Retrieval from Within: An Intrinsic Capability of Attention-Based Models

Elad Hoffer[1], **Yochai Blau**[1], **Edan Kinderman**[1], **Ron Banner**[1], **Daniel Soudry**[1,2], **Boris Ginsburg**[1]

[1]NVIDIA
[2]Department of Electrical Engineering, Technion, Haifa, Israel
{elad.hoffer, daniel.soudry}@gmail.com
{yblau,ekinderman,rbanner,bginsburg}@nvidia.com

Abstract

Retrieval-augmented generation (RAG) typically treats retrieval and generation as separate systems. We ask whether an attention-based encoder-decoder can instead retrieve directly from its own internal representations. We introduce *INTRA* (*INTrinsic Retrieval via Attention*), a framework where decoder attention queries score pre-encoded evidence chunks that are then directly reused as context for generation. By construction, INTRA unifies retrieval and generation, eliminating the retriever-generator mismatch typical of RAG pipelines. This design also amortizes context encoding by reusing precomputed encoder states across queries. On question-answering benchmarks, INTRA outperforms strong engineered retrieval pipelines on both evidence recall and end-to-end answer quality. Our results demonstrate that attention-based models already possess a retrieval mechanism that can be elicited, rather than added as an external module.

1 Introduction

1.1 Motivation

Large language models are increasingly used in settings where the information needed to answer a query is sparse relative to the full available corpus. This is the regime in which retrieval-augmented generation (RAG) has become the default design: a retriever selects candidate evidence, which is then used by a generator to produce an answer [20]. This decomposition is practical because naively concatenating all available context into a single long prompt is computationally expensive, and even large-context models remain brittle when the relevant evidence is sparse and distributed [39, 24].

At the same time, this standard framing encourages a strong architectural separation between retrieval and generation. The retriever operates over indexed text or embeddings, while the language model consumes the selected evidence only after that selection is complete. In practice this modularity is often helpful, but it can obscure an important fact: attention is already a query-conditioned mechanism for selecting and weighting relevant information. This motivates the central question of the paper: can a single pretrained encoder-decoder retrieve the needed evidence and use it to answer a query? More broadly, how much of RAG can be handled inside the model itself, without a separate retriever?

1.2 Retrieval as an intrinsic capability

We study question answering using a fixed knowledge base and ask whether a pretrained encoder-decoder can retrieve, prioritize, and use evidence drawn from its own representation space. Our central hypothesis is that pretrained attention-based models already possess an intrinsic retrieval capability in this setting. We call this regime *INTRA* (*INTrinsic Retrieval via Attention*): rather

Preprint.

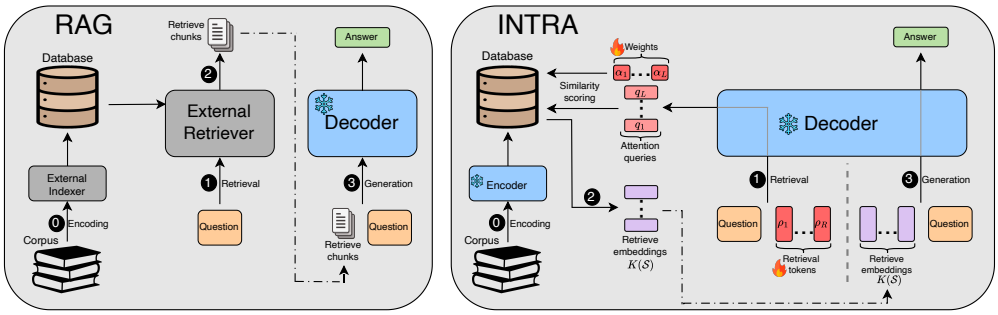

Figure 1: **Left**: Standard RAG pipeline. An external retriever selects documents which are then re-encoded by the decoder to produce an answer. Retrieval and generation operate in separate representation spaces. **Right**: Our method, INTRA, uses a pretrained frozen encoder-decoder for both retrieval and generation. The decoder retrieves relevant chunks through its cross-attention queries, augmented with learnable retrieval tokens. The retriever and generator share a representation space, allowing pre-encoded evidence to be reused across queries. No external retriever is required. Numbers indicate the sequence of operations.

than relying on an external retriever, the model selects evidence and generates answers over the representations produced by its own encoder.

The connection between attention and retrieval is made concrete in Section 2.2: both are query-conditioned matching operations over candidate states. Within this framing, INTRA transforms the decoder's cross-attention queries into scores for chunk-level retrieval. This perspective does not suggest that attention mechanisms constitute a complete solution for large-corpus retrieval. Rather, it suggests that a pretrained encoder-decoder contains the right interface for retrieval: query states that express what the decoder needs, and encoded evidence states that can be selected and consumed without translation into another representation space.

This design has several practical advantages. The same encoded chunk states are used for both evidence selection and answer generation, reducing the mismatch between a separately trained retriever and the generator it serves. Because those states are encoder memories, static evidence can be encoded once and reused across queries instead of being repeatedly packed into a long decoder context. Finally, the retrieval mechanism can be adapted with lightweight decoder-side retrieval queries rather than a separately trained retriever, reducing the need for a dedicated retrieval system.

1.3 Contributions

- We formulate *INTRA*, in which a single pretrained encoder-decoder model uses one shared representation space to couple evidence selection with answer generation.
- We identify a minimal architectural recipe for exposing this capability: the pretrained encoder's native chunk representations are reused directly, encoder-side late interaction performs coarse retrieval, and decoder-side retrieval queries refine evidence without introducing a separate retriever or compression model, as shown in Figure 1.
- We show empirically that this unified retrieval-generation design is especially strong on multi-hop QA. It rivals strong engineered retrieval pipelines despite their use of large-scale training data, while utilizing the same latent evidence for both selection and generation.
- We characterize the computational profile of this design, including the reusable-context regime that emerges when static evidence is encoded once and reused across queries.

2 Method

2.1 Framework formulation

We consider a retrieval-and-generation setting in which the model generates an output from a small set of relevant evidence chunks. Let $\mathcal{T} = \{t_i\}_{i=1}^{M}$ denote the corpus of text chunks, and let

$\mathcal{S} \subseteq \{1, \ldots, M\}$ denote the selected chunk indices. For a selected set, the retrieved text context is

$$T(\mathcal{S}) = \Big[t_i : i \in \mathcal{S} \Big],$$

where $[\cdot]$ denotes concatenation. Given an input x (e.g., a question), a decoder Dec produces the output y conditioned on this context:

$$y = \mathrm{Dec}\big(x, T(\mathcal{S})\big).$$

In a standard RAG pipeline, the selected set $\mathcal{S}$ is obtained from an *external* retrieval function, $\mathcal{S} = \mathrm{retrieve}(x, \mathcal{T})$, and Dec is usually a separate LLM that generates from the retrieved text.

We focus on *encoder-decoder* models, where the same pretrained model can encode evidence and decode the answer. The encoder Enc maps a text sequence t to token representations

$$k = \mathrm{Enc}(t) \in \mathbb{R}^{L_c \times d},$$

where d is the representation dimension. For each corpus chunk, we write $k_i = \mathrm{Enc}(t_i)$ and denote the pre-encoded chunk set by $\mathcal{K} = \{k_i\}_{i=1}^{M}$. For a selected set $\mathcal{S}$, the encoded context is

$$K(\mathcal{S}) = \Big[k_i : i \in \mathcal{S} \Big],$$

where the same concatenation notation is now applied to token representations. Generation in our setting conditions on the encoded context rather than on raw retrieved text, so

$$y = \mathrm{Dec}(x, K(\mathcal{S})).$$

To make the decoder queries explicit, we use a view that isolates the cross-attention computation. Let

$$\mathrm{Attention}(q, k, v) \triangleq \mathrm{softmax}\left(\frac{qk^\top}{\sqrt{d}}\right) v.$$

During the forward pass that computes $\mathrm{Dec}(x, K(\mathcal{S}))$, let $h_0 = x$ and let q_ℓ denote the query-side state consumed by cross-attention in decoder layer ℓ. The simplified internal recurrence is

$$\begin{array}{rcll} q_\ell & = & \Psi_\ell(h_{\ell-1}), & \\ z_\ell & = & \mathrm{Attention}(q_\ell, K(\mathcal{S}), K(\mathcal{S})), & \\ h_\ell & = & \Phi_\ell(h_{\ell-1}, z_\ell), & \ell = 1, \ldots, L. \end{array} \tag{1}$$

Here Ψ_ℓ, Φ_ℓ denotes the layer-specific transformation, including residual updates, self-attention, feed-forward transformations, and normalization. The decoder output is then produced from the final internal state, where Out denotes the model's final text-generation head.

$$y = \mathrm{Dec}(x, K(\mathcal{S})) = \mathrm{Out}(h_L).$$

We also mark $\widetilde{\mathrm{Dec}}$ for the same decoder forward pass, with the intermediate query states exposed:

$$\forall \ell = 1, \ldots, L : q_\ell = \widetilde{\mathrm{Dec}}_\ell(x, K(\mathcal{S})).$$

2.2 Attention-based retrieval

Cross-attention already scores decoder-side query states against encoder-side token representations. We use the same matching signal to rank chunks before generation. Our goal is to convert the token-level comparison in Eq. 1 into a single retrieval score for each encoded chunk k_i. To obtain these scores with a pre-trained *frozen* decoder, we augment the input x with R trainable retrieval tokens $\rho \in \mathbb{R}^{R \times d}$. Given token embeddings $\{x_j\}_{j=1}^{L_q}$, the retrieval input is

$$x_{\mathrm{retrieval}} = \big[x_1, \ldots, x_{L_q}, \rho_1, \ldots, \rho_R\big]. \tag{2}$$

To move from token-level attention scores to chunk-level retrieval scores, we use a scaled ColBERT-style late-interaction score MaxSim [18]*. For sequences $u \in \mathbb{R}^{L_u \times d}$ and $v \in \mathbb{R}^{L_v \times d}$,

$$\mathrm{MaxSim}(u, v) \triangleq \sum_{a=1}^{L_u} \max_{1 \le b \le L_v} \left(\frac{uv^\top}{\sqrt{d}}\right)_{a,b},$$

*The factor $1/\sqrt{d}$ has no effect on MaxSim ranking; it is included only to make the similarity to the attention score explicit.

MaxSim uses the same scaled token-level dot product as attention, but aggregates by taking the best-matching chunk token for each query token rather than applying a softmax over all tokens.

With learned per-layer aggregation weights α_ℓ, the score for chunk i is

$$s_i \triangleq \sum_\ell \alpha_\ell \mathrm{MaxSim}(q_\ell, k_i) \quad \text{where} \quad q_\ell = \widetilde{\mathrm{Dec}}_\ell(x_{\mathrm{retrieval}}, K(\mathcal{S}_0)) \,, \tag{3}$$

where $\mathcal{S}_0$ is the initial chunk selection (See section 2.3 for how $\mathcal{S}_0$ is constructed). We then select the chunks with the largest scores:

$$\mathcal{S}_{\mathrm{INTRA}} = \{i \in \{1, \ldots, M\} : s_i \text{ is among the top-}n \text{ scores}\} \,. \tag{4}$$

Thus, $\mathcal{S}_{\mathrm{INTRA}}$ is the set of selected chunk indices. Then the ordinary decoder generates from that selected context:

$$y = \mathrm{Dec}\left(x, K(\mathcal{S}_{\mathrm{INTRA}})\right). \tag{5}$$

Inference thus consists of two decoder forward passes over the pre-encoded chunk set $\mathcal{K} = \{k_i\}_{i=1}^M$: a retrieval pass $\widetilde{\mathrm{Dec}}(x_{\mathrm{retrieval}}, K(\mathcal{S}_0))$ that exposes the query states $\{q_\ell\}$ used in Eq. 3 to score all chunks, followed by a generation pass over the selected context $K(\mathcal{S}_{\mathrm{INTRA}})$.

2.3 Initial context selection for retrieval

A natural initial chunk set $\mathcal{S}_0$ in our setting is to select chunks whose encoded representations are most similar to the encoded input. Let $k_x = \mathrm{Enc}(x)$. We define

$$s_i^{(0)} = \mathrm{MaxSim}(k_x, k_i), \qquad \mathcal{S}_0 = \left\{ i \in \{1, \ldots, M\} : s_i^{(0)} \text{ is among the top-}n_0 \text{ scores} \right\}. \tag{6}$$

This initialization provides the decoder with a useful starting context, but it does not restrict the final retrieval set. The set $\mathcal{S}_{\mathrm{INTRA}}$ is still selected by scoring the full corpus as in Eq. 4, and can therefore include chunks outside $\mathcal{S}_0$. This differs from reranking methods, which only reorder an initially retrieved candidate set. Another possible initial selection is the empty set $\mathcal{S}_0 = \emptyset$, in which case cross-attention is the identity function and $z_\ell = q_\ell$ in Eq. 1.

3 Practical Implementation

We now describe the practical changes needed to adapt pretrained encoder-decoder attention models to the INTRA framework. Our implementation starts from T5Gemma2 [41] and modifies the decoder cross-attention so that pre-encoded chunk states can be reused directly for retrieval and generation.

3.1 Shared context representations across layers

In T5Gemma2, as in other Transformer-based encoder-decoder models, the cross-attention computation $z_\ell = \mathrm{Attention}(q_\ell, K(\mathcal{S}), K(\mathcal{S}))$ defined in Eq. 1 does not take dot products directly against the raw stored encoder states $K(\mathcal{S})$. Instead, the key inputs to the attention function are subject to layer-specific transformations. The corresponding keys are typically computed by applying an RMSNorm with learned scale $\gamma_{K,\ell}$ and a linear projection matrix $W_{K,\ell}$. Thus, in Eq. 1 and related equations, we would need to replace $K(\mathcal{S})$ with $K_\ell(\mathcal{S}) = (\mathrm{RMSNorm}(K(\mathcal{S})) \odot \gamma_{K,\ell}) W_{K,\ell}$. This would require computing layer-specific representations $K_\ell(S)$ to evaluate MaxSim for retrieval, rather than evaluating against a single reusable context across all layers.

To avoid this overhead, we propose reversed query-key projection *Reverse-QWK* (or *RQWK*), a novel technique that stores one normalized encoder representation $\bar{K}(\mathcal{S}) = \mathrm{RMSNorm}(K(\mathcal{S}))$ and moves the learned key scale $\gamma_{K,\ell}$ and projection matrix $W_{K,\ell}$ to the query side, defining a modified query transformation:

$$\widetilde{q}_\ell = (q_\ell W_{K,\ell}^\top) \odot \gamma_{K,\ell}.$$

Cross-attention can then be computed against the same normalized encoder states for all layers, maintaining equivalence with Eq. 1:

$$z_\ell = \mathrm{Attention}_{\mathrm{RQWK}}(\widetilde{q}_\ell, \bar{K}(\mathcal{S}), \bar{K}(\mathcal{S})) = \mathrm{softmax}\left(\frac{\widetilde{q}_\ell \bar{K}(\mathcal{S})^\top}{\sqrt{d}} \right) \bar{K}(\mathcal{S}) \tag{7}$$

The MaxSim score in Eq. 3 is computed using these same quantities, $\mathrm{MaxSim}(\widetilde{q}_\ell, \bar{k}_i)$ (where $\bar{k}_i = \mathrm{RMSNorm}(k_i)$), so retrieval and attention operate in the same representation space. This preserves the attention scores while allowing retrieval queries from different decoder layers to operate on the same stored chunk pool $K(\mathcal{S})$.

We use Reverse-QWK only as an implementation device; the full derivation including per-head treatment under Group-Query Attention, handling of positional embeddings, and the resulting memory savings are deferred to Appendix A.

3.2 Retrieval training objective

Let $\mathcal{O}(x) \subseteq \{1, \ldots, M\}$ denote the oracle evidence chunks for input x (e.g., a question). When explicit retrieval supervision is available, we train the retrieval scores s_i from Eq. 3 with a soft cross-entropy objective that assigns equal target mass to all oracle chunks:

$$\mathcal{L}_{\text{retrieval}} = -\frac{1}{|\mathcal{O}(x)|} \sum_{j \in \mathcal{O}(x)} \log\left(\mathrm{softmax}(s)_j\right),$$

where $s = [s_1, \ldots, s_M]$. With the decoder frozen, this objective updates the retrieval tokens ρ in Eq. 2 and the layer aggregation weights α in Eq. 3, teaching the induced decoder queries to place probability mass on the oracle evidence set.

3.3 Approximate similarities with pooled chunk embeddings

Computing MaxSim against every token is expensive when chunk length L_c is large. For efficient scoring, we replace each encoded chunk $k_i \in \mathbb{R}^{L_c \times d}$ with a fixed-length mean-pooled sequence $\widehat{k}_i \in \mathbb{R}^{L_p \times d}$ where $L_p \ll L_c$. Retrieval scores are computed using $\widehat{k}_i$ in place of k_i. This approximation is natural for INTRA because the pooled vectors are fixed averages of the model's own encoder states, requiring no additional compressor or compression-specific training (distinct from latent-compression approaches [11]). We find that small values such as $L_p \in \{3, 5, 7\}$ substantially reduce MaxSim cost while preserving the shared-representation design.

4 Benchmarks and Experimental Setup

We evaluate INTRA on four Wikipedia-based QA benchmarks: HotPotQA [38], 2WikiMultihopQA [12], MuSiQue [34], and Natural Questions [19]. Together they span bridge and comparison reasoning, cleaner two-hop evidence chains, compositionally harder multi-hop questions, and single-hop open-domain QA. We build one shared retrieval candidate pool for all four benchmarks under a fixed budget of approximately 100M tokens, containing 759K chunks in total. Full details on the context pool construction and split statistics are provided in Appendix E.

We compare INTRA against nine retrieval baselines, including sparse lexical methods (TF-IDF [29], BM25 [28]), dense single-vector models (BGE-large [37], Qwen3-Embedding-0.6B/4B [42]), reranking (Jina reranker [14]), hybrid RAG (RRF [6]), and a ColBERT-style MaxSim late-interaction baseline [18] (details in Appendix B.2). For retrieval, we report complete-evidence recall at $k \in \{5, 10, 20\}$, defined as the fraction of examples where *all* oracle chunks are retrieved. For end-to-end QA, we take the top-5 retrieved chunks, pack their pre-encoded T5Gemma2 representations as cross-attention context, and generate answers with the T5Gemma2 model, reporting exact match (EM) and token-level F1. All experiments use the open retrieval setting.

Implementation Details. We initialize from a T5Gemma2 4B-4B checkpoint, warm-started on the CLaRa QA pretraining dataset [11] and adapted on our training splits. During retrieval training, the encoder and decoder backbones are frozen, optimizing only the retrieval token embeddings ρ_i ($\sim$ 164K parameters) and layer aggregation weights α_l (272 parameters). Initial context $\mathcal{S}_0$ uses $n_0 = 20$ and pooled chunks of length $L_p = 7$. At evaluation, QA builds a five-chunk context: the top four retrieved chunks from $\mathcal{S}_{\mathrm{INTRA}}$ plus the top initial context chunk from $\mathcal{S}_0$. We then generate with deterministic greedy decoding. Further details and ablations are reported in Appendices B.1 and C.1.

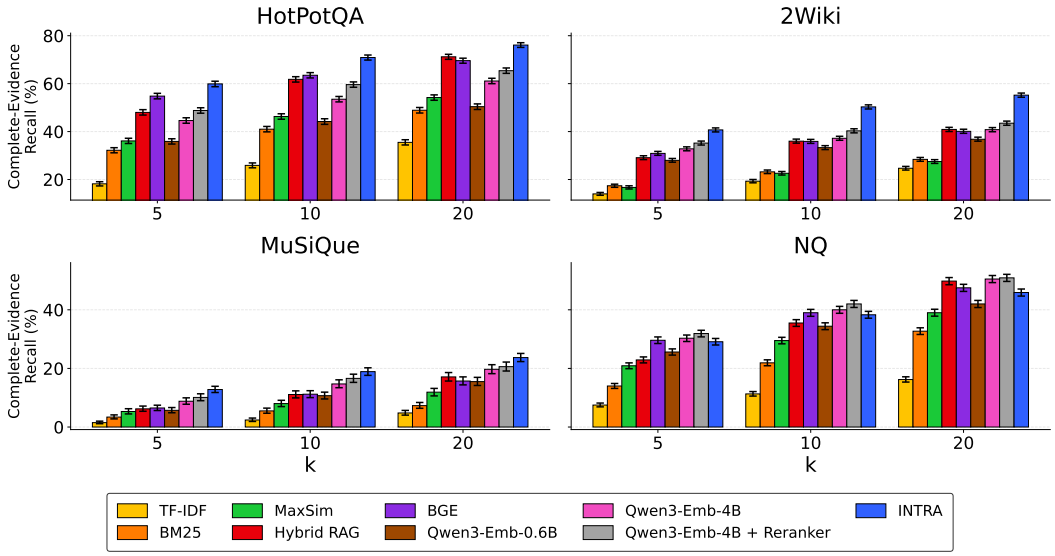

Figure 2: Complete-evidence recall: the percentage of examples for which *all* supporting facts are retrieved. INTRA performs best on multi-hop benchmarks (HotPotQA, 2Wiki, MuSiQue) that require evidence assembly. NQ's single-hop nature minimizes this benefit.

5 Results

We organize the results around the three empirical questions that motivate the paper. First, does INTRA improve retrieval of complete evidence sets (Section 5.1)? Second, do those gains translate into better end-to-end answer quality (Section 5.2)? Third, what efficiency advantage appears once chunk representations are reused rather than re-encoded from raw text (Section 5.3)?

5.1 Retrieval Results

Figure 2 reports complete-evidence recall for $k \in \{5, 10, 20\}$ across the four evaluation benchmarks. Complete-evidence recall@k is the fraction of examples for which *all* annotated supporting chunks are retrieved within the top-k results. We view this metric as the clearest proxy for retrieval quality, because it rewards recovering the full supporting set rather than only incomplete supporting evidence.

The main pattern is that INTRA is strongest on multi-hop retrieval settings that require assembling multiple pieces of evidence (HotPotQA, 2Wiki, MuSiQue). INTRA's ranking leverages decoder attention weights, which serve as a proxy for the informational requirements of the answer generation process. This advantage is less pronounced on NQ, where retrieval often reduces to finding one directly supporting passage, leaving less room for decoder-guided evidence assembly. Appendix C reports the full retrieval results.

In Fig. 3 we also compare three top-5 evidence sets: the initial retrieval set $\mathcal{S}_0$, the same initial set reranked by the decoder scores s_i from Eq. 3, and the final INTRA set $\mathcal{S}_{\text{INTRA}}$ from Eq. 4. The results show that reranking $\mathcal{S}_0$ is beneficial, but full-corpus INTRA scoring yields the largest gains by recovering evidence absent from the initial pool.

5.2 End-to-End Question-Answering Results

Table 1 evaluates the end-to-end retrieval-and-generation behavior of INTRA. We vary the retrieval method while keeping the same T5Gemma2 decoder for generation, reporting both exact match (EM) and token-level F1 (full results are in Appendix C). INTRA surpasses all baselines on multi-hop benchmarks (HotPotQA, 2Wiki, MuSiQue), consistent with the results of Section 5.1. This is notable because INTRA's retrieval signal comes from a frozen decoder pretrained only for generation, whereas baselines such as BGE and Qwen-Embedding are pretrained for retrieval on large-scale retrieval corpora (that include HotPotQA and NQ as supervision [33, 42]).

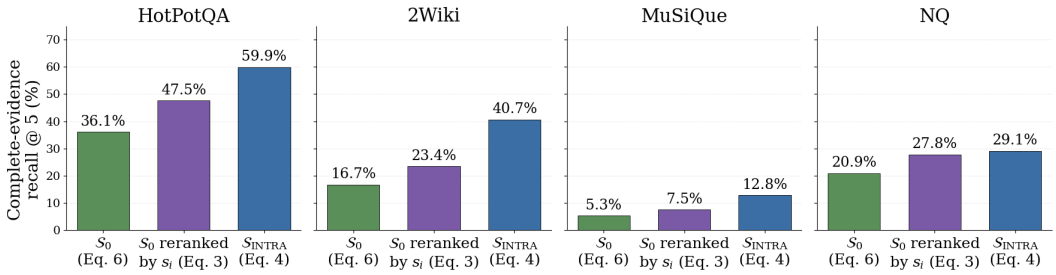

Figure 3: Complete-evidence recall@5 for the initial set $\mathcal{S}_0$, $\mathcal{S}_0$ reranked, and the final set $\mathcal{S}_{\text{INTRA}}$. INTRA performs corpus-wide scoring and can recover evidence outside the initial candidate pool.

Table 1: End-to-end question-answering performance across retrieval methods with a fixed T5Gemma2 generator. INTRA performs best on all multi-hop benchmarks.

	HotPotQA		2Wiki		MuSiQue		NQ		Average	
Retrieval method	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
TF-IDF	34.2	44.5	39.0	42.7	5.3	14.6	34.9	42.9	28.4	36.2
BM25	40.5	52.0	41.7	45.2	7.7	16.8	43.4	51.8	33.3	41.5
MaxSim	40.7	52.2	41.6	45.8	10.1	19.8	48.4	57.8	35.2	43.9
Hybrid RAG	43.4	54.3	46.0	49.9	10.6	20.1	50.5	59.1	37.6	45.8
BGE	41.9	53.0	46.1	49.6	10.8	19.7	52.2	61.2	37.8	45.9
Qwen3-Emb-0.6B	37.0	47.7	45.7	49.8	11.1	20.0	36.4	44.6	32.6	40.5
Qwen3-Emb-4B	40.3	51.2	46.0	50.1	12.7	21.8	54.5	63.7	38.4	46.7
Qwen3-Emb-4B + Reranker	41.6	53.6	46.8	50.8	13.3	22.5	**55.1**	**64.2**	39.2	47.5
INTRA	**46.4**	**58.0**	**49.2**	**53.2**	**14.0**	**23.0**	51.2	60.3	**40.2**	**48.6**

Table 2 compares using a shared decoder for retrieval and generation against coupling an INTRA retriever with a stronger generator. While superior reasoning and parametric knowledge allow more capable generators to boost performance, INTRA retrieval enhances performance by aligning evidence with the decoder's specific attention patterns. To isolate the impact of generator strength, we measure how much of the EM gap between random and complete-evidence contexts is closed by INTRA:

$$\text{GapClosure} = 100 \cdot \frac{\text{EM(INTRA)} - \text{EM(random)}}{\text{EM(complete)} - \text{EM(random)}},$$

where the parenthetical term indicates whether generation uses chunks from $\mathcal{S}_{\text{INTRA}}$, random chunks, or the complete-evidence (oracle) chunks. Utilizing the same T5Gemma2 decoder for both retrieval and generation closes the largest average gap, demonstrating the benefit of coupling the retriever and generator. This highlights the need for stronger INTRA backbones, given that open-source encoder-decoder models are currently scarcer and weaker than decoder-only options.

5.3 Efficiency Results

INTRA's encoder-decoder design also yields a direct efficiency benefit. Standard RAG typically retrieves text, so after retrieval the generator re-encodes the selected chunks before decoding. INTRA retrieves pre-encoded chunks from $\mathcal{K}$ instead, and those states feed into generation as decoder cross-attention memory. Retrieval and generation incur their usual costs[1], but the selected evidence is no longer re-encoded at query time. Table 3 summarizes this computational trade-off (detailed analysis in Appendix D). Furthermore, storing these representations is practical, as storing a 1-billion-token corpus quantized to 8-bit precision requires around 2.5 TB of storage (see Appendix A.3 for details).

We isolate this effect with a time-to-first-token (TTFT) benchmark in Figure 4, which measures the generator-side cost after evidence has been selected. As the number of retrieved chunks k increases, INTRA's prefilling time remains small because it reuses stored chunk states, while standard RAG becomes slower. Appendix D.4 provides the setup and further generation throughput measurements.

[1]Retrieval complexity scales as $\mathcal{O}(\sqrt{M}L_qL_c)$ in practice using inverted file (IVF) approximate nearest-neighbor (ANN) search e.g. with FAISS or cuVS, [18, 15, 27].

Table 2: Generator compatibility with a T5Gemma2-INTRA retriever. Gap closure measures the percentage of the EM gap from random chunks to complete-evidence chunks recovered by INTRA retrieval. Sharing a decoder across retrieval and generation aligns the retrieved evidence with the generator's attention, closing the largest gap.

Generator	HotpotQA	2Wiki	MuSiQue	NQ	Average
Mistral0.3-7B	50.0%	21.6%	24.0%	56.8%	38.1%
Phi4-3.8B	55.9%	18.4%	18.9%	68.7%	40.5%
Llama3.1-8B	61.9%	38.8%	23.1%	71.1%	48.7%
Gemma4-E2B	63.0%	47.5%	25.7%	74.2%	52.6%
Qwen3.5-9B	64.0%	48.0%	26.1%	**78.5%**	54.1%
Qwen3.5-27B	63.2%	51.8%	23.9%	76.5%	53.8%
T5Gemma2-4B (same as retriever)	**66.4%**	**56.8%**	**38.5%**	75.9%	**59.4%**

Table 3: Computational trade-offs across full-context prompting, standard RAG, and INTRA. Variables denote number of corpus chunks M, chunk length L_c, query length L_q, retrieved chunks k, and generation length L_g. INTRA has the same retrieval and generation terms as RAG, but avoids re-encoding retrieved evidence during prefilling when chunk representations are reusable across queries. In our setting where $M \gg k$, full-context prefilling is computationally infeasible.

Model	**Pre-Query**	**Retrieval**	**Prefilling**	**Generation**
Full Context	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}((L_q + ML_c)^2)$	$\mathcal{O}(L_g(L_q + ML_c + L_g))$
Standard RAG	$\mathcal{O}(ML_c^2)$	$\mathcal{O}(\sqrt{M}L_qL_c)$	$\mathcal{O}((L_q + kL_c)^2)$	$\mathcal{O}(L_g(L_q + kL_c + L_g))$
INTRA	$\mathcal{O}(ML_c^2)$	$\mathcal{O}(\sqrt{M}L_qL_c)$	$\mathcal{O}(L_q(L_q + kL_c))$	$\mathcal{O}(L_g(L_q + kL_c + L_g))$

6 Related Work

Retrieval-Augmented Generation Pipelines. Most knowledge-intensive QA systems adopt a modular retrieval-then-generation architecture. REALM incorporates retrieval into language-model pretraining [10], and RA-DIT later instruction-tunes both retriever and generator [23]. DPR established dense passage retrieval as a strong open-domain QA primitive [16], and RAG couples a retriever with a sequence-to-sequence generator [20]. Atlas [13] jointly pre-trained a retriever with an encoder-decoder generator, to optimize few-shot learning from unstructured text. More recently, CLaRa [11] compresses documents into retrievable latent vectors, and jointly optimize reranking and generation. INTRA differs from CLaRa along three axes: (i) it reuses the model's native representation space, (ii) it performs full-corpus scoring, and (iii) it is built upon a frozen encoder-decoder model.

Multi-pass agentic RAG systems instead interleave reasoning and repeated retrieval [21, 1, 31]. INTRA targets the single-pass retrieval block that could be used within such pipelines, rather than the pipeline-level agentic loop itself.

Late Interaction and Representation-Space Retrieval. Our retrieval formulation is close to late-interaction systems such as ColBERT [18], ColBERTv2 [30] and ColPali [8], which compare query and document tokens via MaxSim-style matching over multi-vector representations. Whereas late-interaction systems rely on a dedicated retriever to score query-document matches, INTRA lets the decoder's own cross-attention perform this matching and then consume the matched representations during generation.

Memory, Latent Retrieval, and Unified Retrieval-Generation. Attention has long been viewed as content-based memory access. Memory Networks [32] framed QA as differentiable lookup, while RETRO [4, 36] inject retrieved chunks into the model's computation. Similarly, Titans [2] argues that long-context modeling requires explicit memory mechanisms rather than larger attention windows. This intuition is compatible with our perspective, yet they differ in the source of memory: Titans learns one, while INTRA reuses the model's activations over a evidence pool.

Long-Context Modeling and Efficient Sequence Architectures. Transformer architecture dominance for language modeling can be largely attributed to self-attention, which provides flexible

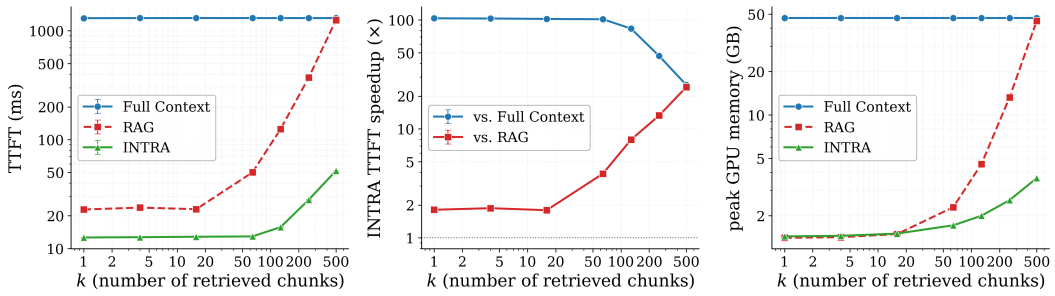

Figure 4: Time to first token vs. number of retrieved chunks k, excluding retrieval time. INTRA reuses pre-encoded evidence, while standard RAG re-encodes the retrieved text before decoding.

token interactions [35]. However, dense attention's quadratic cost hinders long-context scaling, prompting sparse and linear alternatives [3, 40, 5, 17]. More recent models, such as Mamba, explore further reductions in context-processing cost [9, 7, 25, 22]. However, recent long-context benchmarks suggest that merely increasing the window size does not ensure robust evidence use when relevant information is sparse [39, 24]. Our goal is therefore not to replace long-context modeling, but to study a setting in which sparse evidence must be identified and used reliably.

7 Limitations

Our experiments focus on a fixed context pool. We discuss the practicality of a billion-token corpus in App. A.3, but do not position INTRA as a replacement for RAG in open-web retrieval or web-scale settings. Likewise, while we show that a pretrained encoder-decoder exhibits strong retrieval behavior in this regime, we do not show that this mechanism generalizes to a general-purpose retriever. Moreover, we focus on a single implementation family: a T5Gemma2-style encoder-decoder with Reverse-QWK, evaluated on text-QA benchmarks with short answers. INTRA's reliance on encoder-decoder cross-attention also excludes decoder-only models. Extending these findings across scales, modalities, dynamic corpora, and architectures is left for future work.

The systems evaluation isolates a single mechanism. TTFT in Sec. 5.3 excludes retrieval time to isolate the cost of re-encoding versus reuse. Deployment cost also depends on indexing, storage format, and data movement, and token-level memories can be substantially larger than compressed retrieval indices, so end-to-end trade-offs may differ from those analyzed here.

8 Conclusion

Our central conclusion is conceptual: retrieval capabilities, typically outsourced to a separate module, can be carried out within an encoder-decoder's representation space. We introduce INTRA, a framework that unifies retrieval and generation by eliciting the intrinsic retrieval mechanism of attention-based models. By utilizing the same pretrained model for both tasks, INTRA eliminates the representation mismatch between the retriever and generator.

Empirically, INTRA shows strong performance on multi-hop question-answering benchmarks (HotPotQA, 2WikiMultihopQA, and MuSiQue), achieving results competitive with or exceeding several engineered RAG pipelines in both complete-evidence recall and end-to-end answer quality. Moreover, this shared-space formulation can yield a computational advantage: static evidence can be encoded once and reused across queries, reducing prefilling costs and time-to-first-token during generation. Ultimately, these findings suggest that attention-based encoder-decoders may offer a promising, unified alternative to traditional modular RAG architectures.

References

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

[2] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=8GjSf9Rh7Z.

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer, 2020. URL https://arxiv.org/abs/2004.05150.

[4] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan , et al. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

[5] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2022. URL https://arxiv.org/abs/2009.14794.

[6] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759. ACM, 2009. doi: 10.1145/1571941.1572114.

[7] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

[8] Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. Colpali: Efficient document retrieval with vision language models, 2024. URL https://arxiv.org/abs/2407.01449.

[9] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2024. URL https://arxiv.org/abs/2312.00752.

[10] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[11] Jie He, Richard He Bai, Sinead Williamson, Jeff Z. Pan, Navdeep Jaitly, and Yizhe Zhang. CLaRa: Bridging retrieval and generation with continuous latent reasoning, 2026. URL https://arxiv.org/abs/2511.18659.

[12] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625. International Committee on Computational Linguistics, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

[13] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251): 1–43, 2023. URL http://jmlr.org/papers/v24/23-0037.html.

[14] Jina AI. Jina reranker v2 base multilingual, 2024. URL https://jina.ai/models/jina-reranker-v2-base-multilingual. Released June 25, 2024.

[15] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.

[16] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

[17] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

[18] Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pp. 39–48, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075.

[19] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai , et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

[20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

[21] Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models, 2025. URL https://arxiv.org/abs/2501.05366.

[22] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman , et al. Jamba: A hybrid Transformer-Mamba Language Model, 2024. URL https://arxiv.org/abs/2403.19887.

[23] Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. RA-DIT: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=22OTbutug9.

[24] Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schütze. NoLiMa: Long-Context Evaluation Beyond Literal Matching, 2025. URL https://arxiv.org/abs/2502.05167.

[25] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models, 2023. URL https://arxiv.org/abs/2302.10866.

[26] PyTorch Team. TorchTitan. https://github.com/pytorch/torchtitan, 2024. Accessed: 2026-05-03.

[27] RAPIDS Team. *cuVS: GPU-Accelerated Vector Search and Clustering*, 2026. URL https://github.com/rapidsai/cuvs.

[28] Stephen Robertson and Hugo Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

[29] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988. doi: 10.1016/0306-4573(88)90021-0. URL https://doi.org/10.1016/0306-4573(88)90021-0.

[30] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3715–3734, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.272. URL https://aclanthology.org/2022.naacl-main.272/.

[31] Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V. Vasilakos. Agentic retrieval-augmented generation: A survey on agentic rag, 2026. URL https://arxiv.org/abs/2501.09136.

[32] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/8fb21ee7a2207526da55a679f0332de2-Paper.pdf.

[33] Nandan Thakur. BGE training dataset (only retrieval datasets). Hugging Face dataset card, 2023. URL https://huggingface.co/datasets/nthakur/bge-full-data. Ported version of cfli/bge-full-data; lists training splits including hotpotqa and nq. Accessed 2026-05-03.

[34] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[36] Boxin Wang, Wei Ping, Lawrence McAfee, Peng Xu, Bo Li, Mohammad Shoeybi, and Bryan Catanzaro. Instructretro: Instruction tuning post retrieval-augmented pretraining, 2024. URL https://arxiv.org/abs/2310.07713.

[37] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings, 2024. URL https://arxiv.org/abs/2309.07597.

[38] 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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

[39] Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context language models effectively and thoroughly, 2024. URL https://arxiv.org/abs/2410.02694.

[40] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2021. URL https://arxiv.org/abs/2007.14062.

[41] Biao Zhang, Paul Suganthan, Gaël Liu, Ilya Philippov, Sahil Dua, Ben Hora, Kat Black, Gus Martins, Omar Sanseviero, Shreya Pathak, et al. T5gemma 2: Seeing, reading, and understanding longer. *arXiv preprint arXiv:2512.14856*, 2025.

[42] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Supplementary Material

Retrieval from Within: An Intrinsic Capability of Attention-Based Models

A Reverse-QWK

In Transformer encoder-decoder models like T5Gemma2, the standard attention logits for decoder layer ℓ are evaluated against keys computed from static encoder representations $K(\mathcal{S})$:

$$\frac{q_\ell k_\ell^\top}{\sqrt{d}} \quad \text{where} \quad k_\ell = (\bar{k} \odot \gamma_{K,\ell}) W_{K,\ell}, \quad \bar{k} = \mathrm{RMSNorm}(K(\mathcal{S})) \tag{8}$$

Because the projection $W_{K,\ell}$ and learned scale $\gamma_{K,\ell}$ are layer-specific, evaluating these logits requires computing a separate K for every layer. This makes it difficult to leverage a single approximate nearest neighbor (ANN) index built on the shared encoder representation.

By moving the key projection and learned scale to the query side, Reverse-QWK defines a transformed query:

$$\widetilde{q}_\ell = (q_\ell W_{K,\ell}^\top) \odot \gamma_{K,\ell} \tag{9}$$

Substituting this into the standard logit computation, and letting $\mathbf{\Gamma}_{K,\ell} = \mathrm{diag}(\gamma_{K,\ell})$, we see the logits remain identical:

$$\begin{aligned} q_\ell K^\top &= q_\ell \left(\bar{k} \mathbf{\Gamma}_{K,\ell} W_{K,\ell}\right)^\top \\ &= q_\ell W_{K,\ell}^\top \mathbf{\Gamma}_{K,\ell} \bar{k}^\top \\ &= \underbrace{\left((q_\ell W_{K,\ell}^\top) \odot \gamma_{K,\ell}\right)}_{\widetilde{q}_\ell} \bar{k}^\top. \end{aligned} \tag{10}$$

This allows attention scores to be written as direct dot products $(\widetilde{q}_\ell \bar{k}^\top)/\sqrt{d}$ against one normalized encoder pool $\bar{k}$, enabling a single ANN index to be shared across layers and heads.

A.1 Dimensionalities and Group-Query Attention

The simplified equations above suppress head structure for clarity. In T5Gemma2 the decoder uses Group-Query Attention (GQA): each layer has n_h query heads but only $n_{\mathrm{kv}} \le n_h$ key/value heads, with replication factor $n_{\mathrm{rep}} = n_h / n_{\mathrm{kv}}$, so each KV head is shared by n_{rep} Q-heads. With encoder hidden size d and per-head dimension d_h, the relevant tensors are

- $K(\mathcal{S}) \in \mathbb{R}^{N \times d}$ and $\bar{k} = \mathrm{RMSNorm}(K(\mathcal{S})) \in \mathbb{R}^{N \times d}$: a single, head-agnostic, shared encoder pool;
- $W_{K,\ell} \in \mathbb{R}^{d \times n_{\mathrm{kv}} d_h}$, viewed as n_{kv} per-KV-head blocks $W_{K,\ell}^{(g)} \in \mathbb{R}^{d \times d_h}$;
- $\gamma_{K,\ell} \in \mathbb{R}^{d_h}$, the per-head-dim RMSNorm scale shared across KV heads;
- $q_\ell \in \mathbb{R}^{L_q \times n_h \times d_h}$, with per-Q-head slices $q_\ell^{(h)} \in \mathbb{R}^{d_h}$.

At the per-head level the Reverse-QWK transformation in Eq. 9 becomes

$$\widetilde{q}_\ell^{(h)} \;=\; \left(q_\ell^{(h)} \odot \gamma_{K,\ell}\right) \left(W_{K,\ell}^{(g(h))}\right)^\top \;\in\; \mathbb{R}^d, \qquad g(h) = \lfloor h / n_{\mathrm{rep}} \rfloor,$$

so each Q-head h uses the W_K block of the KV-group $g(h)$ it belongs to, and lifts $q_\ell^{(h)}$ from $\mathbb{R}^{d_h}$ into the shared encoder space $\mathbb{R}^d$. The dot product $\widetilde{q}_\ell^{(h)} \bar{k}^\top$ inherits the standard per-head attention scale of Eq. 8 ($1/\sqrt{d_h}$ in the T5Gemma2 implementation).

Because $\bar{k}$ has no head dimension, GQA combines naturally with Reverse-QWK: in standard cross-attention GQA still requires expanding K to n_h heads on the encoder side, costing $\mathcal{O}(N\, n_h\, d_h)$ memory; under Reverse-QWK only the head-agnostic pool $\bar{k}$ is materialized, costing $\mathcal{O}(Nd)$ regardless of n_h, and the GQA replication of $W_K^{(g)}$ across the n_{rep} Q-heads of each group happens entirely on the (small) query side.

A.2 Implementation Details

The following PyTorch-style pseudocode illustrates the Reverse-QWK cross-attention computation, including the GQA structure described in Sec. A.1. Compared to the standard T5Gemma2 cross-attention, the only structural change is that W_k and γ_k are moved to the query side, and K is replaced by the shared, head-agnostic, d-dimensional pool $\bar{k}$ – V and the rest of attention are unchanged.

```
def reverse_qwk_attention(x, kv_x, W_q, W_k, W_v, gamma_k):
    # x:        [batch, q_len,  dim]      decoder hidden states
    # kv_x:     [batch, kv_len, dim]      already-RMSNormed encoder pool: kv_bar
    # W_q:      [n_h  * d_h, dim]         nn.Linear weight, n_h  Q-heads
    # W_k, W_v: [n_kv * d_h, dim]         nn.Linear weight, n_kv KV-heads (GQA, n_kv <= n_h)
    # gamma_k:  [d_h]                     per-head-dim K-norm scale (shared across KV-heads)
    # n_rep = n_h // n_kv                 Q-heads per KV-group

    # Standard per-head Q projection (q_norm and RoPE on q omitted for brevity)
    Q = (x @ W_q.T).view(batch, q_len, n_h, d_h)             # [B, q_len, n_h, d_h]

    # Reverse key projection: gamma_k first (per d_h), then W_k.T (per Q-head, GQA-replicated)
    W_k_per_kv = W_k.view(n_kv, d_h, dim)                    # [n_kv, d_h, dim]
    W_k_per_q  = W_k_per_kv.repeat_interleave(n_rep, dim=0)  # [n_h,  d_h, dim] (GQA replication)
    Q_tilde = einsum('bqhd,hdi->bhqi', Q * gamma_k, W_k_per_q)  # [B, n_h, q_len, dim]

    # kv_x is already the shared pool kv_bar = RMSNorm_d(K(S)); one key per token, head-agnostic
    kv_bar = kv_x.unsqueeze(1)                               # [B, 1, kv_len, dim] (broadcast over heads)

    # Scores: dot product over the full embedding dim d, per-head scaled by 1/sqrt(d_h)
    scores = (Q_tilde @ kv_bar.transpose(-2, -1)) / sqrt(d_h) # [B, n_h, q_len, kv_len]

    # V keeps the standard GQA path: project to n_kv heads, then expand to n_h via repeat_kv
    V = (kv_x @ W_v.T).view(batch, kv_len, n_kv, d_h)
    V = repeat_kv(V, n_rep).transpose(1, 2)                  # [B, n_h, kv_len, d_h]

    output = softmax(scores, dim=-1) @ V                     # [B, n_h, q_len, d_h]
    return output                                            # then reshape and apply W_o
```

A.3 Practical Considerations

Sharing $\bar{k}$ across layers and heads: Standard cross-attention requires materializing layer- and head-specific keys $k_\ell^{(g)} = (\bar{k} \odot \gamma_{K,\ell}) W_{K,\ell}^{(g)}$ per encoder token, for every decoder layer ℓ and KV-group g, and (under GQA) further replicating them across the n_h Q-heads at attention time. Reverse-QWK instead stores the single, head-agnostic pool $\bar{k} \in \mathbb{R}^{N \times d}$ (Sec. A.1), where N is the number of tokens in the corpus and d is the embedding dimension. It then pushes the layer/head-specific factors $\gamma_{K,\ell}$ and $W_{K,\ell}^{(g(h))}$ onto the (small) query side via Eq. 9. A single ANN index over $\bar{k}$ therefore serves all decoder layers and Q-heads.

Position embeddings: As in standard encoder-decoder cross-attention, RoPE is applied to the decoder queries q_ℓ but not to the encoder representations – in T5Gemma2 this is enforced by skipping the encoder prefix when applying RoPE to the merged KV stream. Reverse-QWK preserves this: RoPE is applied to q_ℓ *before* the transformation in Eq. 9, while $\bar{k}$ remains positionally invariant and can be precomputed once per corpus.

Storage footprint and quantization: Because $\bar{k}$ has no head- or layer-specific axis, its size scales as $N \times d$, independent of n_h and the number of decoder layers. For a 1B-token corpus at $d = 2560$ (the hidden size of our T5Gemma2 4B-4B model) and 8-bit precision, this is $10^9 \times 2560 \approx 2.56$ TB, which fits on a single NVMe SSD; product quantization or further compression could shrink this substantially. Values V are not needed for retrieval and only enter cross-attention at generation time, where they are computed on-demand as $V = W_V \cdot \text{encoder\_kv}$ for the (small) top-k set of selected encoder positions, so they need not be precomputed or stored offline.

Compressed cross-attention KV cache for prefill: Beyond enabling a shared ANN index, $\bar{k}$ can be viewed as a heavily compressed cross-attention KV cache. A standard cross-attention KV cache for L decoder layers stores layer-specific projected K and V for every encoder token, totalling $2L\, n_{\text{kv}} d_h$ scalars per token; Reverse-QWK stores a single d-dimensional vector per token, shared

Table 4: Complete-evidence recall: the percentage of examples for which all supporting facts are retrieved. INTRA performs best on multi-hop benchmarks (HotPotQA, 2Wiki, MuSiQue) that require evidence assembly. NQ's single-hop nature minimizes this benefit.

	HotPotQA			2Wiki			MuSiQue			NQ		
Retrieval method	R@5	R@10	R@20	R@5	R@10	R@20	R@5	R@10	R@20	R@5	R@10	R@20
TF-IDF	18.2	25.9	35.5	14.0	19.3	24.7	1.5	2.4	4.8	7.5	11.3	16.2
BM25	32.2	41.0	48.9	17.4	23.2	28.4	3.4	5.5	7.3	14.0	21.9	32.7
MaxSim	36.1	46.3	54.2	16.7	22.6	27.5	5.3	8.0	11.9	20.9	29.5	39.0
Hybrid RAG	48.0	61.8	71.2	29.1	36.0	40.9	6.2	11.1	17.1	22.9	35.5	49.8
BGE	54.8	63.5	69.6	30.9	35.9	40.1	6.5	11.2	15.7	29.6	39.0	47.5
Qwen3-Emb-0.6B	35.9	44.2	50.4	28.0	33.3	36.8	5.7	10.7	15.5	25.6	34.4	42.0
Qwen3-Emb-4B	44.6	53.5	61.1	32.8	37.2	40.8	8.8	14.7	19.7	30.3	40.0	50.5
Qwen3-Emb-4B + Jina reranker	48.8	59.6	65.4	35.4	40.3	43.5	10.1	16.6	20.6	**31.9**	**42.0**	**50.9**
INTRA	**59.9**	**70.9**	**76.1**	**40.7**	**50.3**	**55.2**	**12.8**	**18.9**	**23.7**	29.1	38.3	45.9

across all layers and Q-heads. The resulting compression ratio is $2L\, n_{\mathrm{kv}} d_h / d$, which is exactly $2L$ in the multi-head limit $n_{\mathrm{kv}} d_h = d$ (its theoretical maximum) and somewhat smaller under GQA; for our T5Gemma2 4B-4B model ($L = 34$, $n_{\mathrm{kv}} d_h \approx d/2.5$) it is on the order of $\sim 30\times$. Because $\bar{k}$ already incorporates the encoder forward pass and the only remaining layer-specific work is one query-side multiplication by $W_{K,\ell}$ and $\gamma_{K,\ell}$ (Eq. 9), cross-attention prefill against a cached corpus never re-encodes evidence or recomputes layerwise K – the prefill cost reduces to a per-layer transformed-query/$\bar{k}$ dot product plus on-demand V for the top-k selected positions.

B Retrieval setup

B.1 Retrieval Training Details

We initialize from a T5Gemma2 4B-4B checkpoint whose decoder is first fine-tuned on the CLaRa QA pretraining dataset [11]. This warm-start serves three purposes: aligning the decoder with QA-style generation, adapting generation to chunk-based pre-encoded representations rather than re-encoded text, and incorporating the Reverse-QWK reparameterization of cross-attention. We then jointly adapt on the HotPotQA, 2Wiki, MuSiQue, and NQ training splits using pre-encoded evidence from the shared candidate pool.

During retrieval training, we optimize only the retrieval-specific parameters: 64 trainable retrieval token embeddings $\{\rho_i\}$ corresponding to approximately $164K$ parameters, and the learned layer-head aggregation weights $\{\alpha_{l,g}\}$ used to score chunk vectors corresponding to 272 parameters. The retrieval score is computed from the query states at the R retrieval-token positions only; the query states at the input-token positions are not used for scoring. Initial context can be retrieved with efficient approximate late-interaction search methods such as [30]. Retrieval training runs for 10K optimization steps with AdamW, learning rate 3×10^{-3}, 100 warmup steps, and global batch size 256.

B.2 Baseline Details

We compare INTRA against nine retrieval baselines. As sparse lexical baselines we use TF-IDF [29] and BM25 [28]. As dense single-vector baselines we use BGE-large [37], Qwen3-Embedding-0.6B, and Qwen3-Embedding-4B [42]. For BGE and both Qwen models, we encode the query and each candidate chunk independently and rank all chunks in the shared pool by cosine similarity. We further evaluate Qwen3-Embedding-4B with the Jina reranker [14], and a hybrid RAG baseline that combines BM25, TF-IDF, BGE-large, and Qwen3-Embedding-0.6B rankings using reciprocal rank fusion (RRF) [6]. We also evaluate a ColBERT-style MaxSim late-interaction baseline [18] that scores query embeddings against chunk-vector encodings in the T5Gemma encoder output space.

C Additional Results

Table 4 reports the complete-evidence recall values shown in Fig 2. Tables 5 and 6 present end-to-end question-answering EM and F1 scores, respectively, across various retrieval methods using a fixed T5Gemma generator. These tables include the 95% confidence intervals.

Table 5: End-to-end question-answering EM across retrieval methods with a fixed T5Gemma generator. Errors are 95% CIs.

Retrieval method	HotPotQA	2Wiki	MuSiQue	NQ
TF-IDF	34.2 ± 1.0	39.0 ± 0.8	5.3 ± 0.9	34.9 ± 1.2
BM25	40.5 ± 1.0	41.7 ± 0.9	7.7 ± 1.0	43.4 ± 1.1
MaxSim	40.7 ± 1.1	41.6 ± 0.9	10.1 ± 1.2	48.4 ± 1.2
Hybrid RAG	43.4 ± 1.1	46.0 ± 0.9	10.6 ± 1.2	50.5 ± 1.2
BGE	41.9 ± 1.1	46.1 ± 0.9	10.8 ± 1.2	52.2 ± 1.2
Qwen3-Emb-0.6B	37.0 ± 1.1	45.7 ± 0.9	11.1 ± 1.2	36.4 ± 1.3
Qwen3-Emb-4B	40.3 ± 1.2	46.0 ± 0.9	12.7 ± 1.3	54.5 ± 1.3
Qwen3-Emb-4B + Reranker	41.6 ± 1.2	46.8 ± 1.0	13.3 ± 1.3	$\mathbf{55.1 \pm 1.2}$
INTRA	$\mathbf{46.4 \pm 1.1}$	$\mathbf{49.2 \pm 0.9}$	$\mathbf{14.0 \pm 1.3}$	51.2 ± 1.2

Table 6: End-to-end question-answering F1 across retrieval methods with a fixed T5Gemma generator. Errors are 95% CIs.

Retrieval method	HotPotQA	2Wiki	MuSiQue	NQ
TF-IDF	44.5 ± 1.0	42.7 ± 0.7	14.6 ± 1.1	42.9 ± 1.0
BM25	52.0 ± 0.9	45.2 ± 0.7	16.8 ± 1.2	51.8 ± 1.0
MaxSim	52.2 ± 1.0	45.8 ± 0.8	19.8 ± 1.3	57.8 ± 1.0
Hybrid RAG	54.3 ± 1.0	49.9 ± 0.8	20.1 ± 1.3	59.1 ± 1.1
BGE	53.0 ± 1.0	49.6 ± 0.8	19.7 ± 1.3	61.2 ± 1.1
Qwen3-Emb-0.6B	47.7 ± 1.0	49.8 ± 0.9	20.0 ± 1.4	44.6 ± 1.1
Qwen3-Emb-4B	51.2 ± 1.1	50.1 ± 0.8	21.8 ± 1.4	63.7 ± 1.1
Qwen3-Emb-4B + Reranker	53.6 ± 1.1	50.8 ± 0.8	22.5 ± 1.4	$\mathbf{64.2 \pm 1.2}$
INTRA	$\mathbf{58.0 \pm 1.0}$	$\mathbf{53.2 \pm 0.8}$	$\mathbf{23.0 \pm 1.4}$	60.3 ± 1.1

C.1 Ablation Study

Table 7 isolates the main design choices in INTRA's retrieval path. We vary the initial context $\mathcal{S}_0$ (removing it, using it directly as the final retrieved set, changing its similarity metric, or increasing its size), the pooled chunk length L_p, the number of retrieval tokens, and the method for choosing chunks for generation. The results show that both the initial context and learned retrieval-token scoring are important: removing $\mathcal{S}_0$, replacing it with cosine-only retrieval, or using a single retrieval token substantially reduces complete-evidence recall and EM.

D Supplementary Efficiency Analysis

Let N denote the total number of tokens in the evidence pool, let L_c denote the number of tokens per chunk, let $M \approx N/L_c$ denote the number of chunks, let L_q denote the query length, let $k = |S(q)|$ denote the number of selected chunks, and let L_g denote the output length. As in the main text, we express the query-time retrieval cost as $\mathcal{O}(\sqrt{M} L_q L_c)$, which is the practical scaling of the MaxSim operator [18] when using inverted file (IVF) approximate nearest-neighbor (ANN) indices, such as FAISS or cuVS.

We assume transformer-based encoders and decoders [35], so encoding a sequence of length L costs $\mathcal{O}(L^2)$ under dense attention. The goal of this appendix is not to provide a hardware-faithful latency model, but to make explicit where each family of methods spends computation. We omit layer, head, and hidden-dimension factors.

D.1 Full Context Prompting

Long-context prompting performs no reusable precomputation. The query and all available evidence are concatenated into one input sequence, so the full cost is paid at inference time:

$$\text{Pre-query} = \mathcal{O}(1), \qquad \text{Prefill} = \mathcal{O}((N + L_q)^2), \qquad \text{Generation} = \mathcal{O}(L_g(N + L_q + L_g)).$$

This setting is attractive when the full context must be processed jointly, but it becomes expensive when relevant evidence is sparse relative to the corpus.

Table 7: Ablations of INTRA retrieval and generation components on HotPotQA and 2Wiki with $k = 5$ retrieved chunks. We report complete-evidence recall at 5 (CE-recall@5) and exact match (EM).

	HotPotQA		2Wiki	
Design choice	CE-recall@5	EM	CE-recall@5	EM
INTRA	59.9	46.4	40.7	49.2
$\mathcal{S}_0 = \emptyset$ (no initial retrieval, Sec. 2.2)	26.9 (−33.0)	33.2 (−13.2)	15.4 (−25.3)	38.3 (−10.9)
$\mathcal{S}_{\text{INTRA}} = \mathcal{S}_0$ (only init. retrieval, Sec. 2.3)	37.1 (−22.8)	40.7 (−5.7)	17.7 (−23.0)	41.6 (−7.6)
$\mathcal{S}_0$ based on cosine similarity (Sec. 2.3)	30.1 (−29.8)	35.6 (−10.8)	15.0 (−25.7)	39.2 (−10.0)
Pooled chunk length $L_p = 1$ (Sec. 3.3)	48.9 (−11.0)	43.9 (−2.5)	26.6 (−14.1)	45.0 (−4.2)
16 retrieval tokens (Sec. 2.2)	55.5 (−4.4)	45.2 (−1.2)	35.8 (−4.9)	48.2 (−1.0)
1 retrieval token (Sec. 2.2)	44.9 (−15.0)	42.8 (−3.6)	24.7 (−16.0)	44.2 (−5.0)
Top-5 only from $\mathcal{S}_{\text{INTRA}}$ (Sec. 4)	51.7 (−8.2)	43.5 (−2.9)	35.0 (−5.7)	46.7 (−2.5)

D.2 Standard RAG

Standard retrieval-augmented generation preprocesses the corpus into chunks and encodes them before query time. Under independent fixed-size chunking, this one-time pre-query work is

$$\mathcal{O}\left(\frac{N}{L_c} \cdot L_c^2\right) = \mathcal{O}(NL_c).$$

At inference time, RAG first pays retrieval cost $\mathcal{O}(\sqrt{M}L_qL_c)$ and then re-encodes the retrieved text together with the query:

$$\text{Prefill} = \mathcal{O}((L_q + kL_c)^2), \qquad \text{Generation} = \mathcal{O}(L_g(L_q + kL_c + L_g)).$$

Relative to long-context prompting, this reduces the amount of evidence re-encoded per query, but still requires the generator to process the retrieved raw text from scratch. The main difference from INTRA is the quadratic encoder-side re-encoding term.

D.3 INTRA

INTRA shares the same offline chunk-encoding term as standard RAG,

$$\text{Pre-query} = \mathcal{O}(NL_c),$$

but changes what happens after retrieval. The model retrieves reusable chunk representations rather than raw passages, so query-time work becomes

$$\text{Retrieval} = \mathcal{O}(\sqrt{M}L_qL_c), \qquad \text{Prefill} = \mathcal{O}(L_q(L_q{+}kL_c)), \qquad \text{Generation} = \mathcal{O}(L_g(L_q{+}kL_c{+}L_g)).$$

The key distinction is that evidence is encoded once offline and then reused at query time, which shifts work away from repeated evidence prefilling and toward reusable memory construction.

D.4 Additional Timing Results and Benchmark Details

The efficiency benchmark (Figure 4 in the main text) fixes $L_q = L_c = L_g = 128$ and $N = 65{,}536$, excludes retrieval time, and gives RAG and INTRA the same top-k chunks. It therefore measures only the generator-side cost after evidence has been selected. The long-context baseline provides a reference point, incurring $\mathcal{O}((L_q + N)^2)$ prefill by processing the query with the full evidence pool. Standard RAG pre-fills over the query and k retrieved chunks, $\mathcal{O}((L_q + kL_c)^2)$, whereas INTRA reuses stored chunk states and pays only $\mathcal{O}(L_q^2)$ before the first token.

The measured curves follow the asymptotic comparison. As k increases from 1 to 500, INTRA TTFT grows from 12.8 ms to 65.7 ms, while standard RAG grows from 23.1 ms to 1.25 s; long-context prompting is about 1.31 s in the same setting. Figure 6 shows the same trend as chunk length increases, whereas Figure 5 shows a smaller gap in generation throughput. All methods use the same Reverse-QWK decoder, which lets normalized encoder states serve as reusable cross-attention memory without precomputing separate layer-specific projected key/value states.

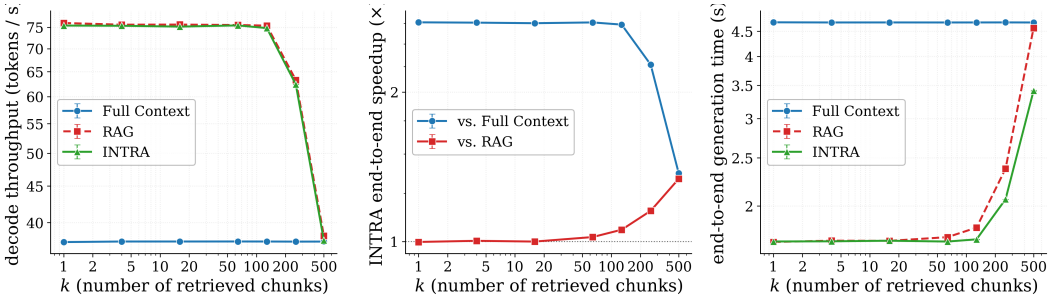

Figure 5: Generation throughput benchmark. Sweep over k values (FC vs RAG vs INTRA)

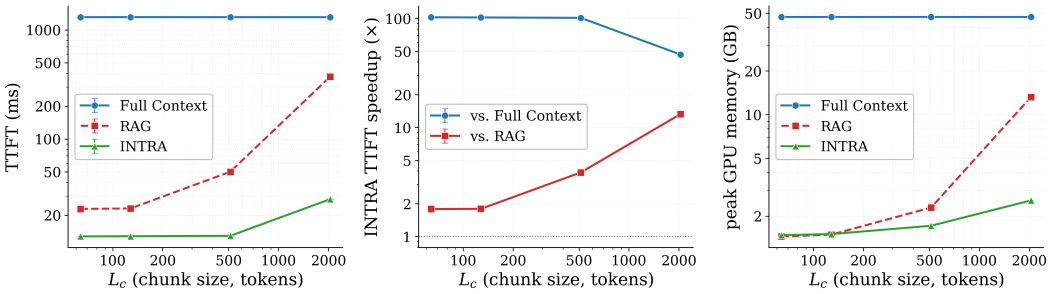

Figure 6: Time-to-first-token benchmark. Sweep over chunk length (FC vs RAG vs INTRA)

D.5 Compute Resources

Training was run on NVIDIA H100 GPUs with 80GB of memory per GPU. The largest training configuration used up to 160 GPUs for the QA pretraining stage and for each retrieval/generation model version. The longest training run was the QA pretraining stage, which took approximately 12 hours, corresponding to about 2,000 GPU-hours. The benchmark-specific retrieval and generation training runs were much shorter, each accounting for less than 200 GPU-hours. Our training code uses a modified implementation based on TorchTitan [26].

E Dataset and Pool Details

We use a deduplicated end-to-end QA dataset derived from the union of the per-benchmark datasets constructed in CLaRa [11], with one training split and one evaluation split for each of HotPotQA, 2WikiMultihopQA, MuSiQue, and Natural Questions.

Following the setup of CLaRa, we build one shared retrieval candidate pool for all four benchmarks under a fixed budget of approximately 100M tokens. We start from the benchmark-specific chunk sets provided for HotPotQA, 2Wiki, MuSiQue, and NQ, merge them into one pool, and de-duplicate chunks globally. We ensure coverage of all oracle chunks referenced by the QA examples and add uniformly sampled non-oracle chunks without replacement until the budget is reached.

Table 8: Split statistics for the deduplicated end-to-end QA dataset used in our experiments.

Split	Examples	Doc refs
2Wiki train	167,454	404,170
HotPotQA train	90,185	180,370
MuSiQue train	277,577	631,153
NQ train	53,301	276,872
2Wiki eval	12,576	30,654
HotPotQA eval	7,384	14,768
MuSiQue eval	2,417	6,404
NQ eval	6,489	42,182

Across all splits, the saved dataset contains 617K QA examples and 1.6M document references. These references cover 514,999 unique oracle chunks. The global context pool contains 758,500 chunks in total: all 514,999 oracle chunks plus 243,501 non-oracle chunks sampled to reach the 100M-token pool budget. The pool contains 69,734,081 whitespace-delimited words (91.94 words per chunk on average, 106 median). Note that while 243K chunks are non-oracle for every example in the dataset, from the perspective of any single example all other $\sim$758K chunks in the pool, including those that serve as oracles for other examples, act as distractors.