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]Technion
{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 [21]. 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 [42, 27].

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

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

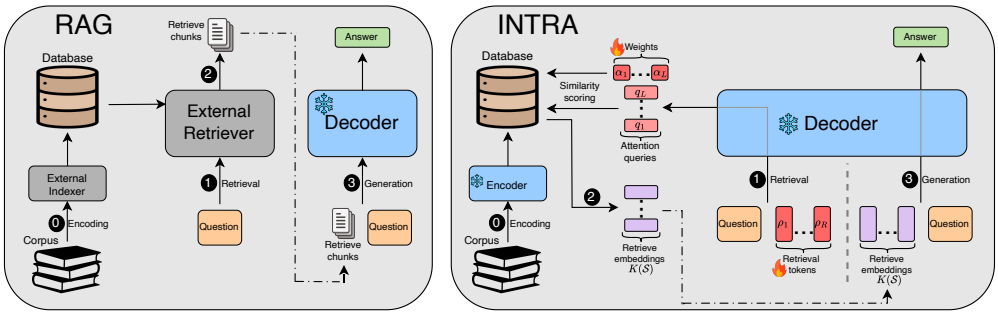

Figure 1: **Left**: Standard RAG pipeline. An external retriever selects documents which are then re-encoded by the decoder to produce an answer. Retrieval and generation operate in separate representation spaces. **Right**: Our method, INTRA, uses a pretrained frozen encoder-decoder for both retrieval 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.

capability in this setting. We call this regime *INTRA* (*INTrinsic Retrieval via Attention*): rather 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, and decoder-side retrieval queries refine evidence without introducing a separate retriever or compression model (see Figure 1).
- We show empirically that this unified retrieval-generation design is especially strong on multi-hop question-answering (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 and Φ_ℓ denotes 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 use $\widetilde{\mathrm{Dec}}$ to denote 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 [19]*. 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

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

where $\mathcal{S}_0$ is the initial chunk selection providing the decoder with useful starting context. $\mathcal{S}_0$ is constructed with training-free BM25 ranking [31], see Appendix B.3 for other choices.

We then select the chunks with the largest scores:

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

and the ordinary decoder generates conditioned on $\mathcal{S}_{\mathrm{INTRA}}$:

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

Inference thus consists of two stages: 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 an autoregressive generation utilizing the selected context $K(\mathcal{S}_{\mathrm{INTRA}})$. Because scoring is over the full corpus, $\mathcal{S}_{\mathrm{INTRA}}$ can include chunks outside $\mathcal{S}_0$, unlike reranking methods which only reorder a fixed candidate set.

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 [44] 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:

$$z_\ell = \mathrm{Attention}_{\mathrm{RQWK}}(\widetilde{q}_\ell, \bar{K}(\mathcal{S}), \bar{K}(\mathcal{S})) = \mathrm{softmax}\left(\frac{\widetilde{q}_\ell \bar{K}(\mathcal{S})^\top}{\sqrt{d}}\right)\bar{K}(\mathcal{S}) \tag{6}$$

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 Grouped-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 chunks for input x (e.g., a question). When retrieval supervision is available, we train the retrieval scores s_i from Eq. 3 with a cross-entropy objective that places uniform target mass on $\mathcal{O}(x)$, with scores normalized over a candidate subset $\mathcal{C}(x)$:

$$\mathcal{L}_{\text{retrieval}} = -\frac{1}{|\mathcal{O}(x)|} \sum_{j \in \mathcal{O}(x)} \log \frac{\exp(s_j/\tau)}{\sum_{n \in \mathcal{C}(x)} \exp(s_n/\tau)},$$

where τ is a temperature parameter. The candidate subset is $\mathcal{C}(x) = \mathcal{O}(x) \cup \mathcal{N}(x)$, where $\mathcal{N}(x)$ holds hard negatives for x and additional in-batch chunks (see Appendix B.4 for details). When scoring the full corpus is feasible, we simply take $\mathcal{C}(x) = \{1, \ldots, M\}$; subsampling is needed only when M makes this prohibitive. 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 [12]).

We treat L_p as a quality-versus-cost tradeoff. The number of scored vectors, and with it both the scoring cost of MaxSim and the size of the retrieval index, grows linearly in L_p, while retrieval quality improves monotonically with L_p up to the no-pooling limit $L_p \approx L_c$, at which $\widehat{k}_i = k_i$ and the score in Eq. 3 is computed exactly. See Appendix C.3 for an ablation of L_p.

4 Benchmarks and Experimental Setup

We evaluate INTRA on four Wikipedia-based QA benchmarks: HotPotQA [41], 2WikiMultihopQA [13], MuSiQue [37], and Natural Questions [20]. 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 $M = 759K$ 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 [32], BM25 [31]), dense single-vector models (BGE-large [40], Qwen3-Embedding-0.6B/4B [46]), reranking (Jina reranker [15]), hybrid RAG (RRF [6]), and a ColBERT-style MaxSim late-interaction baseline [19] (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.

Decoder Warm-Start. The available T5Gemma2 checkpoint is not instruction-tuned, so it can't answer questions out of the box. We first warm-start its decoder on the CLaRa pretraining data [12], with the encoder frozen. We use the Reverse-QWK parameterization of Eq. 6, in which the decoder attends to a shared pool of representations $\{\bar{k}_i\}_{i=1}^{M}$. Training adapts the decoder to this pool, which is then stored and reused at inference. The objective is next-token cross-entropy on answer tokens.

Retrieval training. During retrieval training, the encoder-decoder backbone is frozen, and only the retrieval tokens ρ_i and layer aggregation weights α_l are optimized, for a total of approximately 164K parameters. Initial context $\mathcal{S}_0$ uses $n_0 = 10$ and pooled chunks of length $L_p = 7$.

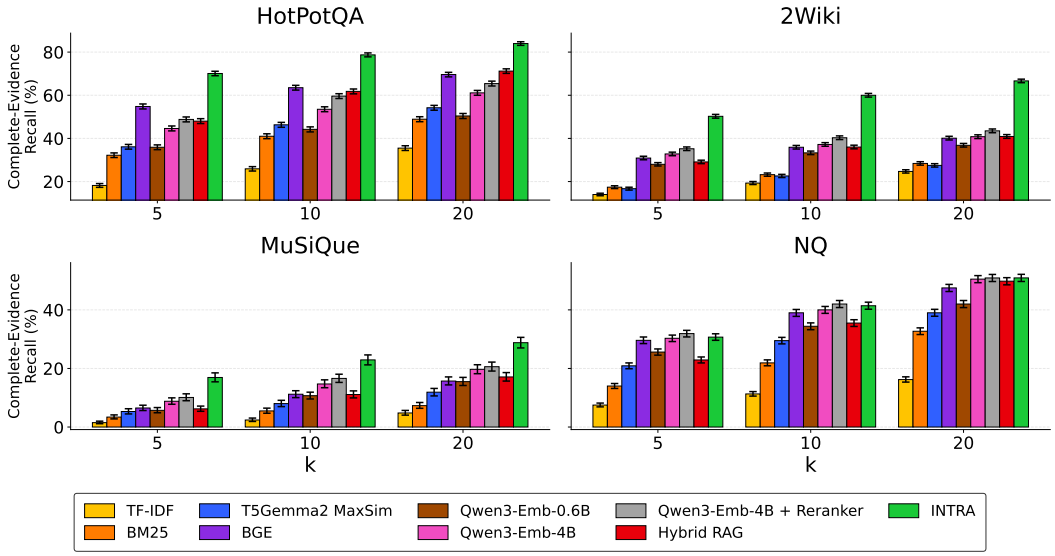

Figure 2: Complete-evidence recall: the percentage of examples for which *all* supporting facts are retrieved. INTRA performs best on multi-hop benchmarks (HotPotQA, 2Wiki, MuSiQue) that require evidence assembly. NQ's single-hop nature minimizes this benefit.

Evaluation. At evaluation, QA builds a five-chunk context: the top four retrieved chunks from $\mathcal{S}_{\text{INTRA}}$ 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.2.

5 Results

We organize the results around the four 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)? Finally, do the retrieval gains persist when the corpus grows to a full Wikipedia scale (Section 5.4)?

5.1 Retrieval Results

Figures 2 and 6 report the complete-evidence recall and average recall, respectively, 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 benchmarks that require assembling multiple pieces of evidence (HotPotQA, 2Wiki, MuSiQue). INTRA ranks chunks by the decoder's own attention queries, which reflect what the decoder needs to generate the answer. A leave-one-out analysis on HotPotQA (Appendix C.4) supports this: the higher INTRA ranks a chunk, the more removing it from the context degrades the generator's answer. This matters most for multi-hop questions, where a *bridge* passage links the question to an *answer-bearing* passage that often shares little wording with the question. A question-similarity retriever tends to find the bridge, whereas INTRA prioritizes the answer-bearing passage and recovers both more often (64.1% vs. 54.3% at $k = 10$, Appendix C.4). On NQ, retrieval often reduces to finding one directly supporting passage, leaving less room for this advantage. See Appendix C for further results, analyses, and ablations.

In Figure 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

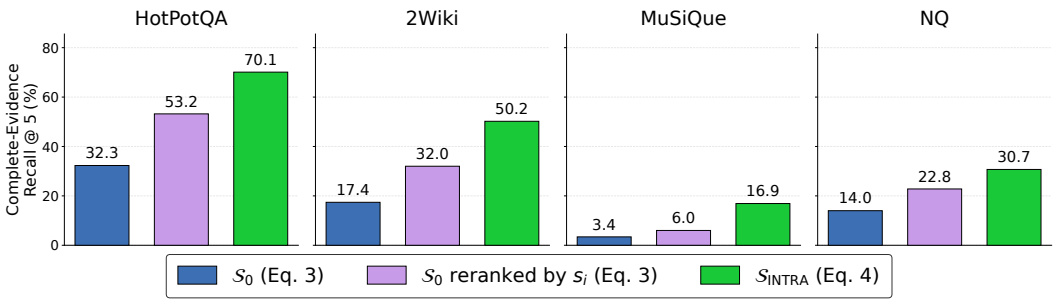

Figure 3: Complete-evidence recall@5 for the initial set $\mathcal{S}_0$, $\mathcal{S}_0$ reranked, and the final set $\mathcal{S}_{\text{INTRA}}$. INTRA performs corpus-wide scoring and can recover evidence outside the initial candidate pool.

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	**48.8**	**60.6**	**52.3**	**56.4**	**13.9**	**23.3**	53.7	63.4	**42.2**	**50.9**

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 generator, 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 retrieval 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 [36, 46]).

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.

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	69.7%	52.4%	27.5%	73.6%	55.8%
Llama3.1-8B	70.1%	51.8%	29.5%	71.9%	55.8%
Phi-4-mini	68.3%	37.9%	23.9%	74.6%	51.2%
Gemma-4-E2B	68.3%	55.7%	27.7%	75.3%	56.8%
Qwen3.5-9B	71.7%	57.0%	30.0%	**76.3%**	58.8%
Qwen3.5-27B	71.8%	58.0%	29.6%	72.6%	58.0%
T5Gemma2-4B (same as retriever)	**73.1%**	**59.8%**	**33.1%**	74.7%	**60.2%**

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	$\mathcal{O}(ML_c^2)$	$\mathcal{O}(\sqrt{M}L_qL_c)$	$\mathcal{O}(L_q(L_q + kL_c))$	$\mathcal{O}(L_g(L_q + kL_c + L_g))$

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).

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 grows, INTRA's prefill time grows linearly because it cross-attends to stored encoder states, whereas standard RAG's grows quadratically because it re-encodes the chunks. Appendix D.4 provides the setup and further generation throughput measurements.

INTRA avoids re-encoding the retrieved chunks at a modest storage cost. Because $\bar{K}(\mathcal{S})$ has no layer or head axis, it stores one d-dimensional vector per token, about 2.5 TB for a 1-billion-token corpus at 8-bit precision (Appendix A.3).

5.4 Retrieval at Full-Wikipedia Scale

Our main experiments use a 100M-token pool, which keeps analyses and ablations tractable. A natural concern is whether INTRA remains discriminative as the number of distractors grows. To address this, we verify INTRA on the full Wikipedia corpus [17], which consists of around 21M chunks (a total of ∼3B tokens). Figure 5 shows the Complete-evidence recall. As before, INTRA outperforms all baselines on multi-hop tasks, showing that its advantage persists as the corpus grows.

[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, [19, 16, 28].

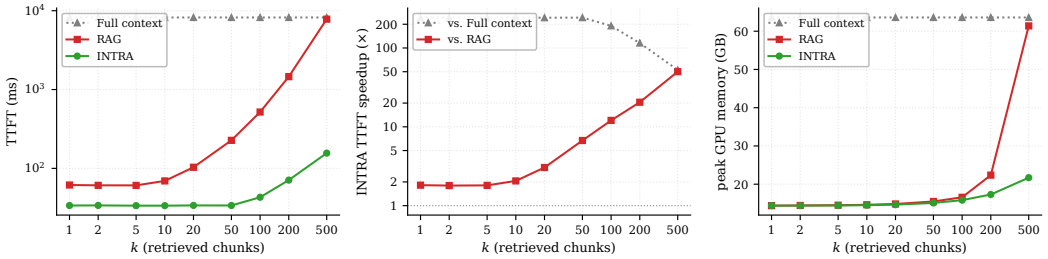

Figure 4: Time to first token vs. number of retrieved chunks k, excluding retrieval time. INTRA reuses pre-encoded evidence, while standard RAG re-encodes the retrieved text before decoding.

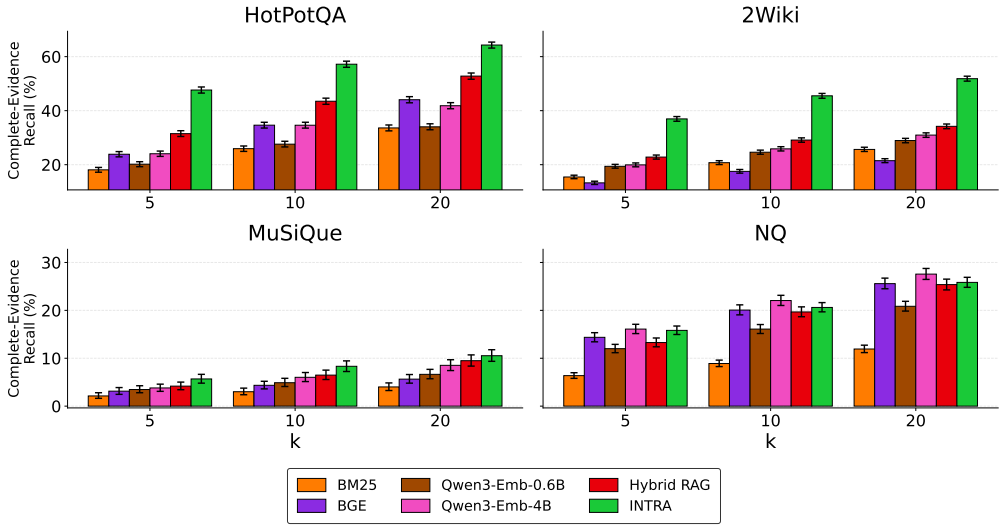

Figure 5: Complete-evidence recall on the full Wikipedia corpus (21M chunks, 3B tokens), 30× larger than the pool in Figure 2. INTRA's advantage on multi-hop benchmarks persists at this scale.

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 [11], and RA-DIT later instruction-tunes both retriever and generator [25]. DPR established dense passage retrieval as a strong open-domain QA primitive [17], and RAG couples a retriever with a sequence-to-sequence generator [21]. Atlas [14] jointly pre-trained a retriever with an encoder-decoder generator, to optimize few-shot learning from unstructured text. More recently, CLaRa [12] compresses documents into retrievable latent vectors, and jointly optimizes 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 [23, 1, 34]. INTRA targets the single-pass retrieval block that could be used within such pipelines, rather than the pipeline-level agentic loop itself. A related line of work retrieves iteratively while drawing on side information such as Wikipedia's hyperlink graph and relies on extractive span predictors for answer extraction [47, 26, 45, 22], whereas we target the case of single-pass retrieval over a plain passage corpus paired with a generative reader; absolute scores are therefore not directly comparable.

Late Interaction and Representation-Space Retrieval. Our retrieval formulation is close to late-interaction systems such as ColBERT [19], ColBERTv2 [33] and ColPali [9], 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 [35] framed QA as differentiable lookup, while RETRO and InstructRetro [4, 39] 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 an 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 token interactions [38]. However, dense attention's quadratic cost hinders long-context scaling, prompting sparse and linear alternatives [3, 43, 5, 18]. More recent models, such as Mamba, explore further reductions in context-processing cost [10, 7, 29, 24]. However, recent long-context benchmarks suggest that merely increasing the window size does not ensure robust evidence use when relevant information is sparse [42, 27]. 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 Appendix 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 Section 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.

Acknowledgments and Disclosure of Funding

The research of DS was funded by the European Union (ERC, A-B-C-Deep, 101039436). Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency (ERCEA). Neither the European Union nor the granting authority can be held responsible for them. The research of DS was also supported by the Council for Higher Education in Israel under the Moonshot Project.

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, 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.

[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] DeepSeek-AI. DeepSeek-V3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

[9] 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.

[10] 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.

[11] 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.

[12] Jie He, Richard He Bai, Sinead Williamson, Jeff Z. Pan, Navdeep Jaitly, and Yizhe Zhang. CLaRa: Bridging retrieval and generation with continuous latent reasoning, 2025. URL https://arxiv.org/abs/2511.18659.

[13] 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. URL https://aclanthology.org/2020.coling-main.580/.

[14] 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.

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

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

[17] 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. URL https://aclanthology.org/2020.emnlp-main.550/.

[18] 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.

[19] 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. URL https://doi.org/10.1145/3397271.3401075.

[20] 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. URL https://aclanthology.org/Q19-1026/.

[21] 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.

[22] Shaobo Li, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, Chengjie Sun, Zhenzhou Ji, and Bingquan Liu. HopRetriever: Retrieve hops over wikipedia to answer complex questions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 13279–13287, 2021.

[23] 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.

[24] 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.

[25] 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.

[26] Kaixin Ma, Hao Cheng, Yu Zhang, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. Chain-of-Skills: A configurable model for open-domain question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1599–1618, 2023.

[27] 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.

[28] NVIDIA Corporation. *cuVS: GPU-Accelerated Vector Search and Clustering*, 2026. URL https://github.com/rapidsai/cuvs.

[29] 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.

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

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

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

[33] 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. URL https://aclanthology.org/2022.naacl-main.272/.

[34] Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V. Vasilakos. Agentic retrieval-augmented generation: A survey on agentic RAG, 2025. URL https://arxiv.org/abs/2501.09136.

[35] 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.

[36] Nandan Thakur. BGE training dataset (only retrieval datasets). Hugging Face dataset card, 2023. URL https://huggingface.co/datasets/nthakur/bge-full-data. Accessed 2026-05-03.

[37] 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. URL https://aclanthology.org/2022.tacl-1.31/.

[38] 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.

[39] 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.

[40] 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.

[41] 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. URL https://aclanthology.org/D18-1259/.

[42] 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.

[43] 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, 2020. URL https://arxiv.org/abs/2007.14062.

[44] 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.

[45] Xinyu Zhang, Ke Zhan, Enrui Hu, Chengzhen Fu, Lan Luo, Hao Jiang, Yantao Jia, Fan Yu, Zhicheng Dou, Zhao Cao, and Lei Chen. Answer complex questions: Path ranker is all you need. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, pp. 449—-458, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. URL https://doi.org/10.1145/3404835.3462942.

[46] 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.

[47] Yunchang Zhu, Liang Pang, Yanyan Lan, Huawei Shen, and Xueqi Cheng. Adaptive information seeking for open-domain question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3615–3626, 2021.

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 $K(\mathcal{S})$:

$$\frac{q_\ell k_\ell^\top}{\sqrt{d}} \quad \text{where} \quad k_\ell = (\bar{k} \odot \gamma_{K,\ell}) W_{K,\ell}, \quad \bar{k} = \mathrm{RMSNorm}(K(\mathcal{S})) \tag{7}$$

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:

$$\widetilde{q}_\ell = (q_\ell W_{K,\ell}^\top) \odot \gamma_{K,\ell} \tag{8}$$

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

$$\begin{aligned} q_\ell K^\top &= q_\ell \left(\bar{k} \mathbf{\Gamma}_{K,\ell} W_{K,\ell}\right)^\top \\ &= q_\ell W_{K,\ell}^\top \mathbf{\Gamma}_{K,\ell} \bar{k}^\top \\ &= \underbrace{\left((q_\ell W_{K,\ell}^\top) \odot \gamma_{K,\ell}\right)}_{\widetilde{q}_\ell} \bar{k}^\top. \end{aligned} \tag{9}$$

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 $n_{\mathrm{kv}} \leq n_h$ 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. 8 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. 7 ($1/\sqrt{d_h}$ 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 Section 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 $\bar{k} \in \mathbb{R}^{N \times d}$ (Section 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. 8. 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. 8, 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

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. 8), 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 [12]; the configuration of this warm-start is given in Section 4. It serves three purposes: aligning the decoder with QA-style generation, which the released checkpoint has not been post-trained for; adapting generation to chunk-based pre-encoded representations rather than re-encoded text; and incorporating the Reverse-QWK reparameterization of cross-attention. The decoder sequence is formatted as `Question: {question} Answer: {answer}`, with the cross-entropy loss taken on the answer tokens only, and an encoder context of 20 context chunks containing both oracle and distractor chunks. We then jointly adapt on the HotPotQA, 2Wiki, MuSiQue, and NQ training splits using pre-encoded evidence from the shared candidate pool. We train for 15K steps at global batch size 320 with AdamW, learning rate 1×10^{-5} and 1,000 warmup steps.

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. Retrieval training runs for 10K optimization steps with AdamW, learning rate 3×10^{-3}, 100 warmup steps, and global batch size 256.

The retrieval input in Eq. 2 appends R learnable tokens to x, each producing a query state per decoder layer and head. 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. To reduce scoring cost, the R query states are average-pooled into $P \leq R$ groups, leaving P vectors per layer-head that are matched against chunk tokens. $P = R$ keeps one query per retrieval token; $P = 1$ pools all tokens into a single query. The layer-head aggregation with weights α can be applied on either side of the MaxSim. In the default *sum → MaxSim* order, the weighted query states are summed first and MaxSim is evaluated once per pooled query, as in Eq. 3. An alternative is *MaxSim → sum*, in which is evaluated per layer and head and the scores are combined afterwards, costing one evaluation per layer-head pair. Summing first is far cheaper and, as Table 7 shows, also more accurate.

B.2 Baseline Details

We compare INTRA against eight retrieval baselines. As sparse lexical baselines we use TF-IDF [32] and BM25 [31]. As dense single-vector baselines we use BGE-large [40], Qwen3-Embedding-0.6B, and Qwen3-Embedding-4B [46]. 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 [15], 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 [19] that scores query embeddings against chunk-vector encodings in the T5Gemma2 encoder output space.

B.3 Initial context selection for retrieval

In our experiments $\mathcal{S}_0$ is selected with BM25 [31]. A natural alternative is to select chunks whose encoded representations are closest to the encoded input x in the encoder's own representation space. With $k_x = \mathrm{Enc}(x)$, we define

$$s_i^{\mathrm{enc}} = \mathrm{MaxSim}(k_x, k_i), \qquad \mathcal{S}_0^{\mathrm{enc}} = \{i \in \{1, \ldots, M\} : s_i^{\mathrm{enc}} \text{ is among the top-}n_0 \text{ scores}\}\,. \quad (10)$$

This variant keeps the entire pipeline inside one model: initialization, retrieval, and generation all run on the same pretrained encoder-decoder, with no external component of any kind. We find it

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	**70.1**	**78.7**	**84.0**	**50.2**	**60.0**	**66.7**	**16.9**	**22.9**	**28.8**	30.7	41.4	**51.0**

Table 5: End-to-end question-answering EM across retrieval methods with a fixed T5Gemma2 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	$\mathbf{48.8 \pm 1.1}$	$\mathbf{52.3 \pm 0.9}$	$\mathbf{13.9 \pm 1.3}$	53.7 ± 1.2

performs on par with BM25 initialization, but at substantially higher cost to construct $\mathcal{S}_0$. A third option 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 (see the ablation study of Table 7.)

B.4 Construction of the candidate set $\mathcal{C}(x)$

For our $\sim$100M-token pool, full-corpus scoring is feasible and we take $\mathcal{C} = \{1, \ldots, M\}$, requiring no negative sampling.

For the full-Wikipedia experiments this is prohibitive, and we build $\mathcal{C}$ per step from the batch. For each query we mine the top-500 chunks by BM25 against its oracle chunks as hard negatives. The per-query pools are merged and deduplicated into a single candidate set shared across the local batch (size 2 in our experiments), giving ~ 1000 candidates per step. Only those rows are gathered from the memory-mapped representation store, so the corpus is never resident in memory.

C Additional Results

C.1 Additional retrieval and end-to-end results

Table 4 reports the complete-evidence recall values shown in Figure 2. Tables 5 and 6 present end-to-end question-answering EM and F1 scores, respectively, across various retrieval methods using a fixed T5Gemma2 generator. These tables include the 95% confidence intervals.

C.2 Ablation Study

Table 7 isolates the main design choices in INTRA's retrieval path. We vary the initial context $\mathcal{S}_0$, the number of retrieval tokens R, the number of groups P into which they are folded, and the order in which heads and layers are aggregated relative to the MaxSim. Removing $\mathcal{S}_0$ costs the most recall, and collapsing the retrieval head to a single token ($P = R = 1$) is nearly as damaging. Recovering the token budget at $P = 1$ closes roughly half the gap, indicating that both the number of retrieval

Table 6: End-to-end question-answering F1 across retrieval methods with a fixed T5Gemma2 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	$\mathbf{60.6 \pm 1.0}$	$\mathbf{56.4 \pm 0.8}$	$\mathbf{23.3 \pm 1.4}$	63.4 ± 1.1

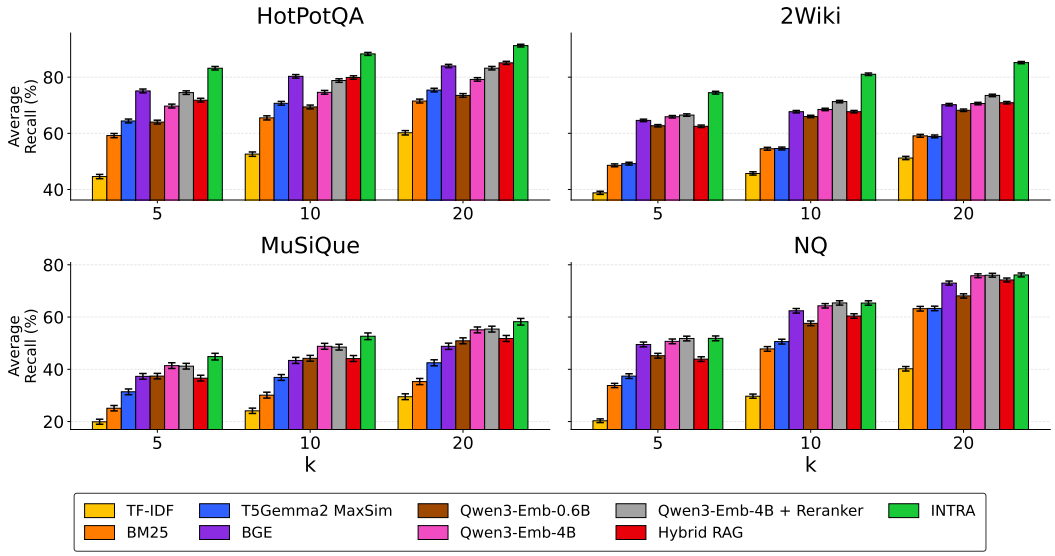

Figure 6: Average recall results on the 100M-token corpus. INTRA performs best on multi-hop benchmarks (HotPotQA, 2Wiki, MuSiQue) that require evidence assembly. NQ's single-hop nature minimizes this benefit.

tokens and their grouping contribute. Aggregating after the MaxSim rather than before is worse, and reducing R from 64 to 16 degrades performance only mildly.

C.3 Pooled Chunk Length

Table 8 sweeps the pooled chunk length L_p of Section 3.3 from $L_p = 1$ to $L_p = 128$. At our chunk length of approximately 130 tokens, $L_p = 128$ is effectively the no-pooling reference, where MaxSim is evaluated against every encoder state and Eq. 3 is exact.

Quality is monotone in L_p over almost the whole range: complete-evidence recall@5 rises with L_p up to $L_p = 64$ on both benchmarks and up to $L_p = 128$ on HotPotQA, and EM follows the same ordering. $L_p = 7$ is therefore not a quality optimum but a cost-driven operating point, and pooling to it costs 11.3 points of complete-evidence recall@5 on HotPotQA and 10.8 points on 2Wiki relative to no pooling.

The cost that grows with L_p is scoring compute and resident GPU memory rather than disk. The pooled index is a mean-pooled view of the same per-token states $\bar{k}$ that the decoder cross-attends to during generation, so it does not contain independent data. For our 759K-chunk, 100M-token pool, storing these states requires 512 GB in `bf16` at $d = 2560$. The pooled copy increases storage by 5% at $L_p = 7$ and 49% at $L_p = 64$. At the no-pooling setting the separate copy is redundant altogether, since retrieval can score the state store itself. What does bind is that the index must be resident for

Table 7: INTRA ablations on HotPotQA and 2Wiki with $k = 5$ retrieved chunks, varying the initial set $\mathcal{S}_0$, the retrieval tokens R, the pooled queries P, and the aggregation order. CE-recall@5 is complete-evidence recall at 5; EM is exact match.

		HotPotQA		2Wiki	
Design choice	Order	CE-recall@5	EM	CE-recall@5	EM
INTRA ($P=16$, $R=64$)	Sum → MaxSim	70.1	48.8	50.2	52.3
$\mathcal{S}_0 = \emptyset$	Sum → MaxSim	40.7 (−29.4)	38.2 (−10.6)	29.6 (−20.6)	43.6 (−8.7)
$P=1$, $R=1$	Sum → MaxSim	38.7 (−31.4)	43.0 (−5.8)	25.0 (−25.2)	46.8 (−5.5)
$P=1$, $R=64$	Sum → MaxSim	53.7 (−16.4)	45.8 (−3.0)	39.8 (−10.4)	50.0 (−2.3)
$P=1$, $R=64$	MaxSim → Sum	62.6 (−7.5)	47.0 (−1.8)	43.5 (−6.8)	50.5 (−1.8)
$P=16$, $R=16$	Sum → MaxSim	67.7 (−2.4)	48.5 (−0.3)	45.7 (−4.6)	51.6 (−0.7)

Table 8: Pooled chunk length L_p on HotPotQA and 2Wiki with $k = 5$ retrieved chunks. CE-recall@5 is complete-evidence recall at 5; EM is exact match. Sizes are for the 759K-chunk, 100M-token pool in `bf16` at $d = 2560$, in decimal GB. Total footprint adds the pooled index to the 512 GB per-token state store that INTRA stores for generation regardless of L_p; at $L_p = 128$ the pooled copy is redundant because retrieval scores that store directly. Quality increases monotonically with L_p, so $L_p = 7$ is a scoring-cost operating point rather than a quality optimum. This sweep was run with $P = 4$.

	HotPotQA		2Wiki		Storage	
Pooled length	CE-recall@5	EM	CE-recall@5	EM	Pooled index	Total
$L_p=1$	48.9	43.9	26.6	45.0	3.9 GB	516 GB
$L_p=4$	57.9	46.0	37.9	48.1	15.5 GB	528 GB
$L_p=7$ (default)	59.9	46.4	40.7	49.2	27.2 GB	539 GB
$L_p=16$	65.5	47.8	48.1	50.7	62.1 GB	574 GB
$L_p=64$	67.0	48.7	**52.0**	**52.5**	248 GB	761 GB
$L_p=128$ (no pooling)	**71.2**	**48.8**	51.5	52.2	–	512 GB

scoring: at $L_p = 7$ it fits on a single device, while the larger settings must be sharded across devices and nodes.

C.4 What the Retrieval Scores Attribute, and Which Hop They Find

Section 5.1 attributes INTRA's multi-hop advantage to scoring evidence by the decoder's own attention. We test that directly with a leave-one-out counterfactual: for every retrieved chunk we remove it from the generator context and re-score the gold answer, recording the drop in the answer's log-likelihood and in exact match. This measures how much the answer depends on a chunk rather than whether it agrees with the annotations. We run it on HotPotQA, with a 20-chunk context, over 7,384 validation questions and 147,680 ablations.

The retrieval scores identify chunks the answer depends on. Table 9 and Figure 7 show that removal cost generally decreases as the retrieval score falls. Removing INTRA's top-ranked chunk lowers exact match (EM) by 7.2 percentage points and the gold answer's log-likelihood by 1.43 nats. For chunks ranked 6–10, both effects are indistinguishable from zero. This pattern does not arise only because top-ranked chunks are more likely to be annotated as supporting evidence. Even among annotated supporting chunks, removal cost increases as the rank falls: from +2.06 nats at rank 1 to +3.54 nats at ranks 11–20. In other words, the decoder relies less on supporting chunks that INTRA ranks lower. Compared with the encoded-question MaxSim baseline, which uses the same representations and generator but a different scoring method, INTRA's advantage is concentrated among the highest-scored chunks. The overall correlation between retrieval score and EM change is the same for both methods ($r = 0.072$). However, in the top score bin, removing an INTRA chunk causes a 0.045-point greater EM drop [+0.010, +0.081]. INTRA also achieves a higher within-question AUC for identifying chunks whose removal lowers log-likelihood by more than 0.5 nats: 0.78 versus 0.66 (Figure 8c).

Table 9: Leave-one-out cost of removing a retrieved chunk from the generator context, by INTRA's rank. Scores are standardized within question; "is oracle" is the share of removed chunks that are annotated supporting chunks. Removal cost decays with the score, so the decoder's attention tracks how much the answer depends on each chunk.

Chunk removed, by INTRA rank	Score (z)	EM drop	Log-lik. drop	Is oracle
rank 1	1.73	**7.2%**	1.43 nats	72%
rank 2	1.26	**5.7%**	0.80 nats	40%
rank 3	1.00	1.3%	0.41 nats	20%
ranks 4–5	0.79	0.1%	0.13 nats	8%
ranks 6–10	0.50	−0.5%	0.07 nats	3%
ranks 11–20	−0.77	−0.1%	0.04 nats	1%

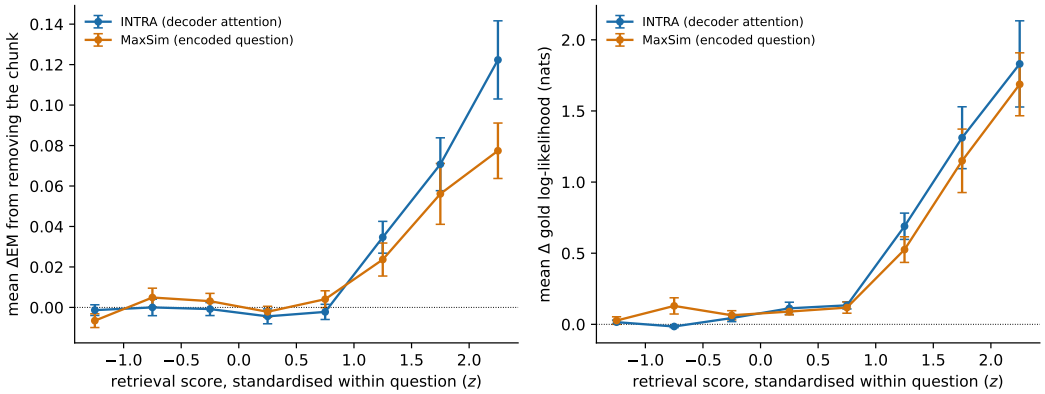

Figure 7: Cost of removing a chunk against its retrieval score, standardized within question. **Left:** exact match. **Right:** gold answer log-likelihood. Error bars are per-question standard errors. Both scorers are flat until $z \approx 0.75$ and then rise, with INTRA separating from MaxSim in the top bins.

The two hops are not equally needed. We divide each two-hop question into an *answer-bearing* chunk, which contains the answer, and a *bridge* chunk. We keep the 5,288 questions for which this distinction is unambiguous and compare INTRA with NV-ColEmbed-v2[2], a fine-tuned late-interaction retriever that ranks the same pool. Removing the answer-bearing chunk lowers the gold answer's log-likelihood by 6.05 nats, compared with 0.16 nats for the bridge. At recall@10, INTRA outperforms the baseline by 12.0 points on the answer-bearing chunk but trails it by 7.1 points on the bridge (Table 10). The baseline favors the bridge because the answer-bearing chunk is reached *through* it and shares little surface form with the question, which favors question-similarity scores. INTRA therefore allocates more of its retrieval budget to the chunk the answer depends on most. This trade-off is favorable even when both hops are needed: INTRA retrieves both at $k = 10$ for 64.1% of questions, compared with 54.3% for the baseline. The scores do not, however, reliably order the selected chunks by how much the answer depends on them. When both supporting chunks are retrieved, the higher-scored chunk is the one the answer depends on more in 47.3% of cases [46.0, 48.6], which is no better than chance. Thus, the useful signal is which chunks enter the top-k, rather than their order within it.

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 $\mathcal{O}(\sqrt{M}L_qL_c)$, which is the practical scaling of the MaxSim operator [19] when using inverted file (IVF) approximate nearest-neighbor (ANN) indices, such as FAISS or cuVS.

[2] `https://huggingface.co/nvidia/llama-nemotron-colembed-vl-3b-v2`

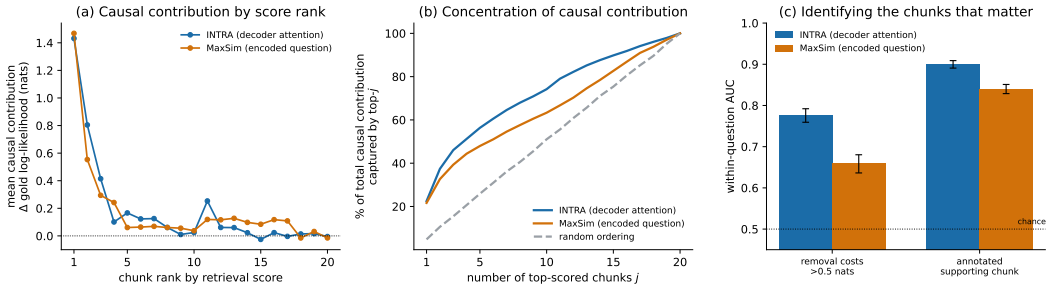

Figure 8: Attribution by rank on the same questions. **(a)** mean log-likelihood drop against rank by retrieval score. **(b)** share of the total drop captured by the top-j chunks, against a random-ordering reference. **(c)** within-question AUC for identifying chunks whose removal costs more than 0.5 nats, and for identifying annotated supporting chunks, with the 0.5 chance line.

Table 10: Per-hop breakdown on the $5{,}288$ HotPotQA questions whose answer localizes to exactly one supporting chunk. Answer dependence is the drop in the gold answer's log-likelihood when that hop is removed; recall is over all such questions, so a hop missing from the pool counts against both retrievers. Brackets are 95% bootstrap intervals; ColEmb. is NV-ColEmbed-v2. The answer depends almost entirely on the answer-bearing hop, which is the hop INTRA gains on; it gives up ground on the bridge, which the answer barely uses, and comes out ahead when both are required.

		recall@5		recall@10		recall@20	
Supporting chunk	Answer dependence (nats)	ColEmb.	INTRA	ColEmb.	INTRA	ColEmb.	INTRA
answer-bearing hop	**+6.05** [+5.75, +6.36]	55.0	**66.1**	61.9	**73.9**	67.3	**76.0**
bridge hop	+0.16 [+0.11, +0.20]	**87.3**	78.3	**91.1**	84.0	**93.9**	86.8
both hops	—	44.5	**53.3**	54.3	**64.1**	61.8	**67.5**

We assume transformer-based encoders and decoders [38], 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.

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 $\mathcal{O}(\sqrt{M} L_q L_c)$ 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.

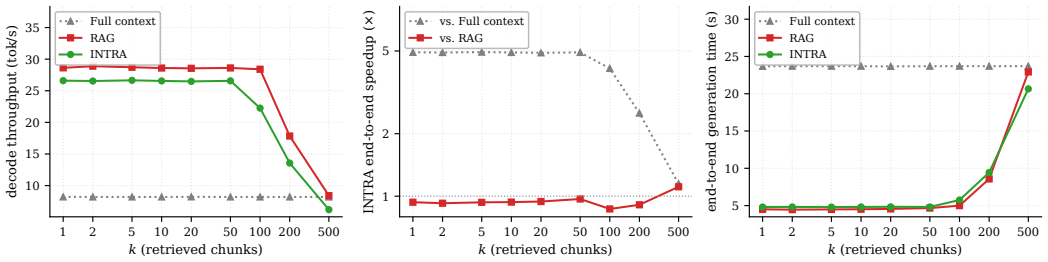

Figure 9: Generation cost against the number of retrieved chunks k, at T5Gemma2 4B-4B on one H100. **Left:** decode throughput, measured after the first token. **Middle:** INTRA's end-to-end speedup, prefill plus 128 generated tokens. **Right:** the end-to-end time that ratio is taken from.

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) starts from $L_q = L_c = L_g = 128$, $N = 65{,}536$ and $k = 20$, sweeps one of these per figure, excludes retrieval time, and gives every system the same top-k chunks. It therefore measures only the generator-side cost after evidence has been selected. Full-context prompting is the reference point, incurring $\mathcal{O}((L_q + N)^2)$ prefill by processing the query with the whole evidence pool; it is independent of both k and L_c, so it appears throughout as a flat line at 8.21 s of prefill, 8.2 tok/s of decode and 63.6 GB of the 80 GB device. 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 33.8 ms to 156 ms, while standard RAG grows from 61.5 ms to 7.81 s, so INTRA's speedup over re-encoding widens from $1.8\times$ to $50\times$ and its advantage over full-context prompting narrows from $243\times$ to $53\times$. Figure 10 shows the same trend as chunk length increases at a fixed $k = 20$, with re-encoding reaching 1.01 s at $L_c = 1024$ against 60.8 ms for INTRA. 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.

Decoding. Figure 9 reports tokens per second after the first token, where the gap is far smaller than in prefill: 26.5 tok/s for INTRA at $k = 20$ against 28.5 for re-encoded RAG, since scoring the shared d-dimensional pool and projecting V on demand is slightly more work per token than attending to re-encoded states. Over 128 generated tokens this outweighs the prefill saving at moderate k, so end-to-end INTRA is $0.94\times$ re-encoded RAG at $k = 20$ and overtakes it only at $k = 500$ ($1.11\times$), while remaining $4.9\times$ faster than full-context prompting throughout. INTRA's efficiency claim is therefore about prefill and storage rather than decoding.

D.5 Comparison Against a Fully Cached Baseline

Re-encoding is not the only way to serve the selected chunks: their cross-attention states can also be precomputed and cached. Table 11 adds this baseline in its strongest form, with every decoder layer's K and V stored so that prefill performs no encoding and no projections. We report it separately from the figures above because it is not a system we compare against on quality, and because the

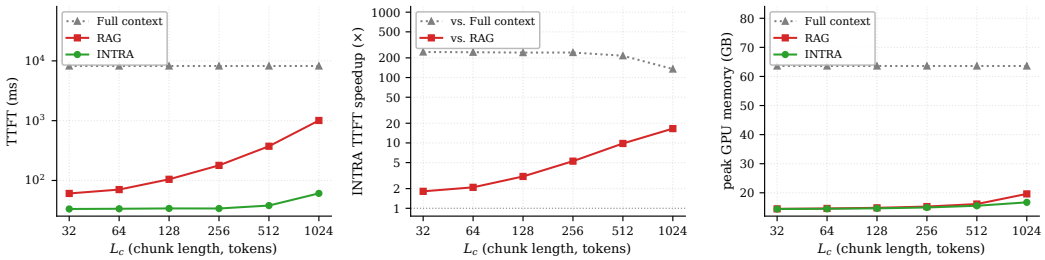

Figure 10: Prefill cost against chunk length L_c at $k = 20$, at T5Gemma2 4B-4B on one H100, in the layout of Figure 4. **Left:** time to first token, where re-encoding grows with kL_c while INTRA stays flat until the memory itself becomes large. **Middle:** INTRA's TTFT speedup. **Right:** peak GPU memory, of which 14.2 GB is model weights.

Table 11: Time to first token and state fetched per query, against a cached baseline that precomputes every decoder layer's cross-attention K and V so that prefill performs no encoding and no projections. Measured at the paper's configuration (T5Gemma2 4B-4B, one H100), including the per-query fetch of selected chunk states from local NVMe, at $L_c = 128$ and 8-bit precision. Caching matches INTRA below $k \approx 8$, but holds that latency only by moving $27.2\times$ more state per query.

Retrieved chunks k	5	10	20	50	100
Time to first token (ms)					
Standard RAG (re-encode)	60.9	69.3	103.3	226.9	518.5
Standard RAG + cached K, V	**32.4**	34.6	38.4	50.3	69.8
INTRA	33.8	**33.9**	**34.4**	**35.1**	**45.5**
State fetched per query (MB)					
Standard RAG + cached K, V	44.6	89.1	178	446	891
INTRA	**1.6**	**3.3**	**6.6**	**16.4**	**32.8**

comparison turns on data movement rather than on the compute those figures measure. At corpus scale neither INTRA's pool nor the cached keys and values fit in HBM, so both fetch the states of their selected chunks from local storage per query, and the measurements include that fetch. Caching removes essentially all of the re-encoding gap, and below $k \approx 8$ it is marginally faster than INTRA; over that range, Figure 4 should be read as an advantage over re-encoding rather than over caching. It sustains that latency only by moving $27.2\times$ more state per query, however, so it crosses over at $k \approx 8$ and is $1.5\times$ slower by $k = 100$: INTRA's latency advantage at practical k is a consequence of its storage advantage. Where that storage gap is small the balance shifts back into a memory-for-latency trade-off, which is the regime of very small k and of architectures whose per-token cache is already thin, such as multi-query or multi-head latent attention [8].

D.6 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 [30].

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 [12], 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 12: 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 759K 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$759K chunks in the pool, including those that serve as oracles for other examples, act as distractors.

F Broader Societal Impact

INTRA is a retrieval-and-generation method rather than a deployed application, but it may affect systems that answer questions over large knowledge bases. On the positive side, reusing model-native evidence representations could reduce the repeated computation needed to serve retrieval-augmented QA over stable corpora, potentially lowering cost and making high-quality evidence access more practical. On the negative side, stronger retrieval-conditioned generation can also make incorrect, biased, private, or otherwise sensitive evidence easier to surface and present fluently if the underlying corpus or deployment safeguards are inadequate. As with other RAG systems, deployment should therefore include corpus governance, access controls, provenance tracking, and application-level evaluation for hallucination, privacy, and fairness risks.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction state the paper's main conceptual claim and empirical contribution: INTRA uses a pretrained encoder-decoder's own representation space for retrieval and generation, with strongest evidence on fixed-pool benchmark QA. The Discussion and Limitations further bound the scope, noting that the work does not claim attention universally replaces modular retrieval and was evaluated on a specific implementation family and task setting.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations section. It discusses the fixed-pool and textual-evidence setting, the fact that INTRA is not shown to replace modular retrieval in open or dynamic corpora, the partial nature of the systems evaluation, memory and indexing trade-offs, and the narrow empirical scope across architectures, scales, and modalities.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper formalizes INTRA and its scoring procedures mathematically, but it does not present formal theoretical results such as theorems, lemmas, or proofs. The main claims are conceptual and empirical rather than theorem-based.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The Method, Practical Implementation, and Benchmarks and Experimental Setup sections specify the INTRA scoring procedure, training objective, model checkpoint, candidate-pool construction, baselines, retrieval and QA metrics, context sizes, and deterministic decoding setup. The appendix provides additional dataset split statistics and retrieval-training details, including optimizer, learning rate, warmup, number of steps, and batch size.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The supplemental material will include instructions to reproduce the main experimental results. The paper also describes the datasets, candidate-pool construction, model setup, training details, baselines, and metrics needed to understand the released reproduction pipeline.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The Benchmarks and Experimental Setup section specifies the evaluated datasets, shared candidate-pool construction, baselines, metrics, checkpoint initialization, frozen and trainable parameters, retrieval context sizes, pooling length, and deterministic decoding setup. The appendix adds the retrieval-training hyperparameters and split statistics, including optimizer, learning rate, warmup, number of steps, batch size, and per-benchmark train/evaluation sizes.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The main result plots report error bars for the retrieval and end-to-end exact-match results. These uncertainty estimates accompany the experimental comparisons that support the paper's retrieval and QA claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix D.6 reports the compute resources used for training, including NVIDIA H100 GPUs with 80GB of memory, the maximum 160-GPU training configuration, the longest pretraining run of approximately 12 hours or about 2,000 GPU-hours, and the shorter benchmark-specific retrieval/generation runs of less than 200 GPU-hours each.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The research was conducted in accordance with the NeurIPS Code of Ethics. The experiments use existing QA benchmark data and model/software assets, do not involve new human-subject data collection, and are reported with reproducibility, limitations, and compute-resource information.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Appendix F discusses potential positive impacts from reducing repeated computation and improving evidence access over stable corpora, as well as potential negative impacts involving incorrect, biased, private, or sensitive evidence being surfaced by retrieval-conditioned generation without appropriate safeguards.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a new high-risk pretrained language model, image generator, or newly scraped dataset. The planned release consists of reproduction code and scripts for constructing the benchmark-derived candidate pool from existing QA resources.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original sources for the benchmark datasets, pretrained models, retrieval baselines, and software assets used in the experiments. Appendix **??** lists the principal existing assets and their licenses or terms of use, and the released reproduction package will include the corresponding attribution and license links.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The supplemental material will include the reproduction code and scripts for constructing the dataset and shared candidate pool used in the experiments. These assets are documented through the experimental setup, implementation details, dataset/pool details, and reproduction instructions described in the paper and supplement.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not conduct crowdsourcing experiments or new research with human subjects. It evaluates on existing QA benchmarks and derived benchmark resources.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or new human subject research, so IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: the core method and experiments explicitly use pretrained language models, including T5Gemma-style encoder-decoder models for INTRA and additional language-model generators in the evaluation. Their roles are described in the Method, Practical Implementation, Benchmarks and Experimental Setup, and Results sections.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.