😈 ImpRAG: Retrieval-Augmented Generation with Implicit Queries

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) systems traditionally treat retrieval and generation as separate processes, requiring explicit textual queries to connect them. This separation can limit the ability of models to generalize across diverse tasks. In this work, we propose a query-free RAG system, named ImpRAG, which integrates retrieval and generation into a unified model. ImpRAG allows models to implicitly express their information needs, eliminating the need for human-specified queries. By dividing pretrained decoder-only language models into specialized layer groups, ImpRAG optimizes retrieval and generation tasks simultaneously. Our approach employs a two-stage inference process, using the same model parameters and forward pass for both retrieval and generation, thereby minimizing the disparity between retrievers and language models. Experiments on 8 knowledge-intensive tasks demonstrate that ImpRAG significantly enhances both retrieval and generation performance, with exact match scores increasing by 3.6-11.5 points and retrieval recalls improving by 5.0-23.2 points for unseen tasks with diverse formats, highlighting its effectiveness in enabling models to articulate their own information needs and generalize across tasks. Our analysis underscores the importance of balancing retrieval and generation parameters and leveraging generation perplexities as retrieval training objectives for enhanced performance.

1 Introduction

Retrieval-Augmented Generation (RAG; Guu et al., 2020; Lewis et al., 2020; Shi et al., 2024) typically involves two key operations: retrieval and generation. RAG systems retrieve relevant information to enhance generation models, enabling them to respond more effectively to prompts by providing long-tail knowledge or up-to-date information. While effective, traditional approaches often treat retrieval and generation as separate processes, connected by queries.[1] Consequently, these approaches usually require explicit specification of textual queries. By definition, queries express one's uncertainties; however, in RAG systems, instead of models expressing their information needs, humans must do this for them. This separation can lead to a disconnect between what large language models

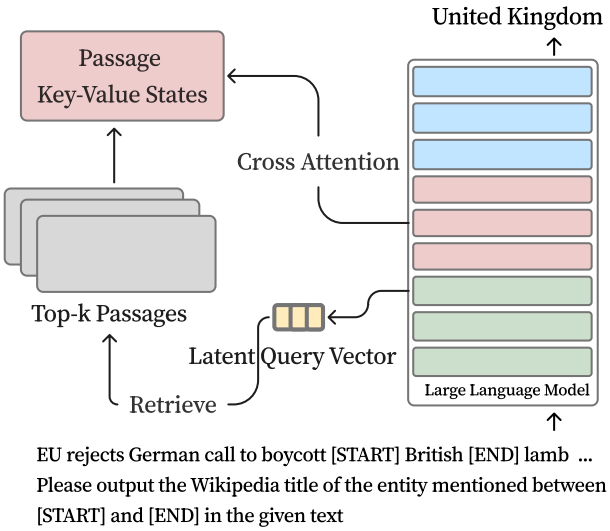

Figure 1: Diagram illustrating the inference process of ImpRAG on the entity linking task. We divide decoder-only LLMs into three layer groups for specialized fine-tuning: bottom (green), middle (red), and top (blue). The bottom layers are optimized for retrieval tasks. The middle and top layers handle the reading of retrieved passages, with cross-attention disabled in the top layers to reduce memory consumption. Standard RAG systems would require a task-specific design of queries (e.g., use the substring "British" as the query in the shown example). In contrast, ImpRAG uses implicit queries, eliminating the need for explicit specification of queries and allowing models to generalize across unseen tasks with varied formats.

[1] In this work, we use the term "queries" to refer to textual queries used in an information retrieval setup, unless otherwise specified. This is distinct from queries in the context of self-attention within Transformer architectures.

(LLMs) require and what retrievers assume is necessary. More importantly, it restricts the models' ability to generalize across diverse, unseen tasks during testing. Therefore, in this work, we explore the development of a *query-free* RAG system, enabling models to articulate their own information needs without additional human intervention.

To achieve this, we introduce ImpRAG, a novel approach that integrates retrieval and generation into a unified model and process. This allows models to convey their own information needs implicitly, reducing the need for prior knowledge of test tasks and for humans to formulate explicit textual queries in advance. At its core, ImpRAG aims to enable retrieval capabilities through retrieval heads in self-attention. Building upon pretrained decoder-only language models, ImpRAG divides the layers into three groups: the bottom group for retrieval and the middle and top groups for reading and generation.

Figure 1 illustrates an example of applying ImpRAG to the entity linking task, where models are tasked with linking the mention "British" to an entity in Wikipedia, given the context paragraph. A typical RAG model would require the design of a separate query template, such as using only the mention text, to achieve reasonable retrieval performance. In contrast, ImpRAG uses implicit queries and can perform retrieval and generation jointly without the need for additional template design, making it more generalizable.

During training, we optimize two objectives simultaneously: generation loss and retrieval loss. The generation loss is the standard causal language modeling loss, while the retrieval loss first utilizes pseudo labels generated by trained retrievers to warm up the retrieval ability and then self-improves using its own generation log likelihood for the remainder of the training.

At inference time, we employ a two-stage process. First, we embed passages using the bottom layer for retrieval, and then utilize the top layer group to read the retrieved passages and generate the final responses. By leveraging the same forward pass and model parameters for both retrieval and generation, ImpRAG reduces the disparity between retrievers and LLMs.

In experiments, we train models on datasets that either require retrieval or do not. The datasets requiring retrieval are used to enhance retrieval performance, while those not requiring retrieval are used to improve models' instruction-following capabilities. We evaluate the models on 8 knowledge-intensive tasks, focusing on different aspects: basic question answering, multihop reasoning, and instruction following. We also establish strong baselines that perform RAG in the retrieve-then-generate paradigm, including RA-DIT (Lin et al., 2024), a method that iteratively updates LLMs and retrievers to better align the two.

Our experiments demonstrate that ImpRAG achieves slightly better performance on 4 tasks with formats similar to the training tasks, with an improvement of 0.2-0.6 points in exact match scores, all without the need for additional model parameters. Moreover, it significantly outperforms previous approaches on unseen test tasks with more diverse formats, achieving improvements of 3.6-11.5 points in exact match scores and 5.0-23.2 points in retrieval recalls. This highlights the effectiveness of enabling models to articulate their own information needs. Our analysis indicates that carefully selecting layer group boundaries that balance the parameters used for retrieval and generation, using both trained retrievers for warmup and then self-improve by leveraging generation perplexities as retrieval training objectives, and instruction tuning training datasets is crucial for achieving superior performance in ImpRAG. Our analysis also reveals that ImpRAG is effective in transferring supervision from generation tasks to retrieval tasks, showing the potential of using an unified model architecture for performing retrieval and generation jointly.

2 Related Work

There has been a lot of work on using the retrieve-then-generate paradigm for RAG (Lewis et al., 2020; Shi et al., 2024, *inter alia*). Many efforts in this line of work have focused on optimizing retrievers using training signals from generation models, and optionally, the reverse (Guu et al., 2020; Lewis et al., 2020; Izacard et al., 2023; Shi et al., 2024). Although the specifics can differ, these approaches generally utilize distinct models and input templates for the retrieval and generation phases. A closely related study is that of Jiang et al. (2022), which seek to use the same model for retrieval and generation. However, their research primarily focuses on encoder-decoder style models and their models still rely on separate input templates for retrieval and generation. Another related work by Zhang et al. (2024a) explores the use of special

tokens for retrieval, but their study emphasizes in-domain task performance rather than unseen task generalization.

This work is also related to research on query formulation in the context of multihop question answering, where previous studies typically generate textual queries by prompting LLMs, followed by retrieval using a separate retriever (Lazaridou et al., 2022; Khattab et al., 2022; Press et al., 2023; Trivedi et al., 2023; Jiang et al., 2023, *inter alia*). Chen et al. (2024) enable LLMs to generate textual queries through synthetic data generation. Additionally, this work is connected to memory architectures in RAG (Yang et al., 2024; Lu et al., 2024), which aim to utilize the key-value (KV) caches of LLMs to reduce computational costs, rather than focusing on minimizing the disparities between generation and retrieval.

Another relevant area of research is instruction tuning for RAG. Lin et al. (2024) perform instruction tuning for both retrievers and LLMs and then align them through iterative updates. Wang et al. (2024) conduct instruction tuning for RETRO-like models (Borgeaud et al., 2022; Wang et al., 2023). Zhang et al. (2024b) align retrievers with LLMs using synthetic data generated by LLMs. Unlike our work, these studies still treat retrieval and generation as separate processes. In a similar vein, researchers have tried to teach retrievers to follow instructions for building general-purpose information retrieval systems (Asai et al., 2023; Lee et al., 2024; Oh et al., 2024; Weller et al., 2025). Since ImpRAG enables its retrieval capabilities by using self-attention, it is related to research on investigating retrieval heads in the context of long context LLMs (Wu et al., 2024).

3 Method

We build on an autoregressive pretrained language model and enable it to perform retrieval and generation jointly. Our model, **ImpRAG**, is based on the LLaMA 3 family (Grattafiori et al., 2024), with architectural modifications to support retrieval and retrieval-augmented generation. At a high level, the layer grouping strategy of ImpRAG is inspired by the observation that LLMs learn distinct functions at different layers (Zhao et al., 2024). Consequently, we have designed the layer groups to align with the capabilities required for retrieval-augmented generation, i.e., retrieval and generation.

3.1 Architecture

Layer Slicing. We partition an N-layer language model vertically into three groups, as illustrated in Figure 1. The bottom group, spanning layers 0 to b, is denoted as $\mathcal{L}_{\mathcal{B}}$. The middle group, from layer b to t, is denoted as $\mathcal{L}_{\mathcal{M}}$, and the top group, from layer $t+1$ to $N-1$, as $\mathcal{L}_{\mathcal{T}}$. Note that $\mathcal{L}_{\mathcal{B}}$ and $\mathcal{L}_{\mathcal{M}}$ share layer b, while $\mathcal{L}_{\mathcal{M}}$ and $\mathcal{L}_{\mathcal{T}}$ are disjoint. The layer boundaries b and t are treated as hyperparameters and can be tuned to optimize performance across different model configurations.

Bottom Layers as Retriever. We repurpose the bottom group $\mathcal{L}_{\mathcal{B}}$ to act as a *retriever*, in addition to its standard decoder functionality. Specifically, we apply pooling last-token pooling over the attention query or key states at the final layer b in $\mathcal{L}_{\mathcal{B}}$. Unlike prior work (Muennighoff et al., 2024), we retain the original causal attention in the bottom layers rather than enabling bidirectional attention, as we do not observe any performance improvement from this modification.

Let h_k be the number of key attention heads, g the number of query attention groups (as in Grouped-Query Attention (Ainslie et al., 2023)), and d_h the head dimension. For a query input, we apply last-token pooling by taking the query attention state of its final token, resulting in a grouped query embedding $\mathbf{E}_q^g \in \mathbb{R}^{(h_k g) d_h}$. We then average the attention heads within each group to obtain the final query embedding $\mathbf{E}_q \in \mathbb{R}^{h_k d_h}$.[2] Similarly, for each corpus passage, we extract the key attention state of its last token to compute the passage embedding $\mathbf{E}_p \in \mathbb{R}^{h_k d_h}$. Similarity between query and passage embeddings is computed via dot product:

$$s(q, p) = \mathbf{E}_q \cdot \mathbf{E}_p \tag{1}$$

We choose to pool over query and key attention states based on the intuition that their dot product underlies the attention mechanism and is pretrained to capture token-wise relevance. By aggregating these signals across tokens, we aim to capture query-passage-level semantic relevance.

Middle Layers as Reader. The middle layer group $\mathcal{L}_{\mathcal{M}}$ functions as a *reader* by enabling cross-attention from the input query tokens to the retrieved passage tokens, thereby incorporating external information into the query representation.

[2] Our preliminary results show that taking average heads works slightly better than using individual heads.

Given k retrieved passages, we jointly encode the concatenation of all k passages to form the key and value states for layers b through t. Cross-attention is then performed from the query's attention states to these key and value states, allowing the model to read and integrate relevant content from the passages. This aligns with prior findings that middle layers of language models are particularly effective at attending to and integrating long-range contextual information (Fang et al., 2024; Yang et al., 2024).

Top Layers Disable Cross-Attention. In the top layer group $\mathcal{L}_{\mathcal{T}}$, we optionally disable cross-attention from the input query tokens to the retrieved passage tokens solely to reduce computational and memory overhead. This design choice is made for efficiency purposes; empirically, we find it results in only a minor performance drop when the layer boundary t is properly tuned as a hyperparameter.

Position IDs. Language models using RoPE (Su et al., 2024) are highly sensitive to position IDs. To prevent interference between the query and passage position encodings during reading, we shift the query's position IDs to the right rather than starting from zero. Let l_{max} denote the maximum passage length and k the number of retrieved passages. We shift the query position IDs by $k \cdot l_{max}$ tokens to account for the total length.

3.2 Training

We train ImpRAG using a multi-task objective that jointly optimizes generation and retrieval:

$$J = J_{\text{gen}}(r \mid q, \mathcal{C}) + \lambda \cdot J_{\text{ret}}(q, \mathcal{C}) \tag{2}$$

Here, $J_{\text{gen}}(r \mid q, \mathcal{C})$ denotes the generation loss, implemented as the standard causal language modeling loss over the response tokens r, conditioned on the input query q and a set of sampled candidate passages $\mathcal{C}$. The term $J_{\text{ret}}(q, \mathcal{C})$ denotes the retrieval loss, computed over the query q and the same set of candidate passages $\mathcal{C}$, and is further detailed in the two-stage formulation described in Section 3.2.1. The hyperparameter λ balances the relative importance of the retrieval loss, allowing us to control the trade-off between retrieval accuracy and generation quality during training.

3.2.1 Retrieval Objective

While the overall training objective remains consistent across both stages—combining generation and retrieval losses as in (2)—the retrieval loss component J_{ret} varies depending on the training phase. In this section, we describe the two-stage training process used to endow ImpRAG with strong retrieval capabilities.

Warmup. Since the pretrained language model is not inherently optimized for retrieval, we begin with a warmup stage that introduces basic retrieval ability. We adopt a Multi-Label NCE loss (Zhang et al., 2022) as the retrieval objective and construct supervision using pseudo-labeled data generated by a strong off-the-shelf retriever, Contriever-MSMARCO (Izacard et al., 2022). For each query q, we retrieve the top-5 passages as pseudo-positive examples, denoted by $\mathcal{P}(q)$. We then sample a small set of pseudo hard negatives, denoted by $\mathcal{N}_h(q)$ (e.g., $|\mathcal{N}_h(q)| < 10$), from passages ranked 10–50.[3] While these passages may still be somewhat relevant, they are less likely to contain the key information necessary to answer the query. This selection introduces meaningful retrieval difficulty. We also use in-batch negatives across devices as additional random negatives $\mathcal{N}_r(q)$. The full negative set is $\mathcal{N}(q) = \mathcal{N}_h(q) \cup \mathcal{N}_r(q)$, and the candidate set is $\mathcal{C} = \mathcal{P}(q) \cup \mathcal{N}(q)$. The retrieval loss for this stage is defined as:

$$J_{\text{ret}}(q, \mathcal{C}) = -\sum_{p \in \mathcal{P}(q)} \log \left(\frac{\exp(s(q,p))}{\exp(s(q,p)) + \sum_{p' \in \mathcal{N}(q)} \exp(s(q,p'))} \right) \tag{3}$$

Self-Distillation. To further enhance retrieval performance, we employ language model perplexity distillation (Izacard et al., 2023), which assesses how much each candidate passage improves the language model's likelihood of generating the ground-truth response, conditioned on the query. Specifically, for each candidate passage $p \in \mathcal{C}$, we compute the log-likelihood of the gold response r given the concatenation of p and q, denoted as $\log P_{\text{LM}}(r \mid p, q)$. This defines a soft target distribution over candidate passages:

$$P_T(p \mid q, r) = \frac{\exp(\log P_{\text{LM}}(r \mid p, q))}{\sum_{p' \in \mathcal{C}} \exp(\log P_{\text{LM}}(r \mid p', q))} \tag{4}$$

[3] We find this approach effective in preliminary experiments, though we did not perform extensive hyperparameter tuning.

We also define the retrieval model's predicted distribution based on the similarity scores:

$$P_R(p \mid q) = \frac{\exp(s(q,p))}{\sum_{p' \in \mathcal{C}} \exp(s(q,p'))} \quad (5)$$

The retrieval loss is then computed as the KL divergence between the target and predicted distributions:

$$J_{\text{ret}}(q, \mathcal{C}) = \text{KL}\left(\overline{P_T(p \mid q, r)} \parallel P_R(p \mid q)\right) \quad (6)$$

Here, $\overline{P_T(p \mid q, r)}$ indicates that gradients are not backpropagated through the target distribution. Note that this stage also involves joint training; the only difference from the warmup phase lies in the retrieval loss J_{ret}.

3.3 Inference

At inference time, we first embed all passages in the knowledge corpus using the bottom layer group $\mathcal{L}_B$ of the model. These embeddings are stored in an approximate nearest neighbor (ANN) index (e.g., FAISS (Douze et al., 2024)) hosted on a remote server for efficient retrieval.

As illustrated in Figure 1, given a query, the ImpRAG model performs the following steps to generate a response:

1. The bottom layers $\mathcal{L}_B$ encode the input query and generate a query embedding, which is sent to the remote ANN search server.
2. The ANN server retrieves the top-k most relevant passages based on the query embedding and returns their passage IDs.
3. The middle layers $\mathcal{L}_M$ continue processing the information by applying cross-attention to the KV states of the retrieved passages.
4. The top layers $\mathcal{L}_T$ complete the encoding and decoding process without cross-attention, generating the next token.
5. The above steps are repeated at each decoding step. Notably, the query embeddings are computed only once at the end of the input prompt, and passage retrieval is not re-triggered thereafter.[4] In subsequent decoding steps, cross-attention continues to use the cached key-value states, and this process repeats until the model reaches a stopping criterion (e.g., an end-of-sequence token).

[4]While ImpRAG is general and can be adapted for iterative retrieval, we intend to focus this work on the single retrieval setup and will leave iterative retrieval for future work.

4 Experiment

4.1 Experimental Setup

Training. For training, we consider two types of datasets: (1) datasets requiring retrieval knowledge: NaturalQuestions (NQ; Kwiatkowski et al., 2019) and HotpotQA (Hopo; Yang et al., 2018); and (2) datasets without requiring retrieval knowledge, where we use the instruction tuning datasets from Lin et al. (2024) (see Appendix D for a complete list of these datasets). Inspired by Chen et al. (2022), we also incorporate two synthetic, retrieval-free tasks into the training to enhance instruction-following capabilities: phrase denoising, and next/previous sentence generation. The training data for phrase denoising is generated by prompting LLMs (we use Llama-3.1 70B) with a paragraph from Wikipedia. For the sentence generation task, we construct it randomly using content from Wikipedia.

For all these datasets, we use a subset of 5,000 examples from their training splits in each dataset. In addition, we use 1,000 examples from the NQ dev split as the development set. We use the December 2021 Wikipedia from Izacard et al. (2023) as our knowledge corpus. Additionally, we spend approximately 10% of training on plain text from Wikipedia to prevent models from overfitting to the downstream tasks.

Evaluation. We evaluate models on 8 different knowledge-intensive tasks to assess their various capabilities, specifically:

- Basic question answering: NQ, SimpleQA (SQA; Wei et al., 2024);
- Multihop reasoning: Hopo, 2WikiMultiHopQA (2WQA; Ho et al., 2020);
- Instruction following: (1) relation extraction: T-Rex (Elsahar et al., 2018), ZsRE (Levy et al., 2017), (2) fact checking: FEVER (FEV; Thorne et al., 2018), and (3) entity linking: AIDA (Hoffart et al., 2011).

For all these datasets, we report exact matches as the evaluation metric for generation tasks and recall rates for retrieval tasks. The retrieval recall is measured by the percentage of instances where the top-retrieved results contain the answers as substrings. We omit retrieval recall for FEV as it is

Task	Template
Knowledge-Intensive Tasks	
NQ, Hopo, SQA, 2WQA	Q: {question} A: {answer}
AIDA	{context} Output the Wikipedia page title of the entity mentioned between [START] and [END] in the given text A: {answer}
FEV	Is this statement true? {statement} A: {answer}
T-Rex, ZsRE	{entity} [SEP] {relation} Provide the answer corresponding to the relation specified after [SEP] for the entity mentioned before [SEP] A: {answer}
Instruction-Tuning Tasks	
Dialogue Completion	$\{turn_1\}$ $\{turn_2\}$ $\{turn_3\}$ ...
Reading Comprehension	{context} Q: {question} A: {answer}
Summarization	{context} Summarize this article: {summary}
Phrase Denoising	{context} Recover the original phrases marked between [START] and [END] in the given text A: {answer}
Sentence Generation	{context} [SEP] next/previous sentence Generate a sentence corresponding to the relation specified after [SEP] for the context mentioned before [SEP] A: {sentence}

Table 1: Prompt templates. We only use retrieval for knowledge-intensive tasks. For simplicity, we list task categories for a subset of the instruction tuning datasets. See Appendix D for more detailed description.

	NQ	SQA	Hopo	2WQA	T-Rex	ZsRE	FEV	AIDA	avg
Llama-3.2$_{3B}$									
+RA-IT	43.2 (77.0)	38.1 (48.2)	35.9 (48.8)	33.4 (43.3)	54.2 (84.3)	58.1 (86.6)	79.2 (-)	40.1 (38.1)	47.8 (60.9)
+RA-DIT	43.4 (77.5)	38.8 (48.7)	36.4 (49.3)	34.0 (43.5)	55.0 (85.0)	59.0 (87.2)	80.5 (-)	41.0 (38.2)	48.5 (61.3)
+RA-DIT-Llama	43.9 (78.0)	39.8 **(49.9)**	37.0 (49.8)	35.1 (44.0)	55.8 (85.9)	60.0 (87.9)	80.2 (-)	41.1 (38.4)	50.4 (64.0)
+ImpRAG	**44.1 (78.4)**	**40.3 (50.0)**	**37.3 (50.2)**	**35.5 (44.5)**	**60.8 (90.2)**	**65.4 (93.2)**	**83.8 (-)**	**52.6 (58.3)**	**52.5 (66.4)**
Llama-3.1$_{8B}$									
+RA-IT	45.1 (77.0)	39.0 (48.2)	36.9 (48.8)	34.4 (43.3)	55.0 (84.3)	59.1 (86.6)	83.2 (-)	41.1 (38.1)	49.2 (60.9)
+RA-DIT	45.7 (77.7)	38.9 (48.9)	37.2 (49.1)	34.9 (44.0)	56.1 (85.4)	60.1 (87.8)	85.1 (-)	41.5 (38.8)	49.9 (61.7)
+RA-DIT-Llama	46.1 (78.7)	40.7 (50.3)	37.9 (50.2)	35.6 (44.8)	57.0 (86.1)	61.2 (88.1)	86.2 (-)	42.1 (39.2)	50.9 (62.5)
+ImpRAG	**46.4 (79.1)**	**41.3 (51.2)**	**38.4 (50.9)**	**36.0 (45.2)**	**62.5 (92.7)**	**67.1 (94.0)**	**89.2 (-)**	**54.2 (62.4)**	**54.4 (67.9)**

Table 2: Evaluation results for 8 knowledge-intensive tasks. We report exact match scores for generation tasks and retrieval recall (shown in parentheses) for retrieval tasks. Retrieval recall is not reported for FEV, as it is a classification task. All these methods use retrieval augmentation.

a classification task where the answer strings are either "True" or "False".

For Hopo, T-Rex, ZsRE, FEV, and AIDA, we use development sets from the KILT benchmark (Lewis et al., 2020). For SQA and NQ, we use the official test set. For 2WQA, we use their development set. For all datasets, we utilize the entire input prompts as queries for the retrievers. We describe our task templates in Table 1.

Baselines. We consider 3 baseline models:

- Retrieval Augmented Instruction Tuning (RA-IT): This approach involves directly incorporating retrieved passages from Contriever-MSMARCO into the context and fine-tuning the language models (LMs) on the training data;
- Retrieval Augmented Dual Instruction Tuning (RA-DIT; Lin et al., 2024): In this method, we first fine-tune the Contriever-MSMARCO on the training subsets of NQ and HotpotQA using Equation 6. Subsequently, we perform fine-tuning as in RA-IT, utilizing the fine-tuned retriever;
- RA-DIT with Llama as the Retriever (RA-DIT-Llama): Here, we replace the Contriever used in RA-DIT with the first 8 layers from the Llama models.[5] To ensure effective retrieval performance, we initially warm up the Llama retrievers with pseudo labels generated by Contriever-MSMARCO using Equation 3.

Hyperparameters. We use Llama-3.2 3B and Llama-3.1 8B as the base models for ImpRAG. For both models, the layer boundary b is set to 7.[6] For Llama-3.2 3B, the layer boundary t is 19, while for Llama-3.1 8B, it is 23. We train for 10 epochs and perform the retrieval warmup in the first 3 epochs. When retrieving passages, we take the top 10 most relevant documents.

See Appendix E for more details on the baselines and computational resources.

[5]We choose to use first 8 layers for fair comparison as ImpRAG uses the same layers for retrieval.

[6]Since we label the first layer of a LLM as layer 0, a layer boundary b of 7 means that the bottom layer group contains the first 8 layers.

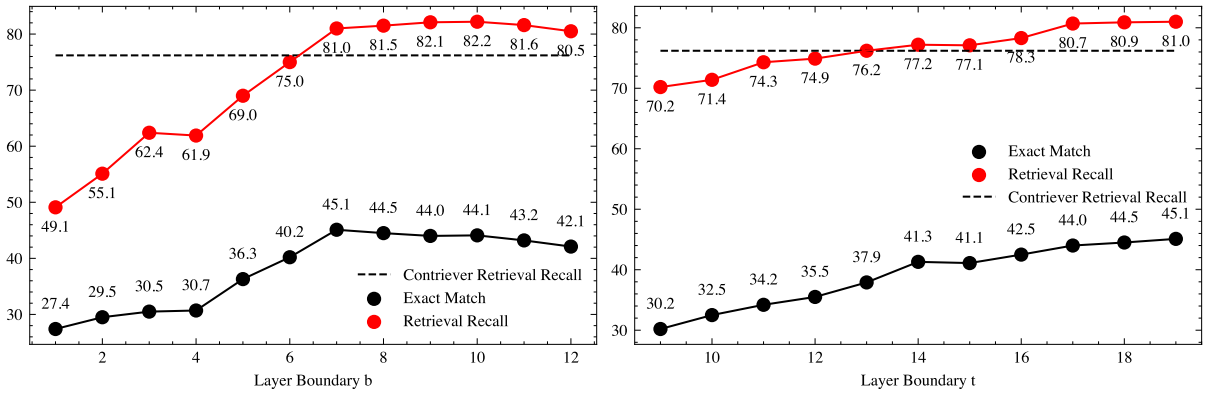

Figure 2: Exact match and retrieval recall on the NQ dev set using Llama-3.2 3B with different values of b (left side) and t (right side). When varying one layer boundary, we keep the other constant.

	NQ	SQA	Hopo	2WQA	T-Rex	ZsRE	FEV	AIDA	avg
self-distillation only	29.9 (61.2)	30.1 (39.9)	29.8 (41.9)	27.5 (37.4)	35.6 (64.9)	40.9 (65.9)	67.7 (-)	28.3 (22.5)	36.2 (47.7)
warmup only	**44.0 (78.3)**	39.9 **(50.0)**	37.1 (50.0)	35.1 (44.2)	56.5 (87.0)	61.2 (88.3)	81.0 (-)	45.2 (42.9)	50.0 (63.0)
warmup+self-distillation	**44.1 (78.4)**	**40.3 (50.0)**	**37.3 (50.2)**	**35.5 (44.5)**	**60.8 (90.2)**	**65.4 (93.2)**	**83.8 (-)**	**52.6 (58.3)**	**52.5 (66.4)**

Table 3: Exact match scores and retrieval recall (shown in parentheses) for ImpRAG using Llama-3.2 3B as the base model, trained with different retrieval objectives.

4.2 Experimental Result

Table 2 presents our main evaluation results. Each model variant—RA-IT, RA-DIT, RA-DIT-Llama, and ImpRAG —exhibits different performance levels, with ImpRAG consistently achieving the highest scores across all tasks. For Llama-3.2 3B, the average exact match score increases from 47.8 with RA-IT to 52.5 with ImpRAG, while for Llama-3.1 8B, the score rises from 49.2 to 54.4. Although RA-DIT shows improvements over RA-IT, ImpRAG further enhances performance. Notably, ImpRAG significantly outperforms RA-DIT-Llama, indicating that the improvements are not merely due to using a more powerful base model (i.e., the first 8 layers of Llama models) for retrieval. Importantly, the enhancements are evident in both exact match scores and retrieval recalls, demonstrating that ImpRAG improves both generation quality and retrieval performance. It is worth noting that compared to the baseline approaches, the most substantial improvements with ImpRAG are seen in tasks that queries need to be formulated more differently from input prompts, such as T-Rex, ZsRE, FEVER, and AIDA. Among these tasks, AIDA shows the most significant improvements, with over a 20-point increase in retrieval recall and more than a 10-point rise in exact match scores for both Llama-3.1 3B and Llama-3.1 8B, likely due to the inadequacy of directly using input prompts as queries in AIDA. This underscores ImpRAG's effectiveness in formulating implicit queries and embedding instruction-following capabilities into retrievers. Overall, these results demonstrate that ImpRAG significantly enhances the models' ability to accurately retrieve and apply knowledge, with improvements more significant in tasks requiring diverse formats.

5 Analysis

5.1 Layer Group Boundary Ablation

In this section, we examine the effects of the layer boundaries b and t. The findings are presented in Figure 2. To facilitate comparison, we vary one layer boundary while keeping the other constant. We note that increasing b reduces the number of layers allocated to the middle layer group, which includes layers for reading and generation. Conversely, increasing t does not affect the retrieval layers. Overall, we find that increasing b enhances retrieval recall, with improvements leveling off once b reaches 7. This plateau is likely due to diminished generation performance, which results in less precise training signals for self-distillation. This underscores the importance of balancing parameters between retrieval and generation. On the other hand, as expected, increasing t consistently yields improvements. Although these improvements seem to plateau at 19, we refrain from further increasing t primarily due to memory constraints. We plan to leave more memory-efficient training of ImpRAG for future exploration.

	T-Rex	ZsRE	FEV	AIDA	avg
No templates	55.8 (85.9)	60.0 (87.9)	80.2 (-)	41.1 (38.4)	59.3 (70.7)
Oracle templates	**61.4 (90.7)**	**66.0 (93.6)**	**83.9 (-)**	**66.1 (72.3)**	**69.4 (85.5)**
ImpRAG	60.8 (90.2)	**65.9 (93.5)**	**83.8 (-)**	52.6 (58.3)	65.8 (80.7)

Table 4: Exact match scores and retrieval recall (shown in parentheses) for RA-DIT-Llama using Llama-3.2 3B as the base model, evaluated with various query templates. In the case of "no templates", the inputs to the LLMs are used directly as queries.

	NQ	SQA	Hopo	2WQA	T-Rex	ZsRE	FEV	AIDA	avg
ImpRAG	**44.1 (78.4)**	**40.3 (50.0)**	**37.3 (50.2)**	**35.5 (44.5)**	**60.8 (90.2)**	**65.4 (93.2)**	**83.8 (-)**	**52.6 (58.3)**	**52.5 (66.4)**
w/o all IT tasks	42.9 (76.4)	38.1 (48.2)	35.2 (47.7)	33.7 (42.0)	43.5 (69.3)	49.5 (70.2)	76.2 (-)	25.4 (20.5)	43.1 (53.5)
w/o PD and SG	**44.0 (78.5)**	**40.1 (50.1)**	**37.4 (50.3)**	**35.4 (44.4)**	53.3 (82.8)	57.1 (84.9)	81.2 (-)	40.5 (41.2)	48.6 (61.7)

Table 5: Exact match scores and retrieval recall (shown in parentheses) for ImpRAG using Llama-3.2 3B as the base model, trained with different combinations of instruction tuning datasets. IT tasks refer to instruction tuning tasks, PD stands for phrase denoising, and SG denotes sentence generation.

5.2 Retrieval Objective Ablation

We conduct experiments to compare the effects of different retrieval training objectives. The results are presented in Table 3. During training, we consistently apply each retrieval objective throughout the entire process. For instance, in the "warmup only" experiment, we extend the use of the warmup objective to 10 epochs instead of limiting it to the initial 3 epochs. Our findings indicate that the warmup objective provides a baseline performance across all tasks and is particularly beneficial for tasks with direct supervision. Self-distillation builds on this baseline, further enhancing model performance on unseen test tasks. Overall, the two training objectives complement each other effectively.

5.3 Effect of Query Templates

We also examine the impact of using different query templates for the baseline approach, RA-DIT-Llama. The results are detailed in Table 4. In these experiments, we omit QA tasks because their "no templates" and "oracle templates" setups are almost the same. Overall, "oracle templates" still provides the best performance. The improvements are particularly notable on AIDA. However, it is important to highlight that ImpRAG achieves highly competitive performance on 3 out of 4 tasks and already shows significant improvement on the remaining task compared to using "no templates."

5.4 Effect of Instruction Tuning for Retrieval

In Table 5, we explore the effects of training on instruction tuning datasets. The table shows that omitting all instruction tuning datasets leads to a decline in model performance on both in-domain tasks (NQ, SQA, Hopo, and 2WQA) and out-of-domain tasks. Notably, removing only phrase denoising and sentence generation has a minimal impact on in-domain tasks but causes more pronounced negative effects on out-of-domain tasks, except for FEV. This exception likely arises because FEV's task format is more similar to the in-domain tasks than other tasks. This suggests that instruction tuning tasks aid models in understanding task formats, and ImpRAG can transfer this knowledge from generation to retrieval due to its unified model architecture.

6 Conclusion

We present ImpRAG, a query-free retrieval-augmented generation (RAG) system that implicitly captures information needs without requiring human-specified queries. Unlike prior work that treats retrieval and generation as separate components with independently trained models, ImpRAG unifies them within a single decoder-only language model by partitioning it into specialized layer groups and jointly optimizing for both retrieval and generation. The same model parameters and forward pass are shared across retrieval and generation, effectively minimizing the mismatch between the retriever and the generator. ImpRAG demonstrates strong performance across eight knowledge-intensive tasks, outperforming traditional RAG systems and delivering substantial gains on unseen tasks with diverse formats.

7 Limitations

One limitation of this work is its focus on a single-pass retrieval setup; we do not explore iterative or multi-hop retrieval, which could further enhance performance on complex reasoning tasks. Adapting ImpRAG to iterative retrieval remains an important direction for future work.

Our method is also evaluated exclusively using the LLaMA 3 family of models. While the approach is broadly applicable, its generalizability to other architectures and model sizes has yet to be validated.

Additionally, the warmup stage relies on pseudo-labeled data generated by Contriever-MSMARCO. Although this provides a strong starting point, we expect that using more powerful retrievers or human-labeled data could lead to further gains by offering higher-quality supervision early in training.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2023. Task-aware retrieval with instructions. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3650–3675, Toronto, Canada. Association for Computational Linguistics.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany. Association for Computational Linguistics.

Mingda Chen, Xilun Chen, and Wen-tau Yih. 2024. Few-shot data synthesis for open domain multi-hop question answering. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 190–208, St. Julian's, Malta. Association for Computational Linguistics.

Mingda Chen, Jingfei Du, Ramakanth Pasunuru, Todor Mihaylov, Srini Iyer, Veselin Stoyanov, and Zornitsa Kozareva. 2022. Improving in-context few-shot learning via self-supervised training. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3558–3573, Seattle, United States. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Junjie Fang, Likai Tang, Hongzhe Bi, Yujia Qin, Si Sun, Zhenyu Li, Haolun Li, Yongjian Li, Xin Cong, Yankai Lin, Yukun Yan, Xiaodong Shi, Sen Song, Zhiyuan Liu, and Maosong Sun. 2024. Unimem: Towards a unified view of long-context large language models. In *First Conference on Language Modeling*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Zhengbao Jiang, Luyu Gao, Zhiruo Wang, Jun Araki, Haibo Ding, Jamie Callan, and Graham Neubig. 2022. Retrieval as attention: End-to-end learning of retrieval and reading within a single transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2336–2349, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Minh Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Alexandrovich Glushkov, Arnav Varma Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Julian Mattick. 2023. Openassistant conversations - democratizing large language model alignment. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*.

Yoonsang Lee, Minsoo Kim, and Seung-won Hwang. 2024. Disentangling questions from query generation for task-adaptive retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4775–4785, Miami, Florida, USA. Association for Computational Linguistics.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. RA-DIT: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Songshuo Lu, Hua Wang, Yutian Rong, Zhi Chen, and Yaohua Tang. 2024. Turborag: Accelerating retrieval-augmented generation with precomputed kv caches for chunked text. *arXiv preprint arXiv:2410.07590*.

Niklas Muennighoff, SU Hongjin, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and

Douwe Kiela. 2024. Generative representational instruction tuning. In *ICLR 2024 Workshop: How Far Are We From AGI*.

Hanseok Oh, Hyunji Lee, Seonghyeon Ye, Haebin Shin, Hansol Jang, Changwook Jun, and Minjoon Seo. 2024. Instructir: A benchmark for instruction following of information retrieval models. *arXiv preprint arXiv:2402.14334*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. 2020. Getting closer to ai complete question answering: A set of prerequisite real tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8722–8731.

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)*, pages 8371–8384, Mexico City, Mexico. Association for Computational Linguistics.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Boxin Wang, Wei Ping, Lawrence Mcafee, Peng Xu, Bo Li, Mohammad Shoeybi, and Bryan Catanzaro. 2024. InstructRetro: Instruction tuning post retrieval-augmented pretraining. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 51255–51272. PMLR.

Boxin Wang, Wei Ping, Peng Xu, Lawrence McAfee, Zihan Liu, Mohammad Shoeybi, Yi Dong, Oleksii Kuchaiev, Bo Li, Chaowei Xiao, Anima Anandkumar, and Bryan Catanzaro. 2023. Shall we pretrain autoregressive language models with retrieval? a comprehensive study. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7763–7786, Singapore. Association for Computational Linguistics.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*.

Orion Weller, Benjamin Chang, Sean MacAvaney, Kyle Lo, Arman Cohan, Benjamin Van Durme, Dawn Lawrie, and Luca Soldaini. 2025. FollowIR: Evaluating and teaching information retrieval models to follow instructions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11926–11942, Albuquerque, New Mexico. Association for Computational Linguistics.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval head mechanistically explains long-context factuality. *arXiv preprint arXiv:2404.15574*.

Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, and 1 others. 2024. Memory3: Language modeling with explicit memory. *arXiv preprint arXiv:2407.01178*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2588–2610.

Jintian Zhang, Cheng Peng, Mengshu Sun, Xiang Chen, Lei Liang, Zhiqiang Zhang, Jun Zhou, Huajun Chen, and Ningyu Zhang. 2024a. OneGen: Efficient one-pass unified generation and retrieval for LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4088–4119, Miami, Florida, USA. Association for Computational Linguistics.

LingXi Zhang, Yue Yu, Kuan Wang, and Chao Zhang. 2024b. ARL2: Aligning retrievers with black-box large language models via self-guided adaptive relevance labeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3708–3719, Bangkok, Thailand. Association for Computational Linguistics.

Wenzheng Zhang, Wenyue Hua, and Karl Stratos. 2022. EntQA: Entity linking as question answering. In *International Conference on Learning Representations*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Zheng Zhao, Yftah Ziser, and Shay B Cohen. 2024. Layer by layer: Uncovering where multi-task learning happens in instruction-tuned large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15195–15214, Miami, Florida, USA. Association for Computational Linguistics.

A Passage Encoding

Given k retrieved passages, we must obtain the key-value (KV) states from the middle layer group $\mathcal{L}_M$ to enable cross-attention. We explore three passage encoding strategies, summarized in Table 6.

First, we consider Independent Encoding, where each passage is encoded separately using position IDs starting from zero, following the parallel encoding strategy in Yen et al. (2024). The resulting KV states are then concatenated across passages.

Second, we examine Concatenated Encoding (Segmented), in which passages are concatenated into a single sequence, but attention across passages is blocked to prevent inter-passage interaction.

Third, we evaluate Concatenated Encoding (Full Attention), where passages are concatenated and full cross-passage attention is allowed throughout the encoding.

We conduct these experiments by finetuning Llama-3.1 8B model on the Natural Questions (NQ) dataset using the top-10 passages retrieved by Contriever-MSMARCO, and report Exact Match (EM) scores on the development set. As shown in Table 6, the two simpler strategies—Independent Encoding and Segmented Concatenation—perform similarly, while Full Attention Concatenation yields a clear performance improvement, highlighting the benefit of modeling inter-passage dependencies.

Encoding Method	Dev EM
Independent Encoding	51.7
Segmented Concatenation	51.4
Full Attention Concatenation	53.3

Table 6: Performance of different passage encoding strategies.

B Freezing Passage Representations

We investigate the impact of freezing passage representations—either hidden states or key-value (KV) states—during inference with a fixed retriever. All experiments are conducted using a fine-tuned LLaMA-3.1 8B model and the top-10 passages retrieved by Contriever-MSMARCO on the Natural Questions (NQ) dataset. Results are reported in Table 7.

We explore two freezing strategies, both using the Independent Encoding approach described in Appendix A. In the first variant, Frozen Hidden States, we freeze the hidden representations of retrieved passages as produced by the initial (untrained) LLaMA-3.1 8B model, and pass them through the trained key/value projection layers to generate the KV states used in cross-attention.

In the second variant, Frozen KV States, we directly freeze the key and value attention states of the passages, also obtained from the initial LLama-3.1 8B model.

We observe that both freezing methods yield comparable performance, slightly underperforming the fully dynamic setting where passage KV states are computed using the trained model.

C Passage KV States Compression

When we use independent encoding strategy in Appendix A, one benefit will be that we can save

Method	Dev EM
No Freezing	51.4
Frozen Hidden States	50.8
Frozen KV States	50.7

Table 7: Performance of freezing different passage representations on NQ dev set with top-10 Contriever-MSMARCO retrieved passages.

the middle layer group key value states for all the passages in knowledge corpus in disk and during inference after retrieval we can load the key value states from disk without recomputation. However, this will result in a large amount of disk spaces. Thus, we consider two compression strategies: token compression and product quantization and we conduct experiments following the same setting as the Frozen KV states in Appendix B. Specifically, take for token compression, we use the Heavy Hitter (Zhang et al., 2023) and only keep half number of tokens for each passage. For production quantization, we use FAISS codec with index type OPQ32x128-PQ32x8 for each key value head, which is trained on 500k randomly sampled wikipedia passages. The compression rate with this quantization is $\frac{128\times 2}{32} = 8$ for original bfloat16 state vector of each attention head. We report the results in Table 8. We can see that both strategies don't hurt the performance much.

Compression	Dev EM
No Compression	50.7
Heavy Hitter	49.9
Product Quantization	50.3

Table 8: The results for various compression techniques.

D Instruction-Tuning Datasets

We use OpenAssistant Conversations Dataset (oasst1; Köpf et al., 2023), Conversational Question Answering (CoQA; Reddy et al., 2019), Discrete Reasoning Over Paragraphs (DROP; Dua et al., 2019), NewsQA (Trischler et al., 2017), PubMedQA (Jin et al., 2019), QA for Artificial Intelligence (Quail; Rogers et al., 2020), SQuAD v2 (Rajpurkar et al., 2018),[7] and CNN DailyMail (Chen et al., 2016) The templates for these datasets are shown in Table 9.

[7]We only use answerable questions from SQuAD v2.

Task	Template
Instruction-Tuning Tasks	
oasst1	{turn_1} {turn_2} {turn_3} ...
CoQA, DROP, NewsQA, PubMedQA, SQuAD	{context} Q: {question} A: {answer}
CNN DailyMail	{context} Summarize this article: {summary}

Table 9: Prompt templates. We only use retrieval for knowledge-intensive tasks.

E Baselines and Computational Resources

Discussions on Baselines. For all these baselines, we use the retrieve-then-generate paradigm, i.e., begin by retrieving candidates using the retrievers and then incorporate them into the context for training and inference. This implies that these baselines require an additional retriever, leading to increased computational costs and a higher number of model parameters compared to ImpRAG. However, since this is a standard practice for retrieval-augmented models, we continue to use them in the baselines to establish stronger comparisons.

Computational Resources. We use NVIDIA H100 GPUs. Each training session requires 8 H100 GPUs, and hosting the index also demands an additional 8 GPUs. Training the baseline approaches takes roughly 96 GPU hours, whereas our models require approximately 160 GPU hours.