Understanding Parametric Knowledge Injection in Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Parametric retrieval-augmented generation (P-RAG) encodes retrieved evidence into model parameters rather than conditioning on passages as input tokens, promising lower inference cost and deeper knowledge integration. However, its advantages over token-based RAG remain insufficiently understood. We conduct a systematic study of P-RAG and its dynamic extension. First, using HasAnswer accuracy instead of length-sensitive F1, we find that P-RAG does not consistently outperform token-based RAG on its own, while combining parametric and token-based context performs best. Second, our analyses show that parametric representations capture document-level semantics and mainly affect deeper feed-forward layers, suggesting that they act as high-level guidance rather than precise evidence storage. Finally, we evaluate P-RAG under knowledge conflicts, retrieval noise, and transfer beyond QA. Parametric injection improves faithfulness under conflicts, is more robust to noisy retrieval, and generalizes to fact verification and slot filling. These results clarify when parametric RAG is beneficial and provide guidance for future context-parameterized generation systems.

1 Introduction

Context-grounded generation is central to many LLM applications, including long-context question answering (QA) (Kočiskỳ et al., 2018; Dasigi et al., 2021), retrieval-augmented generation (RAG) (Lewis et al., 2020; Izacard et al., 2023), and conversational or agentic systems (Packer et al., 2023; Zheng et al., 2025). Most systems incorporate context by directly attending over input tokens, which becomes expensive as context length grows and remains difficult to use reliably even with longer context windows (Google, 2025; Liu et al., 2024; Dou et al., 2026). Recent work therefore explores context **parameterization** (Wang et al., 2024b; Cao et al., 2025; Chen et al.; Liao et al., 2025), which compresses context into trainable parameters such as soft prompts or LoRA adapters (Hu et al., 2022) and injects them into a frozen LLM. This paradigm can incorporate information beyond the model's explicit context window while reducing inference-time context length.

In RAG, this idea gives rise to **P**arametric **RAG** (**P-RAG**) (Su et al., 2025b). Unlike classic **T**oken-based **RAG** (**T-RAG**), which conditions on retrieved passages as text, P-RAG converts retrieved evidence into lightweight parameter updates and injects knowledge through the model's internal computation. Because factual associations are often linked to feed-forward layers (Geva et al., 2021), such injection may offer a direct route for integrating retrieved knowledge, especially when evidence must be consolidated across passages. Prior work reports that P-RAG improves multi-hop QA over T-RAG, reduces inference cost, and performs best when combined with token-based context as **PT-RAG** (Su et al., 2025b), motivating growing interest in parametric RAG (Tan et al., 2025; Chen et al., 2025; Su et al., 2025c; Fan et al., 2025; Su et al., 2025a; Fleshman and Van Durme, 2025).

Despite this promise, P-RAG also introduces important trade-offs compared with T-RAG. Compressing retrieved text into parameters may discard fine-grained evidence, reduce interpretability, depend strongly on training choices, and produce adapters that are tied to a specific base model. It also adds nontrivial training and storage overhead. Moreover, existing P-RAG studies commonly evaluate with F1 (Su et al., 2025b; Tan et al., 2025; Chen et al., 2025), even though F1 is sensitive to response length and can favor concise generations. For answer-centric RAG evaluation, HasAnswer accuracy provides a more direct test of whether the generated response contains the ground-truth answer or an equivalent form. Figure 1 illustrates how these metrics can lead to different conclusions. These issues call for a systematic reassessment of

when parametric knowledge injection helps and where it falls short.

To better understand parametric knowledge injection in RAG and provide practical guidance for future research, we conduct a systematic study of parametric RAG, focusing on P-RAG (Su et al., 2025b) and its extension DyP-RAG (Tan et al., 2025). Specifically, we investigate the following research questions (RQs): (**RQ1**) *Evaluation validity.* Do P-RAG methods still outperform T-RAG under answer-presence accuracy (HasAnswer), rather than length-sensitive F1? (**RQ2**) *Representation and mechanism.* How effectively do parametric representations encode external knowledge, and how does parametric injection influence LLM computation, multi-hop reasoning, and evidence consolidation? (**RQ3**) *Faithfulness, robustness, and generalization.* How does P-RAG behave under key RAG challenges, including knowledge conflicts, retrieval noise, and cross-task generalization?

For **RQ1**, we find that P-RAG does not consistently outperform T-RAG under the HasAnswer metric. However, consistent with prior work, PT-RAG achieves the best overall performance. This suggests that parametric injection alone may be insufficient for capturing fine-grained external knowledge, but can effectively complement token-based context. For **RQ2**, we construct a QA dataset containing up-to-date information beyond the LLM's training cutoff. Our analyses show that (1) parametric representations encode document-level semantic information, with relevant passages exhibiting higher similarity than irrelevant ones; (2) parametric injection primarily affects deeper feed-forward layers, suggesting a role as high-level semantic guidance; and (3) although standalone P-RAG remains weaker than T-RAG for multi-hop reasoning and evidence consolidation, combining parametric and token-based context enables PT-RAG to consistently outperform T-RAG, even allowing smaller models to achieve performance comparable to larger T-RAG systems. For **RQ3**, we show that parametric injection improves faithfulness under knowledge conflicts, is more robust to retrieval noise than T-RAG, and can generalize beyond QA tasks such as fact verification and slot filling.

In summary, we present a systematic analysis of parametric RAG to better understand its behavior and trade-offs relative to token-based RAG, with the main conclusions summarized in Table 1. Our contributions are fourfold: (1) we reassess P-RAG under a more appropriate evaluation protocol based on HasAnswer accuracy; (2) we introduce a QA benchmark containing up-to-date knowledge for controlled analysis; (3) we provide representational and mechanistic analyses of parametric injection, including its semantic encoding behavior and layer-wise effects on model computation; and (4) we evaluate P-RAG under key RAG challenges, including knowledge conflicts, retrieval noise, and cross-task generalization. Overall, our findings clarify the strengths and limitations of parametric RAG and provide practical guidance for future research.

Table 1: Comparison of token-based RAG, parametric RAG, and the approach combining both.

Metric	Token-based	Parametric	Combined
Inference Cost	High	**Low**	High
Long-Context Scalability	Limited	**Strong**	Moderate
Knowledge Fidelity	**High**	Limited	**High**
Multi-hop Consolidation	Moderate	Limited	**Strong**
Noise Robustness	Limited	**Strong**	**Strong**
Faithfulness Under Conflict	Moderate	Moderate	**Strong**
Task Generalization	**Strong**	Moderate	**Strong**
Deployment Flexibility	**High**	Limited	Limited

2 Related Work

Retrieval-augmented generation (RAG) enhances LLMs by incorporating external knowledge (Lewis et al., 2020; Zamani et al., 2022). Existing approaches mainly differ in how retrieved knowledge is integrated: **token-level** augmentation directly inserts retrieved documents into the prompt (Trivedi et al., 2022; Tang et al., 2025); **embedding-level** fusion injects document representations through mechanisms such as cross-attention (Izacard and Grave, 2020; Dong et al., 2025); and recent **parametric** RAG methods encode retrieved documents into lightweight model parameters (Su et al., 2025b; Tan et al., 2025). Representative methods include P-RAG (Su et al., 2025b), which trains document-specific LoRA adapters, DyP-RAG (Tan et al., 2025), which generates adapters through a hyper-network, and PolyP-RAG (Su et al., 2025c), which routes among shared latent LoRA experts. Additional related work is provided in Appendix A.

3 Preliminary

This section introduces the inference formulations of token-based and parametric RAG, together with the parameterization procedures of P-RAG (Su et al., 2025b) and DyP-RAG (Tan et al., 2025).

Standard RAG. Given a query q, a retriever retrieves the top-k relevant documents $\mathcal{D} = \{d_1, d_2, ..., d_k\}$ from an external corpus. In standard token-based RAG, the retrieved documents

are concatenated into a textual context sequence $c = \text{Concat}(d_1, d_2, ..., d_k)$. The LLM with parameters θ then generates a response by attending over both the query q and the explicit context c:

$$y^{\text{T-RAG}} = \arg\max_y P(y \mid c, q; \theta). \quad (1)$$

Parametric RAG. Token-based RAG suffers from finite context windows, high inference cost, and degraded performance on long contexts. Parametric RAG instead encodes retrieved documents into model parameters rather than textual context, allowing the model to access knowledge at parametric level without increasing context length.

Formally, each retrieved document $d_i \in \mathcal{D}$ is encoded into a parametric representation (e.g., a LoRA adapter) $\Delta\theta_i = F(d_i)$, where F can be implemented via offline optimization (e.g., P-RAG) or online hypernetwork generation (e.g., DyP-RAG). At inference time, the representations of the top-k retrieved documents are aggregated into a unified parameter update, typically through linear composition: $\Delta\theta_{\mathcal{D}} = \sum_{i=1}^{k} \Delta\theta_i$. The LLM then generates outputs conditioned only on the query q, with its parameters adapted by the retrieved content:

$$y^{\text{P-RAG}} = \arg\max_y P(y \mid q; \theta + \Delta\theta_{\mathcal{D}}). \quad (2)$$

Parametric RAG can also be combined with token-based RAG. In this hybrid setting, retrieved documents are included in the input context $c = \text{Concat}(d_1, ..., d_k)$ while the parametric representation is simultaneously injected:

$$y^{\text{PT-RAG}} = \arg\max_y P(y \mid c, q; \theta + \Delta\theta_{\mathcal{D}}). \quad (3)$$

Document Parameterization of P-RAG. P-RAG (Su et al., 2025b) encodes each document d_i into a parametric representation $\Delta\theta_i$ (implemented as a LoRA adapter) via gradient-based optimization. To improve knowledge internalization, P-RAG augments each document with synthetic data. Specifically, for each document d_i, P-RAG prompts the target LLM to generate rewritten variants and diverse QA pairs grounded in the document content. These generated samples form an augmented dataset $\mathcal{Z}_i$, where each sample $z \in \mathcal{Z}_i$ concatenates the document, question, and answer. The LoRA parameters $\Delta\theta_i$ are then optimized with next-token prediction over the augmented sequences.

Document Parameterization of DyP-RAG. While effective, P-RAG requires per-document gradient optimization, limiting scalability. DyP-RAG (Tan et al., 2025) addresses this issue with a lightweight hypernetwork $\mathcal{F}_\phi$ that directly maps document representations into adapter weights.

Given a document d_i, DyP-RAG first extracts a semantic representation h_i (e.g., the last hidden state) using the frozen LLM. The hypernetwork $\mathcal{F}_\phi$ then maps h_i into the adapter weight space: $\Delta\theta_i = \mathcal{F}_\phi(h_i)$. To train $\mathcal{F}_\phi$, DyP-RAG uses the offline-optimized P-RAG parameters as supervision signals, combining parameter-level distillation objectives and LLM generative loss.

4 Reassessing Parametric RAG

Prior work reports P-RAG gains mainly with F1-based evaluation (Su et al., 2025b; Tan et al., 2025; Chen et al., 2025). Because F1 is sensitive to response length and lexical overlap, it may favor P-RAG's concise outputs. We therefore ask: *Do P-RAG methods still outperform T-RAG under answer-presence accuracy rather than length-sensitive F1?* We answer this by re-evaluating P-RAG and DyP-RAG with HasAnswer accuracy, which reduces the effect of response verbosity.

4.1 Experimental Setup

Evaluation Metric. We adopt HasAnswer accuracy as the primary metric, which checks whether the generated response contains the ground-truth answer or a semantically equivalent form. We implement this metric using an LLM-as-a-judge framework (Ho et al., 2025) with Qwen2.5-32B-Instruct (Team, 2024). We manually annotated 300 randomly sampled instances and observed a 96.7% agreement rate between the LLM judge and human evaluators, confirming its robustness.

Inference and Parameterization Setup. We adopt a zero-shot setting across all experiments and use greedy decoding for all generations. For parameterization, both P-RAG and DyP-RAG apply LoRA to FFN layers following their original implementations. P-RAG is trained using the augmented data, while DyP-RAG is reproduced using its official training pipeline. Additional implementation details are provided in Appendix B.1.

Datasets. Following prior work (Su et al., 2025b; Tan et al., 2025), we evaluate on four QA benchmarks: 2WikiMultihopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018), PopQA (Mallen et al., 2023), and ComplexWebQuestions (Talmor and Berant, 2018). For each question, we retrieve top-3 relevant passages using BM25. Additional dataset details are provided in Appendix B.2.

Methods and Models. We evaluate six meth-

Table 2: Experimental results under HasAnswer accuracy (%). Bold and underlined numbers denote the best and second-best results. ♣ and ♢ indicate a statistically significant difference ($p < 0.05$) over Vanilla LLM and T-RAG.

LLM	Method	2WikiMultihopQA					HotpotQA			PopQA	CWQ	Average
		Compare	Bridge	Infer	Compose	Total	Bridge	Compare	Total			
LLaMA3.2-1B-Instruct	Vanilla LLM	43.00	43.00	0.66	3.66	21.00	9.00	42.33	16.00	10.66	30.33	21.96
	T-RAG	30.00	34.00	7.33	8.33	21.00	24.00	45.33	30.33	48.66	31.33	28.03
	P-RAG	**45.00**♢	**43.66**♢	2.00♢	5.00	**23.66**	14.66♣♢	48.66♣	21.00♣♢	23.33♣♢	29.66	25.56♣♢
	PT-RAG	36.33♢	37.33	**7.66**♣	**9.66**♣	22.00	25.33♣	**51.33**♣♢	**31.00**♣	48.66♣	**36.66**♣♢	**30.60**♣♢
	DyP-RAG	42.66♢	33.00♣	1.33♢	4.00♢	21.00	11.00♢	44.00	14.66♢	12.33♢	31.66	21.56♢
	DyPT-RAG	35.66♣♢	31.66♣	7.00♣	8.66♣	21.66	**25.66**♣	44.00	28.00♣	**49.00**♣	34.00	28.53♣
Qwen2.5-1.5B-Instruct	Vanilla LLM	**28.33**	29.33	0.66	5.00	14.00	7.00	41.00	13.33	13.00	27.66	17.93
	T-RAG	24.33	21.66	5.66	4.00	14.66	23.33	46.66	27.33	**50.00**	23.33	24.10
	P-RAG	27.66	**34.66**♣♢	2.66♣	5.00	**16.33**	10.00♢	38.00♢	14.33♢	21.33♣♢	31.66♢	20.16♣♢
	PT-RAG	28.00	27.33♢	**7.66**♣	**8.00**♢	15.66	**25.33**♣	**52.33**♣♢	**31.66**♣♢	46.00♣	28.99♢	**27.10**♣♢
	DyP-RAG	23.33	34.33♢	0.66♢	5.00	14.66	7.33♢	39.66♢	13.00♢	14.00♢	**33.33**♣♢	18.53♢
	DyPT-RAG	25.33	25.00	6.33♣	7.66♢	13.66	22.33♣	48.66♣	26.66♣	47.00♣	31.33♢	25.40♣♢
Qwen2.5-7B-Instruct	Vanilla LLM	49.66	47.66	1.66	7.00	25.00	14.00	61.66	20.33	18.00	33.66	27.86
	T-RAG	45.00	41.33	10.66	8.00	22.66	31.66	54.65	34.66	36.66	26.00	31.33
	P-RAG	**56.33**♣♢	**49.00**♢	2.33♢	10.66♣	**28.66**♢	20.00♣♢	**63.00**♢	26.66♣♢	31.33♣	**44.66**♣♢	33.26♣♢
	PT-RAG	46.66	37.33♣	12.00♣	11.33♣♢	25.00	**35.33**♣	57.66	**38.66**♣	43.33♣♢	37.00♢	**34.43**♣♢
	DyP-RAG	49.00	46.33	2.66♢	9.33	26.00	16.00♢	62.33♢	22.66♢	25.33♣♢	35.00♢	29.46♣
	DyPT-RAG	44.00	46.66♢	**12.33**♣	**12.00**♣♢	26.33	33.00♣	56.33	37.33♣	**45.00**♣♢	29.33♢	34.23♣♢

ods spanning token-based, parametric, and hybrid paradigms: (1) Vanilla LLM (generation without retrieval); (2) T-RAG (standard token-based RAG); (3) P-RAG (static parametric RAG with offline-optimized adapters); (4) PT-RAG (hybrid P-RAG with token-based context); (5) DyP-RAG (dynamic parametric RAG with hypernetwork-generated adapters); and (6) DyPT-RAG (hybrid DyP-RAG with token-based context). Experiments are conducted on three LLMs: LLaMA3.2-1B-Instruct (Grattafiori et al., 2024), Qwen2.5-1.5B-Instruct, and Qwen2.5-7B-Instruct (Team, 2024).

4.2 Experimental Results

Table 2 reports the re-evaluation results under semantic accuracy. We observe three key findings: **(1) Hybrid paradigms achieve the best overall performance.** Consistent with prior work (Su et al., 2025b; Tan et al., 2025), combining parametric injection with explicit textual context (PT-RAG and DyPT-RAG) yields the strongest results across nearly all datasets and models. **(2) P-RAG improves over Vanilla LLM but generally underperforms T-RAG.** Although P-RAG consistently improves over the Vanilla LLM baseline, it does not consistently outperform T-RAG. Specifically, P-RAG underperforms T-RAG on smaller models, while achieving competitive results on Qwen2.5-7B-Instruct. These findings differ from prior conclusions based on F1 evaluation, suggesting that length-sensitive metrics may overestimate the effectiveness of parametric injection. **(3) DyP-RAG exhibits limited and less stable improvements.** Compared with P-RAG, DyP-RAG generally achieves weaker or less consistent gains over the Vanilla LLM baseline. This suggests that dynamically generated adapters may struggle to encode knowledge representations effectively.

5 Analyzing Knowledge Encoding

Beyond task performance, an important question is how parametric RAG internally encodes and utilizes external knowledge. This section addresses **RQ2**: *How effectively do parametric representations encode external knowledge, and how does parametric injection influence LLM computation?*

Analyzing parametric knowledge encoding using existing datasets is challenging because many facts may already exist in the LLM's internal memory. To enable controlled analysis, we construct an unseen-knowledge benchmark containing information beyond the models' training cutoff.

Unseen-Knowledge Benchmark. We construct *News25-QA*, a benchmark consisting of 300 news articles published in 2025, together with factual and multi-hop QA pairs grounded in these articles. In total, News25-QA contains 600 QA pairs and 807 passages for knowledge injection. Additional construction details are provided in Appendix B.3.

Inference and Parameterization. During inference, we provide the model with the question and ground-truth passages, bypassing retrieval to eliminate retrieval-related noise. All other settings remain consistent with Section 4.1.

5.1 Evaluation on Unseen-Knowledge

Table 3 reports results on News25-QA, and we observe these key findings: **(1) P-RAG can inject unseen factual knowledge into the model.** Compared with Vanilla LLM, P-RAG consistently improves performance, indicating that parametric

Table 3: Performance comparison on News25-QA dataset. Symbols ♣ and ♢ are defined in Table 2.

LLM	Method	News25-QA		
		Factual QA	Multihop QA	Average
LLaMA3.2-1B-Instruct	Vanilla LLM	9.66	7.33	8.50
	T-RAG	71.66	56.66	64.16
	P-RAG	13.00♣♢	14.00♣♢	13.50♣♢
	PT-RAG	**78.33**♣♢	**64.00**♣♢	**71.16**♣♢
	DyP-RAG	10.00♢	8.33♢	9.16♢
	DyPT-RAG	75.00♣	57.66♣	66.33♣
Qwen2.5-1.5B-Instruct	Vanilla LLM	5.66	8.66	7.16
	T-RAG	70.00	57.99	64.00
	P-RAG	7.33♢	11.00♢	9.16♣♢
	PT-RAG	**75.66**♣♢	**65.00**♣♢	**70.33**♣♢
	DyP-RAG	3.33♣♢	8.66♢	6.00♢
	DyPT-RAG	75.33♣♢	63.66♣♢	69.50♣♢
Qwen2.5-7B-Instruct	Vanilla LLM	13.66	18.00	15.83
	T-RAG	74.00	65.66	69.83
	P-RAG	21.66♣♢	19.00♢	20.33♣♢
	PT-RAG	**80.00**♣♢	**74.33**♣♢	**77.16**♣♢
	DyP-RAG	14.00♢	16.66♢	15.33♢
	DyPT-RAG	76.33♣	68.33♣	72.33♣

representations can encode external knowledge. Figure 1 further shows that P-RAG can answer questions about post-cutoff events that the base LLM fails to answer. **(2) DyP-RAG shows limited knowledge injection ability.** DyP-RAG achieves only marginal gains over Vanilla LLM, with no statistically significant improvements in most settings. This indicates that the generated adapters may provide limited capability for injecting genuinely new knowledge. **(3) Parametric encoding suffers from limited fidelity.** Both P-RAG and DyP-RAG remain far below T-RAG, especially on factual QA. These indicate that current parametric representations still lose considerable fine-grained information compared to explicit textual context. **(4) Hybrid paradigms achieve the best overall performance.** Both PT-RAG and DyPT-RAG consistently outperform T-RAG. This suggests that, while standalone parametric injection is insufficient for precise knowledge integration, the injected parameters still provide useful complementary signals that enhance generation. Notably, smaller models equipped with PT-RAG can even achieve performance **comparable to or better** than larger models using T-RAG, suggesting that parametric injection can partially compensate for limited model capacity. **(5) We also observe a preliminary scaling trend.** The gains of P-RAG over Vanilla LLM generally increase with model size (including Qwen2.5-14B results in Table 7), suggesting that larger models may provide more parameter spaces for absorbing knowledge. In contrast, the gains of PT-RAG over T-RAG do not exhibit a clear scaling trend, likely due to stronger LLMs already utilize textual context effectively.

Passages: ... Tamar Braxton said Tuesday that she "almost died" in a weekend accident that she doesn't remember ... Braxton wrote in an Instagram post. "I fractured my nose, lost some teeth and mobility."...
Question: What did Tamar Braxton fracture in the accident?
Answer: her nose

- **Vanilla LLM Output:** Tamar Braxton fractured her ankle in an accident.
  - [Precision=0.14, Recall=0.5, F1=0.22, HasAnswer=0]
- **T-RAG Output:** her nose, as she fractured it and lost some teeth.
  - [Precision=0.2, Recall=1.0, F1=0.33, HasAnswer=1]
- **P-RAG Output:** her nose.
  - [Precision=1.0, Recall=1.0, F1=1.0, HasAnswer=1]
- **DyP-RAG Output:** 2016. Tamar Braxton fractured her ankle in an accident.
  - [Precision=0.13, Recall=0.5, F1=0.2, HasAnswer=0]

Figure 1: An example from the News25-QA dataset involving events occurred after the LLM's training cutoff.

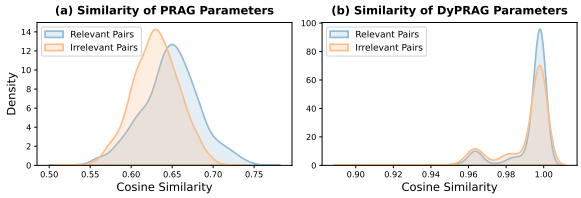

Figure 2: Cosine similarity distributions of parametric representations on Qwen2.5-1.5B-Instruct: (a) flattened P-RAG LoRA parameters (`up_proj`); and (b) flattened DyP-RAG LoRA parameters (`up_proj`).

5.2 Specificity of Parametric Representations

We further investigate the encoding mechanism by examining the specificity of the parametric representations. Intuitively, if injected parameters faithfully encode document knowledge, representations derived from semantically related passages should be more similar than those from unrelated passages. To evaluate this property, we construct two groups of passage pairs from News25-QA and then compute cosine similarities between their parametric representations. For P-RAG and DyP-RAG, we flatten the LoRA parameters and compute cosine similarity averaged across layers. Figure 2 presents the similarity distributions for the `up_proj` module of Qwen2.5-1.5B-Instruct. Results for other models, FFN modules, and embedding-based comparisons are provided in Appendix D.3.

As shown in Figure 2(a), **P-RAG exhibits semantic specificity**: relevant passage pairs consistently achieve higher similarity than irrelevant pairs. This indicates that the parameters indeed encode document-dependent semantic information. However, compared with the embedding similarity in Figure 9 (Appendix D.3), the P-RAG distributions still exhibit substantial overlap. This suggests that parameterization preserves coarse semantic signals but **loses considerable fine-grained information**.

In contrast, the distributions of DyP-RAG in Figure 2(b) almost completely overlap, with similarities for both relevant and irrelevant pairs concentrated near 1.0. Combined with the weak unseen-knowledge performance, this suggests that **the hy-**

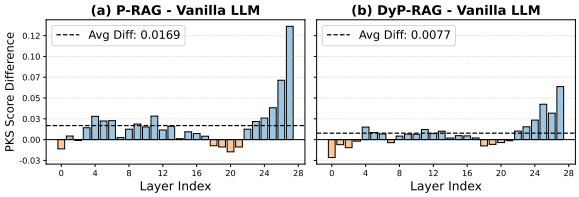

Figure 3: Layer-wise PKS differences on Qwen2.5-1.5B-Instruct. Values denote the PKS difference between the target method and Vanilla LLM. Positive bars indicate stronger parametric knowledge contribution.

pernetwork may fail to effectively condition on document-specific semantics. Instead of generating highly document-dependent parameters, DyP-RAG appears to produce largely homogeneous adapters, indicating a potential form of hypernetwork collapse (Appendix D.4 provides further analysis of this phenomenon).

5.3 Layer-wise Effects of Parametric Injection

To understand how parametric injection influences the internal computation of LLMs, we analyze the **parametric knowledge score (PKS)** (Sun et al., 2024), which quantifies the contribution of each FFN layer to the model's predictive distribution. Following prior work, PKS is computed as the Jensen–Shannon divergence between the token distributions before and after an FFN block. Higher PKS values indicate larger shifts in the predictive distribution and therefore stronger parametric knowledge contribution. We provide the full formulation in Appendix B.4.

We conduct this analysis on News25-QA using Qwen2.5-1.5B-Instruct. Figure 3 shows the layer-wise PKS differences for: (a) P-RAG minus Vanilla LLM; and (b) DyP-RAG minus Vanilla LLM. Results for other models, methods, and comparison settings are provided in Appendix D.5.

We observe two key patterns. **(1) Parametric injection increases parametric knowledge contribution.** Compared with non-parametric baselines, parametric methods generally exhibit positive PKS shifts across layers, indicating stronger parametric contribution during generation (similar trends for hybrid paradigms are reported in Appendix D.5). Moreover, P-RAG produces larger shifts than DyP-RAG, consistent with its stronger knowledge injection ability observed in Section 5.1. **(2) The effect is concentrated in deeper layers.** For both methods, the PKS gains are primarily concentrated in the later layers rather than uniformly distributed across the network. This suggests that parametric injection mainly affects high-level semantic integration, likely encoding abstract document-level priors that help guide the final prediction.

Table 4: Performance on multi-hop QA with gold passages. "With QA-LoRA" adds a generic QA task LoRA to distinguish parametric document knowledge from task adaptation. Symbols ♣ and ♢ follow Table 2.

LLM	Method	Without QA-LoRA		With QA-LoRA	
		2Wiki	HotpotQA	2Wiki	HotpotQA
LLaMA3.2-1B-Instruct	Vanilla LLM	21.00	16.00	20.66	16.66
	T-RAG	39.33	63.66	45.66	63.66
	P-RAG	23.33♢	21.66♣♢	23.66♢	22.33♣♢
	PT-RAG	**46.00**♣♢	**69.33**♣♢	**46.33**♣	**69.33**♣♢
Qwen2.5-1.5B-Instruct	Vanilla LLM	14.00	13.33	24.66	15.66
	T-RAG	28.33	53.66	**47.33**	63.00
	P-RAG	18.33♣♢	15.00♢	18.66♣♢	16.00♢
	PT-RAG	**40.00**♣♢	**67.00**♣♢	40.00♣♢	**67.33**♣
Qwen2.5-7B-Instruct	Vanilla LLM	25.00	20.33	28.00	20.33
	T-RAG	56.99	67.33	54.33	73.00
	P-RAG	30.66♣♢	32.33♣♢	30.66♢	32.66♣♢
	PT-RAG	**64.00**♣♢	**80.66**♣♢	**64.00**♣♢	**80.33**♣♢

5.4 Impact on Multi-hop Reasoning

P-RAG originally motivates parametric injection as a mechanism for deeper knowledge interaction, which may be particularly beneficial for tasks requiring evidence integration and multi-hop reasoning. Our findings above partially support this intuition, showing that injected parameters encode high-level semantic signals and primarily influence deeper layers. However, whether these signals truly improve multi-hop reasoning remains unclear, especially given the loss of fine-grained factual information during parameterization. Meanwhile, since P-RAG is trained with a QA-style objective, part of its gains and high-level semantic effects may also arise from generic task adaptation. We therefore investigate: (1) whether parametric injection benefits multi-hop reasoning; and (2) whether gains stem from generic task adaptation.

We evaluate on two multi-hop QA datasets: HotpotQA (Yang et al., 2018) and 2WikiMultihopQA (Ho et al., 2020). To isolate reasoning ability from retrieval quality, we use gold passages for both parametric encoding and explicit context. To disentangle document knowledge from generic task adaptation, we further introduce **QA-LoRA**: a general-purpose task LoRA trained on 400 QA instances from the same training corpus under identical settings, but without document-specific parameterization. This allows us to evaluate how much of the gains comes from injected document knowledge rather than generic QA adaptation. Given the weak document conditioning observed for DyP-RAG in Section 5.2, we focus on P-RAG here and report DyP-RAG results in Appendix D.6.

Table 4 presents the results. We observe three key findings: **(1) Standalone parametric encod-**

ing struggles with multi-hop reasoning. P-RAG consistently underperforms T-RAG. This suggests that, due to information loss during parameterization, P-RAG alone fails to retain the fine-grained details required for precise multi-step reasoning compared with token-based context. **(2) Parametric injection benefits multi-hop reasoning when combined with explicit context.** PT-RAG consistently achieves the best performance. Moreover, its gains over T-RAG are noticeably larger than those observed in Table 2, suggesting that the injected high-level semantic signals help the model better integrate and reason over multiple pieces of contextual evidence. **(3) The gains cannot be explained solely by generic QA adaptation.** Adding the QA-LoRA to P-RAG or PT-RAG results in only marginal changes, indicating that P-RAG already adapts well to the task format. However, even when Vanilla LLM and T-RAG are equipped with QA-LoRA, they still fail short of PT-RAG. This suggests that the gains primarily stem from document-specific knowledge encoded in the injected parameters, rather than generic task-adaptation effects.

6 P-RAG under Critical RAG Challenges

Beyond knowledge injection performance, a practical RAG system must also remain reliable under challenging conditions, such as conflicting evidence, noisy retrieval, and task transfer. While previous sections show that P-RAG can encode external knowledge to some extent, it remains unclear whether such parametric representations behave robustly under these realistic scenarios. This section addresses **RQ3**: *How does P-RAG behave under key RAG challenges, including knowledge conflicts, retrieval noise, and cross-task generalization?*

Based on the findings in Section 5, which suggest that DyP-RAG struggles to encode document-specific semantics, we focus on P-RAG in this section. Unless otherwise specified, all experiments follow the setup in Section 4.1.

6.1 Faithfulness under Knowledge Conflicts

We evaluate the model's ability to prioritize external evidence over its internal knowledge under factual conflicts, a property commonly referred to as context faithfulness. We use the **ConFiQA** dataset (Bi et al., 2024), which constructs counterfactual contexts by replacing key entities in the original passages with plausible but factually inconsistent alternatives. We use the first 900 instances and apply the counterfactual passages for both parameterization and inference. Performance is measured by accuracy against the counterfactual answers, testing whether the model follows the provided evidence rather than its internal knowledge.

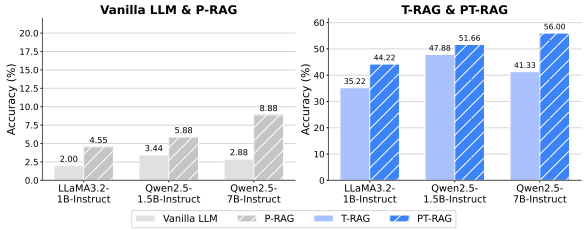

Figure 4: Performance on the ConFiQA dataset. Accuracy is computed with respect to the counterfactual answers. For each model, every pair of methods differs significantly ($p < 0.05$, two-sided paired t-test).

Figure 4 reports results on ConFiQA. We observe two key findings: **(1) P-RAG can alter the model's internal factual priors.** Compared with Vanilla LLM, P-RAG produces significantly more counterfactual answers, showing that parametric injection can override pre-trained knowledge with external information. However, P-RAG still underperforms T-RAG, indicating that parametric representations remain less reliable for precise entity-level knowledge editing. **(2) Parametric injection improves context faithfulness.** PT-RAG consistently achieves the best performance across all models. This suggests that injected parameters help reduce the conflict between the model's internal knowledge and the provided evidence, enabling the model to follow external context more faithfully.

6.2 Robustness to Retrieval Noise

To evaluate robustness against noisy retrieval, we inject controlled artificial noise into the top-3 BM25-retrieved passages on the four QA datasets from Section 4.1. We construct four settings: (1) *BM25-Top3*: original retrieval results; (2) *Rank3Noise*: replacing the third-ranked (lowest-relevance) passage with a random irrelevant document; (3) *Rank1Noise*: replacing the top-ranked passage; and (4) *AllNoise*: replacing all retrieved passages. We report average HasAnswer accuracy across datasets for each noise condition.

Figure 5 illustrates performance trends under increasing retrieval noise. We observe two key findings: **(1) Models can suppress irrelevant parametric knowledge.** As noisy passages increase, P-RAG performance gradually declines. However, even under the *AllNoise* setting, where all injected parameters are derived from irrelevant passages,

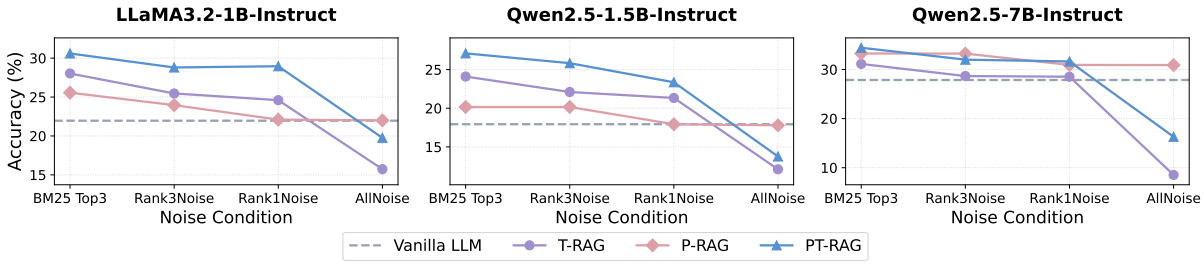

Figure 5: Performance trends under increasing levels of retrieval noise. The x-axis represents noise severity, ranging from the original retrieved passages (BM25-Top3) to completely irrelevant inputs (AllNoise).

P-RAG does not underperform Vanilla LLM. This indicates that the model can identify and largely ignore unhelpful injected parametric knowledge instead of being severely misled by it. **(2) Parametric injection enhances robustness to noisy context.** PT-RAG consistently outperforms T-RAG across all noise conditions. This suggests that parametric injection mitigates the interference from retrieval noise. By internalizing document-level semantics, the model becomes less distracted by irrelevant context during generation.

6.3 Generalization to Non-QA Task

While T-RAG naturally supports diverse tasks through instruction prompting, it remains unclear whether QA-trained parametric representations can generalize beyond QA. We evaluate this transferability on two non-QA tasks: (1) **Fact Verification** using FEVER (Thorne et al., 2018), evaluated by accuracy; and (2) **Slot Filling** using Zero-Shot-RE (Levy et al., 2017), evaluated by F1. For each instance, we retrieve top-3 passages using BM25 and apply the standard P-RAG parameterization pipeline trained with QA-style objectives, while inference uses task-specific prompts.

Figure 6 presents results on fact verification and slot filling. The overall trends are consistent with the QA setting: P-RAG improves over Vanilla LLM but generally remains below T-RAG, especially on slot filling, which requires precise entity-level reproduction. Meanwhile, PT-RAG consistently achieves the best performance. These results further suggest that parametric representations primarily capture transferable high-level semantic information, enabling generalization to downstream tasks beyond the original QA training objective.

6.4 Discussion: Practical Challenges

While parametric RAG offers several advantages, our analysis reveals important practical limitations that currently hinder its scalability and deployment. **(1) Document-specific parameter storage is expensive.** Under the setting used in this work (Qwen2.5-1.5B with LoRA rank $r = 2$), each document-specific adapter contains approximately **1.76M** parameters, requiring around **6.7 MB** storage in FP32 precision. As a result, even a relatively small corpus of 900 documents requires nearly **6 GB** storage, substantially exceeding the storage footprint of vector indices in standard T-RAG. **(2) Practical inference overhead remains high.** Although parametric RAG theoretically reduces inference complexity by removing long document contexts from the prompt, existing serving systems are not optimized for frequent LoRA loading and hot-swapping. In practice, the system-level overhead introduced by dynamically managing document-specific adapters can outweigh the theoretical efficiency gains from shorter contexts, making current P-RAG systems slower than T-RAG in high-throughput scenarios. We provide a more detailed complexity analysis in Appendix C.

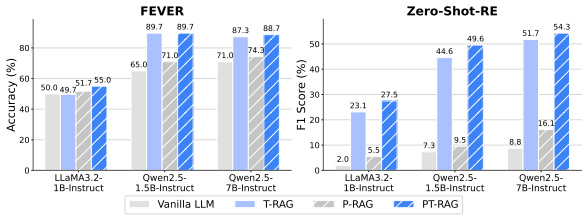

Figure 6: Performance on non-QA tasks: Fact Verification (FEVER) and Slot Filling (Zero-Shot-RE).

7 Conclusion

This work presents a systematic analysis of parametric RAG. We find that parametric injection provides useful high-level semantic guidance, improving multi-hop reasoning, context faithfulness, and robustness to retrieval noise. However, current parametric representations still suffer from substantial information loss and remain less faithful than explicit token-based context. These findings provide insights into the strengths and limitations of parametric RAG, offering guidance for future parameter-based knowledge integration methods.

Limitations

While our study provides systematic insights into parametric RAG, several limitations remain. First, due to computational constraints, our experiments mainly focus on models up to 14B parameters. Although the observed trends are consistent across multiple architectures and settings, whether they fully generalize to much larger models (e.g., 70B+) remains an important direction for future work. Second, our analysis is restricted to text-only scenarios. Extending parametric knowledge injection to multimodal RAG settings is another promising direction for future research. Finally, because current parametric RAG methods incur substantial training and storage overhead, our evaluations are conducted on relatively small-scale datasets. Future work could explore more efficient parameterization pipelines and validate these findings on larger-scale benchmarks.

Ethical Considerations

All models and data used in this study are publicly available and free of personally identifiable information or offensive content, introducing no additional societal risks. In line with the principles of open research and reproducibility, all of our code and data are publicly available at `https://anonymous.4open.science/r/PRAG-Analysis`.

We used ChatGPT solely for language polishing and proofreading. It was not involved in idea generation, experimental design, or scientific analysis.

References

Baolong Bi, Shaohan Huang, Yiwei Wang, Tianchi Yang, Zihan Zhang, Haizhen Huang, Lingrui Mei, Junfeng Fang, Zehao Li, Furu Wei, and 1 others. 2024. Context-dpo: Aligning language models for context-faithfulness. *arXiv preprint arXiv:2412.15280*.

Bowen Cao, Deng Cai, and Wai Lam. 2025. Infiniteicl: Breaking the limit of context window size via long short-term memory transformation. *arXiv preprint arXiv:2504.01707*.

Jinwen Chen, Hainan Zhang, Liang Pang, Yongxin Tong, Haibo Zhou, Yuan Zhan, Wei Lin, and Zhiming Zheng. 2025. Privacy-preserving reasoning with knowledge-distilled parametric retrieval augmented generation. *arXiv preprint arXiv:2509.01088*.

Tong Chen, Hao Fang, Patrick Xia, Xiaodong Liu, Benjamin Van Durme, Luke Zettlemoyer, Jianfeng Gao, and Hao Cheng. Generative adapter: Contextualizing language models in parameters with a single forward pass. In *The Thirteenth International Conference on Learning Representations*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*.

Qian Dong, Qingyao Ai, Hongning Wang, Yiding Liu, Haitao Li, Weihang Su, Yiqun Liu, Tat-Seng Chua, and Shaoping Ma. 2025. Decoupling knowledge and context: An efficient and effective retrieval augmented generation framework via cross attention. In *Proceedings of the ACM on Web Conference 2025*, pages 4386–4395.

Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, Jiayi Chen, Yuchen Ni, Junjie Ye, Cheng Zhang, and 1 others. 2026. Cl-bench: A benchmark for context learning. *arXiv preprint arXiv:2602.03587*.

Xingyu Fan, Feifei Li, Wenhui Que, and Hailong Li. 2025. One agent to serve all: a lite-adaptive stylized ai assistant for millions of multi-style official accounts. *arXiv preprint arXiv:2509.17788*.

William Fleshman and Benjamin Van Durme. 2025. Lora-augmented generation (lag) for knowledge-intensive language tasks. *arXiv preprint arXiv:2507.05346*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Google. 2025. A new era of intelligence with gemini 3.

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*.

Xanh Ho, Jiahao Huang, Florian Boudin, and Akiko Aizawa. 2025. Llm-as-a-judge: Reassessing the performance of llms in extractive qa. *arXiv preprint arXiv:2504.11972*.

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.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

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.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and l others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Zihan Liao, Jun Wang, Hang Yu, Lingxiao Wei, Jianguo Li, and Wei Zhang. 2025. E2llm: Encoder elongated large language models for long-context understanding and reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 19212–19241.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. Memgpt: Towards llms as operating systems.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Weihang Su, Qingyao Ai, Jingtao Zhan, Qian Dong, and Yiqun Liu. 2025a. Dynamic and parametric retrieval-augmented generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4118–4121.

Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. 2025b. Parametric retrieval augmented generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1240–1250.

Zhan Su, Fengran Mo, and Jian-yun Nie. 2025c. Parametric retrieval-augmented generation using latent routing of lora adapters. *arXiv preprint arXiv:2511.17044*.

Zhongxiang Sun, Xiaoxue Zang, Kai Zheng, Yang Song, Jun Xu, Xiao Zhang, Weijie Yu, and Han Li. 2024. Redeep: Detecting hallucination in retrieval-augmented generation via mechanistic interpretability. *arXiv preprint arXiv:2410.11414*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. 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 641–651.

Yuqiao Tan, Shizhu He, Huanxuan Liao, Jun Zhao, and Kang Liu. 2025. Dynamic parametric retrieval augmented generation for test-time knowledge enhancement. *arXiv preprint arXiv:2503.23895*.

Minghao Tang, Shiyu Ni, Jiafeng Guo, and Keping Bi. 2025. Injecting external knowledge into the reasoning process enhances retrieval-augmented generation. *arXiv preprint arXiv:2507.19333*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Xi Wang, Taketomo Isazawa, Liana Mikaelyan, and James Hensman. 2024a. Kblam: Knowledge base augmented language model. *arXiv preprint arXiv:2410.10450*.

Yan Wang, Dongyang Ma, and Deng Cai. 2024b. With greater text comes greater necessity: Inference-time training helps long text generation. *arXiv preprint arXiv:2401.11504*.

Zihao Wang, Anji Liu, Haowei Lin, Jiaqi Li, Xiaojian Ma, and Yitao Liang. 2024c. Rat: Retrieval augmented thoughts elicit context-aware reasoning in long-horizon generation. *arXiv preprint arXiv:2403.05313*.

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.

Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. 2022. Retrieval-enhanced machine learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2875–2886.

Hengran Zhang, Keping Bi, Jiafeng Guo, Jiaming Zhang, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. 2025a. Distilling a small utility-based passage selector to enhance retrieval-augmented generation. *arXiv preprint arXiv:2507.19102*.

Hengran Zhang, Minghao Tang, Keping Bi, Jiafeng Guo, Shihao Liu, Daiting Shi, Dawei Yin, and Xueqi Cheng. 2025b. Utility-focused llm annotation for retrieval and retrieval-augmented generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1683–1702.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*.

Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and Qianli Ma. 2025. Lifelong learning of large language model based agents: A roadmap. *arXiv preprint arXiv:2501.07278*.

A Related Work

A.1 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) improves large language models (LLMs) by incorporating external knowledge, enhancing performance on knowledge-intensive tasks while reducing hallucinations (Lewis et al., 2020; Zamani et al., 2022; Izacard et al., 2023). Its effectiveness largely depends on two factors: retrieval quality (Shi et al., 2023; Zhang et al., 2025b,a) and knowledge integration capability (Zhang et al., 2024; Sun et al., 2024). Existing approaches for integrating retrieved knowledge into LLMs can be broadly divided into three categories: **(1) Token-level augmentation** (Trivedi et al., 2022; Wang et al., 2024c; Tang et al., 2025): Retrieved documents are directly inserted into the prompt, often with additional context refinement. Although simple and effective, this paradigm is constrained by context length limits and incurs substantial inference overhead for long contexts. Moreover, LLMs often struggle to effectively utilize information in extremely long inputs. **(2) Embedding-level fusion** (Izacard and Grave, 2020; Dong et al., 2025; Wang et al., 2024a): Documents are encoded into embeddings and injected into the LLM through mechanisms such as cross-attention, reducing token-level context overhead. However, the interaction between the model and retrieved knowledge remains relatively shallow, often limiting performance, especially with limited number of documents. **(3) Parametric-level adaptation** (Su et al., 2025b; Tan et al., 2025; Chen et al., 2025; Su et al., 2025c): A recent paradigm that converts retrieved documents into pluggable model parameters (e.g., LoRA adapters) and injects them directly into the LLM. P-RAG (Su et al., 2025b) argues that parameter-level integration enables deeper knowledge interaction while avoiding long-context inference costs. Our work focuses on this third direction. Specifically, we conduct a systematic analysis of parametric RAG, centering on P-RAG (Su et al., 2025b) and DyP-RAG (Tan et al., 2025), to better understand how parametric injection encodes and utilizes external knowledge.

A.2 Parametric RAG

Parametric RAG (Su et al., 2025b; Tan et al., 2025; Chen et al., 2025; Su et al., 2025c) incorporates external knowledge by injecting document-dependent parameters into LLMs instead of appending retrieved documents to the input context. Typically, each document is converted into a lightweight parameter module such as a LoRA adapter (Hu et al., 2022), which is dynamically attached during inference. P-RAG (Su et al., 2025b) obtains document-specific parameters by generating synthetic training data for each document and optimizing a dedicated LoRA module. While effective, this approach requires training and storing a separate LoRA for every document, resulting in substantial storage and preprocessing costs. To improve efficiency, DyP-RAG (Tan et al., 2025) introduces a hypernetwork that directly maps documents to LoRA parameters at inference time, eliminating per-document training. PolyP-RAG (Su et al., 2025c) further proposes representing an entire corpus using a small set of shared latent LoRA experts that are dynamically

routed according to the query. Despite promising empirical results, existing studies mainly focus on downstream performance while leaving the underlying mechanism of parametric injection underexplored. In particular, it remains unclear whether injected parameters truly preserve document knowledge, or primarily function as generic task adapters. In contrast, our work systematically investigates the representation properties and knowledge integration behavior of parametric RAG.

B Implementation Details

B.1 Inference and Parameterization Setup

While the original P-RAG setup utilized few-shot prompts during both parameterization and inference on specific datasets, we argue that this practice risks conflating task-specific patterns with the factual content of documents. To strictly evaluate the effectiveness of knowledge injection, we enforce a zero-shot setting across all experiments.

Both P-RAG and DyP-RAG apply LoRA adapters to FFN layers with rank $r = 2$ and scaling factor $\alpha = 32$.

For P-RAG, we follow the original training pipeline (Su et al., 2025b). Each document is augmented with one rewritten variant and three QA pairs. LoRA adapters are optimized for one epoch with a learning rate of 3×10^{-4}.

For DyP-RAG, we use the official pre-trained checkpoints released by the authors (Tan et al., 2025) for Qwen2.5-1.5B-Instruct and LLaMA3.2-1B-Instruct. For Qwen2.5-7B-Instruct, we reproduce the model using the official training code and pipeline under the same LoRA configuration.

B.2 Dataset Details in Reproduction

Following prior work (Su et al., 2025b; Tan et al., 2025), we evaluate on the first 300 questions from each dataset. For 2WikiMultihopQA and HotpotQA, the "Total" column in Table 2 reports results on the first 300 questions of the full dataset, while each sub-task column reports results on the first 300 questions within that sub-task. For retrieval, we use BM25 over the Wikipedia corpus from DPR (Karpukhin et al., 2020) and retrieve the top-3 passages for each query.

B.3 Construction of News25-QA

We collect 300 news articles published in 2025 from major news outlets (e.g., BBC News[1]), ensuring that the content falls beyond the knowledge cutoff dates of all evaluated LLMs. We then use Qwen2.5-32B-Instruct (Team, 2024) to generate two types of QA pairs grounded in each article: (1) factual QA targeting specific details; and (2) multi-hop QA requiring reasoning over multiple facts within the article. This process produces 600 QA pairs in total. Finally, articles are segmented into non-overlapping passages based on sentence boundaries, with chunk sizes matched to the retrieval setup in Section B.2. The resulting corpus contains 807 passages used for parametric encoding and explicit context.

B.4 Details of PKS Computation

To analyze how parametric injection affects the internal computation of LLMs, we adopt the **Parametric Knowledge Score (PKS)** proposed by Sun et al. (Sun et al., 2024). PKS measures the contribution of each FFN layer by quantifying how much the token prediction distribution changes after passing through the FFN block.

Formally, for generation step n at transformer layer l, let $\mathbf{h}^{\text{in}}_{n,l}$ and $\mathbf{h}^{\text{out}}_{n,l}$ denote the hidden states before and after the FFN module, respectively. The predictive distribution associated with hidden state $\mathbf{h}$ is defined as:

$$q(\mathbf{h}) = \text{softmax}(\text{LayerNorm}(\mathbf{h})W_U), \quad (4)$$

where W_U denotes the unembedding matrix. The PKS value for token position n and layer l is then computed as the Jensen–Shannon divergence (JSD) between the two predictive distributions:

$$P^l_n = \text{JSD}\left(q(\mathbf{h}^{\text{in}}_{n,l}) \parallel q(\mathbf{h}^{\text{out}}_{n,l})\right). \quad (5)$$

A larger PKS value indicates that the FFN block causes a greater shift in the token distribution, suggesting a stronger contribution of parametric knowledge at that layer. The final PKS score for layer l is obtained by averaging over all generated tokens:

$$\text{PKS}(l) = \frac{1}{N}\sum_{n=1}^{N} P^l_n, \quad (6)$$

where N denotes the number of generated tokens.

Following prior work (Sun et al., 2024), we compute PKS on the generated responses for each method and compare the layer-wise differences between parametric and non-parametric settings.

C Complexity Analysis

We first instantiate the storage overhead using Qwen2.5-1.5B-Instruct ($L = 28$, $h = 1536$, $d = 8960$) with LoRA rank $r = 2$. For FFN-only adap-

[1] https://www.bbc.com/news

Table 5: Results under F1 score (%). Bold and underlined numbers denote the best and second-best results.

LLM	Method	2WikiMultihopQA					HotpotQA			PopQA	CWQ	Average
		Compare	Bridge	Infer	Compose	Total	Bridge	Compare	Total			
LLaMA3.2-1B-Instruct	Vanilla LLM	41.15	44.41	16.65	4.27	22.39	11.21	39.97	15.59	4.34	36.28	23.63
	T-RAG	24.41	33.19	25.65	9.47	20.92	20.22	41.31	25.95	15.95	37.49	25.46
	P-RAG	**45.70**	**45.71**	19.57	5.93	26.46	15.05	**48.62**	21.30	17.64	34.30	28.03
	PT-RAG	34.03	38.49	25.14	**11.07**	23.65	22.70	46.78	**28.47**	**31.87**	**40.39**	**30.26**
	DyP-RAG	45.34	45.09	20.29	5.53	**27.28**	13.55	44.74	16.85	9.52	36.66	26.49
	DyPT-RAG	29.42	36.89	**25.78**	10.27	24.08	**22.74**	40.99	25.55	30.08	38.37	28.42
Qwen2.5-1.5B-Instruct	Vanilla LLM	19.71	31.21	13.49	5.07	15.00	8.66	27.90	11.14	5.55	34.31	17.20
	T-RAG	21.02	28.84	19.45	6.52	16.74	20.43	44.65	24.72	9.99	28.23	22.06
	P-RAG	20.70	33.92	19.17	5.86	**17.62**	11.65	32.53	15.45	16.40	**35.83**	20.91
	PT-RAG	**23.43**	29.08	22.41	**9.17**	17.23	**22.31**	**45.64**	**26.61**	19.07	30.89	**24.58**
	DyP-RAG	17.49	**35.59**	16.27	5.54	17.04	9.46	34.69	12.18	10.55	35.13	19.39
	DyPT-RAG	18.94	25.33	**22.91**	9.01	14.85	20.54	42.94	24.29	**22.71**	33.57	23.51
Qwen2.5-7B-Instruct	Vanilla LLM	46.43	46.03	20.60	6.83	27.15	15.14	52.10	20.47	4.20	36.23	27.52
	T-RAG	46.11	41.92	24.84	8.44	23.98	28.29	52.03	32.05	8.57	32.07	29.83
	P-RAG	**53.28**	**48.64**	22.07	11.58	**30.77**	18.27	**57.57**	24.72	19.92	**45.77**	33.26
	PT-RAG	47.62	39.71	29.59	10.83	27.11	**31.50**	53.62	**35.08**	**27.58**	41.96	**34.46**
	DyP-RAG	48.53	45.04	24.83	10.06	29.48	16.02	56.55	22.84	7.40	38.10	29.89
	DyPT-RAG	47.72	46.85	**30.54**	**12.71**	28.55	29.29	53.56	34.06	17.31	35.89	33.65

Table 6: Inference and storage complexity comparison between different methods. Average latency denotes the wall-clock inference time per query.

Method	Storage Cost	Inference Cost	Avg. Latency (s)
Vanilla LLM	–	$\lvert R\rvert \times (\lvert q\rvert^2 \cdot \text{ATTN} + \lvert q\rvert \cdot \text{FFN})$	1.91
T-RAG	–	$\lvert R\rvert \times ((k\lvert p\rvert + \lvert q\rvert)^2 \cdot \text{ATTN} + (k\lvert p\rvert + \lvert q\rvert) \cdot \text{FFN})$	2.05
P-RAG	$M \times 3Lr(h+d)$	$\lvert R\rvert \times (\lvert q\rvert^2 \cdot \text{ATTN} + \lvert q\rvert \cdot \text{FFN}) + O(\text{LoRA Loading Overhead})$	2.29
DyP-RAG	$n \times 3Lr(h+d) + O(\text{Hypernet Parameters})$	$k \times (\lvert p\rvert^2 \cdot \text{ATTN} + \lvert p\rvert \cdot \text{FFN}) + k \times O(\text{Hypernet Translation}) + \lvert R\rvert \times (\lvert q\rvert^2 \cdot \text{ATTN} + \lvert q\rvert \cdot \text{FFN}) + O(\text{LoRA Loading Overhead})$	3.20

tation in SwiGLU-based architectures (Shazeer, 2020) (e.g., LLaMA and Qwen), LoRA is applied to the three FFN projections (`gate`, `up`, and `down`). Since each projection contains both a down-projection and an up-projection, the total parameter count for a document-specific FFN LoRA is: $3Lr(h+d)$. Substituting the Qwen2.5-1.5B-Instruct configuration gives:

$$3Lr(h+d) = 3 \times 28 \times 2 \times (1536 + 8960) \approx 1.76\text{M} \quad (7)$$

LoRA parameters per document. Under FP32 precision, this corresponds to approximately 6.7 MB storage per document. A corpus containing 900 documents therefore requires nearly 6 GB storage solely for the parametric representations.

Table 6 further compares the inference and storage complexity between methods. Let $|q|$ denote the query length, $|p|$ the average passage length, k the number of retrieved passages, and $|R|$ the generated response length. For the LLM, let L denote the number of transformer layers, h the hidden dimension, d the FFN intermediate dimension, and r the LoRA rank. We abstract the per-layer self-attention and FFN costs as ATTN $= O(h)$ and FFN $= O(h^2)$, respectively. M denotes the corpus size, and n denotes the number of passages LoRAs used to train the DyP-RAG hypernetwork. Average latency is measured as the wall-clock inference time per query using the HuggingFace Transformers framework on a single NVIDIA A100 GPU.

For T-RAG, retrieved passages are directly concatenated into the prompt, resulting in a decoding context length of $(k|p| + |q|)$. P-RAG replaces retrieved passages with injected document-specific LoRA parameters, reducing the decoding context to $|q|$. However, P-RAG additionally incurs practical serving overhead from dynamically loading and hot-swapping LoRA adapters during inference. Compared with P-RAG, DyP-RAG avoids storing document-specific LoRAs for the entire corpus. Instead, it maintain a hypernetwork trained on n passage-specific LoRAs and dynamically generates document-conditioned LoRA parameters during inference. This substantially reduces storage cost, but also introduces additional inference overhead from online document encoding and hypernetwork-based parameter translation.

D Additional Experimental Results

D.1 Reproduction Results under F1 Score

Table 5 presents experimental results evaluated using F1 score. Compared to the LLM-judged results in Table 2, these results are much closer to those reported in the original work (Su et al., 2025b; Tan et al., 2025). We attribute this discrepancy to the limitations of the F1 metric.

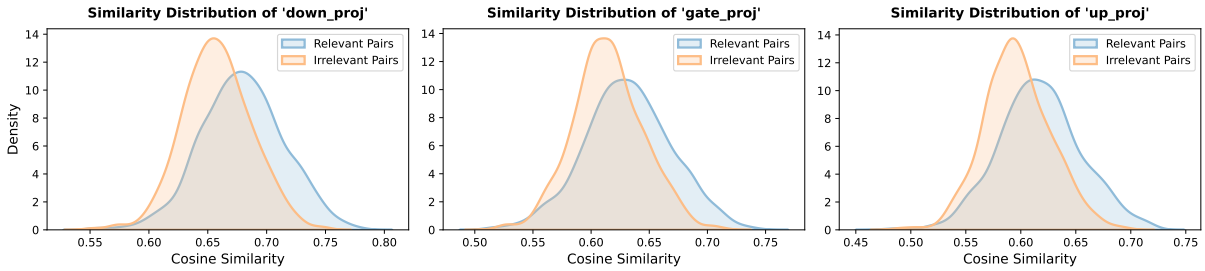

Figure 7: Cosine similarity distributions of flattened P-RAG LoRA parameters on LLaMA3.2-1B-Instruct across different FFN modules. Relevant passage pairs consistently exhibit higher similarity than irrelevant pairs.

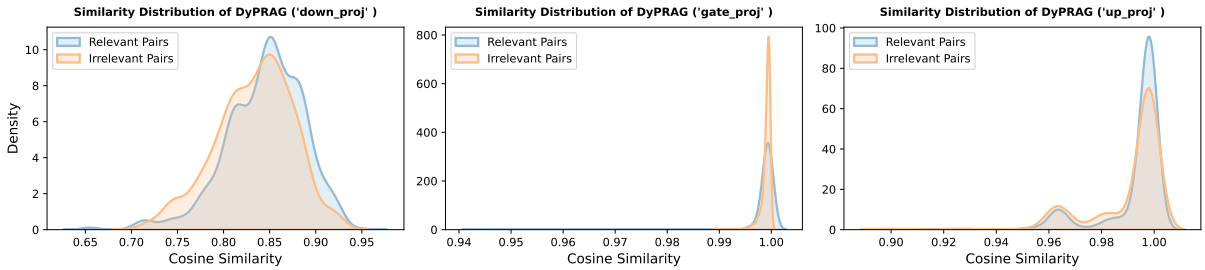

Figure 8: Cosine similarity distributions of flattened DyP-RAG LoRA parameters on Qwen2.5-1.5B-Instruct across different FFN modules. The distributions of relevant and irrelevant pairs largely overlap, with cosine similarities concentrated near 1.0.

Table 7: Performance comparison on News25-QA dataset. Symbols ♣ and ♢ are defined in Table 2.

LLM	Method	News25-QA		
		Factual QA	Multihop QA	Average
Qwen2.5-14B-Instruct	Vanilla LLM	15.66	15.00	15.33
	T-RAG	76.00	69.33	72.66
	P-RAG	22.66♣♢	24.33♣♢	23.50♣♢
	PT-RAG	**80.00**♣	**73.33**♣	**76.66**♣♢

D.2 Performance of Larger LLMs

Table 7 reports additional results on Qwen2.5-14B-Instruct. Consistent with the observations in Section 5.1, P-RAG achieves larger improvements over Vanilla LLM on stronger LLMs, while PT-RAG continues to obtain the best overall performance. These results further support the scaling trend discussed in the main text.

D.3 Similarity Analysis

To further analyze the specificity of parametric representations, we report additional similarity distributions for LLM embeddings (the last hidden state of the LLM given the passage input), different backbone models, and different FFN modules.

Figure 9 presents the cosine similarity distributions of LLM embeddings on News25-QA.

Figure 7 shows the similarity distributions of P-RAG parameters on LLaMA3.2-1B-Instruct across different FFN modules (`down_proj`, `gate_proj`, and `up_proj`). Across all modules, relevant passage pairs consistently exhibit higher similarity than irrelevant pairs, indicating that P-RAG preserves a certain degree of semantic specificity.

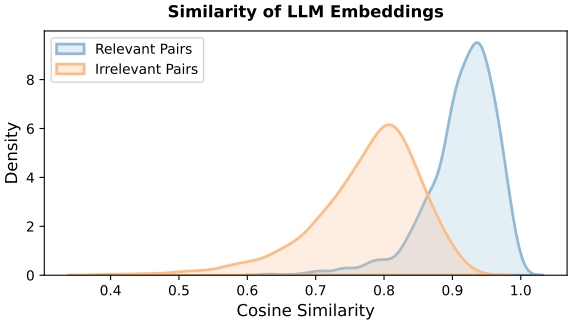

Figure 9: Cosine similarity distributions of LLM embeddings (the last hidden state) for relevant and irrelevant passage pairs on News25-QA.

Figure 8 reports the corresponding results for DyP-RAG on Qwen2.5-1.5B-Instruct. Unlike P-RAG, the similarity distributions of relevant and irrelevant pairs almost completely overlap, while the cosine similarities remain close to 1.0 across all modules. This suggests that the dynamically generated adapters exhibit very limited semantic distinguishability.

D.4 Ablation on Document Conditioning in DyP-RAG

The analyses in Sections 5.1 and 5.2 suggest that DyP-RAG may fail to effectively condition on document-specific semantics, potentially leading to a form of hypernetwork collapse. To further verify this phenomenon, we conduct an additional ablation study. Specifically, instead of feeding the original passages into the DyP-RAG hypernetwork for LoRA generation, we replace all hypernetwork

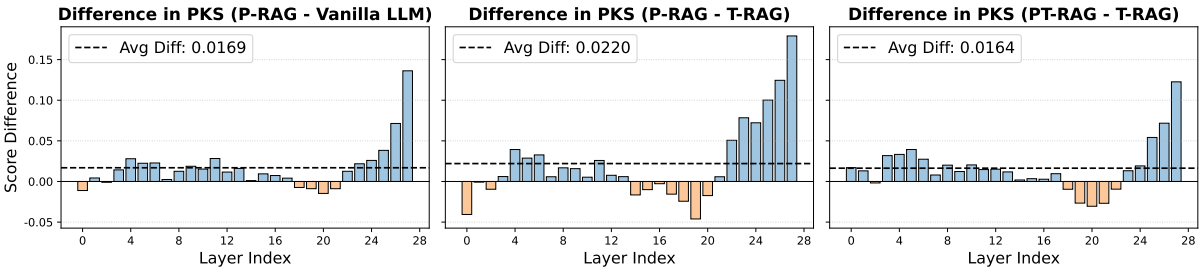

Figure 10: Layer-wise PKS differences on Qwen2.5-1.5B-Instruct. Positive values indicate a stronger parametric knowledge contribution than the baseline.

Table 8: Ablation on document conditioning in DyP-RAG. "Dummy" replaces the passages used for hypernetwork-based LoRA generation with unrelated placeholder passages.

LLM	Passages	Method	News25-QA		
			Factual QA	Multihop QA	Average
Qwen2.5-7B-Instruct	-	Vanilla LLM	13.66	18.00	15.83
		T-RAG	74.00	65.66	69.83
	Normal	DyP-RAG	14.00	16.66	15.33
		DyPT-RAG	76.33	68.33	72.33
	Dummy	DyP-RAG	12.66	19.66	16.16
		DyPT-RAG	76.66	68.00	72.33

inputs with unrelated dummy passages:

`"passage_{i}: I don't know anything."`

The generated LoRA adapters are then injected into the model during inference. For DyPT-RAG, the textual context provided to the LLM remains unchanged, and only the passages used for LoRA generation are replaced.

Table 8 reports the results on News25-QA. Surprisingly, replacing the original passages with unrelated content causes almost no performance degradation. This suggests that the generated adapters contribute little document-specific knowledge during inference. Instead, the hypernetwork appears to produce largely homogeneous adapters regardless of the input passage. Combined with the similarity analysis in Section 5.2, these findings provide further evidence that DyP-RAG may suffer from a hypernetwork collapse issue.

D.5 Layer-wise PKS Results

This section provides additional layer-wise PKS results for other models and comparison settings. Figure 11 reports the PKS differences on LLaMA3.2-1B-Instruct, comparing P-RAG against T-RAG and DyP-RAG against T-RAG. Figure 10 reports the PKS differences on Qwen2.5-1.5B-Instruct, comparing P-RAG against Vanilla LLM, P-RAG against T-RAG, and PT-RAG against T-RAG.

D.6 DyP-RAG on Multi-hop Reasoning

Table 9 reports the multi-hop QA results of DyP-RAG under gold-passage settings. DyP-RAG alone

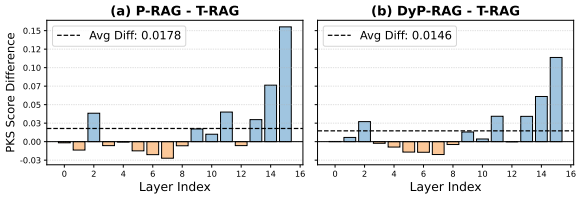

Figure 11: Layer-wise PKS differences on LLaMA3.2-1B-Instruct. Positive values indicate a stronger parametric knowledge contribution than the baseline.

achieves only marginal improvements over Vanilla LLM, while DyPT-RAG provides relatively limited gains over T-RAG compared with the stronger improvements observed for PT-RAG in Table 4. Moreover, after introducing QA-LoRA, the performance of DyP-RAG and DyPT-RAG becomes more close to the corresponding QA-LoRA-enhanced Vanilla LLM and T-RAG baselines.

Table 9: Performance of DyP-RAG on multi-hop QA with gold passages. "With QA-LoRA" adds a generic QA task LoRA to distinguish document-specific parametric knowledge from generic task adaptation.

LLM	Method	Without QA-LoRA		With QA-LoRA	
		2Wiki	HotpotQA	2Wiki	HotpotQA
LLaMA3.2-1B-Instruct	Vanilla LLM	21.00	16.00	20.66	16.66
	T-RAG	39.33	**63.66**	**45.66**	**63.66**
	DyP-RAG	23.33	15.33	19.33	16.00
	DyPT-RAG	**41.66**	62.00	44.33	62.33
Qwen2.5-1.5B-Instruct	Vanilla LLM	14.00	13.33	24.66	15.66
	T-RAG	28.33	53.66	47.33	**63.00**
	DyP-RAG	14.33	12.66	24.66	15.00
	DyPT-RAG	**37.66**	**68.33**	**47.66**	**63.00**
Qwen2.5-7B-Instruct	Vanilla LLM	25.00	20.33	28.00	20.33
	T-RAG	56.99	67.33	54.33	**73.00**
	DyP-RAG	26.33	23.00	28.66	19.33
	DyPT-RAG	**59.33**	**68.33**	**55.66**	72.66