B^2 = \theta_2 - \theta_B$, capture the skills infused in them. If we combine the two task vectors, we get a model that possibly possesses both the skills -

$$\theta^c = \theta_B + \alpha\Delta\theta_B^1 + \gamma\Delta\theta_B^2 \tag{4}$$

Here, θ^c is the combined model capturing the skills of both the finetuned models. α and γ are hyperparameters.

Now, let $\mathcal{D}_s$ be the data used for the instruction tuning of the instruct LLM. Then the corresponding '*instruct task vector*' would be -

$$\begin{aligned}\theta_I - \theta_B &= \arg\min_{\Delta\theta} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_s} -\log \mathbf{Pr}(\mathbf{y} \mid \mathbf{x}; (\theta_B + \Delta\theta)) \\ &= \Delta\theta_B^s\end{aligned}$$

We can think of our knowledge adapter $\Delta\theta_B^k$ as '*knowledge task vector*' capturing all the knowledge from the corpus. Now, combining the '*instruct task vector*' with '*knowledge task vector*', we get -

$$\theta^* = \theta_B + \alpha\Delta\theta_B^k + \gamma\Delta\theta_B^s \tag{5}$$

Substituting $\gamma = 1$ and $\Delta\theta_B^s = \theta_I - \theta_B$ from section 3, we get -

$$\theta^* = \theta_B + \alpha\Delta\theta_B^k + \theta_I - \theta_B = \theta_I + \alpha\Delta\theta_B^k \tag{6}$$

which is exactly same as eq. (3).

Adding a small amount of synthetic QA: Allen-Zhu and Li, 2024 observe that having a few question-answer pairs in the pre-training data significantly enhances the recall of the ingested knowledge. These QAs do not necessarily have to span the entire corpus. Accordingly, we enhance our training corpus with a small amount of synthetically generated QAs. Unlike instruction finetuning, where loss is backpropagated only over the answer tokens conditioned on the question, we concatenate the question, answer and treat it as part of the new knowledge to be ingested. Accordingly, let $\mathcal{D}_{qa} = \{\mathbf{x}^i = (\mathbf{sys}, \mathbf{q}^i, \mathbf{a}^i)\}_{i=1}^{nq}$ be the training data obtained from the synthetic QAs. Here **sys** is a common system prompt that asks the model to answer the question; $(\mathbf{sys}, \mathbf{q}^i, \mathbf{a}^i)$ represents the concatenation of system prompt, question and answer; and nq is the number of synthetically generated QAs. We train our knowledge adapter on top of the base LLM using $\mathcal{D}_{k\cup qa} = \mathcal{D}_k \bigcup \mathcal{D}_{qa}$.

$$\Delta\theta_B^{k\cup qa} = \arg\min_{\Delta\theta} \sum_{\mathbf{x}\in\mathcal{D}_k \bigcup \mathcal{D}_{qa}} -\log \mathbf{Pr}(\mathbf{x}; (\theta_B + \Delta\theta)) \tag{7}$$

Using token embeddings of the instruct LLMs: We observe that for certain tokens, embeddings in the base and instruct LLMs are quite different. Often, they correspond to the tokens introduced during the instruction fine-tuning phase, *e.g.*, '[INST]' in instruct versions of Mistral. The system prompt used in the synthetic QA dataset may also introduce special tokens not seen during training of the base LLM, *i.e.*, the parameters θ_B are oblivious to these tokens. This creates a mismatch: the knowledge adapter is trained with the base model's token embeddings but during inference it is used with the instruct LLM's entirely different embeddings. To mitigate this, we propose to use the token embeddings of the instruct LLM instead of the base LLM. Concretely, during knowledge adapter training, we replace the base LLM's token embeddings (and the `lm_head`, if separate) with those of the instruct LLM. We claim that this enhances the adaptability of the knowledge adapter, trained on the base LLM but used with an instruct LLM. Intuitively, it gives the adapter parameters early exposure to the inference-time environment and vocabulary, reducing the risk of distribution shift.

If we represent model parameters θ_B as $(\theta_{Be}, \theta_{Br})$ and θ_I as $(\theta_{Ie}, \theta_{Ir})$, where θ_{Be}, θ_{Ie} are the token embeddings and θ_{Br}, θ_{Ir} are the remaining parameters in the base and instruct LLMs, respectively, then we learn our knowledge adapter on top of $(\theta_{Ie}, \theta_{Br})$ -

$$\Delta\theta_{(Ie,Br)}^{k\cup qa} = \arg\min_{\Delta\theta} \sum_{\mathbf{x}\in\mathcal{D}_k \bigcup \mathcal{D}_{qa}} \mathcal{L} \tag{8}$$

Where,

$$\mathcal{L} = -\log \mathbf{Pr}\left(\mathbf{x}; (\theta_{Ie}, \theta_{Br} + \Delta\theta)\right)$$

Algorithm 1 in appendix presents our method that returns the trained knowledge adapter. One can load it on top of the instruct LLM θ_I to obtain the final model parameters as -

$$\begin{aligned}\theta^* &= \theta_I + \alpha\left(\mathbf{0}, \Delta\theta_{(Ie,Br)}^{k\cup qa}\right) \\ &= (\theta_{Ie}, \theta_{Ir}) + \alpha\left(\mathbf{0}, \Delta\theta_{(Ie,Br)}^{k\cup qa}\right) \\ &= \left(\theta_{Ie}, \theta_{Ir} + \alpha\Delta\theta_{(Ie,Br)}^{k\cup qa}\right)\end{aligned} \tag{9}$$

	RedBook 1					QuALITY				
	QA	RAG			Train Time (in mins)	QA	RAG			Train Time (in mins)
		All	Ret. Success	Ret. Fail.			All	Ret. Success	Ret. Fail.	
Instruct	53.67 ±2.82	71.76 ±2.54	86.27 ±1.95	54.17 ±2.82		17.62 ±1.98	50.41 ± 2.60	82.31 ± 2.73	14.66 ± 2.68	
RAFT	56.87 ±2.80	79.87 ±2.27	88.20 ±1.82	68.89 ±2.62	19	11.65 ± 1.67	43.22 ± 2.57	67.95 ± 3.34	15.52 ± 2.74	120
PA-RAG	64.22 ±2.71	84.66 ±2.04	**92.70** ±1.47	74.07 ±2.48	43	10.84 ± 1.61	38.75 ± 2.53	58.97 ± 3.52	16.09 ± 2.78	330
Chat Vector	71.47 ±2.55	83.98 ±2.07	90.56 ±1.65	78.65 ±2.32	7	11.65 ±2.15	45.52 ±3.34	71.79 ±3.02	16.09 ±2.47	34
DKL	**73.80** ±2.49	**86.58** ±1.93	92.13 ±1.52	**79.26** ±2.29	**7**	**25.75** ± 2.27	**54.61** ± 2.59	**83.85** ± 2.63	**21.84** ± 3.13	**34**

Table 1: Comparing DKL with various baselines. The table reports the fraction of test samples where the LLM Judge rated the predicted response as good as the gold response. **QA**: performance in the QA setup; **All**: performance over the entire test set in RAG setup; **Ret. Success**: performance over test queries where retriever succeeds (match@5=1); **Ret. Fail.**: performance over test queries where retriever fails (match@5=0). These results are for Mistral-7B-Instruct-v0.3. Please see Table 10 for results on Llama-3.1-8B-Instruct. **Train Time**: Approx. training time in minutes. All scores are reported as mean +/- standard error.

4 Experimental Setup

Datasets: We show the efficacy of our method on three datasets: 2 technical RedBooks (Bhushan et al., 2025) and a non-technical dataset, QuALITY (Pang et al., 2022). The first two datasets consist of text from technical Redbooks[1] along with corresponding test question answers. The QuALITY dataset consists of a large number of long-form articles from the open domain. We randomly sample 10 articles to act as a knowledge base for our experiments. See appendix B for more details.

Models: We ingest the knowledge from all the datasets into *Mistral-7B-Instruct-v0.3* and *LLama-3.1-8B-Instruct*. In addition, to demonstrate the robustness of DKL to various backbone architectures and model sizes, we experiment with *SmolLM2-1.7B-Instruct*, and *Qwen3-0.6B* on one of the datasets. **Evaluation Metrics:** We evaluate our models in two setups – QA and RAG. In the QA setup, the model is prompted with only the question, and in the RAG setup, we provide the top 5 retrieved passages along with the question. We use Llama-3.3-70B-Instruct as a judge to evaluate the correctness of the predicted answer *w.r.t.* the given gold answer. For each test sample, we provide the judge with the question, gold answer, and generated answer, and the judge returns a binary score (0/1) after reasoning across multiple criteria. See appendix K for full prompts.

To ensure that our LLM judge is aligned with human judgement, we conduct a small-scale human study in which we evaluate the responses generated by the Instruct LLM. See appendix C for more details on the human study.

[1]Book 1: Do More with Less: Automating IBM Storage FlashSystem Tasks with REST APIs, Scripting, and Ansible. Book 2: Red Hat OpenShift Container Platform on IBM Z and LinuxONE.

Baselines: We compare DKL with RAFT (Zhang et al., 2024b), PA-RAG (Bhushan et al., 2025), and Chat-Vector (Huang et al., 2024). Both RAFT and PA-RAG rely on synthetically generated QAs for knowledge ingestion. We prompt Mixtral-8x22B-Instruct-v0.1 to generate synthetic QAs and use the same prompt as described in Bhushan et al., 2025. See appendix K for the exact prompt.

Size of the synthetic training data: The number of question–answer pairs in the synthetically generated training dataset depends on the corpus size. For RAFT and PA-RAG we need to cover the entire corpus with the generated synthetic data. Therefore, we generate pairs such that the total number of generated words is twice the number of words in the corpus. PA-RAG additionally requires multiple answers per question. For each training question, we generate four additional answers using Mixtral-8x22B-Instruct-v0.1. Consequently, the synthetic dataset for PA-RAG contains about ~ 10 times as many words as the corpus. See appendix D for more details.

Recall that DKL also requires a small amount of synthetic QAs, but without the need to cover the full corpus. Consequently, for DKL, we randomly select question-answer pairs such that the total number of selected words is only 50% of the number of words in the corpus.

Training Details: We run all experiments with Hugging Face's `SFTTrainer`. To train the knowledge vector, we use Low Rank Adapters (LoRA) for *all* linear layers in the model with rank $r = 16$. For DKL, after loading the base model, we replace its token embeddings with those of the corresponding instruct model as explained in section 3.

For baseline methods, model selection is based on validation loss with early stopping. For DKL we instead train until convergence of the training

Over-representation of concepts in synthetic QA generation

Source passage: *AI: what's the worst that could happen?*

Concept 1: Short-term AI risk (over-represented)

- **Q1:** What short-term risk associated with AI development is mentioned in the passage?
  A1: A strong societal backlash ("GMO moment") that could block the technology's benefits.
- **Q2:** To what historical event does the speaker compare a potential public backlash against AI?
  A2: A "GMO moment" where strong opposition prevents technological progress.
- **Q3:** Which short-term danger of AI progress does the passage highlight?
  A3: A severe public backlash that could halt deployment of AI technologies.

Concept 2: Long-term AI risk (under-represented)

- **Q4:** What long-term risk associated with AI does the speaker highlight?
  A4: Extreme dependence on AI leading to widespread deskilling of human abilities.

Figure 2: Example illustrating uneven concept coverage in synthetically generated QA pairs for the QuALITY dataset. Multiple questions are generated about the same short-term AI risk concept from a single passage, while the long-term risk discussed in the passage is sampled only once. Such imbalances lead training methods that rely heavily on synthetic QA data (e.g., PA-RAG and RAFT) to overfit to over-represented concepts.

loss and control overfitting via the scaling hyperparameter α. We sweep $\alpha \in \{0.25, 0.5, 0.75, 1.0\}$ and select the best value using validation performance. See appendix G for an alternate way of selecting hyperparameter α.

5 Experimental Results

We seek to answer the following research questions through our experiments:

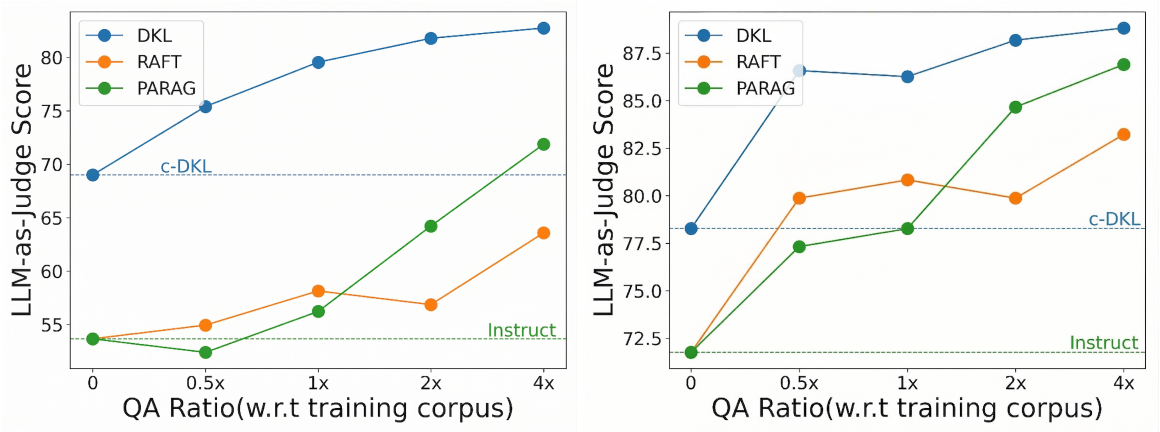
1. Can DKL effectively ingest knowledge into instruct LLM's parameters? To test this, we evaluate the knowledge ingested model in the QA setup where it is provided with only the question and it has to answer from its parametric knowledge.
2. Can DKL effectively combine the knowledge ingested in its parameters with additional knowledge present in its context? To test this, we compare DKL in the RAG setup with SFT based methods such as RAFT (Zhang et al., 2024b) and PA-RAG (Bhushan et al., 2025) that rely heavily on an enormous amount of synthetic data.
3. What is the role of synthetic QA data in DKL? Specifically, is our method robust to the size of the synthetic QA dataset and its coverage of the corpus? To this end, we run two ablations – (1) We vary the size of the synthetic dataset and compare DKL with RAFT and PARAG in both QA and RAG setups. (2) We systematically bias the synthetic QA dataset by generating training QAs from a specific subset of documents and then measure the impact on performance.
4. Is DKL robust to various model architectures and sizes?
5. Finally, we seek to quantify the importance of using token embeddings of the instruct LLM instead of base LLM while training the knowledge adapter, *i.e.*, what happens if we train the adapter on $\theta_B = (\theta_{Be}, \theta_{Br})$ (eq. (7)) instead of $(\theta_{Ie}, \theta_{Br})$ (eq. (8)).

5.1 Comparison with the baselines

In table 1, we compare the performance of several baselines with DKL, highlighting its robustness across different domain corpora. For the Book1 dataset, both RAFT and PA-RAG show the expected improvements over the base instruct model. However, this is not true for the QuALITY dataset.

Inspection of the QuALITY corpus reveals that related concepts appearing in comparable proportions in the source documents (e.g., short-term vs. long-term risks) are unevenly represented in the synthetic QA data, leading to skewed priors over these concepts. See fig. 2 for concrete examples illustrating this imbalance. This imbalance results in failures at test time. When presented with questions about long-term risks, models trained on such skewed synthetic data often default to generating answers about short-term risks. We noticed this behaviour even when the retrieved passages in RAG explicitly contain information about long-term risks. This indicates that methods that rely heavily on synthetic QA generation, such as PA-RAG and RAFT, do not faithfully ground their responses in the provided context but are instead influenced by the training-induced biases.

In contrast, DKL consistently outperforms the baselines across all datasets, as it minimises the



(a) QA

(b) RAG

Figure 3: Impact of scaling synthetic QA on the Redbook1 dataset with Mistral-Instruct-v0.3. Blue horizontal line corresponds to c-DKL– our model trained in an unsupervised manner only using the document text. Green horizontal line corresponds to Mistral-Instruct-v0.3.

reliance on synthetic data generation. This is because corpus coverage is ensured by design: the EPT stage exposes the model to the entire domain corpus, while merging it with the instruct model preserves its ability to effectively utilize this knowledge at inference time. This behavior is reflected in the RAG setting reported in table 1. When the retriever succeeds, DKL effectively leverages the retrieved passages, achieving performance gains up to 92.13 and 83.85. When retrieval fails, the model can ignore the misleading context and answer using knowledge encoded in its parameters.

Comparing DKL with Chat-Vector, we observe that DKL significantly outperforms Chat-Vector on both datasets, demonstrating the importance of swapping the base model's embeddings with the corresponding embeddings of the Instruct model during finetuning and combining the knowledge vector in the optimal ratio. We further analyze it in detail in section 5.5. We observe trends similar to RedBook 1 on RedBook 2. See appendix F for the exact numbers.

5.2 Impact of the size of the synthetic data

In this experiment, we study how scaling the synthetic QA dataset affects model performance. For this, we progressively increase the number of synthetic QA pairs and compare DKL with baselines in both QA and RAG setups. We define '*QA ratio*' as the ratio of number of words in the synthetic QA dataset to the words in the training document. For DKL, QA ratio of 0 corresponds to training only on the corpus (eq. (3)) and we call it corpus-DKL or c-DKL in short. For RAFT and PA-RAG , 0 corresponds to the instruct model. Note that for PA-RAG , we need to generate multiple answers for each question, and the QA ratio does not account for it. Therefore, the actual synthetic QA dataset used for PA-RAG would contain about $5\times$ as many words as RAFT and DKL. Note that the performance numbers in our main experiments correspond to a ratio of 2 for RAFT and PA-RAG , and 0.5 for DKL. For this analysis, we generate additional data and scale up to a ratio of 4.

Figures 3a-3b presents the analysis. We first observe that c-DKL(dotted blue horizontal line) outperforms both baselines in the QA setup, demonstrating the capability of our method to efficiently ingest knowledge.

As seen in Figure-3b DKL can achieve near optimal performance with only $0.5\times$ of synthetic data where the difference of performance is only 2.24% between $0.5\times$ vs $4\times$ of synthetic data. In contrast, PA-RAG improves by more than 10% going from $0.5\times$ to $4\times$ synthetic data, implying PA-RAG indeed needs comprehensive volume of synthetic data for effective knowledge ingestion. A similar trend appears in the QA setup, where DKL not only outperforms the baselines but also shows a smoother saturation curve, unlike the sharp jumps with more synthetic data as seen in PA-RAG. These results empirically establish that DKL is indeed much more lightweight yet the new state-of-the-art scalable knowledge ingestion recipe, which does not need extensive synthetic data generation.

	Ch. 1-3		Ch. 4-5	
	QA	RAG	QA	RAG
c-DKL	52.80	74.40	65.96	80.85
b-DKL	+26.40	+8.00	+ 2.66	+5.25

Table 2: Comparison between models trained without synthetic QA (c-DKL i.e., corpus-DKL) and models trained with chapter-biased synthetic QA(b-DKL)

	SmolLM2-1.7B		Qwen3-0.6B		Llama-3.1-8B		Mistral-7B-v0.3	
	QA	RAG	QA	RAG	QA	RAG	QA	RAG
Instruct	15.53	39.33	18.35	40.65	52.40	67.41	53.67	71.76
RAFT	18.35	40.94	16.47	41.65	60.06	77.96	56.87	79.87
PA-RAG	18.47	38.59	20.00	42.59	65.81	82.75	64.22	84.66
Chat-Vec.	23.52	40.47	23.29	44.94	71.05	78.91	71.47	83.98
DKL	**24.47**	**46.12**	**25.41**	**48.94**	**72.20**	**80.19**	**73.80**	**86.58**

Table 3: Comparing DKL with baselines using 4 different model architectures and sizes on Redbook1.

5.3 Robustness of DKL to corpus coverage by synthetic QAs

In the previous experiment, we observed that DKL is robust to the amount of the synthetic QA dataset, and its RAG performance begins to saturate even with a QA ratio of 0.5. Here, we systematically study the impact of partial knowledge coverage on our method. Allen-Zhu and Li, 2024 note that "*partially augmenting data can improve knowledge extraction for non-augmented data*". Here augmentation refers to adding QAs corresponding to the knowledge being ingested. To systematically study this, we run a control experiment – we train a version of DKL using the document along with QA data generated from only chapters 1 to 3 of Book1 (we call it biased-DKL, or b-DKL) and compare its performance with c-DKL (trained without any QA data). Table 2 shows the results. We observe that adding synthetic QAs from chapters 1 to 3 improves the performance even on chapter 4-5, demonstrating that even if we have access to QA from only a part of the corpus, DKL will still show gains over the remaining data.

5.4 Robustness to Architectures and Sizes

Here we establish that DKL is robust to various model architectures and sizes. In addition to Mistral-7B and Llama-8B, we finetune *SmolLM2-1.7B-Instruct*, and *Qwen3-0.6B* on Redbook1. Table 3 presents the results. We see that DKL consistently outperforms all the baselines. See appendix H for detailed results.

	QA	RAG		
		All	Ret. Success	Ret. Failure
PA-RAG	64.22 ±2.71	84.66 ±2.04	92.70 ±1.47	74.07 ±2.48
DKL	73.80 ±2.49	86.58 ±1.93	92.13 ±1.52	79.26 ±2.29
e-DKL	72.76 ±2.52	84.57 ±2.04	92.13 ±1.52	73.13 ±2.51

Table 4: Effectiveness of using instruct LLM's embeddings during training. Comparing DKL with a version trained directly on top of base LLM (e-DKL) for RedBook1 using Mistral-7B-Instruct

5.5 Impact of using instruct LLM's token embeddings during training

Recall that in DKL we replace the frozen token embeddings θ_{Be} in the base LLM with those from the corresponding instruct LLM. *I.e.*, we train the knowledge LoRA adapter on top of $(\theta_{Ie}, \theta_{Br})$ instead of $(\theta_{Be}, \theta_{Br})$. Here, we quantify its impact by comparing the models trained using eq. (7) and eq. (8), respectively. Table 4 and table 13 shows the results for Mistral and Qwen3-0.6B, respectively.

We find that using instruct LLM's token embeddings improves performance in both QA and RAG setups. Without them, performance drops significantly under retriever failure cases and approaches that of PA-RAG. Thus, replacing the base model's embeddings with those of the instruct model is crucial for DKL to outperform PA-RAG in the RAG setup. Overall, this ablation confirms that using instruct token embeddings is a simple yet effective intervention: it resolves the vocabulary mismatch between the base and instruct LLMs, thereby improving the adaptability of knowledge adapters during inference. See appendix I for a detailed study on the impact of swapping token embeddings.

6 Conclusion

In this work, we introduced DKL, a lightweight and efficient approach for knowledge infusion in Instruct LLMs. By training a knowledge adapter through extended pretraining on the base LLM and transferring it to the instruct LLM, DKL enables effective knowledge ingestion without costly IFT.

Our experiments and ablation study show that DKL consistently outperforms state-of-the-art SFT-based knowledge infusion methods, such as RAFT and PA-RAG, while requiring substantially less synthetic data. These results highlight DKL as a practical and scalable alternative for rapidly incorporating domain-specific knowledge into LLMs.

7 Limitations

Despite the efficacy of DKL for ingesting domain specific corpora into model parameters, it suffers from some fundamental limitations. First, our method relies on the availability of a base model that has not yet undergone instruction fine-tuning. As mentioned in the paper, it is the base model that is more susceptive to extended pre-training as a means of absorbing domain-specific knowledge. The instruction-tuned checkpoints are typically more brittle and prone to overfitting or catastrophic forgetting particularly under the unsupervised training regime. In practice, however, most open weight models are released in instruction-tuned form, limiting the applicability of our approach. Second, the merging procedure itself requires an extensive hyperparameter search to obtain the optimal merging weights for the knowledge and task vectors and the best checkpoint to use for the merge. These hyperparameters are highly sensitive to the underlying data distribution, and we currently lack both a principled theoretical framework and an automated method to select them. Although our proposed strategy for choosing these parameters reliably yields performance gains, extracting the full potential of the method still requires substantial additional experimentation and careful tuning.

References

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of language models: Part 3.1, knowledge storage and extraction. In *Forty-first International Conference on Machine Learning*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Kushagra Bhushan, Yatin Nandwani, Dinesh Khandelwal, Sonam Gupta, Gaurav Pandey, Dinesh Raghu, and Sachindra Joshi. 2025. Systematic knowledge injection into large language models via diverse augmentation for domain-specific RAG. In *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 5922–5943. Association for Computational Linguistics.

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, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Clément Christophe, Tathagata Raha, Svetlana Maslenkova, Muhammad Umar Salman, Praveen K. Kanithi, Marco AF Pimentel, and Shadab Khan. 2024. Beyond fine-tuning: Unleashing the potential of continuous pretraining for clinical llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 10549–10561. Association for Computational Linguistics.

Nico Daheim, Nouha Dziri, Mrinmaya Sachan, Iryna Gurevych, and Edoardo Ponti. 2024. Elastic weight removal for faithful and abstractive dialogue generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7096–7112, Mexico City, Mexico. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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

Shih-Cheng Huang, Pin-Zu Li, Yu-chi Hsu, Kuang-Ming Chen, Yu Tung Lin, Shih-Kai Hsiao, Richard Tsai, and Hung-yi Lee. 2024. Chat vector: A simple approach to equip LLMs with instruction following and model alignment in new languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10943–10959, Bangkok, Thailand. Association for Computational Linguistics.

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proc. of EMNLP*.

Zixuan Ke, Yifei Ming, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. 2025. Demystifying domain-adaptive post-training for financial llms. *Preprint*, arXiv:2501.04961.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pre-training of language models. In *Proceedings of The Eleventh International Conference on Learning Representations*.

Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms. *arXiv preprint arXiv:2404.13081*.

Patrick S. H. 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. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Haokun Liu, Derek Tam, Muqeeth Mohammed, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*.

Wei Lu, Rachel K. Luu, and Markus J. Buehler. 2025. Fine-tuning large language models for domain adaptation: exploration of training strategies, scaling, model merging and synergistic capabilities. *npj Computational Materials*, 11(1).

Shirong Ma, Shen Huang, Shulin Huang, Xiaobin Wang, Yangning Li, Hai-Tao Zheng, Pengjun Xie, Fei Huang, and Yong Jiang. 2023. Ecomgpt-ct: Continual pre-training of e-commerce large language models with semi-structured data. *arXiv preprint arXiv:2312.15696*.

Mistral AI. Mistral tokenization guide. `https://docs.mistral.ai/guides/tokenization`.

Mistral AI. 2024. Model card: Mistral instruct v0.3. `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3`.

Yatin Nandwani, Vineet Kumar, Dinesh Raghu, Sachindra Joshi, and Luis Lastras. 2023. Pointwise mutual information based metric and decoding strategy for faithful generation in document grounded dialogs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10335–10347, Singapore. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Oded Ovadia, Menachem Brief, Rachel Lemberg, and Eitam Sheetrit. 2025. Knowledge-instruct: Effective continual pre-training from limited data using instructions. *arXiv preprint arXiv:2504.05571*.

Gaurav Pandey, Yatin Nandwani, Tahira Naseem, Mayank Mishra, Guangxuan Xu, Dinesh Raghu, Sachindra Joshi, Asim Munawar, and Ramón Fernandez Astudillo. 2024. BRAIn: Bayesian reward-conditioned amortized inference for natural language generation from feedback. In *Forty-first International Conference on Machine Learning*.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. 2022. Quality: Question answering with long input texts, yes! *Preprint*, arXiv:2112.08608.

Huachuan Qiu, Hongliang He, Shuai Zhang, Anqi Li, and Zhenzhong Lan. 2023. Smile: Single-turn to multi-turn inclusive language expansion via chatgpt for mental health support. *arXiv preprint arXiv:2305.00450*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Spurthi Setty, Harsh Thakkar, Alyssa Lee, Eden Chung, and Natan Vidra. 2024. Improving retrieval for rag based question answering models on financial documents.

Zhang Shengyu, Dong Linfeng, Li Xiaoya, Zhang Sen, Sun Xiaofei, Wang Shuhe, Li Jiwei, Runyi Hu, Zhang

Tianwei, Fei Wu, and 1 others. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Zayne Rea Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. MuSR: Testing the limits of chain-of-thought with multistep soft reasoning. In *The Twelfth International Conference on Learning Representations*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *CoRR*, abs/2210.09261.

Xiangru Tang, Tianyu Hu, Muyang Ye, Yanjun Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou, Pan Lu, Zhuosheng Zhang, Yilun Zhao, and 1 others. Chemagent: Self-updating library in large language models improves chemical reasoning. In *The Twelfth International Conference on Learning Representations*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Weidi Xie, and Yanfeng Wang. 2024. Pmc-llama: toward building open-source language models for medicine. *Journal of the American Medical Informatics Association*, 31(9):1833–1843.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2023. Pixiu: A large language model, instruction data and evaluation benchmark for finance. *arXiv preprint arXiv:2306.05443*.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation.

Xianjun Yang, Junfeng Gao, Wenxin Xue, and Erik Alexandersson. 2024. Pllama: An open-source large language model for plant science. *arXiv preprint arXiv:2401.01600*.

Zitong Yang, Neil Band, Shuangping Li, Emmanuel J. Candès, and Tatsunori Hashimoto. 2025. Synthetic continued pretraining. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Wanli Ouyang, and 1 others. 2024a. Chemllm: A chemical large language model. *arXiv preprint arXiv:2402.06852*.

Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. 2023. Composing parameter-efficient modules with arithmetic operation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. 2024b. RAFT: Adapting language model to domain specific RAG. In *First Conference on Language Modeling*.

A DKL Algorithm

Algorithm 1 presents the algorithm for training knowledge adapter using DKL.

Algorithm 1 DKL: Knowledge Adapter Training

Input: Base $(\theta_{Be}, \theta_{Br})$, Instruct $(\theta_{Ie}, \theta_{Ir})$, Corpus $\mathcal{D}_k$, QA set $\mathcal{D}_{qa}$, learning rate η, epochs T
Output: Knowledge adapter $\Delta\theta^{k\cup qa}_{(Ie,Br)}$
Model Init.: $\theta \leftarrow (\theta_{Ie}, \theta_{Br})$
Adapter Init.: $\Delta\theta \leftarrow (\mathbf{0}, \Delta\theta_{(Ie,Br)})$
Union data: $\mathcal{D}_{k\cup qa} \leftarrow \mathcal{D}_k \cup \mathcal{D}_{qa}$
for $t = 1$ **to** T **do**
  for *mini-batch* $\mathcal{B} \subset \mathcal{D}_{k\cup qa}$ **do**
    // Compute loss as in eq. (8)
    $\mathcal{L}_{\mathcal{B}} \leftarrow \sum_{\mathbf{x}\in e\mathcal{B}} -\log \mathbf{Pr}\left(\mathbf{x}; \left(\theta_{Ie}, \theta_{Br} + \Delta\theta_{(Ie,Br)}\right)\right)$
    $\Delta\theta_{(Ie,Br)} \leftarrow \Delta\theta_{(Ie,Br)} - \eta\nabla\mathcal{L}_{\mathcal{B}}$
  end
end
Set $\Delta\theta^{k\cup qa}_{(Ie,Br)} \leftarrow \Delta\theta_{(Ie,Br)}$;
return $\Delta\theta^{k\cup qa}_{(Ie,Br)}$

B More details on Test Datasets

Our test dataset consists of technical Redbooks and their accompanying QAs, as introduced in Bhushan et al., 2025. While manually inspecting the test QAs, we observed that some questions are either incomplete or not properly decontextualized. Therefore, we decided to clean up the test data by prompting Llama-3.1-70B-Instruct to evaluate each QA pair on various dimensions and assign a rating from 1 to 10. We filtered all QA pairs with a score less than 10. The resulting datasets have 313 and 1554 test samples, dropping 26% and 32% of the QAs in the original version. Our small-scale human study reveals that our LLM filter is able to recall 70% of the improper QAs from the test data, thereby improving its quality. See appendix K for the prompt used for cleaning the test data. Below we provide details of the human study. The QuALITY benchmark is originally formulated as a multiple-choice QA task. However, both PA-RAG and RAFT are trained using long-form question–answer pairs rather than multiple-choice. Evaluating these methods directly in a multiple-choice setting would therefore introduce a mismatch between the training and evaluation formats. To ensure a fair comparison, we instead use a curated long-form QA version of the QuALITY test set, following the procedure described in Bhushan et al., 2025.

C Human Annotation and LLM-as-a-Judge Alignment

The objective of our human study is two-fold: (1) To evaluate the efficacy of test data filtering, and (2) To evaluate the correlation between LLM Judge and human judgment.

C.1 Human Annotation Setup

To validate the reliability of our evaluation protocol, we conduct a human annotation study using 50 examples sampled from the Book 1 and Book 2 test splits. Responses were generated with *Mistral v0.3 Instruct* under both QA and RAG setups. For each instance, annotators were provided with the question, the gold answer, and the model-generated answer. Each example was independently rated by three domain experts according to the rubric below:

- **Fully Correct (1):** Response covers all statements in the gold, introduces no contradictions, and may include additional relevant information.
- **Incorrect (0):** Response contradicts the gold, fails to answer the question, or is incomplete/-vague.
- **Ill-formed QA (–1):** The question or gold answer is itself vague, incomplete, or not properly decontextualized.

In cases where all three annotators disagreed, a fourth expert adjudicated to obtain the final label. The final human score was determined via majority vote.

C.2 Human Annotation Results

Annotation statistics are shown in Table 5. We annotate 50 model responses for both QA and RAG setups in Book 2, and an additional 50 responses for the QA setup on Book 1, since scores of 0 were over-represented in the QA annotations of Book 2. Inter-annotator agreement is strong for the RAG setup, with consistently high percent agreement and Krippendorff's α values, reflecting stable human judgments. The QA setup shows a lower agreement ($\alpha \approx 0.66$ compared to ≈ 0.92 for RAG), which we attribute to the longer and more verbose responses (196 words on average vs. 135 in

RAG). These longer responses often include hallucinations or extraneous details, making annotation more challenging.

Setup	Agreement	Krippen. α	AC2	Annotators	Examples	Response Word Count
QA	0.78	0.66	0.68	3	100	196
RAG	0.95	0.92	0.92	3	50	135

Table 5: Human annotation agreement statistics

During annotation, a notable fraction of examples were identified as Ill-formed QA pairs, reflecting limitations of the synthetic test sets (Table 6).

Dataset	Ill-formed	Valid	Total
Book 1	9	41	50
Book 2	15	35	50

Table 6: Filtered examples by humans

C.3 LLM-as-a-Judge for Filtering

To mitigate dataset noise, we employ *Llama 3.1 70B Instruct* as an automatic judge. Each evaluation instance provided the judge with the question and gold answer, and the judge assigns a rating (1–10) based on Accuracy, Relevance, Clarity, and Usefulness (see appendix K for the prompt). QA pairs with ratings < 10 were filtered out. We adapted our prompt from Synthetic Data Kit

This automatic filtering removes $\sim 71\%$ of the Ill-formed QA pairs identified by humans. Extending this procedure to the full test dataset yields the results in Table 7.

Dataset	Before	After
Book 1	425	313
Book 2	2269	1554

Table 7: Dataset size before and after filtering

Examples of removed QA pairs are provided in appendix K.

C.4 LLM-as-a-Judge for Evaluation

We use *Llama 3.3 70B Instruct* as the LLM-as-a-Judge to evaluate the generated responses for all of our experiments. To verify its reliability, we compared the judge's binary decisions (0/1) against the human majority labels on the annotated examples after filtering. The results, shown in Table 8, demonstrate a strong alignment between the LLM-as-a-Judge and human judgments, indicating that the prompt (detailed in appendix K) used produces consistent evaluations throughout the data set.

Dataset	Accuracy	Precision	Recall	TN	FP	FN	TP	Total
QA	0.84	0.86	0.76	39	4	8	25	76
RAG	0.97	1.00	0.94	18	0	1	16	35

Table 8: Alignment of LLM-as-a-Judge with human annotations

C.5 Discussion

Overall, the LLM-as-a-Judge demonstrates strong alignment with human annotations, achieving $\sim 84\%$ accuracy on QA and $\sim 97\%$ on RAG. It is interesting to note the difference in agreement rates between the two setups. We attribute this difference to the fact that in the QA setup, instruct LLM's responses are not grounded on any text. The model's responses generated solely from its parameteric memory tend to be more verbose, confusing the LLM and human judges alike. The comparatively lower accuracy in QA reflects the inherent ambiguity in evaluating context-free generations. This also explains why inter-annotator agreement amongst humans is lower in the QA setup than in the RAG setup.

These findings suggest that (i) the synthetic test sets contain a non-trivial proportion of **Ill-formed QA pairs**, and (ii) LLM-as-a-Judge provides a reliable and scalable mechanism for filtering and evaluating examples in large-scale experiments.

D Data Statistics

Please refer to table 9 for details about both the datasets used in the paper. As mentioned in section 4, PA-RAG and RAFT train sets were created with 2x the amount of words in the domain documents.

E Results on Llama

The main table with the results of DKL as well as various other baselines using LLaMA 3.1 8B model are presented in table 10.

F Results on RedBook 2

The results of DKL and various other baselines on RedBook 2 are presented in table 11.

Dataset	Chapters	Words	Train Samples PA-RAG	Train Samples RAFT	Num. Test Samples	Avg. words per QA	RAFT No. QA words / No. words
RedBook 1	5	15,225	1,107	286	313	106	2
RedBook 2	6	33,795	2,980	770	1,554	87	2
QuALITY	10	43,254	6,973	1,721	738	11	2

Table 9: Data statistics for the datasets used in the paper.

	RedBook 1				QuALITY			
	QA	RAG			QA	RAG		
		All	Ret. Success	Ret. Failure		All	Ret. Success	Ret. Failure
Instruct	52.40 ±2.82	67.41 ±2.65	83.71 ±2.09	45.93 ±2.82	4.07 ±1.32	47.43 ±2.59	81.79 ±2.58	8.91 ±1.90
RAFT	60.06 ±2.77	77.96 ±2.34	88.76 ±1.79	63.70 ±2.72	4.07 ±1.32	45.80 ±2.59	76.15 ±2.86	11.78 ±2.16
PA-RAG	65.81 ±2.68	82.75 ±2.14	92.70 ±1.47	69.63 ±2.60	3.47 ±1.23	47.70 ±2.60	74.87 ±2.91	17.24 ±2.54
Chat Vector	71.05 ±2.56	78.91 ±2.31	89.02 ±1.77	64.63 ±2.70	12.73 ±2.24	49.45 ±3.35	81.53 ±2.61	13.50 ±2.30
DKL	**72.20** ±2.53	**80.19** ±2.25	**89.89** ±1.70	**67.41** ±2.65	**12.74** ±2.24	**52.03** ±3.35	**85.64** ±2.34	**14.37** ±2.36

Table 10: Main table comparing the performance of various baselines descibed in the paper using Llama 8b model.

G Stopping Criteria Ablation

In extended pre-training, a practical challenge is determining when to stop training. Stopping too early risks underfitting, while stopping too late may lead to overfitting to the training corpus. This decision is particularly relevant when merging the pre-trained base model with an instruction-tuned model. The objective of this ablation is to illustrate how the choice of stopping point affects downstream performance.

To study this effect, we conduct experiments on the Book 1 corpus by performing extended pre-training on the *LLaMA 3.1 8B* base model for 60 epochs on DKL's training data mixture. At intermediate checkpoints, we perform task-arithmetic merges with the instruct model using four different merge weights (0.25–1.0) applied to the knowledge-ingested base model. At each checkpoint, we selected the optimal merge according to the LLMaJ Score under RAG setup. We conduct evaluations under both QA and RAG setups. For RAG, the Book 1 validation set was split into two subsets: **(i) Ret. Success**, where the retrieved context passages contain the answer, and **(ii) Ret. Fail**, where the context does not contain the answer.

The resulting performance trends are shown in Figures 4a-4d.

Across all setups, we observe a consistent trend: performance improves substantially in the early and mid stages of training, peaks at intermediate checkpoints, and then gradually declines as training continues to convergence. For the sake of uniformity across baselines and experimental conditions, we opted to train until convergence before performing merges. As a result, the scores presented in the main article should be viewed as *conservative estimates*. More careful stopping criteria could further enhance performance.

H Robustness to Model Architecture and Sizes

The goal of this experiment is to establish that our proposed method DKL works across model architectures and sizes. Consequently, we experiment using two additional models, varying both the size and model family. Specifically, we train *SmolLM2-1.7B-Instruct* and *Qwen3-0.6B* on Redbook1. Table 12 presents the results. We observe that DKL outperforms all baselines across model sizes and architectures, demonstrating its robustness.

I Token Swapping Ablation

First, we present the ablation results with Qwen3-0.6B in table 13 and observe similar trends as observed with Mistral.

Next, we conducted experiments to determine where the major performance boost originates during embedding swap: from the OOD tokens in the base model, such as those introduced in the chat template, or from the tokens already trained in the base model. We use two strategies to automatically select the probable OOD tokens:

1. top-k: Select top-k tokens w.r.t. the L-2 norm of the difference between instruct and base models' token embeddings

2. top-p: A nucleus sampling-like approach

	Mistral-7B-v0.3				LLaMA 3.1-8b			
	QA	RAG			QA	RAG		
		All	Ret. Success	Ret. Fail.		All	Ret. Success	Ret. Fail.
Instruct	27.51 ±1.13	61.23 ±1.24	77.96 ±1.05	36.13 ±1.22	26.61 ±1.12	59.86 ±1.24	78.97 ±1.03	31.13 ±1.17
RAFT	27.23 ±1.13	62.95 ±1.23	79.16 ±1.03	38.65 ±1.24	31.61 ±1.18	65.44 ±1.21	80.34 ±1.01	**43.06 ±1.26**
PA-RAG	27.23 ±1.13	62.23 ±1.23	78.60 ±1.04	37.64 ±1.23	31.87 ±1.18	64.13 ±1.22	79.03 ±1.03	41.77 ±1.25
Chat-Vector	29.83 ±1.16	64.41 ±1.21	76.74 ±1.07	45.91 ±1.26	33.46 ±1.20	62.82 ±1.22	73.01 ±1.13	42.25 ±1.25
DKL	**40.98 ±1.25**	**66.86 ±1.19**	**80.67 ±1.00**	**46.13 ±1.26**	**34.92 ±1.21**	**64.54 ±1.21**	**80.58 ±1.00**	40.39 ±1.24

Table 11: Results of Mistral-7b and LLaMA 3.1-8b on RedBook 2.

	SmolLM2-1.7B				Qwen3-0.6B			
		RAG				RAG		
	QA	All	Ret. S.	Ret. F	QA	All	Ret. S.	Ret. F
Instruct	15.53 ±2.04	39.33 ±2.75	50.65 ±2.83	22.35 ±2.36	18.35 ±2.18	40.65 ±2.77	58.73 ±2.79	19.81 ±2.25
RAFT	18.35 ±2.18	40.94 ±2.78	51.93 ±2.82	27.60 ±2.52	16.47 ±2.11	41.65 ±2.79	53.22 ±2.82	27.60 ±2.52
PA-RAG	18.47 ±2.18	38.59 ±2.74	43.78 ±2.81	32.29 ±2.64	20.00 ±2.26	42.59 ±2.79	53.65 ±2.82	29.17 ±2.57
Chat-Vector	23.52 ±2.39	40.47 ±2.77	51.07 ±2.83	27.60 ±2.52	23.29 ±2.38	44.94 ±2.81	61.80 ±2.75	24.47 ±2.43
DKL	**24.47 ±2.43**	**46.12 ±2.82**	**57.94 ±2.79**	**31.77 ±2.63**	**25.41 ±2.45**	**48.94 ±2.83**	**65.24 ±2.69**	**29.17 ±2.57**

Table 12: Comparing DKL with baselines using different model architectures and sizes on Redbook1.

	QA	RAG		
		All	Ret. Success	Ret. Failure
Instruct	18.35	40.65	58.73	19.81
DKL-e	11.76	46.35	60.94	28.65
DKL	**25.41**	**48.94**	**65.24**	**29.17**

Table 13: Effectiveness of using instruct LLM's embeddings during training. Comparing DKL with a version trained directly on top of base LLM (e-DKL) for RedBook1 using Qwen3-0.6B

	QA	RAG		
		All	Ret. Success	Ret. Failure
DKL	73.80	86.58	92.13	79.26
top_k-50-DKL	74.12	87.22	94.38	77.78
top_p-0.9-DKL	74.12	84.66	91.01	76.30
bottom_k-50-DKL	72.52	83.39	92.13	71.85
bottom_p-0.9 DKL	73.48	81.79	88.20	73.33
e-DKL	72.76	84.57	92.13	73.13

Table 14: Ablation to study the effect of embeddings. top_*-50/bottom_*-50 refer to the embeddings with the most/least distance between the instruct and base models as chosen by the sampling method mentioned above. e-KnitLM refers to the run without doing any embedding swaps.

where we select the top-p tokens upto a threshold of 0.9.

Row 2 and 3 (top_k-50 and top_p-0.9) represents the version where only the most different embeddings were swapped, and Row 4 and 5 (bottom_k-50 and bottom_p-0.1) represents the version where the most different embeddings were excluded, and only the rest of the embeddings were swapped. Recall that e-DKL(Row 6) represents the version where none of the embeddings were swapped.

We see that clearly the top-50 embeddings influence the final trained model much more than the rest, achieving performance equivalent to the full DKL regime with just the top 50 embeddings swapped. On the other hand, we can see Row 4 (bottom_k-50-DKL) and Row 6 (e-DKL) having comparable performance indicating that the embeddings which were the same in both the instruct and base models have little impact on the final performance. Another interesting observation is the drop in performance between rows 2(top_k-50-DKL) and 3(top_p-0.9). One would expect that adding more meaningful tokens from the instruct model would improve performance, this is not what we see. Our hypothesis as to why this happens can be explained due to 2 major phenomena:

- the inherent value of each token from the instruct model i.e. how useful a particular token is to the training.

- the consistency of the entire embedding layer i.e., how consistent the embedding layer is with respect to its tokens (an embedding layer with only tokens from one model is said to be highly consistent where as a layer with 50% tokens from one model and the rest from another is said to be highly inconsistent).

We posit that the OOD tokens help to a certain extent but soon start interfering with the other tokens as inconsistency within the layer increases. However, swapping the entire layer preserves consistency(as all tokens in the base model are swapped with instruct model's) as well as utilising the more useful tokens during training. As swapping does not introduce any computational bottleneck we advise to always switch the entire layer as opposed to targeted tokens.

J Performance on general tasks

We compare DKL with Llama 3.1 8B Instruct (Instruct), RAFT, and PARAG on several benchmarks:

- **Big Bench Hard** (Suzgun et al., 2022): 23 challenging tasks spanning language understanding and reasoning.
- **GPQA** (Rein et al., 2024): Google-Proof Graduate-level STEM questions.
- **MATH-Hard** (Hendrycks et al., 2021): Difficult math competition questions.
- **MMLU-Pro** (Wang et al., 2024): 12k questions across diverse fields, measuring general knowledge.
- **MUSR (Multistep Soft Reasoning)** (Sprague et al., 2024): Evaluates reasoning capabilities of LLMs.

table 15 reports the performance for RedBook 1.

DKL maintains competitive performance across all general benchmarks, while RAFT and PARAG show regression on general tasks relative to Instruct.

K Prompts and Examples

This appendix presents the prompts used for three purposes: (i) filtering low-quality QA pairs from the dataset, (ii) evaluating responses generated by LLMs, and (iii) generating synthetic QA pairs. We also provide examples of QA pairs that were removed during the filtering process, along with sample responses from our method and the baseline models.

K.1 Filtering Prompt

The following prompt was used to identify Ill-formed QA pairs during dataset filtration. The filtering judge receives a question and its gold answer as input. It considers multiple criteria such as accuracy, relevance, clarity and usefulness and outputs a score from 1–10.

Filtering Prompt

```
Rate each question-answer pair on
    a scale from 1-10, based on:
- Accuracy (0-3): factual
    correctness
- Relevance (0-2): relevance to
    content
- Clarity (0-2): clear language
- Usefulness (0-3): value for
    model learning

YOU MUST RETURN A VALID JSON
    OBJECT OR ARRAY WITH THIS EXACT
     SCHEMA:
{{
  "question": "Exact question text
      ",
  "answer": "Exact answer text",
  "explanation": {{
    "Accuracy": "Short explanation
         of factual correctness",
    "Relevance": "Short
        explanation of relevance",
    "Clarity": "Short explanation
        of clarity",
    "Usefulness": "Short
        explanation of usefulness"
  }},
  "Accuracy": 2,
  "Relevance": 2,
  "Clarity": 2,
  "Usefulness": 2,
  "rating": 8
}}

OR FOR MULTIPLE PAIRS:
[
  {{
    "question": "Q1",
    "answer": "A1",
    "explanation": {{
      "Accuracy": "Explanation for
           Accuracy",
      "Relevance": "Explanation
          for Relevance",
      "Clarity": "Explanation for
          Clarity",
      "Usefulness": "Explanation
          for Usefulness"
    }},
```

	Big Bench Hard	GPQA	MATH-Hard	MMLU Pro	MUSR	Aggregate
Instruct	29.88	5.36	17.47	37.83	8.73	19.85
RAFT	29.75	6.22	14.87	37.67	6.01	18.90
PA-RAG	30.28	4.85	14.63	36.95	6.73	18.69
DKL	29.69	7.59	17.11	38.08	6.75	19.84

Table 15: General Task Performance

```
    "Accuracy": 2,
    "Relevance": 2,
    "Clarity": 2,
    "Usefulness": 2,
    "rating": 8
  }},
  {{
    "question": "Q2",
    "answer": "A2",
    "explanation": {{
      "Accuracy": "Explanation for
           Accuracy",
      "Relevance": "Explanation
          for Relevance",
      "Clarity": "Explanation for
          Clarity",
      "Usefulness": "Explanation
          for Usefulness"
    }},
    "Accuracy": 3,
    "Relevance": 2,
    "Clarity": 2,
    "Usefulness": 2,
    "rating": 9
  }}
]

*** YOUR RESPONSE MUST BE VALID
    JSON AND NOTHING ELSE - NO
    EXPLANATION, NO MARKDOWN ***

QA pairs to rate:
{pairs}
```

K.2 LLM-as-a-Judge Prompt

The following prompt was used to evaluate model-generated responses. The model is provided with the question, gold answer and model generated answer, and it outputs a binary rating (0/1) according to the specified evaluation rules.

LLM Evaluation Prompt

```
You are an evaluator. Your task is
     to compare a Ground-truth
    Answer and a Prediction to
    decide if the Prediction
    correctly answers the given
    Question.

Evaluation Rules:
(1) Correctness: A correct
    prediction must include all
    essential information from the
    Ground-truth Answer. Extra
    information is allowed if it
    does not contradict the Ground-
    truth. If the Prediction states
     something as a possibility,
    treat it as a definitive
    statement.

(2) Function, Tool Names, and API
    Calls: If the Ground-truth
    Answer contains specific
    function names, tool names, API
     calls, or exact command
    identifiers, the Prediction
    must contain the same
    identifier(s) or clearly
    equivalent forms. Minor
    syntactic or formatting
    variations that do not change
    meaning should be treated as
    equivalent. For example,
    leading flag prefixes such as
    -, --, or no prefix at all when
     they clearly refer to the same
     option;  underscore vs hyphen
    differences in identifiers when
     the intent is identical;
    surrounding punctuation or
    formatting differences such as
    backticks, quotes, parentheses,
     or code block notation; small
    whitespace differences or
    capitalization differences that
     do not change the identifier's
     meaning etc. However,
    replacements that change the
    actual function/tool/API name,
    or substitute a different
    command that would change the
    behavior are considered
    incorrect. Do not penalize a
    prediction if it contains
    additional function / tool /
    API names as long as the ones
    present in the Ground-Truth are
     covered.

(3) URLs: If the Ground-truth
    Answer contains specific URLs,
    the Prediction should reference
     the same URL or an equivalent
```

```
    canonical form. Minor
    differences that do not change
    the target resource (for
    example, presence or absence of
     a trailing slash, or http vs
    https when both resolve to the
    same canonical resource) should
     be treated as equivalent.
    Altering the domain, path, or
    query such that the resource is
     different is incorrect.

Scoring Rules:
If the Prediction is correct
    according to the above rules,
    output <score>1</score>. If the
     Prediction is incomplete or
    incorrect, output <score>0</
    score>.

Output Format:
<explanation>
...
</explanation>
<score>
...
</score>

First provide reasoning inside <
    explanation> and </explanation>
     tags. Then output the score as
     specified above within <score>
     and </score> tags. Do not
    include any extra text outside
    these tags.
```

K.3 Prompt for generating synthetic QA

The prompt generates fully contextualized question–answer pairs from a document, covering the entire content and formatted with specific tags.

QA Generation Prompt

```
Create question answer pairs from
    the document given below within
     <document> tags. Title of the
    document is given in the first
    line of the document. Do not
    use co-referencing and pronouns
     at all in the questions. Do
    not refer to the document in
    the question like "according to
     the document ..." or any
    similar paraphrasing. When
    needed, contextualize the
    question by using the topic
    that the question is about. You
     can use the title of the
    document as well for
    contextualizing. There are
    several figures in the document
    , while referring to the figure
     in any question, contextualize
     it by mentioning the title of
    the passage it was present in.
    Put questions within <question>
     and </question> tags and
    answers within <answer> and </
    answer> tags. Ensure that the
    question and answers cover the
    entire document. When you are
    done generating QA pairs,
    generate </done> token.
```

K.4 Examples of QA Pairs Removed During Filtering

Below are three representative examples of QA pairs that were filtered out by the LLM-as-a-Judge. Each example shows the question and gold answer.

Question: What is the state of the second volume to be mapped to the hostcluster?
Gold Answer: The state of the second volume to be mapped to the hostcluster is present.

× **Vague**

Question: What was the status, memory, and CPU usage of the voting-app-worker-py-1 pod in the Red Hat OpenShift Cluster on IBM LinuxONE?
Gold Answer: The status, memory, and CPU usage of the voting-app-worker-py-1 pod in the Red Hat OpenShift Cluster on IBM LinuxONE were as follows:
- Status: Completed
- Memory: Not specified
- CPU: Not specified

× **Incomplete**

Question: What are the settings for the infra nodes in the deployment of Red Hat OpenShift on IBM Z?
Gold Answer: The infra nodes have settings such as disk size, RAM, vCPU, VM name, IP address, and hostname. These settings are defined in Section 10 of the all.yaml file.

× **Contextual**

K.5 Example Model Responses

Question: What command is used to upgrade the IBM Storage Virtualize collection to the latest version from Ansible Galaxy?
Gold Answer: The command used to upgrade the IBM Storage Virtualize collection to the latest version from Ansible Galaxy is: *ansible-galaxy collection install ibm.storage_virtualize –force*

To upgrade the IBM Storage Virtualize collection to the latest version from Ansible Galaxy, you can use the following command:
“‘ *ansible-galaxy collection install ibm.storwize:latest* “‘

× **Instruct**

The command used to upgrade the IBM Storage Virtualize collection to the latest version from Ansible Galaxy is:
“‘*bash ansible-galaxy collection install -r requirements.yml –force* “‘

× **RAFT**

The command to upgrade the IBM Storage Virtualize collection to the latest version from Ansible Galaxy is:
“‘*bash ansible-galaxy collection install -r requirements.yml* “‘

× **PA-RAG**

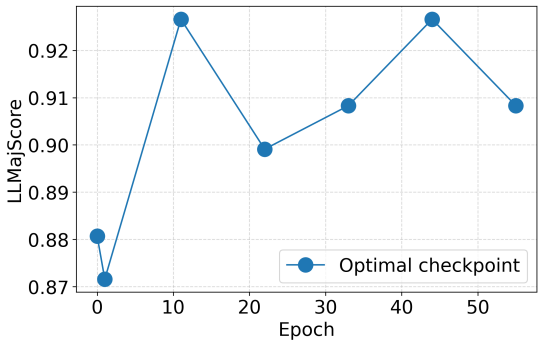
"The command to upgrade the IBM Storage Virtualize collection to the latest version from Ansible Galaxy is:
ansible-galaxy collection install ibm.storage_virtualize –force"

✓ **DKL**

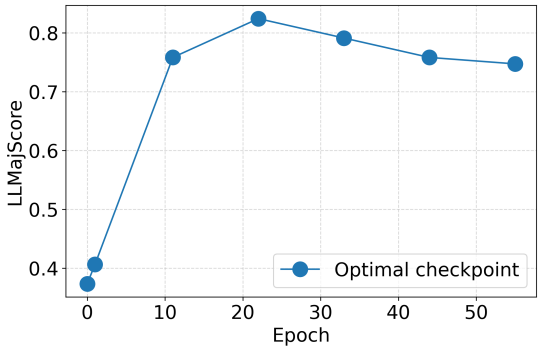
Only DKL provides the exact command that correctly upgrades the IBM Storage Virtualize collection. The instruct's response contains a typo in the collection name (‘storwize‘ instead of ‘storage_virtualize‘), while RAFT and PA-RAG incorrectly rely on a requirements file, which is not specified in the ground truth.

L LLM Usage

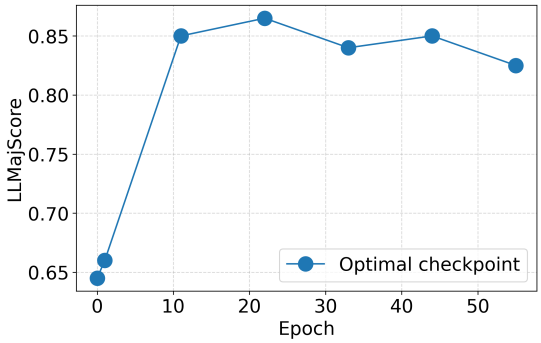
During the preparation of this manuscript, we employed a Large Language Model (LLM) as a writing support tool. Specifically, LLM was used to polish the phrasing, improve grammatical accuracy, and provide paraphrased alternatives to enhance clarity and readability. The LLM's role was limited to language refinement, and all suggested edits were reviewed and verified by the authors before inclusion.



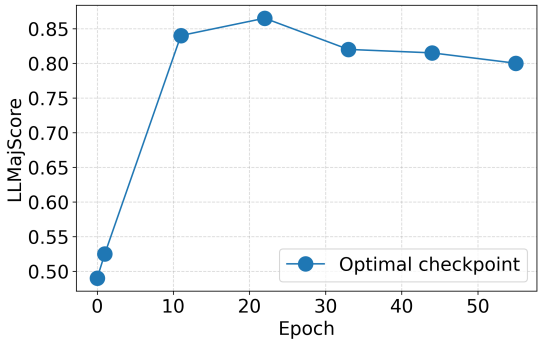
(a) RAG: Some overlap



(b) RAG: No overlap



(c) RAG: All



(d) QA setup

Figure 4: Stopping criteria ablation: best LLMajScore across checkpoints.