# FUSEPRAG: FUSE-THEN-PROJECT FOR PARAMETRIC RETRIEVAL-AUGMENTED GENERATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Parametric retrieval-augmented generation incorporates retrieved evidence into language models through adapter parameters. However, existing document-wise adapter generators produce an adapter independently for each document and combine the resulting updates afterward, so cross-document relationships cannot directly shape what each adapter encodes. We investigate query-conditioned evidence fusion before parameter projection for multi-document question answering. Thus, we propose FusePRAG, a "Fuse-then-Project" framework in which a shallow fusion module jointly processes query and document embeddings, and a parameter translator maps the resulting representation to a single query-specific LoRA adapter. Training proceeds in two phases. First, the fusion module is aligned with embeddings of answer-aware evidence summaries: answer-relevant facts drawn from gold passages using training questions and answers. Second, the parameter translator is optimized for answer generation. These summaries are used only to construct Phase-1 training targets and are neither generated nor required at inference. Across four question-answering benchmarks and three backbone models, FusePRAG improves macro-average F1 over DyPRAG by 3.0–4.7 points with the same retrieved context. A capacity-matched control yields consistent incremental gains from direct document–document attention, while the single-projection design lowers adapter-construction latency at larger retrieval depths.

## 1 INTRODUCTION

Retrieval-Augmented Generation (RAG) gives language models external evidence for knowledge-intensive tasks (Guu et al., 2020; Lewis et al., 2020; Peng et al., 2024). Input-based methods provide retrieved text (Figure 1(a)) or its representations through attention-based integration (Izacard & Grave, 2021; Jiang et al., 2022) or context compression (Ge et al., 2023; Cheng et al., 2024). However, long or noisy contexts can be difficult for language models to use reliably (Liu et al., 2024; Shi et al., 2023). Parametric RAG instead encodes retrieved knowledge into model updates. PRAG (Su et al., 2025) trains and stores document-specific LoRA adapters (Hu et al., 2022) offline, whereas DyPRAG (Tan et al., 2025) uses a shared parameter translator to generate one adapter per retrieved document at inference time (Figure 1(b,c)).

The information to encode from one passage may depend on the query and on evidence in other passages. For example, a passage that does not directly answer the question may provide a link in a multi-hop reasoning chain (Yang et al., 2018; Ho et al., 2020). DyPRAG (Tan et al., 2025), however, generates each document's adapter without conditioning on the other retrieved documents and combines the updates afterward. We call this ordering "Project-then-Fuse." It does not use cross-document relationships when deciding what to encode in each update. These observations motivate two questions: does query-conditioned Fuse-then-Project improve adapter generation, and what is the incremental contribution of direct document–document attention?

We propose **FusePRAG**, a "Fuse-then-Project" framework that generates one query-specific adapter from jointly processed query and document embeddings (Figure 1(d)). A shallow fusion module first models query–document and document–document interactions, and a parameter translator maps the fused representation to LoRA parameters. Training has two phases. Phase 1 aligns the fused representation with embeddings of answer-aware evidence summaries generated from the training question, gold passages, and gold answer. These summaries define Phase 1 targets only and are not



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

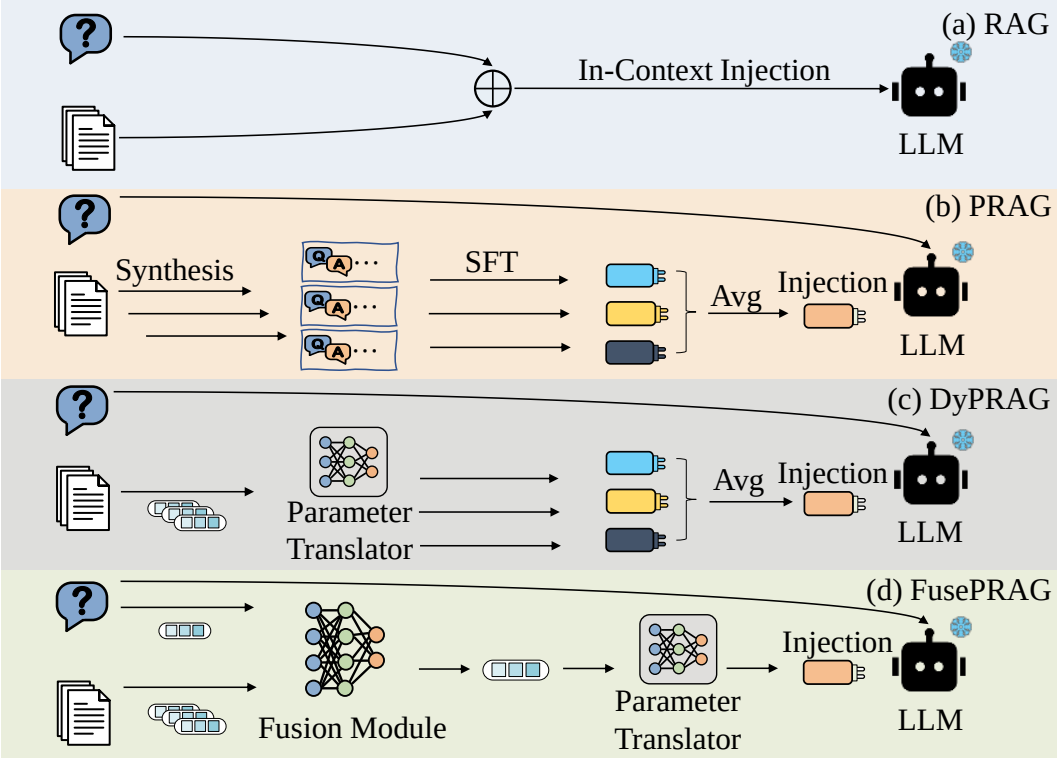


Figure 1: Comparison of RAG paradigms. (a) Standard RAG injects retrieved text into the context. (b) PRAG encodes documents into offline adapters. (c) DyPRAG independently projects documents before combining their updates. (d) FusePRAG fuses the retrieved evidence before generating a query-specific adapter.

generated or used at inference. Phase 2 trains the translator for answer generation using the gold answer tokens as supervised labels. For $k$ retrieved documents, FusePRAG replaces $k$ document-wise projections with one projection, although document encoding and fusion still scale with $k$.

Across four question-answering benchmarks and three backbones, FusePRAG improves macro-average F1 over DyPRAG when both methods receive the same retrieved text during generation. Its pure parametric variant, **FusePRAG-w/o C**, remains more task dependent. FusePRAG also reduces adapter-construction latency at larger retrieval depths by avoiding repeated translator calls.

Our contributions are summarized as follows:

- We introduce FusePRAG, a query-conditioned Fuse-then-Project framework that jointly processes the query and retrieved documents before generating a single query-specific adapter.
- We combine evidence fusion with answer-aware evidence-summary alignment and analyze direct document–document attention through single-document and capacity-matched controls.
- Across four benchmarks and three backbones, FusePRAG improves macro-average F1 over DyPRAG in the context-augmented setting and reduces adapter-construction latency at larger retrieval depths.

# 2 Related Work

## 2.1 Retrieval-Augmented Generation and Evidence Fusion

Retrieval-augmented generation conditions language models on external evidence rather than relying solely on knowledge acquired during pretraining (Guu et al., 2020; Lewis et al., 2020). Beyond retrieving relevant passages, a central challenge is combining their information for answer generation. Fusion-in-Decoder (FiD) (Izacard & Grave, 2021) encodes question–passage pairs independently and aggregates their representations through decoder cross-attention. Other approaches modify the retrieval process: FLARE (Jiang et al., 2023) and DRAGIN (Su et al., 2024) retrieve evidence according to generation-time information needs, while IRCoT (Trivedi et al., 2023) interleaves retrieval with intermediate reasoning steps. These methods primarily change when or how evidence is retrieved. FusePRAG instead assumes a retrieved document set and studies how query–document and document–document interactions within that set should condition adapter generation. This focus complements input-based evidence integration; our context-augmented setting retains retrieved text alongside the generated adapter.

## 2.2 Parametric Knowledge Injection

Parametric RAG incorporates retrieved evidence through model updates. PRAG (Su et al., 2025) trains document-specific LoRA adapters (Hu et al., 2022) offline and merges retrieved adapters at inference. DyPRAG (Tan et al., 2025) instead learns a shared parameter translator that generates adapters from document embeddings, avoiding per-document adapter training. More broadly, Text-to-LoRA generates adapters from natural-language task descriptions (Charakorn et al., 2025), whereas Doc-to-LoRA distills an input context into adapter parameters (Charakorn et al., 2026).

These methods differ in what conditions parameter generation. Text-to-LoRA conditions on a task description, and Doc-to-LoRA focuses on distilling a provided context. FusePRAG instead addresses query-conditioned multi-document retrieval: it jointly models a query and its retrieved documents before generating a single query-specific adapter. A central architectural distinction from DyPRAG is the ordering of evidence integration and parameter projection. FusePRAG jointly models the query and retrieved documents before projection, whereas DyPRAG combines independently generated document updates afterward.

## 2.3 Context Compression and Representation Learning

Context compression reduces the evidence or instructions processed by the generator. Selective Context (Li et al., 2023) and LongLLMLingua (Jiang et al., 2024) prune input text, whereas ICAE (Ge et al., 2023), AutoCompressors (Chevalier et al., 2023), and gist tokens (Mu et al., 2023) learn compact continuous representations. Particularly relevant is xRAG (Cheng et al., 2024), which maps a dense retrieval embedding into the language model's input representation space, replacing document text with a single token representation. FusePRAG likewise produces a compact representation, but projects it into adapter-parameter space rather than supplying it as compressed context. During training, the fused representation is aligned with embeddings of answer-aware evidence summaries generated from each training question, its gold passages, and its gold answer. No summary is generated or supplied to the model at inference, distinguishing this training-time alignment objective from methods that inject generated summaries into the inference context.

# 3 Methodology

Figure 2 summarizes FusePRAG. During offline training, Phase 1 learns a query-conditioned fused representation through answer-aware evidence-summary alignment, and Phase 2 freezes the fusion module while training the parameter translator for answer generation. At inference, a single translator pass maps the fused representation of the query and retrieved documents to one query-specific adapter. This workflow realizes Fuse-then-Project, in contrast to DyPRAG's document-wise Project-then-Fuse.

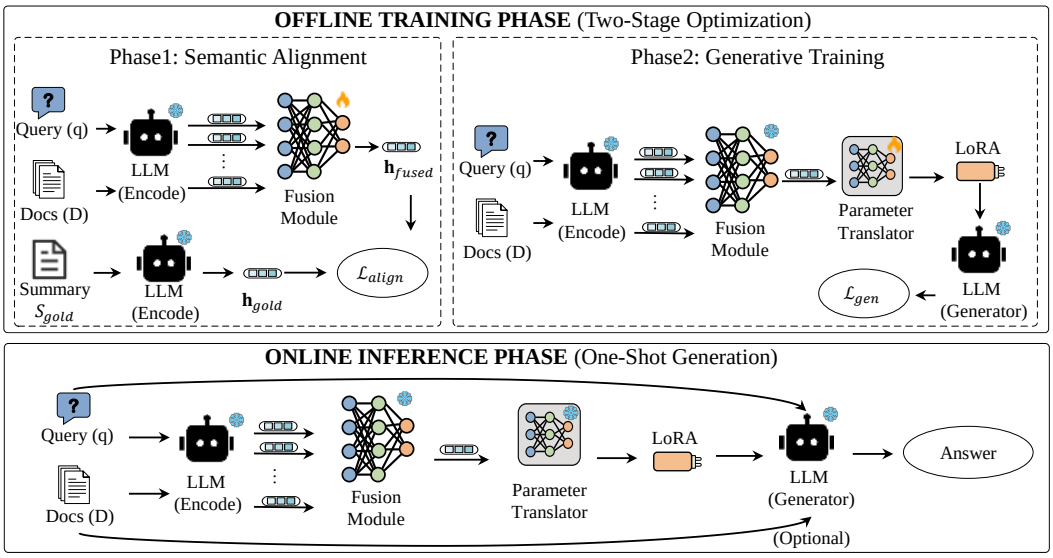


Figure 2: Overview of FusePRAG. **Offline training:** Phase 1 aligns the fusion module ($\mathcal{M}_\psi$) with answer-aware evidence-summary embeddings. Phase 2 freezes fusion and trains the translator ($\mathcal{F}_\phi$) to generate LoRA parameters ($\Delta\Theta$). **Online inference:** The query and retrieved documents are fused into one representation and projected to a single adapter.

### 3.1 Preliminaries and Problem Formulation

**Parametric RAG.** Given a query $q$ and corpus $\mathcal{C}$, let $\mathcal{D}$ denote the documents supplied to the adapter generator: $\mathcal{D} = \mathcal{D}_{\text{ret}}(q)$, the retrieved top-$k$, at inference, and $\mathcal{D} = \mathcal{D}_{\text{train}}$, gold passages plus synthetic distractors, during training (Section 4.1). Let $\Theta$ denote the frozen backbone parameters and $\Delta\Theta(q, \mathcal{D})$ a document-conditioned update. We write the answer distribution as

$$P_{\text{LLM}}\big(y \mid q, C_{\text{gen}}; \Theta + \Delta\Theta(q, \mathcal{D})\big), \tag{1}$$

where FusePRAG sets $C_{\text{gen}}$ to the textual form of $\mathcal{D}$ and FusePRAG-w/o C sets $C_{\text{gen}} = \varnothing$; both generate the update from $q$ and $\mathcal{D}$. We implement the update with LoRA Hu et al. (2022). At inference, no reference answer or per-query optimization is used.

**The DyPRAG Baseline.** DyPRAG Tan et al. (2025) uses a parameter translator (a hypernetwork), denoted as $\mathcal{F}_\phi$, to generate document-specific adapters without training each adapter online. It independently maps each document embedding $\mathbf{e}_{d_i}$ to a parameter update $\Delta\Theta_i = \mathcal{F}_\phi(\mathbf{e}_{d_i})$. For multiple retrieved documents ($k > 1$), DyPRAG takes the arithmetic mean of the projected updates, an ordering we call "Project-then-Fuse":

$$\Delta\Theta_{\text{DyPRAG}} = \frac{1}{k}\sum_{i=1}^{k} \mathcal{F}_\phi(\mathbf{e}_{d_i}), \tag{2}$$

**Limitations of Document-wise Parameter Generation.** In Eq. (2), each update is generated without conditioning on the other retrieved documents. A passage may provide an intermediate fact needed to interpret evidence elsewhere, yet these relationships do not explicitly inform its parameter generation. Similarly, an irrelevant passage can contribute an update without being assessed jointly with the query and the other evidence. These observations motivate joint evidence modeling before parameter generation; they do not imply that the adapted LLM cannot reason across documents or that independent generation necessarily fails in the presence of noise.

**Joint Evidence-conditioned Parameter Generation.** FusePRAG uses a fusion module $\mathcal{M}_\psi$ to jointly process the query embedding $\mathbf{e}_q$ and document embeddings $\{\mathbf{e}_{d_1}, \ldots, \mathbf{e}_{d_k}\}$. A parameter translator then maps the fused representation to a single update:

$$\mathbf{h}_{\text{fused}} = \mathcal{M}_\psi(\mathbf{e}_q, \{\mathbf{e}_{d_1}, \ldots, \mathbf{e}_{d_k}\}), \tag{3}$$

$$\Delta\Theta_{\text{Ours}} = \mathcal{F}_\phi(\mathbf{h}_{\text{fused}}). \tag{4}$$

This "Fuse-then-Project" architecture allows the translator's input to reflect query-document and document-document interactions. We omit the arguments $(q, \mathcal{D})$ from $\Delta\Theta_{\text{Ours}}$ below for brevity.

### 3.2 Cross-Document Semantic Fusion

The fusion module $\mathcal{M}_\psi$ produces a single query-conditioned representation from the query and retrieved document embeddings.

**Input Construction.** The fusion module operates on one vector per query or document, rather than their concatenated token sequences. Following DyPRAG Tan et al. (2025), we use the frozen backbone LLM as the encoder $\mathcal{E}_{\text{LLM}}$. We independently encode the query $q$ and each document $d_i$, using the final-token hidden state as the dense representation:

$$\mathbf{e}_q = \mathcal{E}_{\text{LLM}}(q), \quad \mathbf{e}_{d_i} = \mathcal{E}_{\text{LLM}}(d_i), \tag{5}$$

where $\mathbf{e}_q, \mathbf{e}_{d_i} \in \mathbb{R}^d$. To distinguish the query from document inputs, we add two learned type embeddings, $\mathbf{t}_q, \mathbf{t}_d \in \mathbb{R}^d$. The same document-type embedding $\mathbf{t}_d$ is shared across all retrieved documents. The fusion input is

$$\mathbf{H}^0 = [\mathbf{e}_q + \mathbf{t}_q,\ \mathbf{e}_{d_1} + \mathbf{t}_d,\ \ldots,\ \mathbf{e}_{d_k} + \mathbf{t}_d] \in \mathbb{R}^{(k+1)\times d}. \tag{6}$$

Here, $[\cdot]$ denotes concatenation along the sequence dimension. The type embeddings mark input roles; they do not encode document relevance or retrieval rank. We add no positional embeddings at the fusion stage.

**Shallow Transformer Fusion.** We process $\mathbf{H}^0$ with a Transformer encoder using pre-layer normalization and GELU feed-forward activations. The main model uses unrestricted self-attention among the query and non-padded document positions, so both query–document and document–document interactions can influence the output. The training objectives encourage the resulting representation to capture relevant evidence; attention alone does not guarantee the removal of distractors or correct multi-hop reasoning. A key-padding mask excludes padded document entries while keeping the query unmasked. We evaluate a controlled variant without direct document–document attention in Appendix D.7.

**Fused Representation.** After $L$ encoder layers, we read the output at the query position and apply a final layer normalization:

$$\mathbf{h}_{\text{fused}} = \text{LayerNorm}_{\text{out}}\big(\text{Transformer}(\mathbf{H}^0)_0\big) \in \mathbb{R}^d. \tag{7}$$

This query-conditioned vector is the input to the parameter translator. All fusion parameters, including the type embeddings and output normalization, are included in $\psi$.

### 3.3 Parameter Projection

An MLP-based parameter translator $\mathcal{F}_\phi$ maps $\mathbf{h}_{\text{fused}}$ to LoRA parameters. Unlike document-wise projection, its input is conditioned on the query and the retrieved documents jointly. For a target weight $W \in \mathbb{R}^{m\times n}$, it generates low-rank matrices $A_W \in \mathbb{R}^{r\times n}$ and $B_W \in \mathbb{R}^{m\times r}$ ($r \ll \min(m, n)$):

$$\{A_W, B_W\} = \mathcal{F}_\phi^W(\mathbf{h}_{\text{fused}}), \qquad \Delta W = \frac{\alpha}{r} B_W A_W, \tag{8}$$

where $\alpha$ is the LoRA scaling hyperparameter. We use $\Delta\Theta$ to denote the collection of updates $\{\Delta W\}$ over all target matrices. Following DyPRAG, we generate LoRA updates for the FFN matrices and add the resulting $\Delta\Theta$ to the frozen LLM; details are in Appendix B.

### 3.4 Two-Phase Training Strategy

We use embeddings of answer-aware evidence summaries to supervise evidence fusion before training the translator for answer generation. This gives the fusion module an intermediate target in addition to the final answer task.

**Phase 1: Answer-Aware Evidence Alignment.** This phase aligns fused evidence with a dense, answer-aware semantic target. The summaries capture answer-relevant facts supported by gold passages; they do not constitute verified step-by-step reasoning traces. For each training example, GPT-3.5-Turbo receives the question, gold passages, and gold answer and produces an answer-aware evidence summary $S_{\text{gold}}$ constrained to facts supported by the gold passages (prompt in Appendix A). These summaries are generated offline during data construction and used only as Phase-1 supervision targets; they are not inputs to the fusion module. At inference, FusePRAG uses neither summaries nor reference answers. To anchor the fused representation within the native semantic space of the backbone LLM, we employ the same frozen backbone LLM $\mathcal{E}_{\text{LLM}}$ (defined in Section 3.2) to encode the summary. Specifically, we obtain the target embedding $\mathbf{h}_{\text{gold}} = \mathcal{E}_{\text{LLM}}(S_{\text{gold}})$. We optimize the fusion parameters $\psi$ by minimizing a hybrid objective comprising Mean Squared Error (MSE) and Cosine Embedding Loss:

$$\begin{aligned}\mathcal{L}_{\text{align}} = \frac{1}{d}\|\mathbf{h}_{\text{fused}} - \mathbf{h}_{\text{gold}}\|_2^2 \\ + \lambda \cdot (1 - \cos(\mathbf{h}_{\text{fused}}, \mathbf{h}_{\text{gold}})),\end{aligned} \tag{9}$$

where $\cos(\cdot,\cdot)$ denotes cosine similarity and $\lambda$ is a balancing hyperparameter (set to 0.5 in our experiments). The squared-error term encourages agreement in embedding coordinates, while the cosine term encourages agreement in direction. This objective provides an intermediate target for fusion, rather than a guarantee that the fused vector preserves every fact or reasoning step.

**Phase 2: Translator Training for Answer Generation.** To enable a controlled downstream comparison, we initialize the translator from the same pre-trained DyPRAG checkpoint (Tan et al., 2025) used to initialize DyPRAG-Matched and optimize it using the answer-generation loss. The Phase-1 summaries are not used in this phase. The backbone parameters $\Theta$ and fusion parameters $\psi$ remain frozen; only the translator parameters $\phi$ are updated. Gradients from the generation loss flow through the adapted LLM and the generated LoRA updates to the translator:

$$\mathcal{L}_{\text{gen}} = -\sum_{t=1}^{|y|} \log P_{\text{LLM}}(y_t \mid q, C_{\text{gen}}, y_{<t}; \Theta + \Delta\Theta_{\text{Ours}})\,. \tag{10}$$

Here, $C_{\text{gen}}$ concatenates $\mathcal{D}_{\text{train}}$ for FusePRAG and is empty for FusePRAG-w/o C, as defined in Eq. (1). The translator thus maps the aligned $\mathbf{h}_{\text{fused}}$ into low-rank updates $\Delta\Theta_{\text{Ours}}$ for answer generation.

### 3.5 Inference Process

At inference, we set $\mathcal{D} = \mathcal{D}_{\text{ret}}(q)$, encode the query and documents, and run the fusion module in evaluation mode to obtain $\mathbf{h}_{\text{fused}}$. One translator pass generates the LoRA parameters. No summaries or reference answers are used. In Eq. (1), FusePRAG uses $C_{\text{gen}} = \text{concat}(\mathcal{D}_{\text{ret}})$ and FusePRAG-w/o C uses $C_{\text{gen}} = \varnothing$. The translator processes one fused vector, reducing $k$ projections to one, while encoding and fusion remain $k$-dependent.

## 4 Experiments

### 4.1 Experimental Setup

**Training Data Construction.** We construct 1,200 balanced examples from **2WikiMultihopQA (2WQA)** Ho et al. (2020) and **HotpotQA (HQA)** Yang et al. (2018), sampling 200 examples from each task subcategory and splitting them into 960 training and 240 validation examples. GPT-3.5-Turbo (`gpt-3.5-turbo-0125`) Ouyang et al. (2022) generates hard negatives that join the gold passages in $\mathcal{D}_{\text{train}}$, as well as answer-aware evidence summaries grounded in the gold passages. $\mathcal{D}_{\text{train}}$ conditions both phases and forms $C_{\text{gen}}$ in context-augmented Phase 2; the summaries supervise only Phase 1. When an original example provides multiple answer aliases, we select the label most consistent with its gold passages. The resulting training data are disjoint from the official evaluation sets, and no reference answers are provided to the model during evaluation. Prompts and further curation details are provided in Appendix A.

Table 1: Main results for LLaMA3.2-1B-Instruct and Qwen2.5-1.5B-Instruct (EM/F1, %). Avg denotes the macro-average over eight evaluation categories (Section 4.1). Within each backbone, the upper/lower groups omit/include retrieved text at generation; parametric w/o C variants still use retrieval for adapters. Best/second-best values per column within each backbone are **bold**/underlined.

| Method | 2WQA | | HQA | | PQA | | CWQ | | Avg | |
|---|---|---|---|---|---|---|---|---|---|---|
| | EM | F1 | EM | F1 | EM | F1 | EM | F1 | EM | F1 |
| **LLaMA3.2-1B-Instruct** | | | | | | | | | | |
| Vanilla | 16.67 | 21.94 | 16.50 | 23.30 | 1.00 | 2.55 | 23.33 | 35.39 | 15.50 | 21.54 |
| SFT | 20.08 | 22.57 | 12.83 | 16.53 | 1.67 | 2.22 | 17.67 | 24.83 | 15.67 | 18.80 |
| PRAG-w/o C | 19.75 | 25.47 | 19.17 | 25.45 | 12.00 | 23.58 | 26.00 | 35.86 | 19.42 | 26.53 |
| DyPRAG-w/o C | 24.08 | 29.75 | 19.83 | 26.13 | 8.33 | 9.52 | 27.67 | 36.35 | 21.50 | 27.14 |
| **FusePRAG-w/o C (ours)** | **28.58** | 33.29 | 21.33 | 27.71 | 8.67 | 9.11 | 28.33 | 37.31 | 24.25 | 29.38 |
| Standard RAG | 17.58 | 25.34 | 21.33 | 30.85 | 11.33 | 20.11 | 23.67 | 37.07 | 18.50 | 27.53 |
| SFT + RAG | 22.58 | 29.20 | 16.83 | 23.02 | 20.00 | 30.89 | 19.33 | 27.84 | 20.42 | 27.70 |
| PRAG | 20.33 | 26.11 | 23.50 | 31.06 | 20.33 | 31.07 | 28.33 | 39.63 | 22.13 | 29.65 |
| DyPRAG | 26.17 | 32.51 | 23.33 | 30.80 | 22.67 | 31.87 | 28.00 | 38.78 | 25.25 | 32.79 |
| **FusePRAG (ours)** | 27.58 | **35.24** | **26.33** | **34.40** | **25.67** | **37.13** | **33.33** | **45.03** | **27.75** | **36.49** |
| **Qwen2.5-1.5B-Instruct** | | | | | | | | | | |
| Vanilla | 20.83 | 27.28 | 17.67 | 25.82 | 0.67 | 2.87 | 18.00 | 26.47 | 17.17 | 23.76 |
| SFT | 12.75 | 15.22 | 10.83 | 13.74 | 0.00 | 0.13 | 12.67 | 19.68 | 10.67 | 13.52 |
| PRAG-w/o C | 22.25 | 29.54 | 18.83 | 26.85 | 1.33 | 3.87 | 22.33 | 30.82 | 18.79 | 25.82 |
| DyPRAG-w/o C | 22.33 | 28.95 | 20.00 | 27.55 | 3.00 | 6.64 | 22.67 | 31.94 | 19.38 | 26.19 |
| **FusePRAG-w/o C (ours)** | 26.17 | 31.53 | 22.17 | 28.76 | 4.33 | 6.47 | 25.00 | 33.89 | 22.29 | 28.00 |
| Standard RAG | 16.17 | 23.97 | 16.67 | 26.66 | 0.67 | 9.97 | 18.33 | 28.23 | 14.63 | 23.43 |
| SFT + RAG | 13.08 | 17.92 | 12.67 | 16.60 | 6.33 | 9.07 | 10.67 | 18.47 | 11.83 | 16.55 |
| PRAG | 20.58 | 26.94 | 23.33 | 30.81 | 2.67 | 12.61 | 21.67 | 32.13 | 19.17 | 26.77 |
| DyPRAG | 18.83 | 26.04 | 21.67 | 31.40 | 7.33 | 22.69 | 23.67 | 33.57 | 18.71 | 27.90 |
| **FusePRAG (ours)** | **26.92** | **32.62** | **24.50** | **32.90** | **17.67** | **29.67** | **25.67** | **34.75** | **25.00** | **32.59** |

**Evaluation Protocols and Metrics.** Following PRAG Su et al. (2025) and DyPRAG Tan et al. (2025), we evaluate the first 300 examples from each of the four 2WQA categories and two HQA categories, and 300 examples each from **PopQA (PQA)** Mallen et al. (2023) and **ComplexWebQuestions (CWQ)** Talmor & Berant (2018), yielding 2,400 evaluation examples in total. Phase-1 alignment and Phase-2 QA training use only 2WQA and HQA, while PQA and CWQ provide cross-dataset evaluations of these stages. All methods use the same evaluation examples and answer-normalization procedure. Unless otherwise specified, $\mathcal{D}_{\mathrm{ret}}(q)$ contains the same BM25 top-3 passages for every retrieval-based method; context-augmented methods also use these passages as $C_{\mathrm{gen}}$. F1 is the primary metric and EM is also reported. For each metric, Avg equally weights the four 2WQA categories (Compare, Bridge, Inference, and Compose), two HQA categories (Bridge and Compare), PQA, and CWQ.

**Baselines.** We evaluate eight baseline configurations spanning five baseline families. **Vanilla** uses the unchanged backbone without retrieved context, whereas **Standard RAG** supplies the retrieved passages directly to the backbone (Lewis et al., 2020). **SFT** trains one static LoRA adapter (Hu et al., 2022) on the QA training data; we evaluate it without retrieved text (SFT) and with retrieved text at inference (SFT+RAG). **PRAG** trains and stores document-specific LoRA adapters offline (Su et al., 2025). **DyPRAG** dynamically generates one adapter per retrieved document and averages the resulting updates (Tan et al., 2025). For PRAG, DyPRAG, and FusePRAG, the suffix **-w/o C** denotes the parametric-only configuration: retrieved documents condition the adapter but are omitted from the generation prompt. The corresponding names without this suffix additionally provide the same retrieved text to the generator.

Table 2: **Ablation results in the context-augmented setting using LLaMA3.2-1B.** All values are F1 scores (%). Avg follows the eight-category macro-average defined in Section 4.1, and Δ Avg is measured relative to the full model.

| Method | 2WQA | HQA | PQA | CWQ | Avg | Δ Avg |
|---|---|---|---|---|---|---|
| **FusePRAG (Full Model)** | **35.24** | **34.40** | **37.13** | **45.03** | **36.49** | – |
| Doc. Mean Pool (w/o Query) | 32.56 | 32.62 | 32.50 | 40.51 | 33.56 | -2.93 |
| w/o Phase 1 (End-to-End) | 24.40 | 30.54 | 17.03 | 37.05 | 26.60 | -9.89 |
| w/o Type Embeddings | 35.06 | 34.02 | 36.87 | 44.63 | 36.22 | -0.27 |

The main PRAG and DyPRAG rows follow their original training procedures. Because FusePRAG additionally uses 960 QA examples in Phase 2, Appendix D.3 reports **DyPRAG-Matched**, which retains Project-then-Fuse while matching FusePRAG's Phase-2 QA supervision, translator initialization, and optimization budget. This control does not receive FusePRAG's Phase-1 summary supervision.

**Implementation Details.** We use LLaMA3.2-1B-Instruct, Qwen2.5-1.5B-Instruct, and LLaMA3-8B-Instruct. The two-layer fusion Transformer is trained for three epochs with AdamW at $10^{-5}$; the translator is then trained for three epochs at $10^{-6}$ while the backbone and fusion module remain frozen. LoRA uses rank $r = 2$, scaling factor $\alpha = 32$, and no dropout. All context-augmented methods share a 3,000-token input limit and the same truncation policy. Seed sensitivity and complete implementation and baseline-training details are reported in Appendices D.5 and B.

### 4.2 Main Results

Table 1 reports results for the two smaller backbones; LLaMA3-8B results are in Appendix D.2. In the context-augmented setting, FusePRAG improves macro-average F1 over DyPRAG by 3.70, 4.69, and 2.98 points on LLaMA3.2-1B, Qwen2.5-1.5B, and LLaMA3-8B, respectively. The gains hold on every benchmark for all three backbones, including the cross-dataset evaluations on PQA and CWQ. This consistent pattern supports fused adapters as a complement to explicit retrieved evidence rather than as a replacement for it.

Without textual context, FusePRAG-w/o C also improves the overall average over DyPRAG-w/o C on all three backbones, but the dataset-level results are less uniform: it is slightly below DyPRAG-w/o C on PQA for the two smaller backbones and can remain behind PRAG-w/o C. The comparison highlights a trade-off between storage-intensive offline document adapters and dynamically generated query-specific adapters. We therefore treat the context-augmented setting as the primary result and the pure parametric setting as evidence about how much retrieved information can be conveyed through generated parameters alone.

On Qwen2.5-1.5B, FusePRAG exceeds DyPRAG-Matched by 5.37, 4.41, 6.19, and 4.61 F1 points with context on 2WQA, HQA, PQA, and CWQ, respectively (Appendix D.3). This control matches Phase-2 QA data, translator initialization, and optimization budget, but not Phase-1 alignment or query inputs to the translator. Matching Phase-2 training therefore does not close the gap; this comparison supports the complete pipeline rather than isolating projection order.

### 4.3 Ablation Study

Table 2 compares three variants on LLaMA3.2-1B with context: (1) **Doc. Mean Pool (w/o Query)** replaces learned fusion with document mean pooling, removing query conditioning; (2) **w/o Phase 1 (End-to-End)** jointly trains fusion and translation using only the answer-generation loss, with the same initial weights and total optimization steps; (3) **w/o Type Embeddings** removes query/document type embeddings while retaining fusion and two-phase training.

**Analysis of Evidence Fusion.** Mean pooling reduces average F1 by 2.93 points, reflecting the joint removal of learned fusion and query conditioning. On LLaMA3.2-1B, a capacity-matched attention-mask control preserves architecture and training while disabling direct document–document attention; full attention improves context-augmented F1 by 0.62–1.29 points (Appendix D.7). On

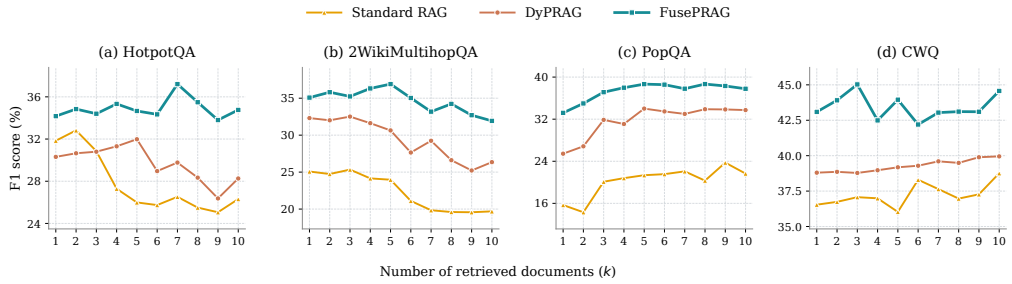


Figure 3: **Performance across retrieval depths.** F1 on four benchmarks using LLaMA3.2-1B-Instruct as top-$k$ varies from 1 to 10. All methods use fixed checkpoints in the context-augmented setting; only $k$ changes at inference.

Qwen2.5-1.5B, FusePRAG outperforms both query-aware attention pooling and cross-attention fusion on all four benchmarks with context, exceeding the latter by 0.74–4.59 points (Appendix D.8). Together, these results support query-conditioned fusion and the incremental contribution of direct document–document attention.

Removing Phase 1 causes the largest drop, 9.89 points, supporting two-phase training over answer-only end-to-end training under the same optimization budget. This comparison changes both supervision and scheduling, rather than isolating the summary target. Type embeddings contribute a smaller 0.27 points. Appendix D.9 reports summary-based controls.

### 4.4 Analysis and Discussion

FusePRAG outperforms Standard RAG and DyPRAG on all four benchmarks at $k = 1, \ldots, 10$ (Figure 3). At $k = 1$, where direct document–document attention is absent, FusePRAG still exceeds DyPRAG by 2.77–7.76 F1 points. Both methods generate one adapter at $k = 1$, so fewer projections cannot explain this gap. Query-conditioned representation learning and two-phase training may contribute; this comparison does not isolate their individual effects. Performance varies non-monotonically with $k$, but the advantage persists.

**Robustness to Retrieval Noise.** Appendix Table 6 adds up to eight distractors to the gold passages on 2WQA. FusePRAG remains ahead at every noise level, and its F1 decreases by 8.27 points from zero to eight distractors, compared with 10.40 for DyPRAG. This controlled synthetic-distractor test supports improved robustness in the context-augmented setting.

**Efficiency and Scalability.** FusePRAG adds query-encoding and fusion costs but avoids DyPRAG's $k$ translator executions. The adapter-construction measurement excludes document encoding for both methods and includes query encoding for FusePRAG. The crossover occurs between $k = 3$ and $k = 5$; at $k = 15$, latency falls from 146.2ms to 35.4ms (4.1×). End-to-end gains are smaller because encoding and generation still scale with retrieval depth. The protocol and component breakdown are in Appendices C.1 and C.3.

## 5 Conclusion

This work introduces **FusePRAG**, a query-conditioned framework that fuses retrieved evidence before generating one adapter. Across four benchmarks and three backbones, it consistently improves over DyPRAG in the context-augmented setting. The $k = 1$ and capacity-matched analyses show that the overall advantage extends beyond direct document–document attention while providing evidence for its incremental contribution. Results without context remain task dependent; future work should disentangle supervision, query conditioning, and projection order while improving parameter-only transfer.

### AI USE STATEMENT

We used generative AI tools for synthetic data construction. Specifically, GPT-3.5-Turbo generated answer-aware evidence summaries for Phase-1 answer-aware evidence-summary alignment and hard-negative distractor passages used in training and controlled retrieval-noise evaluation. We additionally used LLaMA3.2-1B-Instruct to regenerate training summaries for the supervision-quality experiment in Appendix D.10. The Phase-1 training prompt receives the question, gold passages, and gold answer to construct answer-aware evidence-summary targets. Summary generation is not required at inference time by FusePRAG. The summary-based baselines use a separate inference prompt that receives only the test question and retrieved passages, without gold passages or reference answers.

The data-generation prompts and human verification procedure are described in Appendix A. We manually reviewed a randomly sampled subset of 100 instances, checking whether summaries preserved answer-relevant supporting facts without introducing unsupported information and whether distractor passages satisfied the intended constraints. This was a sample-based assessment rather than exhaustive verification of all generated data.

We take responsibility for the final content of this work, including the text, claims, and artifacts produced with the assistance of generative AI.

### ETHICS STATEMENT

This work investigates retrieval-augmented question answering using existing benchmarks and pretrained language models. FusePRAG may inherit or amplify biases and factual errors in backbone models, retrieved evidence, and synthetic supervision. Mapping evidence to adapter parameters does not ensure factual correctness and may make individual sources' contributions harder to trace; retaining textual context does not by itself resolve this limitation. Answer-aware evidence summaries are generated for training, while synthetic distractors are used in training and controlled evaluation, as detailed in the AI use statement and Appendix A. Their sample-based manual verification does not rule out errors or systematic biases in the remaining data. Our retrieval-noise experiments assess robustness under the tested conditions, not resistance to adversarial manipulation, privacy leakage, or safety in high-stakes applications. Deployment would require application-specific evaluation, source validation and provenance tracking, safeguards for sensitive information in retrieved data and generated adapters, and appropriate human oversight. Any reuse or distribution of datasets, model weights, and derived artifacts should respect the applicable licenses and data-use restrictions.

### REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 describes the FusePRAG architecture, training objectives, and two-phase optimization procedure. Section 4.1 specifies the training data construction and split, evaluation benchmarks, metrics, and baseline configurations. Appendix A provides the prompts used to generate answer-aware evidence summaries and hard negative distractors, together with the human verification procedure. Appendix B documents the computing infrastructure, retrieval corpus and implementation, embedding extraction, fusion architecture, LoRA configuration, translator initialization, and context truncation strategy. The latency measurement protocol and post-retrieval inference breakdown are described in Appendices C.1 and C.3, respectively. We will clean up and publicly release our code upon acceptance of the paper.

## REFERENCES

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-lora: Instant transformer adaption. *arXiv preprint arXiv:2506.06105*, 2025.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-lora: Learning to instantly internalize contexts. *arXiv preprint arXiv:2602.15902*, 2026.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xRAG: Extreme context compression for retrieval-augmented generation with one token. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pp. 3829–3846. Association for Computational Linguistics (ACL), 2023.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

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

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

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

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 874–880. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.74.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024.

Zhengbao Jiang, Luyu Gao, Zhiruo Wang, Jun Araki, Haibo Ding, Jamie Callan, and Graham Neubig. 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*, pp. 2336–2349, 2022.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*, 2023.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen Tau Yih. Dense passage retrieval for open-domain question answering. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pp. 6769–6781. Association for Computational Linguistics (ACL), 2020.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474, 2020.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 6342–6353, 2023.

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

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 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)*, pp. 9802–9822, 2023.

Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352, 2023.

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

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*, 2024.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pp. 31210–31227. PMLR, 2023.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. Dragin: dynamic retrieval augmented generation based on the information needs of large language models. *arXiv preprint arXiv:2403.10081*, 2024.

Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. Parametric retrieval augmented generation. *arXiv preprint arXiv:2501.15915*, 2025.

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*, 2018.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 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)*, pp. 10014–10037, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

# A Data Construction Details

To enhance reproducibility, we provide the prompts used to construct the Phase-1 answer-aware evidence summaries, generate inference-time summaries for the summary baselines, and synthesize hard negative passages.

## A.1 Answer-Aware Evidence Summary Generation

In Phase 1, the summary generator receives the training question, its gold passages, and its gold answer. The answer guides summary-target construction toward answer-relevant evidence, while the prompt requires the generated summary to remain grounded in the gold passages. These answer-aware summaries supervise only Phase 1 and are neither generated nor supplied at inference. In Phase 2, the gold answer tokens provide the supervised labels for the standard next-token prediction objective. The complete Phase-1 prompt is shown in Figure 4.

## A.2 Inference-Time Summary Generation for Baselines

The summary-based baselines in Appendix D.9 use the separate prompt in Figure 5. At evaluation time, the summary generator receives only the test question and its retrieved passages. It has no access to the reference answer, gold passages, or Phase-1 training summary. The resulting summary is either appended to the generator input or encoded for parameter projection, depending on the baseline.

**Prompt for generating answer-aware evidence summaries**

**User:**
You will receive a Question and its Golden Passages. Your task is to extract a compact summary containing only the information necessary to answer the Question.

Requirements:
1. The summary must be strictly based on the Golden Passages. Do NOT add any new facts or outside knowledge.
2. Keep only the essential information required to answer the Question.
3. Compress the content into 2--4 concise sentences.
4. Maintain objective and factual style.
5. Output format must be EXACTLY:
*Summary: <your concise extracted paragraph>*

Question: {question}
Golden Passages: {golden_passages}
Answer: {answer}

Figure 4: Prompt used to generate answer-aware evidence summaries ($S_{gold}$) as Phase-1 training targets. This prompt is invoked only during offline target construction and is not used at inference.

### A.3 Hard Negative Distractor Generation

To simulate realistic retrieval noise, we synthesize "hard negative" documents that share high lexical overlap with the query but contain no relevant answer. The prompt is shown in Figure 6.

**Training and Validation Set Composition.** Across the 1,200-example supervision pool (960 training and 240 validation examples), we retain all dataset-provided gold passages and up to two generated distractors per example, yielding 2–7 input documents. In total, 927 examples contain two gold passages and two distractors, 199 contain four and two, and 72 contain two and one; the remaining two contain $(2, 0)$ and $(5, 2)$ gold–distractor pairs. Variable-length document sets are padded within minibatches, with padded positions excluded by the key-padding mask.

**Human Verification.** To assess the quality of the synthesized data, we conducted a manual review on a randomly sampled subset of 100 instances. For *evidence summaries*, we verified that they retained answer-relevant supporting facts without introducing unsupported information. For *hard negative distractors*, we confirmed that they shared high lexical overlap with the query while strictly excluding the ground-truth answer. Our review indicated a pass rate of over 95%, providing a limited sample-based quality check of the generated data.

## B Implementation Details

### B.1 Computing Infrastructure

All experiments are implemented using **PyTorch** and the **Hugging Face Transformers** library. Training and inference are conducted on NVIDIA A100 GPUs (80GB) and V100 (32GB) GPUs.

**Prompt for generating inference-time summaries**

**User:**
You will receive a Question and its Passages. Your task is to extract a compact summary containing only the information necessary to answer the Question.

Requirements:
1. The summary must be strictly based on the Passages. Do NOT add any new facts or outside knowledge.
2. Keep only the essential information required to answer the Question.
3. Compress the content into 2--4 concise sentences.
4. Maintain objective and factual style.
5. Output format must be EXACTLY:
*Summary: <your concise extracted paragraph>*

Question: {question}
Passages: {passages}

Figure 5: Prompt used to generate summaries for the inference-time summary baselines. It receives only the test question and retrieved passages; no reference answer or gold passages are provided.

### B.2 Retrieval Implementation

For retrieval, motivated by recent findings that BM25 performs on par with state-of-the-art dense models while offering superior efficiency Su et al. (2025); Tan et al. (2025), we adopt BM25 implemented via Elasticsearch. Regarding the retrieval corpus, we utilize the standard English Wikipedia dump pre-processed by DPR Karpukhin et al. (2020). We specifically use the `psgs_w100.tsv` dump, consistent with Tan et al. (2025), serving as the unified external knowledge source for all comparative methods.

### B.3 Hyperparameter Configuration

**Document and Query Encoding.** Following DyPRAG Tan et al. (2025), we use the frozen backbone LLM as the encoder. For each query $q$ and retrieved document $d_i$, we extract the last hidden state at the final token position before the vocabulary projection as the dense representation. The query, document, and summary embeddings therefore use the same frozen backbone encoder.

**Fusion Module Architecture.** We use the `TypeAwareTransformerFusion` implementation: a Transformer encoder with 2 layers, 8 attention heads, hidden dimension $d$ matching the backbone LLM, and feed-forward dimension $2d$. Each layer uses pre-layer normalization (`norm_first=True`) and GELU activations. The implementation defaults to dropout 0.1. Two learned type embeddings distinguish query and document inputs, with the document-type embedding shared across all passages. Their entries are initialized independently from $\mathcal{N}(0, d^{-1})$, corresponding to standard deviation $d^{-1/2}$. These embeddings add $2d$ trainable parameters and no additional sequence positions. The fusion encoder uses no positional embeddings or causal mask. An optional key-padding mask excludes padded document entries from attention. The query-position output is passed through a separate LayerNorm before parameter projection. Type embeddings and normalization parameters are trained with the fusion module in Phase 1 and frozen with it in Phase 2; dropout is disabled at inference. With dropout disabled, permuting the document embeddings and their padding mask leaves the fused output unchanged up to numerical precision. Shared document-type embeddings and the absence of fusion-level positional encodings therefore make fusion permutation-

Prompt for generating hard negative distractors

**User:**
You are a data construction assistant for retrieval-augmented QA.
**[Task]**
Given a question, its answer, and several gold relevant passages that are enough to answer the question, generate additional NOISE passages.
Noise passages must:
1. Look topically or entity-wise related to the question (for example, same domain, similar type of entity).
2. Not be useful to answer the question or derive the correct answer.
3. Do not repeat or closely paraphrase the input passages.
4. Not contain the answer text or obvious paraphrases of the answer.
5. Each passage should be about 80--200 words, in one plain paragraph (no lists).
**[Input]**
Question: {question}
Answer: {answer}
Gold relevant passages: {gold_passages}
**[Output format]**
Return ONLY a JSON array of strings, each string is one noise passage.

Figure 6: The prompt template used for generating synthetic hard negative distractors (noise passages).

invariant over documents, while token order within each document is encoded by the backbone before fusion.

**Parameter Translator and LoRA Configuration.** We adopt the same parameter translator architecture as DyPRAG Tan et al. (2025) to ensure a fair comparison. LoRA modules are integrated exclusively into the feed-forward network (FFN) matrices of the backbone LLM, while the query, key, and value (QKV) projections are excluded. The LoRA rank is set to $r = 2$ with a scaling factor $\alpha = 32$, and no dropout is applied. We initialize the parameter translator with pre-trained DyPRAG weights to provide a common translator prior; DyPRAG-Matched uses the same initialization, enabling a controlled comparison under matched Phase-2 supervision and optimization. Subsequently, during the generation phase (Phase 2), the translator parameters are fine-tuned to adapt the mapping function to the fused cross-document semantic representations.

### B.4 Training Protocols and Supervision Matching

Table 3 distinguishes method-faithful system comparisons from the controlled comparison used to isolate downstream QA supervision. The main PRAG and DyPRAG baselines preserve their published method-specific training procedures; forcing PRAG into the same query-level training procedure would change its defining offline per-document adaptation mechanism. DyPRAG's original objective trains a shared translator on document–LoRA alignment pairs, enabling each retrieved document to be projected independently before the resulting updates are combined. DyPRAG-Matched preserves this Project-then-Fuse architecture while further optimizing the shared translator through the answer-generation loss on the same Phase-2 QA data used by FusePRAG. It therefore controls for downstream QA supervision without introducing pre-projection multi-document fusion.

Table 3: Training and supervision protocols. The main PRAG and DyPRAG rows follow their original training objectives. DyPRAG-Matched retains Project-then-Fuse while matching FusePRAG's Phase-2 QA data, translator initialization, and optimization budget. For each backbone, all methods use the same LoRA target matrices, rank, and scaling configuration.

| Method / stage | Initialization | Training supervision | Trainable parameters | Optimization | Checkpoint / role |
|---|---|---|---|---|---|
| PRAG (main) | Randomly initialized document LoRAs; method-faithful reproduction | Per-document offline LoRA objective following PRAG | Document-specific LoRA parameters | lr=$3\times10^{-4}$; 1 epoch (PQA: 2) | Final scheduled epoch; system baseline, not equal-supervision |
| DyPRAG (main) | Method-faithful reproduced translator | Document–LoRA alignment pairs following DyPRAG; 4,800 pairs for 1B/1.5B and 2,400 for 8B (2WQA: 480); no training on our 960 QA examples | Shared parameter translator | 1 epoch, lr=$10^{-5}$ | Final scheduled epoch; system baseline, not equal-supervision |
| DyPRAG-Matched | Same pre-trained DyPRAG translator used to initialize FusePRAG Phase 2 | Our 960 QA training examples | Shared parameter translator only; backbone frozen | 3 epochs, AdamW, lr=$10^{-6}$ | Same 240-example validation protocol as FusePRAG; equal-QA-supervision control |
| FusePRAG Phase 1 | Randomly initialized fusion module; frozen backbone encoder | Summary embeddings constructed for the same 960 training examples | Fusion module, type embeddings, and output normalization | 3 epochs, AdamW, lr=$10^{-5}$ | Same 240-example validation split; semantic-alignment stage |
| FusePRAG Phase 2 | Aligned fusion module plus the same pre-trained DyPRAG translator initialization | Our 960 QA training examples | Shared parameter translator only; backbone and fusion module frozen | 3 epochs, AdamW, lr=$10^{-6}$ | Same validation protocol as DyPRAG-Matched; full method |

For DyPRAG-Matched, each retrieved document is projected independently and the resulting updates are averaged,

$$\Delta\Theta_{\text{matched}} = \frac{1}{k}\sum_{i=1}^{k}\mathcal{F}_\phi(\mathbf{e}_{d_i}). \tag{11}$$

The averaged update is injected into the frozen backbone, and the answer-generation loss is back-propagated through the average to the shared translator. Thus, averaging does not block gradients; rather, it supplies the same query-level loss to document-wise projections that were produced without query or cross-document conditioning. This control is distinct from the mean-pooling ablation, which computes $\mathcal{F}_\phi(\frac{1}{k}\sum_i \mathbf{e}_{d_i})$ and therefore fuses representations before projection.

**Context Truncation Strategy.** For the context-augmented setting, we adopt a unified truncation protocol. Following Tan et al. (2025), the maximum context length is set to **3000 tokens**. We prioritize preserving the query and truncate the concatenated retrieved documents from the tail to fit this limit. This constraint is applied uniformly across all context-augmented methods (Standard RAG, SFT+RAG, PRAG, DyPRAG, and FusePRAG), preventing variations in effective context-window size from confounding the comparison.

# C Detailed Evaluation Setup

## C.1 Efficiency Analysis Protocol

Figure 7 reports two complementary measurements. Panel (a) measures the method-specific computation required to construct an adapter after the document representations have been obtained. It excludes retrieval, document encoding, and answer generation. DyPRAG does not require query encoding for parameter generation, so its measurement includes $k$ translator executions, parameter aggregation, and adapter injection. FusePRAG requires a query representation for query-conditioned fusion, so its measurement includes query encoding, the fusion module, one translator execution, and adapter injection. Denoting these components by $T_{\text{q-enc}}$, $T_{\text{translator}}$, $T_{\text{aggregate}}$, $T_{\text{fuse}}(k)$, and $T_{\text{inject}}$, respectively, the measured stage is

- **DyPRAG**: $T_{\text{adapter}} \approx kT_{\text{translator}} + T_{\text{aggregate}} + T_{\text{inject}}$,
- **FusePRAG**: $T_{\text{adapter}} \approx T_{\text{q-enc}} + T_{\text{fuse}}(k) + T_{\text{translator}} + T_{\text{inject}}$.

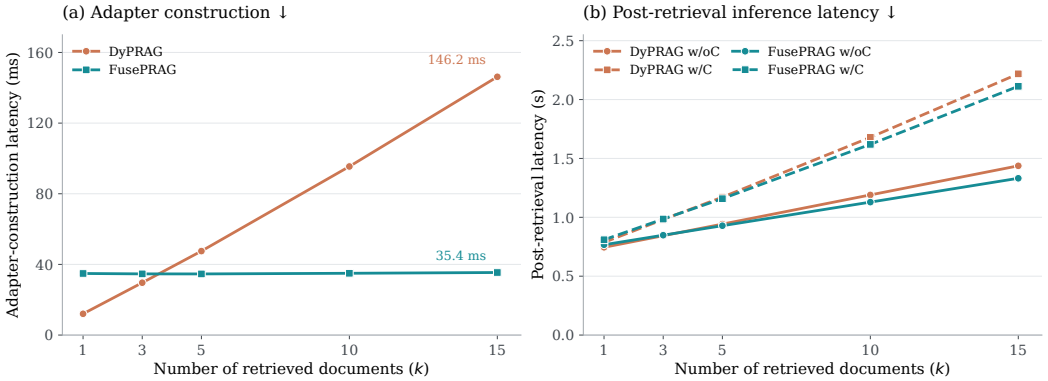


Figure 7: Latency scaling with retrieval depth on CWQ using LLaMA3.2-1B, averaged over 300 examples with retrieval excluded and generation fixed to 32 new tokens. Panel (a) reports adapter-construction latency after document representations are available; panel (b) reports post-retrieval inference latency including dense encoding, adapter construction, and answer generation. Lower is better.

This comparison includes the additional query-conditioning cost required by FusePRAG while exposing the reduction from $k$ translator executions to one.

Panel (b) reports post-retrieval inference latency, which additionally includes dense query/document encoding and autoregressive answer generation. Retrieval latency remains excluded because all methods use the same retrieved passages. We fix generation to 32 new tokens and disable sampling to prevent output-length differences from confounding the comparison. Method execution order is shuffled for each example, CUDA synchronization is applied around every timed operation, and each method is warmed up before measurement. Both panels are averaged over 300 CWQ examples.

### C.2 Extended Scalability Analysis

To empirically examine the scaling behavior of FusePRAG during adapter construction, we conduct an extended latency analysis at retrieval depths $k \in \{1, 3, 5, 10, 15, 30, 50, 100\}$.

The fusion module applies self-attention to the $k$ document-level representations and therefore has $O(k^2)$ complexity in retrieval depth. Because this operation uses a two-layer Transformer, the experiment evaluates whether its cost becomes material relative to a translator execution over the observed range. Following Figure 7(a), this extended stress test begins after the document representations have been obtained. It includes query encoding, fusion, parameter projection, and adapter injection for FusePRAG, and repeated projection, aggregation, and adapter injection for DyPRAG.

As shown in Table 4, FusePRAG latency increases from 34.86 ms at $k = 1$ to 36.84 ms at $k = 100$, a 5.7% increase. Thus, within the evaluated range, fusion does not dominate the measured adapter-construction cost. Its fixed query-encoding and fusion overhead makes it slower than DyPRAG at $k = 1$ and $k = 3$, but it becomes faster from $k = 5$ onward. DyPRAG latency increases from 12.03 ms to 1003.41 ms and scales approximately linearly because it invokes the translator once per document. These measurements characterize the observed range and do not imply constant scaling for FusePRAG beyond $k = 100$.

### C.3 Post-Retrieval Inference Latency Breakdown

As a separate component-level analysis, Table 5 reports a post-retrieval inference latency breakdown using Qwen2.5-1.5B at retrieval depth $k = 3$, the default setting in our main experiments. We report the average latency over 100 randomly sampled examples from ComplexWebQuestions (CWQ), with generation continuing until natural termination. This protocol differs from Figure 7, which uses LLaMA3.2-1B, 300 examples, and a fixed generation length of 32 new tokens. The component boundaries also differ: Figure 7(a) includes FusePRAG's query encoding in adapter

Table 4: Adapter-construction latency at retrieval depths $k \in \{1, 3, 5, 10, 15, 30, 50, 100\}$ under the same measurement boundary as Figure 7(a). Document encoding is excluded for both methods; query encoding is included for FusePRAG because it is required for query-conditioned fusion. FusePRAG latency increases by 5.7% over the evaluated range, whereas DyPRAG scales approximately linearly.

| $k$ | **DyPRAG (ms)** | **FusePRAG (ms)** |
|---|---|---|
| 1 | 12.03 | 34.86 |
| 3 | 29.64 | 34.64 |
| 5 | 47.54 | 34.61 |
| 10 | 95.45 | 34.98 |
| 15 | 146.22 | 35.41 |
| 30 | 293.97 | 35.53 |
| 50 | 495.14 | 35.90 |
| 100 | 1003.41 | 36.84 |

Table 5: Post-retrieval inference latency (seconds) on Qwen2.5-1.5B at $k = 3$, averaged over 100 randomly sampled CWQ examples with natural output termination. Enc., LoRA Gen., and LLM Gen. denote dense encoding, retrieval-conditioned adapter generation, and autoregressive generation, respectively; Total sums these components, and Tok. is the average output length. Retrieval latency is excluded, and "–" indicates that a component is not applicable.

| **Method** | **Enc.** | **LoRA Gen.** | **LLM Gen.** | **Total** | **Tok.** |
|---|---|---|---|---|---|
| Vanilla | – | – | 0.4292 | 0.4292 | 22.0 |
| Standard RAG | – | – | 0.5020 | 0.5020 | 22.7 |
| DyPRAG-w/o C | 0.1038 | 0.0419 | 0.3725 | 0.5182 | 16.8 |
| FusePRAG-w/o C | 0.1231 | 0.0208 | 0.2695 | 0.4134 | 12.5 |
| DyPRAG | 0.1037 | 0.0397 | 0.4055 | 0.5489 | 13.1 |
| FusePRAG | 0.1228 | 0.0213 | 0.3551 | 0.4992 | 11.6 |

construction, whereas Table 5 reports it under Enc. and excludes it from LoRA Gen. The absolute latency values in Table 5 therefore should not be directly compared with those in Figure 7; the table instead provides a within-protocol comparison of the measured components. Following common evaluation practice in parametric RAG, we focus on the computation after the retrieved passages are obtained, including dense encoding, retrieval-conditioned LoRA generation, and autoregressive answer generation. Retrieval latency is excluded because all methods use the same retriever, retrieved passages, and evaluation instances. Moreover, retrieval latency can vary substantially with the search backend, index implementation, and deployment environment, whereas our goal is to compare the computational overhead introduced by different knowledge-integration mechanisms.

For Vanilla and Standard RAG, dense encoding and LoRA generation are not applicable. For DyPRAG, Enc. includes document encoding only, because its parameter translator is conditioned independently on document representations. For FusePRAG, Enc. additionally includes query encoding because the fusion module jointly processes the query and document representations. For DyPRAG, LoRA Gen. includes repeated translator executions over the retrieved documents and parameter aggregation. For FusePRAG, LoRA Gen. includes the cross-document fusion module and a single translator execution after the query and document representations have been obtained. For the context-augmented configurations (DyPRAG and FusePRAG), retrieved passages are additionally concatenated into the input context, so their LLM generation latency reflects long-context decoding.

As shown in Table 5, replacing three document-wise translator executions with one fused projection reduces LoRA-generation latency by 50.4%, from 0.0419 s to 0.0208 s, without textual context and by 46.3%, from 0.0397 s to 0.0213 s, in the context-augmented setting. After including the query encoding required for fusion, the encoding-plus-LoRA cost remains comparable at $k = 3$: 0.1457 s for DyPRAG-w/o C versus 0.1439 s for FusePRAG-w/o C, and 0.1434 s for DyPRAG versus 0.1441 s for FusePRAG. This reflects the trade-off at shallow retrieval depth between the fixed cost of query-conditioned fusion and the saving from reducing three translator executions to one; the latter becomes dominant at larger $k$ (Appendix C.2).

Table 6: **Robustness to Retrieval Noise.** F1 and EM scores on the 2WQA Total task using LLaMA3.2-1B-Instruct across varying numbers of noise documents ($N_{noise}$). Both methods include retrieved textual context in the generation input.

| Noise | DyPRAG | | FusePRAG | |
|---|---|---|---|---|
| ($N_{noise}$) | F1 | EM | F1 | EM |
| 0 | 37.89 | 30.67 | **50.91** | **43.33** |
| 1 | 35.31 | 29.33 | **48.76** | **42.00** |
| 3 | 32.59 | 25.67 | **45.55** | **38.67** |
| 5 | 33.13 | 26.00 | **45.54** | **38.00** |
| 8 | 27.49 | 20.33 | **42.64** | **36.00** |

Table 7: Results for LLaMA3-8B-Instruct (EM/F1, %). Grouping, averaging, and highlighting follow Table 1.

| Method | 2WQA | | HQA | | PQA | | CWQ | | Avg | |
|---|---|---|---|---|---|---|---|---|---|---|
| | EM | F1 | EM | F1 | EM | F1 | EM | F1 | EM | F1 |
| **LLaMA3-8B-Instruct** | | | | | | | | | | |
| Vanilla | 28.33 | 35.05 | 20.83 | 30.01 | 4.67 | 7.96 | 30.33 | 42.58 | 23.75 | 31.35 |
| SFT | 5.08 | 8.16 | 1.00 | 1.58 | 0.00 | 0.00 | 4.00 | 8.72 | 3.29 | 5.57 |
| PRAG-w/o C | 32.92 | 40.19 | 38.33 | 48.68 | 20.00 | 25.83 | 32.00 | 42.53 | 32.54 | 40.81 |
| DyPRAG-w/o C | 31.17 | 38.25 | 25.17 | 38.01 | 10.00 | 14.44 | 31.00 | 41.41 | 27.00 | 35.61 |
| **FusePRAG-w/o C (ours)** | 31.83 | 38.69 | 29.67 | 39.26 | 18.00 | 21.57 | **38.33** | **48.95** | 30.38 | 37.98 |
| Standard RAG | 28.08 | 33.70 | 17.17 | 26.58 | 5.67 | 16.13 | 25.33 | 35.45 | 22.21 | 29.94 |
| SFT + RAG | 4.42 | 8.45 | 2.33 | 5.70 | 0.33 | 0.85 | 3.00 | 8.81 | 3.21 | 6.86 |
| PRAG | 34.83 | 42.61 | **41.50** | **53.82** | 11.33 | 26.23 | 28.00 | 36.44 | 32.71 | 42.59 |
| DyPRAG | 35.08 | 44.09 | 34.83 | 45.40 | 21.33 | 33.35 | 30.67 | 40.24 | 32.75 | 42.59 |
| **FusePRAG (ours)** | **38.75** | **46.92** | 39.00 | 49.49 | **22.67** | **35.19** | 32.33 | 42.74 | **36.00** | **45.57** |

FusePRAG also records lower total latency in both settings under naturally terminated generation. Because it produces fewer output tokens in these runs, the total values characterize the realized latency of this protocol rather than an architecture-only speedup. The component breakdown therefore provides direct evidence of reduced translator-related overhead, while the Enc. + LoRA Gen. and Total columns report the corresponding adapter-construction and post-retrieval inference costs, respectively.

# D Supplementary Experimental Results

## D.1 Robustness to Retrieval Noise

Table 6 reports robustness to controlled retrieval noise. We evaluate fixed DyPRAG and FusePRAG checkpoints by augmenting the gold 2WQA passages with $N_{noise} \in \{0, 1, 3, 5, 8\}$ synthetic hard negatives constructed as described in Appendix A. At each noise level, both methods receive the same passages for adapter construction and as textual generation context; no retraining is performed.

## D.2 Results with LLaMA3-8B-Instruct

Table 7 reports the full results with LLaMA3-8B-Instruct. The grouping, evaluation metrics, averaging protocol, and highlighting convention follow the main results in Table 1.

The static SFT baseline degrades substantially on this backbone when the same training pipeline used for the two smaller models is applied without backbone-specific retuning. DyPRAG (Tan et al., 2025) reports a similar pattern on LLaMA3-8B, with average F1 decreasing from 31.85 for the unchanged backbone to 7.80 after SFT; our corresponding values are 31.35 and 5.57. These results indicate that this fixed SFT configuration performs poorly when transferred to the 8B backbone without retuning;

Table 8: Matched-supervision comparison on Qwen2.5-1.5B-Instruct (EM/F1, %). Method-faithful DyPRAG follows its original training procedure. DyPRAG-Matched retains Project-then-Fuse while matching FusePRAG's Phase-2 translator initialization, 960-example QA supervision, and optimization budget; it does not use Phase-1 summary alignment. "w/o C" omits retrieved text from the generator while retaining retrieval-conditioned adapter generation. Bold marks the best result within each setting.

| Method | 2WQA | | HQA | | PQA | | CWQ | |
|---|---|---|---|---|---|---|---|---|
| | EM | F1 | EM | F1 | EM | F1 | EM | F1 |
| *Pure Parametric Setting (w/o C)* | | | | | | | | |
| DyPRAG (method-faithful) | 22.33 | 28.95 | 20.00 | 27.55 | 3.00 | **6.64** | 22.67 | 31.94 |
| DyPRAG-Matched | 24.00 | 29.49 | 19.33 | 25.57 | 3.67 | 6.21 | 19.00 | 27.30 |
| **FusePRAG-w/o C** | **26.17** | **31.53** | **22.17** | **28.76** | **4.33** | 6.47 | **25.00** | **33.89** |
| *Context-Augmented Setting* | | | | | | | | |
| DyPRAG (method-faithful) | 18.83 | 26.04 | 21.67 | 31.40 | 7.33 | 22.69 | 23.67 | 33.57 |
| DyPRAG-Matched | 21.92 | 27.25 | 22.50 | 28.49 | 12.33 | 23.48 | 22.00 | 30.14 |
| **FusePRAG** | **26.92** | **32.62** | **24.50** | **32.90** | **17.67** | **29.67** | **25.67** | **34.75** |

they do not establish a general limitation of SFT for larger models. Our primary comparisons remain Standard RAG, PRAG, and DyPRAG.

### D.3 Matched-Supervision Comparison with DyPRAG

The main DyPRAG results use a method-faithful reproduction of its document-to-LoRA translator training. We additionally construct DyPRAG-Matched on Qwen2.5-1.5B-Instruct to control for the downstream QA supervision introduced in FusePRAG Phase 2. As detailed in Appendix B.4, DyPRAG-Matched uses the same pre-trained translator initialization, 960 QA examples, three training epochs, and learning rate of $10^{-6}$ as FusePRAG Phase 2, while keeping the backbone frozen. It retains Project-then-Fuse: each top-3 document is translated independently, and the resulting LoRA updates are averaged before injection. Both the pure parametric and context-augmented configurations use the same retrieved passages and evaluation protocol as the main experiment.

Phase 1 has no direct counterpart in method-faithful DyPRAG because its target supervises a jointly fused representation. DyPRAG-Matched therefore controls Phase-2 initialization, QA supervision, and optimization while retaining document-wise Project-then-Fuse, testing whether matched downstream training closes the performance gap.

On Qwen2.5-1.5B, FusePRAG outperforms DyPRAG-Matched on every reported dataset and metric. The F1 gains are 2.04, 3.19, 0.26, and 6.59 points without textual context, and 5.37, 4.41, 6.19, and 4.61 points with textual context on 2WQA, HQA, PQA, and CWQ, respectively. The method-faithful DyPRAG remains the primary baseline because it preserves the original training procedure, whereas DyPRAG-Matched serves as a separate control for Phase-2 supervision. Across the eight dataset–setting F1 comparisons, matched QA fine-tuning improves three and degrades five relative to method-faithful DyPRAG. Thus, matching FusePRAG's Phase-2 QA fine-tuning does not reproduce its performance on this backbone. Under Project-then-Fuse, each translator input remains a single document representation. Although the answer-generation loss is query conditioned, it updates the translator through the average of independently generated document updates rather than providing the query and other documents as explicit inputs to each projection. Together with the Phase-1 and fusion ablations, this comparison supports the contribution of the complete FusePRAG training and fusion pipeline.

### D.4 Qualitative Case Study

Table 9 illustrates a CWQ example where evidence is distributed across two retrieved documents. Answering the question requires linking *Warren Moon* to the team associated with the fight song "*Skol, Vikings*". DyPRAG predicts the bridge entity itself, suggesting difficulty in composing the

two pieces of evidence. FusePRAG instead produces the correct answer, indicating that semantic interaction before projection can help combine dispersed evidence in this case.

Table 9: Case study on a CWQ example requiring cross-document evidence composition. The backbone model is LLaMA3.2-1B-Instruct. Yellow highlights key entities used for reasoning bridges, while green and red backgrounds indicate correct and incorrect answers, respectively.

| | | |
|---|---|---|
| **Question:** Who did Warren Moon play for whose fight song was "Skol, Vikings" ? | | |
| **Ground truth:** Minnesota Vikings | | |
| **Retrieved Documents:**<br>**[Doc 1]** Pro Football Hall of Fame quarterback Warren Moon served as the starting signal-caller for the NFL franchise whose fight song is "Skol, Vikings" for three seasons from 1994 to 1996.<br>**[Doc 2]** "Skol, Vikings" is the fight song of the Minnesota Vikings of the National Football League. It was introduced... | | |
| **Method** | **Answer** | **Status** |
| **DyPRAG** | Warren Moon | ✗ |
| **FusePRAG** | Minnesota Vikings | ✓ |

### D.5 Results Across Random Seeds

To assess sensitivity to random seeds, we repeat FusePRAG and FusePRAG-w/o C with seeds 42, 43, and 44 for each backbone. Table 10 reports the mean and sample standard deviation of F1. Across the six backbone–setting combinations, the standard deviation of average F1 ranges from 0.06 to 0.47, indicating limited seed sensitivity in aggregate performance. Although dataset-level variation is larger for LLaMA3-8B without textual context on PQA and CWQ, its average F1 remains stable across the three runs.

### D.6 Ablation in the Pure Parametric Setting

Table 11 extends the component analysis to the pure parametric setting. Replacing fusion with mean pooling reduces average F1 by 1.75 points, while removing Phase 1 produces a larger 6.60-point decrease. Removing the type embeddings has a small net effect (a 0.29-point decrease in average F1), consistent with the context-augmented results: explicit input roles provide a modest refinement, whereas removing the learned fusion module or Phase 1 leads to substantially larger performance drops.

### D.7 Isolating Direct Document–Document Attention

To more directly isolate the contribution of document–document interaction, we construct a capacity-matched variant, **w/o Direct Document–Document Attention**. It retains the same two-layer Transformer, parameter count, initialization procedure, type embeddings, and two-phase training protocol as FusePRAG. The only architectural change is an attention mask that prevents a document position from directly attending to a different document position. For three retrieved documents, the allowed attention pattern is

| | $q$ | $d_1$ | $d_2$ | $d_3$ |
|---|---|---|---|---|
| $q$ | ✓ | ✓ | ✓ | ✓ |
| $d_1$ | ✓ | ✓ | × | × |
| $d_2$ | ✓ | × | ✓ | × |
| $d_3$ | ✓ | × | × | ✓ |

where rows denote receiving positions and columns denote positions that may be attended to. The query can therefore attend to every document, while each document can attend only to itself and the query. Direct attention between distinct documents is disabled in every Transformer layer during both training and inference. Because the network has multiple layers, indirect information exchange through the updated query representation remains possible; this experiment isolates direct document–document attention rather than removing every possible path between documents.

Table 10: Random-seed sensitivity of FusePRAG. We report F1 as mean ± sample standard deviation over seeds 42, 43, and 44. "w/o C" omits retrieved text from the generator while retaining retrieval-conditioned adapter generation; "with C" additionally supplies retrieved text to the generator. Avg follows the eight-category macro-average defined in Section 4.1.

| Backbone | Setting | 2WQA | HQA | PQA | CWQ | Avg |
|---|---|---|---|---|---|---|
| LLaMA3.2-1B | w/o C | $32.64 \pm 0.56$ | $27.09 \pm 0.54$ | $9.95 \pm 0.79$ | $36.92 \pm 0.39$ | $28.95 \pm 0.38$ |
| | with C | $35.19 \pm 0.31$ | $34.43 \pm 0.03$ | $36.85 \pm 0.27$ | $44.74 \pm 0.37$ | $36.40 \pm 0.13$ |
| Qwen2.5-1.5B | w/o C | $31.84 \pm 0.27$ | $28.92 \pm 0.16$ | $6.63 \pm 0.17$ | $33.98 \pm 0.36$ | $28.23 \pm 0.20$ |
| | with C | $32.29 \pm 0.34$ | $33.23 \pm 0.49$ | $29.62 \pm 0.31$ | $34.87 \pm 0.31$ | $32.51 \pm 0.06$ |
| LLaMA3-8B | w/o C | $38.86 \pm 0.17$ | $39.27 \pm 0.13$ | $20.73 \pm 1.40$ | $48.80 \pm 1.01$ | $37.93 \pm 0.27$ |
| | with C | $46.44 \pm 0.65$ | $49.71 \pm 0.45$ | $35.26 \pm 0.37$ | $43.17 \pm 0.61$ | $45.45 \pm 0.47$ |

Table 11: **Ablation results in the pure parametric setting using LLaMA3.2-1B.** All values are F1 scores (%). Retrieved text is omitted from the generator, while retrieval still conditions adapter generation. Avg follows the eight-category macro-average defined in Section 4.1, and Δ Avg is measured relative to FusePRAG-w/o C. The best value in each score column is bolded.

| Method | 2WQA | HQA | PQA | CWQ | Avg | Δ Avg |
|---|---|---|---|---|---|---|
| **FusePRAG-w/o C** | **33.29** | **27.71** | 9.11 | **37.31** | **29.38** | – |
| Doc. Mean Pool (w/o Query) | 31.66 | 24.63 | 8.80 | 36.36 | 27.63 | -1.75 |
| w/o Phase 1 (End-to-End) | 23.31 | 26.16 | 2.36 | 34.31 | 22.78 | -6.60 |
| w/o Type Embeddings | 33.03 | 27.10 | **9.44** | 36.97 | 29.09 | -0.29 |

Removing direct document–document attention reduces F1 on every benchmark in both settings. The decreases are 1.35, 1.44, 0.67, and 1.15 points without textual context, and 0.62, 0.87, 0.97, and 1.29 points with textual context on 2WQA, HQA, PQA, and CWQ, respectively. Since architecture size and training are held fixed, these results provide evidence for an incremental contribution from direct document–document attention. They do not measure the total contribution of Fuse-then-Project: the masked variant retains query conditioning, pre-projection fusion, Phase-1 alignment, and indirect exchange through the query.

### D.8 Stronger Fusion Baselines

To assess whether the gains over mean pooling arise merely from adding query conditioning, we compare against two query-conditioned fusion alternatives:

- **Query-Aware Attention Pool**: Employing a learned attention pooling layer conditioned on the query representation to aggregate document features.
- **Cross-Attention Fusion**: Utilizing the query embedding to attend to document embeddings via a multi-head cross-attention layer.

We evaluate these variants on all four benchmarks using the Qwen2.5-1.5B-Instruct backbone, with the remaining training and evaluation settings held fixed. As shown in Table 13, cross-attention consistently improves over mean pooling, whereas query-aware attention pooling is more task dependent and performs particularly well on PQA. FusePRAG achieves the highest F1 in seven of the eight setting–benchmark combinations. In the context-augmented setting, it exceeds Cross-Attention Fusion by 0.74–4.59 points across the four benchmarks; without textual context, it performs best on 2WQA, HQA, and CWQ, while query-aware attention pooling obtains the highest PQA score. Within this controlled backbone comparison, the full fusion module generally outperforms the simpler query-conditioned alternatives, while the pure-parametric PQA result shows that the preferred fusion mechanism can remain task dependent.

Table 12: Controlled ablation of direct document–document attention using LLaMA3.2-1B-Instruct. All values are F1 scores (%). The variants differ only in the attention mask; architecture size, type embeddings, training data, two-phase objectives, optimization budget, and evaluation protocol are unchanged. "w/o C" omits retrieved text from the generator while retaining retrieval-conditioned adapter generation.

| Method | 2WQA | HQA | PQA | CWQ |
|---|---|---|---|---|
| *Pure Parametric Setting (w/o C)* | | | | |
| w/o Direct Doc–Doc Attn. | 31.94 | 26.27 | 8.44 | 36.16 |
| **FusePRAG-w/o C** | **33.29** | **27.71** | **9.11** | **37.31** |
| *Context-Augmented Setting* | | | | |
| w/o Direct Doc–Doc Attn. | 34.62 | 33.53 | 36.16 | 43.74 |
| **FusePRAG** | **35.24** | **34.40** | **37.13** | **45.03** |

Table 13: Comparison of fusion mechanisms using Qwen2.5-1.5B-Instruct. All values are F1 scores (%). "w/o C" omits retrieved text from the generator while retaining retrieval-conditioned adapter generation. All variants use the same training and evaluation protocol. The best value in each column within each setting is bolded.

| Method | 2WQA | HQA | PQA | CWQ |
|---|---|---|---|---|
| *Pure Parametric Setting (w/o C)* | | | | |
| Mean Pooling | 28.92 | 27.59 | 6.51 | 32.02 |
| Query-Aware Attention Pool | 28.37 | 26.86 | **7.52** | 27.56 |
| Cross-Attention Fusion | 30.84 | 27.95 | 6.73 | 32.78 |
| **FusePRAG-w/o C (Ours)** | **31.53** | **28.76** | 6.47 | **33.89** |
| *Context-Augmented Setting* | | | | |
| Mean Pooling | 26.20 | 30.94 | 22.82 | 33.57 |
| Query-Aware Attention Pool | 26.56 | 27.03 | 26.85 | 30.41 |
| Cross-Attention Fusion | 29.36 | 31.63 | 25.08 | 34.01 |
| **FusePRAG (Ours)** | **32.62** | **32.90** | **29.67** | **34.75** |

### D.9 Summary-based Baselines

To compare learned fusion against explicit inference-time summarization, we introduce two summary-based baselines. Unlike the answer-aware evidence summaries used as Phase-1 training targets, their summaries are generated at evaluation time from only the test question and retrieved passages using Figure 5; the summary generator never receives the reference answer or gold passages. We evaluate the following variants:

- **Summary → Context**: Explicitly appending the LLM-generated summary into the input context (a non-parametric, explicit integration approach).
- **Summary → Projector**: Encoding the generated summary into a dense embedding and feeding it directly into the parameter translator to generate adapter weights.

As shown in Table 14, FusePRAG-w/o C exceeds Summary → Projector by 1.10 F1 points, while FusePRAG exceeds the strongest context-augmented summary baseline by 3.40 points. Under the evaluated setup, the complete FusePRAG strategy outperforms the evaluated inference-time summarization strategies. The Phase-1 targets are constructed with gold answers, whereas the inference summaries use only the test question and retrieved passages; the experiment consequently compares complete deployable strategies rather than serving as an ablation of answer-aware target construction.

Table 14: Comparison with summary-based baselines on CWQ using LLaMA3.2-1B. Inference summaries are generated from only the test question and retrieved passages, without access to reference answers or gold passages.

| Method | F1 | EM |
|---|---|---|
| *Pure Parametric Setting (w/o C)* | | |
| Summary → Projector | 36.21 | 26.33 |
| **FusePRAG-w/o C (Ours)** | **37.31** | **28.33** |
| *Context-Augmented Setting* | | |
| Summary → Context | 38.11 | 26.33 |
| Summary → Projector (with C) | 41.63 | 31.00 |
| **FusePRAG (Ours)** | **45.03** | **33.33** |

### D.10 Robustness to Synthetic Supervision Quality

In our main experiments (Section 4.1), we use GPT-3.5-Turbo to synthesize answer-aware evidence summaries for Phase-1 semantic alignment. To examine whether FusePRAG is sensitive to the choice of summary generator, we replace GPT-3.5-Turbo with the open-weight **LLaMA3.2-1B-Instruct** model and regenerate the training summaries. We then retrain both FusePRAG-w/o C and FusePRAG using the **LLaMA3.2-1B-Instruct** backbone while keeping the remaining experimental setup fixed.

As shown in Table 15, the effect of replacing GPT-3.5-Turbo with LLaMA3.2-1B is small but task dependent. In the pure parametric setting, F1 changes by $-0.27$, $-0.78$, $+0.67$, and $-0.98$ points on 2WQA, HQA, PQA, and CWQ, respectively. In the context-augmented setting, the corresponding changes are $+0.75$, $+0.04$, $-0.41$, and $-0.85$ points. Thus, the open-weight generator yields broadly comparable results rather than a uniform degradation, suggesting that the alignment strategy is not tied to a proprietary summary generator. The remaining variation indicates that summary quality can still affect individual tasks.

Table 15: Performance (F1 score) of FusePRAG-w/o C and FusePRAG with the LLaMA3.2-1B-Instruct backbone when Phase-1 answer-aware evidence summaries are generated by different models. All other training and evaluation settings are held fixed.

| Generator | 2WQA | HQA | PQA | CWQ |
|---|---|---|---|---|
| *Pure Parametric Setting (w/o C)* | | | | |
| GPT-3.5-Turbo | 33.29 | 27.71 | 9.11 | 37.31 |
| LLaMA3.2-1B | 33.02 | 26.93 | 9.78 | 36.33 |
| *Context-Augmented Setting* | | | | |
| GPT-3.5-Turbo | 35.24 | 34.40 | 37.13 | 45.03 |
| LLaMA3.2-1B | 35.99 | 34.44 | 36.72 | 44.18 |