Systematic Knowledge Injection into Large Language Models via Diverse Augmentation for Domain-Specific RAG

Kushagra Bhushan*†[1], Yatin Nandwani*[2], Dinesh Khandelwal[2], Sonam Gupta[2],
Gaurav Pandey[2], Dinesh Raghu[2], and Sachindra Joshi[2]

`20je0506@mc.iitism.ac.in, {yatin.nandwani@, dikhand1@in., sonam.gupta7@, gpandey1@in.,diraghu1@in., jsachind@in.}ibm.com`

[1]IIT(ISM) Dhanbad [2]IBM Research

Abstract

Retrieval-Augmented Generation (RAG) has emerged as a prominent method for incorporating domain knowledge into Large Language Models (LLMs). While RAG enhances response relevance by incorporating retrieved domain knowledge in the context, retrieval errors can still lead to hallucinations and incorrect answers. To recover from retriever failures, domain knowledge is injected by fine-tuning the model to generate the correct response, even in the case of retrieval errors. However, we observe that without systematic knowledge augmentation, fine-tuned LLMs may memorize new information but still fail to extract relevant domain knowledge, leading to poor performance. In this work, we present a novel framework that significantly enhances the fine-tuning process by augmenting the training data in two ways – context augmentation and knowledge paraphrasing. In context augmentation, we create multiple training samples for a given QA pair by varying the relevance of the retrieved information, teaching the model when to ignore and when to rely on retrieved content. In knowledge paraphrasing, we fine-tune with multiple answers to the same question, enabling LLMs to better internalize specialized knowledge. To mitigate catastrophic forgetting due to fine-tuning, we add a domain-specific identifier to a question and also utilize a replay buffer containing general QA pairs. Experimental results demonstrate the efficacy of our method over existing techniques, achieving up to 10% relative gain in token-level recall while preserving the LLM's generalization capabilities.

1 Introduction

Large Language Models (LLMs) have transformed natural language processing, excelling across various tasks (Brown et al., 2020). To tailor LLMs for domain-specific applications, such as question answering over specialized corpora, Retrieval-Augmented Generation (RAG) has emerged as a popular approach (Lewis et al., 2020; Karpukhin et al., 2020). Although RAG enhances the relevance of generated answers, it is prone to hallucinations (Ji et al., 2023; Nandwani et al., 2023), especially when the retriever fails to fetch relevant documents from the corpus.

To address this issue, *knowledge injection* techniques (Zhang et al., 2024a; Yoran et al., 2024) have been proposed. Knowledge injection has two main objectives: (1) when the retriever succeeds in fetching correct documents, then the LLM should be able to leverage that information to generate an appropriate response, and (2) when the retriever fails, the LLM should recall the domain specific information from the infused parametric knowledge to generate the response. To achieve them, existing techniques fine-tune LLMs with domain-specific data, embedding the knowledge directly into the LLM's parameters. Specifically, these techniques fine-tune LLMs to generate the correct response to a question irrespective of the relevance of the retrieved documents. However, existing knowledge injection techniques suffer from two main issues:

1) *Conditional Memorization Bias:* In the training data, each question is assigned to either a retrieval success or a retrieval failure scenario. *I.e.*, the relevant information to answer the question is either present or absent in the given context for a question. This static assignment determines how the LLM learns the knowledge. For example, consider a scenario where all training questions from a particular document are assigned to the 'retrieval success' bucket, *i.e.*, information required to answer is present in the accompanying context. In this case, the LLM is encouraged to rely on the external retrieved context, and it may not memorize information from such documents. Conversely, if all train-

---

[*] Equal contribution.

[†] Work done during Kushagra Bhushan's internship at IRL

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 5937–5958
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

ing questions from a particular document are assigned to 'retrieval failure' bucket, the LLM will be forced to memorize its content during fine-tuning and may learn to ignore the provided information for questions from that document. As a result, the LLM learns different sections of the domain data in different ways. This inconsistency can be problematic, as the LLM might struggle when faced with opposite scenarios during inference. We confirm this behaviour in our experiments.

2) *Canonical Answer Overfitting:* Each question in the fine-tuning dataset is associated with only one canonical answer. This singular association leads the LLM to learn and replicate spurious patterns (Allen-Zhu and Li, 2024), treating the answer as a fixed representation for that specific question. As a result, the LLM's ability to generate nuanced or diverse responses based on varying contextual factors is constrained.

To mitigate these issues, we propose PA-RAG: Paraphrase Augmentation for Retrieval-Augmented Generation, a novel fine-tuning framework that improves knowledge injection into LLMs for domain-specific RAG tasks. PA-RAG introduces two different ways of training data augmentation to mitigate conditional memorization bias and canonical answer overfitting.

First, PA-RAG uses context augmentation to simulate both retrieval success and retrieval failure scenarios for all the training questions. This prevents conditional memorization bias by teaching the model to identify whether the given retrieved information is relevant or not and then accordingly decide to ignore or utilize it.

Second, PA-RAG synthetically generates multiple answers for each training question to mitigate canonical answer overfitting. It is inspired by a recent study (Allen-Zhu and Li, 2024) demonstrating that paraphrasing knowledge during pre-training significantly enhances LLMs' ability to inject and extract that information in the downstream tasks. However, they focus on paraphrasing only during pre-training and not during fine-tuning. Building on this, we propose to systematically augment the fine-tuning data by synthetically generating multiple answers for each question. This encourages LLMs to memorize and extract the domain knowledge effectively while minimizing the overfitting on stylistic features.

Furthermore, to address the challenge of *catastrophic forgetting* (Zhang and Wu, 2024; Ke et al., 2023; Jang et al., 2022)—where an LLM's general language understanding deteriorates as domain-specific fine-tuning overwrites prior knowledge—we introduce a *self-selective rehearsal replay buffer* (Gupta et al., 2024; Huang et al., 2024). This buffer contains samples from an instruction tuning dataset. But unlike traditional replay buffers, the self-selective approach uses LLM's own predictions rather than the gold response to retain the LLM's versatility across tasks when injecting knowledge.

In addition, we also introduce the novel use of *Domain Identifiers*—phrases pre-pended to questions—to help the LLM distinguish new knowledge from its existing skills. This improves domain-specific accuracy and also mitigates catastrophic forgetting, preserving LLM's general competence during fine-tuning.

To validate PA-RAG's ability to inject new domain-specific knowledge, we require a corpus that the LLM hasn't seen during pre-training or instruction tuning. In the absence of such a corpus, we create two datasets using domain-specific books that were published in 2024[1]. In our experiments, PA-RAG achieves a significant increase in performance against other methods on this domain-specific benchmark while maintaining general reasoning capabilities that we measure using various benchmarks such as MMLU (Hendrycks et al., 2021), TruthfulQA (Lin et al., 2022), Hellaswag (Zellers et al., 2019), and GSM8k (Cobbe et al., 2021).

2 Related Work

Retrieval Augmented Generation: RAG enhances Large Language Models (LLMs) by integrating external data sources, such as knowledge bases, to improve relevance and accuracy (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020). Recent advancements have extended its applicability across domains (Asai et al., 2024; Kim et al., 2024; Yan et al., 2024; Liu et al., 2024), but RAG systems still face key challenges: hallucinations due to mismatches between retrieved data and the LLM's pre-existing knowledge (Setty et al., 2024; Jin et al., 2024), difficulty with complex multi-document reasoning (Setty et al., 2024), and an inability to fully leverage fixed-domain settings where all domain-specific documents are available beforehand because typically neither the retriever

---

[1] after the cutoff date of the fine-tuned LLM

nor the generator LLM are trained on the domain data.

Domain-Aware Fine-Tuning for RAG: Joint training of the retriever and LLM has been proposed as a way to improve RAG's domain-specific performance (Guu et al., 2020; Sachan et al., 2021; Siriwardhana et al., 2023; Shi et al., 2024). By jointly training the retriever and LLM, the system can better adapt to domain-specific contexts. However, this approach introduces complexities, including the need for specialized loss functions and frequent retriever updates.

Another line of work (Mecklenburg et al., 2024; Zhang et al., 2024b) focuses solely on adding domain knowledge to LLMs as an alternative to RAG. These approaches fine-tune LLMs using question-answer (QA) pairs derived from domain data and aim to answer any new test query without retrieving any document. As a result, they fail to leverage access to the domain documents during inference.

Recently, Zhang et al. introduced *Retrieval-Augmented Fine-Tuning (RAFT)*, a fine-tuning method for LLMs to incorporate domain knowledge and enhance in-domain RAG performance. RAFT combines RAG and fine-tuning by training LLMs on domain data using a mixture of oracle and distractor document contexts. However, it suffers from conditional memorization bias and canonical answer overfitting. On the other hand, PA-RAG uses context augmentation and answer paraphrasing to address these issues.

Catastrophic forgetting: Catastrophic Forgetting (French, 1999; Zheng et al., 2024) occurs when new domain-specific fine-tuning overwrites previously learned general knowledge, reducing performance on earlier tasks. Replay-based methods (de Masson d'Autume et al., 2019; Rolnick et al., 2019), help mitigate this by rehearsing prior task data during training. Recent advances in replay-based approaches for language models (Scialom et al., 2022; Mok et al., 2023) have shown promise in reducing catastrophic forgetting. The Self-Synthesized Rehearsal (SSR) (Gupta et al., 2024; Huang et al., 2024) framework uses the LLM to generate synthetic rehearsal data, reducing reliance on stored instances.

3 Problem Definition

Given a domain-specific corpus $D = \{d_i\}_{i=1}^{n}$, where each document contains domain-relevant knowledge, the goal is to fine-tune Large Language Models (LLMs) for enhanced performance in domain-specific Retrieval-Augmented Generation (RAG) systems. Unlike previous works (Lin et al.; Wang et al.; Mecklenburg et al., 2024), which handle changing test-time domains or documents, our approach, like RAFT, assumes a fixed target domain with known access to domain-specific documents.

Our proposed method, PA-RAG described in Section 4, addresses the following key challenges:

1) Canonical answer overfitting: In Section 4.1, we discuss how paraphrased answer augmentation is used to generate multiple variations of answers per train question, ensuring the LLM learns the underlying knowledge and avoids overfitting to fixed answers.
2) Conditional memorization bias: In Section 4.2, we introduce a method that simulates both successful and failed retrieval scenarios for each question. This helps the LLM learn how to handle various retrieval conditions, preventing it from memorizing answers in static settings and improving its generalization.
3) Catastrophic forgetting: In Sections 4.3 and 4.4, we outline how a self-selective rehearsal replay buffer and domain-specific identifiers help balance domain-specific knowledge acquisition while retaining the LLM's general knowledge and capabilities.

Through these strategies, PA-RAG enables more effective knowledge integration, improving domain-specific performance while maintaining generalization capabilities in RAG systems.

4 Approach

In this section, we outline our approach, which involves – 1). generating a synthetic QA dataset from the documents of the given domain; 2). augmenting it to improve knowledge extraction and; 3). incorporating domain identifiers to clearly define boundaries between distinct knowledge areas.

4.1 Synthetic QA Generation

Given a domain-specific corpus $D = \{d_i\}_{i=1}^{n}$ of n documents, our objective is to generate diverse question-answer (QA) pairs that maximize information coverage across the documents using an LLM. We generate these QA pairs using Mixtral-8x22B-Instruct-v0.1[2] as the LLM.

---

[2] mistralai/Mixtral-8x22B-Instruct-v0.1

Document Chunking: If a document d_i has more than T tokens, we split it into chunks of size $T//2$. Otherwise, it is processed as a single chunk. Here T is a hyperparameter. We denote the set of chunks for document d_i as $C^{(i)} = \{c_1^{(i)}, c_2^{(i)}, \dots, c_{m_i}^{(i)}\}$, where m_i is the total number of chunks for document d_i. Each $c_s^{(i)}$ represents the s-th chunk of document d_i, where $1 \leq s \leq m_i$.

QA Pair Generation: For each chunk $c_s^{(i)} \in C^{(i)}$, we prompt the LLM N_c times using an instruction P_1 along with the chunk $c_s^{(i)}$. Each call to the LLM using prompt P_1 generates multiple QA pairs, ensuring that most of the tokens in $c_s^{(i)}$ are covered. Multiple calls with nucleus sampling (Holtzman et al., 2020) ensure adequate token coverage. Let $\{(q_s^j, a_s^j)\}_{j=1}^k$ be the set of QA pairs generated for chunk $c_s^{(i)}$. We call this dataset of QA pairs as $\mathcal{D}_{\text{base}} = \{\{(q_s^j, a_s^j)\}_{j=1}^k, \forall c_s^{(i)} \in C^{(i)}, \forall d_i \in D\}$. This approach ensures broad content coverage, but the lack of variability in the generated answers can lead to Canonical Answer Overfitting, as described in Section 1.

Adding Answer Multiplicity: To address canonical answer overfitting, we introduce answer multiplicity by re-prompting the LLM to generate multiple answers for each generated question q_s^j using a separate prompt P_2. For each chunk-question pair $(c_s^{(i)}, q_s^j)$, we generate a set of paraphrased answers $\{a_s^{j1}, a_s^{j2}, \dots, a_s^{jp}\}$, where p is the number of paraphrases:

$\text{LLM}(P_2, c_s^{(i)}, q_s^j) \rightarrow \{a_s^{j1}, a_s^{j2}, \dots, a_s^{jp}\}$

This results in the augmented dataset $\mathcal{D}_{\text{aug.}}$, where each question q is linked to diverse paraphrased answers. The added variability mitigates canonical answer overfitting and improves generalization. As shown in ablation studies, the model fine-tuned on $\mathcal{D}_{\text{aug.}}$ outperforms the one fine-tuned on the base dataset $\mathcal{D}_{\text{base}}$. All prompts used for dataset creation, including P_1 and P_2, are provided in Section E.

4.2 Fine-tuning Strategy

In the original RAFT approach (Zhang et al., 2024a), each data point used for fine-tuning consists of a question (q), a context (a collection of documents), and an answer (a). The context is classified as either *relevant*, *i.e.*, it contains at least one document (alongside distractor documents) that provides the information needed to deduce the answer, or *irrelevant* where the entire context consists of distractor documents. Baseline RAFT, referred to as RAFT(p), presents a fraction $(1 - \text{p})$ of questions with a relevant context. We call this training subset as the 'retriever success' bucket. For the remaining p fraction of questions, only an irrelevant context (composed entirely of distractor documents) is provided. We call such a training subset as the 'retriever failure' bucket. However, this fine-tuning setup leads to conditional memorization bias, where the LLM either relies on relevant contexts or stores the necessary information in its parametric memory when presented with irrelevant contexts. As a result, the model may excel at handling specific retrieval scenarios but struggle to generalize effectively across varied contexts during inference. For some parts of the document, the LLM memorizes the content, while for others, it relies on the retrieved documents, resulting in inconsistent performance.

To address conditional memorization bias, we introduce CA-RAFT (Context-Augmented RAFT), which combines RAFT(0) and RAFT(1) with RAFT(p) for more granular control over relevant and irrelevant contexts. In RAFT(0), all questions are paired with irrelevant contexts, teaching the model to rely solely on its internal memory. In RAFT(1), only relevant contexts are provided. By mixing RAFT(0), RAFT(1), and RAFT(p), CA-RAFT(p) effectively mitigates Conditional Memorization Bias.

Our complete approach, PA-RAG significantly enhances CA-RAFT by incorporating paraphrased answer augmentation to address Canonical Answer Overfitting. While CA-RAFT ensures the model learns how to handle both relevant and irrelevant contexts without over-relying on one or the other, PA-RAG goes further by exploiting diverse paraphrased answers during training to effectively inject new knowledge. This augmentation ensures that the model does not overfit to a single canonical answer for each question, promoting deeper learning of underlying knowledge.

Interestingly, answer augmentation implicitly takes care of conditional memorization bias as well. When we randomly assign each training QA pair in $\mathcal{D}_{\text{aug.}}$ to 'retriever success' or 'retriever failure' bucket, it automatically distributes different QA pairs for the same question into different buckets. Thus, when training with $\mathcal{D}_{\text{aug.}}$, we do not need context augmentation separately.

4.3 Domain Identifiers

LLMs store vast amounts of information across different domains, which can lead to confusion when handling specialized questions. To ensure accurate and relevant responses, we introduce *domain identifiers* that establish boundaries within the model's parametric memory by specifying the context of each question. A domain identifier is a simple token or phrase prepended to each question during training and evaluation. For example:

```
"This question is from {domain_name}.
{Question}"
```

Here, the placeholder `{domain_name}` is replaced with the specific domain (e.g., Healthcare, Legal), and `{Question}` is replaced with the actual question. This template ensures that each question is clearly tied to its domain, reducing ambiguity and helping the model differentiate between similar questions from different fields. Additionally, domain identifiers have minimal impact on general performance, as removing them may allow the model to return to its original state, preserving its generalization capabilities.

4.4 Self-Selective Replay Buffer

Typically, a replay buffer contains old samples from previous tasks on which the LLM has been trained. This helps the model retain general knowledge and mitigate catastrophic forgetting (Zhang and Wu, 2024; Ke et al., 2023; Jang et al., 2022) while fine-tuning on new domain-specific data. However, since we do not have access to the original instruction-tuning data of the target LLM, we introduce a self-selective rehearsal replay buffer (Gupta et al., 2024; Huang et al., 2024).

This approach differs from traditional replay buffers in that it generates new outputs for old inputs. Using the technique introduced in (Sudalairaj et al., 2024), we generate diverse inputs that belong to different categories, such as code, math, reasoning, extraction, safety, writing *etc.*. We then pass each input (x) through the LLM and use (x,LLM(x)) as an auxiliary dataset during the fine-tuning process.

By combining this self-selective replay buffer with the current synthetic QA data, we ensure that the model retains general knowledge while fine-tuning on domain-specific tasks, thereby mitigating catastrophic forgetting.

5 Experimental Setup

5.1 Objectives

The main objectives of our experiments are to:

1) demonstrate that PA-RAG successfully injects new knowledge in LLM while preserving its generic capabilities and show that it performs better than the other fine-tuning methods in both dimensions – injecting new knowledge and preserving generic capabilities.
2) demonstrate the importance of each of the components of PA-RAG, *viz.*, data identifier, replay-buffer, and answer multiplicity.
3) systematically analyze the issues with RAFT (Zhang et al., 2024a), *viz.*, contextual overfitting and canonical answer overfitting.

All the code and test datasets used for our experiments are available on GitHub. [3]

5.2 Base model and Datasets

We use Mistral-7B-Instruct-v0.2[4] as our base LLM and inject new knowledge in it using LoRA(Hu et al., 2022) tuning. To demonstrate the capability of PA-RAG to inject new knowledge, we would need information published after the cutoff date of the base model. Accordingly, we chose two Redbooks[5] published on 23rd July and 16th May 2024 as our two different corpora. We use LlamaIndex[6] to parse the PDFs into markdown and extract the text from the 5 chapters of the first book and the 6 chapters of the second book, respectively.

Base data: We consider each chapter as a document and arbitrarily set $T = 8000$ to split the chapters into chunks for creating synthetic QAs. Using base–data prompt P_1, we get a total of 5122 and 33570 QA pairs for book1 and book2, respectively, and we randomly split them into train/val/test splits. As a post-facto analysis, we compute the coverage of each chapter by the generated QA pairs and obtain 84.4% and 82.5% average coverage at the token level by the train data for book1 and book2, respectively.

Answer multiplicity in training data: We use multiplicity prompt P_2 to generate multiple answers for each question in the base dataset. We restrict the number of answers per question to 5 and get 4.6 answers per question on average in the training data of both books. Generating multiple answers also results in an increase in the token-

---

[3] https://github.com/kushagrabhushan/Systematic-Knowledge-Injection

[4] mistralai/Mistral-7B-Instruct-v0.2

[5] 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

[6] LlamaIndex

level coverage to 93.6% and 92.4%, respectively, for the two books.

Test Sets: Along with the base data test set, we also created an additional factoid test set consisting of questions with up to eight words as answers. In the base data test set, answers often consist of multiple sentences and can be expressed in various paraphrased forms, making evaluation challenging. In contrast, factoid answers have a limited set of acceptable responses, making it easier to assess model performance. Evaluating models on this factoid test set provides a more direct measure of how many facts from the corpus have been injected into the model. Additionally, we observed that several questions in the test sets contained phrases such as "in the above passage" and "in the given context." To address this, we removed such contextual questions by prompting a large language model (LLM) to filter them out. Table 19 provides the exact prompt for filtering the test set. All results in the paper are on the filtered base and factoid test set, with results on the factoid dataset presented in Appendix C. Please see Table 5 in the Appendix A.1 for detailed statistics of all the training and test datasets.

5.3 Training Details:

Please see Appendix A.2 for training details.

5.4 Evaluation criteria

We evaluate all the methods under the RAG setup, *i.e.*, we fetch the top 5 passages that are similar to the question and provide them as context along with the question. To test under a realistic scenario, instead of setting up the index on passages from only book1 and book2, we downloaded 34 red papers [7] and indexed all of them. This results in the indexing of a total of 4,765 passages, each with 512 tokens. We use LlamaIndex with BGE Embedding (Xiao et al., 2023) to create the index. See Table 16 for the exact prompt used while fine-tuning.

Following (Adlakha et al., 2023), we use token level Recall *w.r.t.* the gold response as an automated metric to evaluate the generated response. In addition, we prompt Mixtral-8x22B-Instruct-v0.1[8] and LLaMA-3.3-70B-Instruct[9] to identify if the predicted answer conveys the same message as gold while answering the question or not. See Table 18 for the prompt used. We report these metrics for the test split of base data. To quantify catastrophic forgetting and model's generic reasoning capabilities, we use a diverse set of standard benchmarks, *viz.*, MMLU (Hendrycks et al., 2021), GSM8k (Cobbe et al., 2021), Hellaswag (Zellers et al., 2019), and, TruthfulQA (Lin et al., 2022).

5.5 Baselines

We compare PA-RAG against the following baselines. The first three are inspired by Zhang et al., and the last one is an augmented version of RAFT.

1) Domain-specific fine-tuning (DSF): We train the base LLM using base data to generate a response to a question without accessing any retrieved passages. Since the QAs in the training data cover most of the corpus content, the LLM should be able to answer test queries, provided it has injected the knowledge shown during fine-tuning. See Table 17 for the prompt used.
2) DSF+RAG: We prompt the DSF model with the top 5 retrieved passages along with the question. The prompt used here is the same as in the other baselines and PA-RAG (Table 16).
3) RAFT($\mathrm{p}*$): method proposed in (Zhang et al., 2024a), trained using base data. We treat corruption probability p in RAFT as a hyper-parameter and select amongst $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ based on the validation loss.
4) CA-RAFT($\mathrm{p}*$): Here we augment the training data of RAFT(p) with RAFT(0) and RAFT(1). As earlier, we treat corruption probability p as a hyper-parameter and select the optimum $\mathrm{p}*$ based on validation loss.

6 Experimental Results

6.1 Comparison with baselines

Table 1 compares the performance of our method with various baselines. PA-RAG performs the best in both dimensions: knowledge injection (measured by recall and LLM Judge), as well as maintaining the base model's general capabilities, as measured by the drop in average regression score. We observe that the base LLM performs better than most of the baselines when the retrieved passage overlaps with the gold document (we call it "some overlap" subset), demonstrating that the instruction-tuned Mistral has decent reading comprehension capabilities. However, when the retrieved passages do not overlap with the gold document ("No overlap" subset), the model has to recognize this and then answer from its memory. We observe that all

---

[7] `https://www.redbooks.ibm.com/`

[8] mistralai/Mixtral-8x22B-Instruct-v0.1

[9] meta-llama/Llama-3.3-70B-Instruct

	Overall			No Overlap			Some Overlap			Reg. Score
	Recall	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	
Book 1										
Base+RAG	68.9	80.7	79.3	57.7	61.1	56.4	76.1	93.4	94.1	56.4
DSF	71.1	84.0	83.3	72.3	83.8	81.9	70.3	84.1	84.1	-5.7
DSF+RAG	68.1	80.9	82.7	68.0	71.9	79.5	68.2	86.8	84.8	-5.7
RAFT	70.6	82.8	84.0	67.0	76.0	78.4	73.0	87.2	87.6	-5.4
CA-RAFT	72.5	85.2	83.9	69.2	80.2	78.3	74.6	88.4	87.5	-6.4
PA-RAG	**77.6**	**95.3**	**93.6**	**73.4**	**91.0**	**89.8**	**80.3**	**98.1**	**96.1**	**-1.2**
Book 2										
Base+RAG	69.5	67.8	67.5	67.1	58.6	58.9	71.0	73.2	72.5	56.4
DSF	68.2	70.3	71.5	68.1	69.4	70.7	68.2	70.7	71.9	-5.6
DSF+RAG	66.0	65.8	66.3	64.0	61.7	61.9	67.2	68.3	68.9	-5.6
RAFT	70.9	76.2	77.2	70.0	73.1	75.1	71.4	78.0	78.5	-4.8
CA-RAFT	71.4	78.2	78.0	70.2	77.0	74.8	72.2	78.9	79.9	-5.6
PA-RAG	**76.4**	**86.2**	**85.7**	**75.9**	**83.7**	**82.8**	**76.6**	**87.7**	**87.4**	**-1.8**

Table 1: Comparing PA-RAG with various baselines defined in Section 5.5. **Overall**: performance over the entire test set; **No overlap**: the subset of test split where retriever fails; **Some overlap**: subset where the retriever fetches at least one passage from the gold document; **Reg. Score**: average performance on various benchmarks (GSM8k, Hellaswag, MMLU, and TruthfulQA). Mixtral-J and LLaMA-J stand for Mistral-8x22B-Instruct-v0.1 and Llama-3.3-70B-Instruct as judges, respectively. For the base model(Mistral-Instruct-v2), we show the average regression score, and for others, we show the drop from the base model. For RAFT, the optimal corruption probability, $\mathrm{p}* = 0.6$ for both books; For CA-RAFT, $\mathrm{p}* = 0.2, 0.4$, and for PA-RAG $\mathrm{p}* = 0.4, 0.6$, for book1 and book2, respectively.

the baselines perform poorly in such a scenario. On the other hand, PA-RAG beats all the baselines in both cases, demonstrating that: 1. When the retriever succeeds, PA-RAG can identify this and leverage the retrieved information to respond. 2. When the retriever fails, PA-RAG can ignore the retrieved information and answer correctly from its own parametric memory.

Next, we observe that CA-RAFT performs better than RAFT, confirming the presence of conditional memorization bias. Recall that such a bias occurs due to the static assignment of each question to retriever success or failure case during training. In CA-RAFT, when we augment RAFT(p) with RAFT(0) and RAFT(1), we ensure that each question is seen with and without correct passages during training. This forces the model to learn that irrespective of the question, it has to leverage the information present in the context only when it is correct and rely on its own parametric knowledge when the retriever fails.

Next, we observe that using a replay buffer, data identifier, and multiple answers for the same question significantly improves the model's performance without impacting its generic capabilities. We note that PA-RAG implicitly takes care of conditional memorization bias. In Section 6.3, we empirically validate that this is indeed the case.

6.2 Ablations

To understand the value addition by different components of PA-RAG, we run ablations by removing each of the components and retraining the model. Specifically, we ablate on replay buffer, Data Identifier (DI), and multiple answers (Multi Ans.). Table 2 shows the results. For each ablation, we show the change in the metric *w.r.t.* PA-RAG.

We first observe that removing the replay buffer has a significant impact on the average regression score that measures the model's generic capabilities (a drop of 1.9 from PA-RAG and an overall drop of 3.1 from the base model). Interestingly, removing the replay buffer negatively impacts the model's performance on the "some overlap" subset. This demonstrates that the replay buffer helps in bolstering the reading comprehension capabilities of the model as well. On the other hand, performance on the "no overlap" subset improves slightly, demonstrating that the reduced burden of remembering existing skills helps in memorizing and recalling new information.

Next, we observe that removing the Data Identifier (DI) results in an overall drop in the model's

	Overall			No Overlap			Some Overlap			Reg. Scores
	Recall	Mixtral-J	LLaMA-J	Recall	Mixtral-J	LLaMA-J	Recall	Mixtral-J	LLaMA-J	
PA-RAG	77.6%	95.3%	93.6%	73.4%	91.0%	89.8%	80.3%	98.1%	96.1%	55.2
- DI	76.5%	94.6%	94.8%	72.3%	92.2%	90.3%	79.1%	96.1%	97.6%	-0.1
- Replay Buffer	77.0%	96.0%	94.8%	73.6%	92.8%	91.6%	79.2%	98.1%	96.9%	-1.9
- Multi Ans.	72.0%	84.9%	84.6%	65.8%	79.0%	78.9%	76.0%	88.8%	88.3%	0.6

Table 2: Ablation results on book 1. The first row has absolute performance metrics of PA-RAG whereas the remaining rows show the change in the metric by removing the corresponding component from PA-RAG. DI: Data Identifier; Multi Ans: multiple answers. Mixtral-J and LLaMA-J stand for Mistral-8x22B-Instruct-v0.1 and Llama-3.3-70B-Instruct as judges, respectively.

Ch.	Method	Overall	No overlap	Some overlap
1-3	Ch.w. bucket	74.1	71.3	76.0
	PA-RAG	76.3	70.9	79.9
4-5	Ch.w. bucket	74.8	68.1	79.1
	PA-RAG	78.4	75.0	80.5

Table 3: Comparing chapter-wise assignment into 'retriever success' and 'retriever failure' buckets during training with PA-RAG that does random assignment.

performance. The drop is more prominent over "no overlap" subset of the test data, demonstrating that the DI helps in injecting and recalling the domain knowledge when the model has to answer from its own parametric memory.

Finally, we observe that using multiple paraphrases of the same answer significantly impacts the performance – recall drops by 5.65 pts from 77.0. As expected, dropping multiple answers has no negative impact on the model's generic capabilities.

6.3 Analysis of conditional memorization bias

In this section, we empirically demonstrate: 1). Conditional memorization bias exists, and it can significantly hamper the learning process, and 2). Answer multiplicity during training implicitly results in context augmentation when we randomly assign each augmented QA pair to either of the two buckets ('retriever success' or 'retriever failure').

To do so, we systematically assign each QA pair to the 'retriever success' and 'retriever failure' buckets while training. In the first analysis, all QA pairs from chapters 1 to 3 from book 1 are assigned to the 'retriever failure' bucket, and all QA pairs from chapters 4 and 5 are assigned to the 'retriever success' bucket. Table 3 separately compares the token level recall for the test queries from chapters 1-3 and chapters 4-5. We observe that when there is no overlap between the retrieved passages and the gold documents, the performance over questions from chapters 4-5 is significantly poor for the chapter wise bucketing method. This happens because during training, all questions from chapters 4-5 had relevant context, and thus the model relied only on the given context to answer questions from chapters 4-5. Hence, when provided with irrelevant passages during testing, the model fails to recall that information and thus performs poorly. We do not see any such pattern for questions from chapters 1-3. Interestingly, when relevant information is provided during testing for questions from chapters 1-3, the model fails to leverage that, demonstrating that it learnt to always ignore the context when the question is from chapters 1-3.

Recall that our PA-RAG finetunes LLM using $\mathcal{D}_{aug.}$ that has multiple QA pairs for the same question. We show that in such a scenario, random assignment of train QA pairs to the 'retriever success' or 'retriever failure' bucket alleviates the need for explicit context augmentation in PA-RAG. To do so, we systematically assign all QA pairs with the same question to the same bucket. For a question, the 'retriever failure' bucket is chosen with probability $\mathrm{p} = 0.4$. In such a setup, we have systematically removed the context augmentation but have kept multiple answers. Table 4 compares the token level recall of question-wise bucketing with PA-RAG that has QA-wise bucketing. We observe that the overall token level recall worsens with question-wise bucketing. Notice that the gap is wider (2.5 pt) when there is no overlap between retrieved passages and the gold document for a test question. This demonstrates that the model struggles to either inject all the knowledge or recall it when required. We attribute this to the lack of context augmentation in question-wise bucketing.

	Overall	No overlap	Some overlap
Q.w. bucket	76.2	71.1	79.5
PA-RAG	77.6	73.4	80.3

Table 4: Comparing question-wise assignment into 'retriever success' and 'retriever failure' buckets during training with PA-RAG that randomly assigns QA pairs.

6.4 Effect of model size and model family

All the results presented above are obtained by fine-tuning a Mistral-7B model. To evaluate the robustness of our method to variations in model size and architecture, we train two additional models from the LLaMA family, LLaMA-2-7B-Chat[10] and LLaMA-2-13B-Chat[11]. Table 8 reports our findings. In both cases, we observe trends similar to Mistral-7B, demonstrating that PA-RAG is robust to varying architectures and model sizes.

6.5 Effect of ingesting already seen knowledge

We conduct additional experiments using a small subset (50 documents and 50 queries) of the recently proposed CLAPNQ dataset (Rosenthal et al., 2025). It is derived from NQ dataset (Kwiatkowski et al., 2019) and consists of a human-annotated corpus of actual user queries and corresponding answers from Wikipedia articles. Mistral may not have seen the specific QA pairs during training, but it likely encountered the underlying information from Wikipedia pages in various paraphrased forms. As a result, there is no new knowledge to be ingested, and we do not expect our method to yield significant performance gains over the base model. See table 13 for the results.

Surprisingly, RAFT performs worse than the base model, while PA-RAG's performance remains at par with the base model. We hypothesize two causes for this: First, RAFT is trained using only a single answer per question, which may result in overfitting to canonical answers and losing its ability to recall knowledge already seen during pretraining. In contrast, PA-RAG trains on multiple answers and varied contexts, preserving and leveraging this ability. Second, our synthetically generated QAs cover only 49% of a document on average. However, it is plausible that the base model would have seen the remaining 51% during pretraining, giving it an edge. RAFT results in overfitting to 49% of the information, whereas PA-RAG generalises better by retaining the preexisting knowledge and skills due to the replay buffer.

7 Conclusion

We introduced PA-RAG, a novel framework designed to enhance the fine-tuning process for domain-specific RAG tasks. By incorporating context augmentation and answer multiplicity through paraphrasing, our approach effectively mitigates both conditional memorization bias and canonical answer overfitting. It results in a more adaptable LLM that is robust to retriever errors on domain-specific questions. We use a novel replay buffer technique along with a data identifier that mitigates catastrophic forgetting. Our experimental results demonstrate that PA-RAG outperforms baseline methods, offering a promising direction for improving domain-specific knowledge injection in LLMs.

8 Limitations

While PA-RAG demonstrates promising results in reducing catastrophic forgetting, it does not completely eliminate the issue. The retained parametric knowledge may still degrade when new domain-specific information is introduced due to domain-specific fine-tuning. Additionally, the effectiveness of PA-RAG depends on a strong LLM to generate high-quality QA pairs from domain documents. If the LLM struggles to generate accurate or contextually rich pairs, the overall performance may be affected. Moreover, the enhancements introduced by PA-RAG such as paraphrased answer augmentation and dynamic retrieval simulation, require additional computational resources, which can be a concern when injecting large-scale domain data. Despite these limitations, PA-RAG shows considerable promise, though further work is needed to address these challenges and make it more robust and widely applicable.

Acknowledgements

We thank anonymous reviewers for their insightful comments and suggestions that helped in further improving our paper. We also thank Meghanadh Pulivarthi for reviewing our code independently and Ojas Gramopadhye for helping out with many teething issues during the experimental setup.

---

[10] meta-llama/Llama-2-7b-chat-hf

[11] meta-llama/Llama-2-13b-chat-hf

References

Vaibhav Adlakha, Parishad BehnamGhader, Xing Han Lu, Nicholas Meade, and Siva Reddy. 2023. Evaluating correctness and faithfulness of instruction-following models for question answering. *ArXiv preprint*.

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*.

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. 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*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *ArXiv preprint*.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, (4).

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Sonam Gupta, Yatin Nandwani, Asaf Yehudai, Mayank Mishra, Gaurav Pandey, Dinesh Raghu, and Sachindra Joshi. 2024. Selective self-rehearsal: A fine-tuning approach to improve generalization in large language models. *ArXiv preprint*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proc. of ICML*, Proceedings of Machine Learning Research.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *Proc. of ICLR*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

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*.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. 2024. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. *ArXiv preprint*.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. 2022. Towards continual knowledge learning of language models. In *Proc. of ICLR*.

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).

Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, Xiaojian Jiang, Jiexin Xu, Li Qiuxia, and Jun Zhao. 2024. Tug-of-war between knowledge: Exploring and resolving knowledge conflicts in retrieval-augmented language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*.

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, 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. In *The Twelfth International Conference on Learning Representations*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

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*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proc. of ACL*.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. Radit: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Yanming Liu, Xinyue Peng, Xuhong Zhang, Weihao Liu, Jianwei Yin, Jiannan Cao, and Tianyu Du. 2024. RA-ISF: Learning to answer and understand from retrieval augmentation via iterative self-feedback. In *Findings of the Association for Computational Linguistics ACL 2024*.

Nick Mecklenburg, Yiyou Lin, Xiaoxiao Li, Daniel Holstein, Leonardo Nunes, Sara Malvar, Bruno Silva, Ranveer Chandra, Vijay Aski, Pavan Kumar Reddy Yannam, et al. 2024. Injecting new knowledge into large language models via supervised fine-tuning. *ArXiv preprint*.

Jisoo Mok, Jaeyoung Do, Sungjin Lee, Tara Taghavi, Seunghak Yu, and Sungroh Yoon. 2023. Large-scale lifelong learning of in-context instructions and how to tackle it. In *Proc. of ACL*.

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.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*.

Sara Rosenthal, Avirup Sil, Radu Florian, and Salim Roukos. 2025. Clapnq: C ohesive l ong-form a nswers from p assages in natural questions for rag systems. *Transactions of the Association for Computational Linguistics*, 13:53–72.

Devendra Singh Sachan, Siva Reddy, William L. Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. In *Proc. of EMNLP*.

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

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. *Transactions of the Association for Computational Linguistics*.

Shivchander Sudalairaj, Abhishek Bhandwaldar, Aldo Pareja, Kai Xu, David D Cox, and Akash Srivastava. 2024. Lab: Large-scale alignment for chatbots. *ArXiv preprint*.

Boxin Wang, Wei Ping, Lawrence McAfee, Peng Xu, Bo Li, Mohammad Shoeybi, and Bryan Catanzaro. Instructretro: Instruction tuning post retrieval-augmented pretraining. In *Forty-first International Conference on Machine Learning*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2023. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*.

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

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making retrieval-augmented language models robust to irrelevant context. In *The Twelfth International Conference on Learning Representations*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proc. of ACL*.

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

Xiao Zhang and Ji Wu. 2024. Dissecting learning and forgetting in language model finetuning. In *The Twelfth International Conference on Learning Representations*.

Xiaoying Zhang, Baolin Peng, Ye Tian, Jingyan Zhou, Yipeng Zhang, Haitao Mi, and Helen Meng. 2024b. Self-tuning: Instructing llms to effectively acquire new knowledge through self-teaching. *ArXiv preprint*.

Junhao Zheng, Shengjie Qiu, and Qianli Ma. 2024. Learn or recall? revisiting incremental learning with pre-trained language models. In *Proc. of ACL*.

A Experimental Details

A.1 Data statistics

See Table 5 for detailed statistics of all the training and test datasets used in our experiments.

	#Chapters/ Pages	#Trn	#Val	Test (Raw)	Test (Filtered)	%cover. trn	# Aug. trn.	%cover. aug.
Book1	5 / 76	4,122	515	515	425	84.4	18,986	93.6
Book2	6 / 158	27,056	3,382	3,132	2,269	82.5	126,213	92.4

Table 5: Data statistics. #Trn: number of training QA pairs in base data; #Val: number of validation QA pairs. #Test (Raw): number of test QA pairs before filtering. #Test (Filtered): number of test QA pairs after filtering. %cover. trn: average coverage of the chapters by the train QA pairs in the base data. It is computed as token level overlap between the chapter and all QA pairs from that chapter. #Aug. trn.: number of training QA pairs in the augmented train data. %cover. aug.: average coverage of the chapters by the train QA pairs in the augmented base data.

A.2 PA-RAG training details

We use Huggingface's SFTTrainer[12] to finetune the base model separately for book1 and book2 using LoRA adapters with rank 16 and 32, respectively. We experimented with the learning rates of $1e-5$ and $1e-4$ and selected $1e-4$ based on validation set performance. We train for 400 and 1200 steps for book1 and book2, respectively, with an effective batch size of 256 (16 gradient accumulation steps on 4 A100 80GB GPUs with 4 batch-size on each GPU) and select the best checkpoint based on validation loss. Training time for the two models is less than 5 and 15 hours, respectively.

A.3 Effect of domain identifiers and replay buffer on catastrophic forgetting

	GSM8k		HellaSwag	MMLU	TQA		TQA_Gen
	Flexible	**Strict**	**Acc**	**Acc**	**TQA_MC1 (acc)**	**TQA_MC2 (acc)**	**RougeL (acc)**
Base Model	41.9	41.6	66.0	59.0	52.4	66.8	54.1
DSF	35.7	35.4	63.9	57.6	39.0	56.1	48.7
DSF + DI	37.1	36.8	64.2	57.7	39.8	57.0	49.3
DSF + RB	42.8	42.3	64.8	59.2	47.7	65.2	55.2
DSF + DI + RB	41.7	41.2	64.8	58.7	48.1	65.7	54.5

Table 6: Regression Scores For Book 1: Performance of the training methods discussed in paper on various benchmark tasks using Mistral-7B-v0.1. **Flexible, Strict** are the different criteria for assessing the exact match metric. **TQA** refers to the TruthfulQA benchmark, where **MC1** is the subset of multiple choice question with one answer and MC2 is the subset of questions with more than one answer

Here, we demonstrate the effectiveness of both Domain Identifiers and Replay Buffer to mitigate catastrophic forgetting. Tables 6 and 7 report the individual regression scores for each task. We use github repo(Gao et al., 2024) to compute all the scores. For GSM8k, we take the average of Flexible and Strict Match. For TruthfulQA (TQA), we take the average accuracy over MC1 and MC2. We consider TruthfulQA_Gen separately from MC1 and MC2 and use RougeL to quantify the performance. Reported average score is the average of these five scores.

B Experimental Analysis of LLaMA Models

Tables 8 and 9 show the results of , LLaMA-2-7B-Chat and , LLaMA-2-13B-Chat respectively on both book 1 and book 2. We see in both the cases that PA-RAG outperforms all baselines, proving its robustness to model architecture and size.

---

[12] docs/trl/sft_trainer

	GSM8k		HellaSwag	MMLU	TQA		TQA_Gen
	Flexible	**Strict**	**Acc**	**Acc**	**TQA_MC1 (acc)**	**TQA_MC2 (acc)**	**RougeL (acc)**
Base Model	41.9	41.6	66.0	59.0	52.4	66.8	54.1
DSF	37.6	37.3	62.8	55.6	37.6	55.1	51.7
DSF + DI	36.5	36.3	62.8	56.5	40.4	57.8	53.0
DSF + RB	41.0	40.7	64.1	58.9	48.3	64.5	54.0
DSF + DI + RB	40.7	40.4	64.2	58.4	48.7	65.0	53.2

Table 7: Regression Scores for Book 2: Performance of the training methods discussed in paper on various benchmark tasks using Mistral-7B-v0.1. **Flexible, Strict** are the different criteria for assessing the exact match metric. **TQA** refers to the TruthfulQA benchmark, where **MC1** is the subset of multiple choice question with one answer and MC2 is the subset of questions with more than one answer

	Overall			No Overlap			Some Overlap			Reg. Scores
	Recall	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	
					Book 1					
Base+RAG	69.7	73.2	70.9	60.7	53.3	48.8	75.5	86.0	84.9	41.9
DSF	66.0	72.9	74.2	67.0	70.7	73.5	65.3	74.4	74.7	-3.1
DSF+RAG	57.6	64.9	68.8	56.6	59.3	64.8	58.3	68.6	71.4	-3.1
RAFT	68.9	77.2	78.5	65.4	68.3	69.3	71.2	82.9	84.4	-1.9
CA-RAFT	71.8	79.5	78.8	67.1	70.7	70.7	74.8	85.3	84.0	-2.9
PA-RAG	**73.7**	**89.6**	**88.7**	**69.7**	**83.8**	**83.8**	**76.3**	**93.4**	**91.8**	-0.2
					Book 2					
Base+RAG	70.9	60.2	57.9	68.0	50.7	49.6	72.7	65.7	62.8	41.9
DSF	67.7	68.0	69.0	67.4	66.1	66.2	67.9	69.1	70.7	-2.9
DSF+RAG	55.3	54.3	55.9	55.1	49.6	52.8	55.4	57.0	57.7	-2.9
RAFT	69.0	70.5	71.1	68.0	66.5	68.2	69.6	72.8	72.8	-2.9
CA-RAFT	70.5	72.7	73.1	69.9	70.8	70.3	70.9	73.8	74.7	-4.2
PA-RAG	**73.7**	**82.4**	**82.0**	**73.2**	**78.6**	**78.1**	**74.1**	**84.7**	**84.3**	-0.7

Table 8: Comparison of , LLaMA-2-7B-Chat trained using PA-RAG against baselines mentioned in 5.5.

	Overall			No Overlap			Some Overlap			Reg. Scores
	Recall	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	
					Book 1					
Base+RAG	70.8	74.4	73.5	61.6	54.5	54.2	76.7	87.2	86.0	45.4
DSF	70.2	78.6	78.9	70.2	74.3	73.7	70.2	81.4	82.4	-1.9
DSF+RAG	58.3	65.2	71.4	56.6	58.1	66.5	59.3	69.8	74.6	-1.9
RAFT	70.3	79.3	79.4	66.0	69.5	71.7	73.1	85.7	84.4	-1.1
CA-RAFT	72.7	83.5	83.6	67.5	74.3	77.0	76.1	89.5	87.8	-1.9
PA-RAG	**77.1**	**92.0**	**92.0**	**73.6**	**89.2**	**89.8**	**79.4**	**93.8**	**93.4**	-1.8
					Book 2					
Base+RAG	72.9	63.1	62.4	70.2	53.9	53.5	74.4	68.4	67.6	45.4
DSF	67.5	68.8	69.9	67.7	68.7	69.7	67.4	68.9	70.1	-1.9
DSF+RAG	59.3	55.7	59.8	58.6	52.2	56.6	59.7	57.8	61.6	-1.9
RAFT	68.7	71.4	72.7	67.7	67.9	69.7	69.3	73.5	74.4	-1.5
CA-RAFT	68.8	72.6	73.3	67.4	68.9	68.9	69.6	74.8	75.9	-2.3
PA-RAG	**74.7**	**83.6**	**83.3**	**74.0**	**81.0**	**80.4**	**75.1**	**85.2**	**85.0**	-0.3

Table 9: Comparison of , LLaMA-2-13B-Chat trained using PA-RAG against baselines mentioned in 5.5.

	Overall			No Overlap			Some Overlap		
Book 1									
	Recall	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**
Base+RAG	56.1	73.0	69.7	30.1	50.0	44.0	76.1	93.4	87.6
DSF	69.2	83.4	81.0	66.4	79.9	77.5	70.3	84.1	83.4
DSF+RAG	66.1	80.7	79.5	56.0	71.6	67.9	68.2	86.8	87.6
RAFT	70.3	84.1	84.1	62.1	75.7	75.0	80.3	98.1	90.5
CA-RAFT	72.5	85.6	85.4	64.4	74.6	76.3	73.0	87.2	**91.7**
PA-RAG	**76.7**	**88.0**	**87.3**	**71.3**	**81.7**	**81.7**	**74.6**	**88.4**	91.3
Book 2									
Base+RAG	56.3	61.8	60.5	46.3	47.2	46.4	62.6	70.9	69.4
DSF	53.9	62.5	60.1	51.4	58.6	56.4	55.5	64.9	62.5
DSF+RAG	57.5	66.1	65.7	51.2	57.9	57.3	61.3	71.2	70.9
RAFT	61.5	71.6	70.7	55.4	64.6	63.9	65.3	76.0	75.0
CA-RAFT	62.2	72.1	71.1	56.9	65.4	64.1	65.6	76.3	75.5
PA-RAG	**68.5**	**80.3**	**79.2**	**64.8**	**75.2**	**73.9**	**70.8**	**83.4**	**82.6**

Table 10: Results of PA-RAG and other baselines using Mistral-7B-v0.1 on the factoid dataset

	Overall			No Overlap			Some Overlap		
Book 1									
	Recall	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**
Base+RAG	52.5	65.6	60.7	26.2	37.3	30.8	71.0	85.5	81.7
DSF	61.5	73.2	73.2	61.0	73.4	71.6	61.9	73.0	74.3
DSF+RAG	53.0	68.5	68.5	46.0	62.7	60.7	57.9	72.6	73.9
RAFT	65.0	79.8	78.8	55.4	70.4	70.4	71.8	86.3	84.6
CA-RAFT	69.6	82.7	81.0	**65.0**	76.9	75.1	72.8	86.7	85.1
PA-RAG	**71.4**	**86.8**	**86.3**	62.3	**79.3**	**79.8**	**77.8**	**92.1**	**90.9**
Book 2									
Base+RAG	53.6	58.5	56.3	43.1	42.9	41.2	60.1	68.3	65.8
DSF	51.5	58.5	57.1	46.8	52.7	51.2	54.4	62.2	60.7
DSF+RAG	50.0	57.1	57.0	42.9	46.6	46.5	54.4	63.7	63.5
RAFT	58.3	67.1	65.4	51.0	57.4	55.2	62.9	73.3	71.8
CA-RAFT	59.2	68.2	66.6	54.0	62.1	59.7	62.5	72.0	70.9
PA-RAG	**62.8**	**73.8**	**73.0**	**57.2**	**65.0**	**65.2**	**66.2**	**79.3**	**77.9**

Table 11: Results of PA-RAG and other baselines using , LLaMA-2-7B-Chat on the factoid dataset

C Results on Factoid Dataset

As mentioned in 5.2, we curated a factoid dataset consisting of QA pairs containing only factoid answers, i.e., short, factual questions. Tables 10, 11 & 12 show the comparisons between PA-RAG and various other baselines, on all three models, Mistral-7B-v0.1 , LLaMA-2-7b-chat-hf and LLaMA-2-13b-chat-hf.

D Results on CLAPNQ Dataset

As mentioned in 6.5, here we show the result of PA-RAG along with Base+RAG and RAFT on the CLAPNQ dataset which is a subset from the Natural Questions dataset. We see in table 13 that token level recall for RAFT is lower than both the baseline and PA-RAG.

	Overall			No Overlap			Some Overlap		
	Book 1								
	Recall	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**
Base+RAG	54.1	65.8	62.7	29.1	41.6	36.1	71.7	82.5	81.3
DSF	67.9	80.7	79.2	67.3	77.5	75.1	68.3	83.0	82.1
DSF+RAG	55.5	69.0	69.3	43.6	58.6	60.4	63.8	76.3	75.5
RAFT	66.9	81.0	81.7	63.7	73.4	75.1	69.1	86.3	86.3
CA-RAFT	70.3	82.7	82.2	67.8	78.7	78.1	72.0	85.5	85.1
PA-RAG	**74.9**	**86.1**	**86.8**	**68.1**	**78.7**	**79.8**	**79.7**	**91.3**	**91.7**
	Book 2								
Base+RAG	56.3	59.8	57.3	45.5	44.3	42.4	63.0	69.5	66.7
DSF	52.9	61.0	59.4	49.5	54.9	53.5	55.0	64.9	63.2
DSF+RAG	47.0	51.7	54.5	41.0	43.6	46.9	50.8	56.8	59.3
RAFT	58.7	67.9	66.7	52.0	58.2	57.1	63.0	74.0	72.8
CA-RAFT	59.1	68.2	66.8	52.7	59.2	57.8	63.1	73.9	72.5
PA-RAG	**66.5**	**77.7**	**77.0**	**61.5**	**69.9**	**69.7**	**69.7**	**82.6**	**81.6**

Table 12: Results of PA-RAG and other baselines using , LLaMA-2-13B-Chat on the factoid dataset

	Overall			No Overlap			Some Overlap			Reg. Scores
	Recall	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	**Recall**	**Mixtral-J**	**LLaMA-J**	
Base+RAG	46.7	**78.0**	**78.0**	34.8	65.4	69.2	**59.6**	**91.7**	**87.5**	56.4
RAFT	35.7	74.0	**78.0**	25.1	69.2	73.1	47.1	79.2	83.3	-8.4
PA-RAG	**47.7**	**78.0**	**78.0**	**40.4**	**80.8**	**80.8**	55.6	75.0	75.0	-1.2

Table 13: Comparison of PA-RAG with base model and RAFT on the CLAPNQ dataset to see the effect of trying to ingest already known knowledge.

E Prompts

This section enumerates all the prompts that are used in our experiments. We use LLaMA-3.3-70B-Instruct[13] as judge and employ Mixtral-8x22B-Instruct-v0.1[14] both for data generation and as a judge to evaluate the fine-tuned model's predicted answers against the ground truth.

E.1 Prompt for generating question-answer pairs

The prompt in Table 14 instructs the model to create question-answer pairs from a given document, ensuring that the questions do not use co-referencing or pronouns and that they are fully contextualized.

> <s> [INST] 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. [/INST]

Table 14: Prompt for generating question-answer pairs from a document.

---

[13] https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct

[14] https://huggingface.co/mistralai/Mixtral-8x22B-Instruct-v0.1

E.2 Prompt for generating multiple answers

The prompt in Table 15 enables the generation of multiple distinct answers for a question based on the provided document, encouraging diverse answer styles and formats.

```
<s> [INST] You are provided with a document within <document> and </document> tags. Followed by the document, you are provided with a question within <question> and </question> tags. Using only the information provided in the document, generate multiple correct, complete and comprehensive answers for the given question, varying the style and format. Each of your answers should be within separate <answer> and </answer> tags so that I can parse and extract them using a code. Each answer must be complete, correct, comprehensive and only from the provided document.
Start by first generating <answer> token. Then generate one valid answer following any one style or format. Then generate </answer> token. Then again generate <answer> token, followed by another valid answer. Then generate </answer> token. Keep on repeating this till you generate as many correct answers as possible. In the end, generate </done> token once you are done generating all the answers in all possible styles and formats.
Some example styles and format that you must use:
1. List Answer: Answer can be in the form of a list.
2. Extractive answer: Answer with extracted sentences from the document.
3. Inferential answer: Answer that summarizes the information in the document but is not directly quoted.
4. Definitions: Answer that gives definitions related to the question.
5. Examples: Providing examples that can be valid responses to the question.
Ensure that the answers are not just paraphrases but differ in their content and are factually correct based on the document. [/INST]
```

Table 15: Prompt for generating multiple answers to a question from a document.

E.3 Prompt used while fine-tuning

Table 16 shows the prompt used while fine-tuning PA-RAG using RAFT and PA-RAG. Table 17 shows the prompt for the DSF baseline that trains the model to generate a response to a given question without any retrieved passages in the context.

```
<s> [INST] You are an AI assistant who is provided with a conversation between a user and an agent. User utterances start with "User:" and agent utterances start with "Agent:". Your task is to generate agent's response to the last user utterance. Enclosed within <passage> tags, you will find various excerpts. These passages may or may not contain the answer. You may or may not use the information in them to generate your response. Present your response within <response> and </response> tokens.
<passage_0> {document_0} </passage_0>
<passage_1> {document_1} </passage_1>
<passage_2> {document_2} </passage_2>
<passage_3> {document_3} </passage_3>
<passage_4> {document_4} </passage_4>
User: {data_identifier} {question}
[/INST]
### Answer:
```

Table 16: Prompt used while fine-tuning the base model. {document_i} $\forall i = 1 \cdots 5$ are replaced by the retrieved passages; {question} is replaced by the user question. For both book1 and book2, {data_identifier} is replaced by "This query is with reference to IBM Redbooks".

```
<s> [INST] You are an AI assistant who is provided with a conversation between a user and an agent.
User utterances start with "User:" and agent utterances start with "Agent:". Your task is to generate agent's
response to the last user utterance. Present your response within <response> and </response> tokens.
User: {data_identifier} {question}
[/INST]
### Answer:
```

Table 17: Prompt used during Domain-specific fine-tuning (DSF) using only QA pairs without any passages; {question} is replaced by the user question. For both book1 and book2, {data_identifier} is replaced by "This query is with reference to IBM Redbooks".

E.4 Prompt for LLM-as-Judge evaluation

The prompt in Table 18 is used to evaluate the fine-tuned model's predicted answers against the ground truth answers, with Mixtral-8x22B-Instruct-v0.1 and LLaMA-3 70b serving as the evaluation judge.

```
<s> [INST] You are given a question, the corresponding ground-truth answer and two predictions from
two different AI models. Compare the "Ground-truth answer" with both the predictions to determine
which of them is better at answering the given question.
There should be no contradicting statements in a good prediction. A good prediction may contain extra
information. If the prediction states something as a possibility, treat it as a definitive answer.
A good prediction must contain all the important information presented in the ground truths, but doesn't
have to fully match it word by word.
Consider an incorrect or incomplete prediction as bad.
To make your decision, first read the question and Ground-truth answer carefully. Then compare the two
predictions with the Ground-truth answer in the light of the question.
Start with a crisp explanation and reasoning for your evaluation within <explanation> and </explanation>
tags.
Then, within <score> and </score> tokens, generate your evaluation according to the followin rubric:
If "Prediction 1" is preferred over "Prediction 2", then generate "1 better".
If "Prediction 2" is preferred over "Prediction 1", then generate "2 better".
If both "Prediction 1" and "Prediction 2" are equally good, then generate "both good".
If both "Prediction 1" and "prediction 2" are equally bad, then generate "both bad".
Question: {question}
Ground-truth answer: {gold_response}
Prediction 1: {predicted_response1}
Prediction 2: {predicted_response2} [/INST]
```

Table 18: Prompts for Mixtral-8x22B-Instruct-v0.1-based evaluation. Here, {question}, {ground-truth-answer}, and {predicted-answer} are placeholders that are replaced by the actual question, ground truth answer, and predicted answer, respectively.

E.5 Prompt for filtering test dataset

The prompt in Table 19 is used to filter dependent questions from the test set of the synthetic datasets. We used Mixtral-8x22B-Instruct-v0.1 to do the filtering.

F Example Predictions

F.1 Example where the Retriever Fails

As seen in Table 20, PA-RAG is able to correctly recall the information necessary for answering the question even when the retrieved passages did not contain any relevant information. The RAFT method is

<s> [INST] You are given a question, and your task is to determine whether the question is complete and makes sense in isolation. A complete question should not contain text like "based on given example" or "mentioned in the chapter" because only question text is there to answer it, not the chapter or examples. Based on this, assign a final score of **Complete** or **Incomplete** for the question. Only assign **Incomplete** if the question uses phrases like these: "based on given example" or "mentioned in the chapter".
Scoring:
- **Complete:** The question contains all necessary details.
- **Incomplete:** The question uses phrases like "based on given example", "mentioned in the chapter", "mentioned in the passage", "given in the document", or something similar.

Provide your feedback in the following format:
“‘
Feedback:::
Evaluation: (Provide your rationale for the scoring in detail)
Scoring: (State "Complete" or "Incomplete")
“‘

Important: You must provide values for both ’Evaluation:’ and ’Scoring:’ in your answer.

Now, here is the question:
Question: {question}

Provide your feedback.

Feedback:::
Evaluation: [/INST]

Table 19: Prompt for filtering test data using Mixtral-8x22B-Instruct-v0.1. Here, {question} is a placeholder that will be replaced by the actual question.

expected to perform poorly conditional memorization bias, where as DSF is unable to recall the relevant information.

F.2 Example where the Retriever is Successful

Refer to Table 21 for responses to a question where the retrieved passages contained the gold answer. We see that although all the methods are able to extract some relevant information from the provided context, they are unable to extract all the necessary information. However, PA-RAG captures all necessary information from the retrieved passages demonstrating its effectiveness in utilizing the retrieved data for accurate responses.

Question	How can extended data tables be retrieved in IBM Storage Virtualize?
Gold Answer	Many ls commands in IBM Storage Virtualize can leverage the -gui parameter to retrieve an extended data table similar to the output displayed in the IBM Storage Virtualize GUI. This provides more detailed information for all objects of the requested type and can be helpful for automation and scripting.
Base Model	IBM Storage Virtualize does not directly support the concept of extended data tables as described in the provided passages, which involve creating external tables in Presto using data in IBM Storage Scale directories. Instead, IBM Storage Virtualize is a software-defined storage solution that provides virtualization of physical storage resources, enabling the consolidation and management of multiple storage systems into a single, unified storage pool. To retrieve data from IBM Storage Virtualize, you can use various access methods such as block I/O (iSCSI), file I/O (POSIX compliant), or object storage (S3). The data can then be accessed using appropriate tools or applications based on the chosen access method. If you need to work with data in a distributed file system like HDFS or use Spark for data processing, you may consider using IBM Watson for Data and AI solutions, such as IBM Watson Studio or IBM Watson Machine Learning, which can interact with IBM Storage Scale and provide the required functionality for managing and processing extended data tables.
DSF+RAG	Extended data tables can be retrieved in IBM Storage Virtualize by using the lsdeltas command with the -extended parameter. This command returns more detailed information about the changed objects, including the old and new values of the object attributes.
RAFT	Extended data tables can be retrieved in IBM Storage Virtualize by using the -gui parameter with the lsdrive command. This will retrieve more detailed information about each drive, including its capacity, status, and type.
PA-RAG	To retrieve extended data tables in IBM Storage Virtualize, you can use the -gui parameter in many ls commands. This parameter allows you to retrieve an extended data table similar to the output displayed in the IBM Storage Virtualize GUI. For example, the lsdrive command with the -gui parameter will provide more detailed information about each drive, such as UID, block_size, vendor_id, product_id, FRU_part_number, FRU_identity, RPM, firmware_level, FPGA_level, quorum_id, port_1_status, port_2_status, was_spare, interface_speed, write_endurance_used, transport_protocol, compressed, physical_capacity, physical_used_capacity, effective_used_capacity, date_of_manufacture, protection_enabled, write_endurance_usage_rate, replacement_date, anomaly_detection_active, and so on.

Table 20: An example where the retriever fails and the models have to recall domain-specific information from their parametric knowledge.

Question	What are the options for configuring password policy in IBM Storage Virtualize?
Gold Answer	The following attributes can be defined in password policy in IBM Storage Virtualize: - Minimum password length: 6-64 characters. - Minimum number of: - Uppercase characters: 0-3. - Lowercase characters: 0-3. - Special characters: 0-3. - Digits: 0-3. - History check (0-10) before password reuse. - Password expiry: 0-365 days. - Password expiry warning (0-30 days) which is displayed on CLI at login only. - Password age (1-365 days) which is the minimum age before a password can change.
Base Model	The password policy in IBM Storage Virtualize can be configured through the GUI or CLI. Here are the steps to configure it through the GUI: 1. Select ‘Settings‘ > ‘Security‘. 2. Select the ‘Password Policies‘ tab. 3. Set the required password parameters in the ‘Password creation‘ and ‘Password expiration and account lockout‘ sections. 4. Click ‘Save‘. Alternatively, you can use the ‘chsecurity‘ command in the CLI to change the password policy. For example, to set the minimum password length to 8 characters, you can use the following command: “‘bash IBM_FlashSystem:ITSO:securityadmin > chsecurity -minpasswordlength 8“‘. The ‘chsecurity‘ command also allows you to configure other password policy attributes such as minimum number of uppercase, lowercase, special characters, digits, history check, password expiry, and password expiry warning. . . . Note that if you disable password reset and you do not have local or remote Security Administrator users except superuser, there is no way to restore management access if superuser password is lost.
DSF+RAG	The options for configuring password policy in IBM Storage Virtualize include setting password expiration, password expiry warning, password history check, minimum password length, minimum number of uppercase characters, lowercase characters, special characters, digits, and minimum password age.
RAFT	The options for configuring password policy in IBM Storage Virtualize include setting minimum password length, minimum number of uppercase characters, lowercase characters, special characters, and digits, history check before password reuse, password expiry, password expiry warning, and password age.
PA-RAG	The options for configuring password policy in IBM Storage Virtualize are as follows: - Minimum password length: 6-64 characters. - Minimum number of: - Uppercase characters: 0-3. - Lowercase characters: 0-3. - Special characters: 0-3. - Digits: 0-3. - History check (0-10) before password reuse. - Password expiry: 0-365 days. - Password expiry warning (0-30 days) which is displayed on CLI at login only. - Password age (1-365 days) which is the minimum age before a password can change. These attributes can be set in the GUI by selecting Settings > Security, then selecting the Password Policies tab. The chsecurity command can also be used in the CLI to set the password policy.

Table 21: An example where the retriever is successful and the models must leverage the retrieved information efficiently to answer the questions.

G Licenses

Tool Used	License
LlamaIndex	The MIT License Copyright (c) Jerry Liu
LM Evaluation Harness	MIT License, Copyright (c) 2020 EleutherAI
HuggingFace	Apache License, Version 2.0, January 2004

Table 22: Licenses of the different tools we used for running our experiments.