

ArcAligner: Adaptive Recursive Aligner for Compressed Context Embeddings in RAG

Jianbo Li*, Yi Jiang*, Sendong Zhao[†], Bairui Hu, Haochun Wang, Bing Qin

Harbin Institute of Technology, China
 {jianboli,yjiang,sdzhao,brhu,hcwang,qinb}@ir.hit.edu.cn

Abstract

Retrieval-Augmented Generation (RAG) helps LLMs stay accurate, but feeding long documents into a prompt makes the model slow and expensive. This has motivated context compression, ranging from token pruning and summarization to embedding-based compression. While researchers have tried “compressing” these documents into smaller summaries or mathematical embeddings, there is a catch: the more you compress the data, the more the LLM struggles to understand it. To address this challenge, we propose ArcAligner (Adaptive recursive context Aligner), a lightweight module integrated into the language model layers to help the model better utilize highly compressed context representations for downstream generation. It uses an adaptive “gating” system that only adds extra processing power when the information is complex, keeping the system fast. Across knowledge-intensive QA benchmarks, ArcAligner consistently beats compression baselines at comparable compression rates, especially on multi-hop and long-tail settings. The source code is publicly available¹.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Guu et al., 2020) is now the standard approach for keeping Large Language Models (LLMs) (Achiam et al., 2023; Touvron et al., 2023) factually accurate. By grounding answers in external documents rather than relying on memory alone, RAG helps models avoid “hallucinations” and answer difficult, niche questions more reliably. However, the standard way of doing RAG—simply pasting long documents into the prompt—creates a bottleneck. As we ask models to handle longer histories, deeper user profiles, or massive libraries of evidence, the input quickly becomes too long

*Equal Contribution.

[†]Corresponding Author.

¹<https://github.com/liunian-Jay/ArcAligner.git>

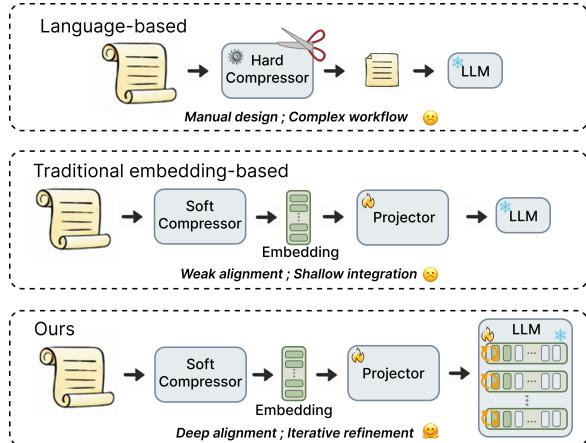


Figure 1: Comparison of different methods.

for the model to process efficiently. This makes effective context compression no longer just an optimization, but a necessity for building responsive, long-memory LLM agents.

To solve this problem, recent studies explore context compression—essentially shrinking the content before the LLM reads it (Liu et al., 2025). Current research generally follows one of two paths to achieve this. The first is **text-level compression**, which involves pruning unnecessary words or summarizing documents into a shorter prompt (Pan et al., 2024). The second is **embedding-based compression**, which translates long text into a compact “shorthand” of mathematical signals (embeddings) that the model can digest quickly (Cheng et al., 2024; Ge et al.; Rau et al., 2025). These methods suggest a compelling possibility: if we can improve an LLM’s in-model alignment to compressed context, RAG could maintain answer quality while significantly reducing time and cost.

However, as shown in Figure 1, these approaches still face significant limitations (Nagle et al., 2024; Deng et al., 2025; Chen et al., 2025). Text-level compression is often unpredictable; because it relies on picking and choosing specific words to keep,

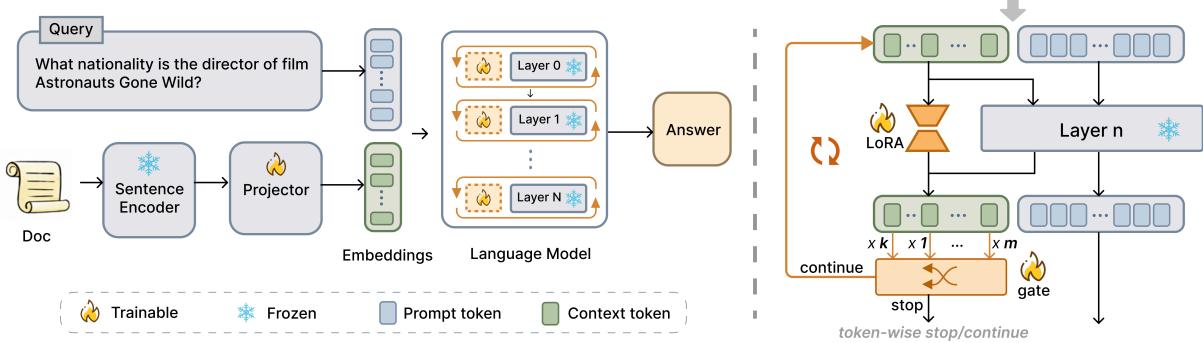


Figure 2: Illustration of the ArcAligner framework. The left part is an illustration of our training and inference process, where the context is compressed and progressively refined within the LLM while the input query remains unchanged. The right part is a flow of the token through each layer, passing through the LoRA and the gate, where the context tokens are adaptively and recursively refined.

even a small error can result in the loss of crucial evidence. This makes it particularly unreliable for complex, multi-step questions where every detail matters. On the other hand, embedding-based compression avoids deleting text but creates a “language barrier.” The compressed signals are mathematically different from what the LLM is used to seeing, making it difficult for the model to actually use that information to generate a grounded answer. Despite progress on compression objectives and encoder-decoder interfaces, performance under strong compression still hinges on whether the generator can consistently exploit and progressively decode compressed evidence for answering.

We propose **ArcAligner**, a lightweight module that enables LLMs to better read and reason over compressed context representations. ArcAligner operates within the model layers to progressively align compressed signals with the representations used during generation, reducing the mismatch between compressed inputs and the model’s native processing space. It further introduces an adaptive alignment mechanism with a learned gate that controls, layer by layer, how much additional refinement is applied to each piece of compressed information. This design helps preserve task-relevant evidence and supports multi-step reasoning on complex questions. Across a range of knowledge-intensive benchmarks, ArcAligner consistently improves answer quality and reliability over prior compression-based baselines under the same context budget. Our main contributions are:

- **The ArcAligner Framework:** We propose ArcAligner, a parameter-efficient alignment framework that performs deep semantic alignment from within the language model through

multi-stage training.

- **Adaptive Recursive Alignment:** We propose a novel “gating” mechanism that selectively processes information. By applying additional refinement where needed, ArcAligner improves the use of compressed context without uniformly increasing computation.
- **Robust Performance Gains:** We provide extensive experiments and ablations showing consistent gains over strong compression baselines, with clear improvements in “difficult” scenarios, such as multi-hop and long-tail QA.

2 Related Work

2.1 Retrieval-Augmented Generation

RAG improves factuality by conditioning LLM on retrieved evidence. The standard pipeline concatenates top- k passages into the prompt and relies on attention to integrate evidence. Existing approaches enhance evidence utilization with fusion-in-decoder readers (Jiang et al., 2025c) and end-to-end retrieval-generation training (Izacard and Grave, 2021; Izacard et al., 2023; Jiang et al., 2025a), and also investigate reliability and controllable reliance on retrieved signals (Asai et al., 2024). However, long prompts incur substantial inference overhead, motivating methods that reduce the context burden while preserving evidence usability.

2.2 Compressing Retrieved Evidence

Context compression reduces context before it reaches the generator, via *hard* compression in text space or *soft* compression in embedding space.

Hard Compression (Text Space). Hard compression shortens the prompt by selecting/pruning tokens or sentences (Li et al., 2023; Jiang et al., 2023b; Pan et al., 2024) or rewriting/distilling passages into shorter textual evidence (Xu et al.). These methods preserve a text-only interface but require query-time decisions and can be sensitive to downstream alignment; under aggressive budgets, small mistakes may remove answer-critical evidence and disproportionately hurt difficult or long-tail questions (Pan et al., 2024).

Soft Compression (Embedding Space). Soft compression replaces retrieved text with compact continuous representations, e.g., encoding passages into a small set of projected embedding tokens/slots (Cheng et al., 2024), or injecting learned context/memory representations via reconstruction-style objectives and downstream fine-tuning (Ge et al.; Rau et al., 2025; Pilchen et al., 2025). While avoiding long prompts, it shifts the bottleneck to alignment and usability: compressed embeddings are heterogeneous to the generator’s hidden space and typically require a projector, while training must ensure reliable decoding under strong compression (Rau et al., 2025; Pilchen et al., 2025).

2.3 From Compression to Usable Evidence

A key issue is that *compressing or preserving information* does not guarantee *effective use* during decoding. Related work tackles adjacent aspects: parameter-efficient adaptation (adapters, prefix/prompt tuning, low-rank updates) improves behavior under constrained or non-standard inputs (Houlsby et al., 2019; Li and Liang, 2021; Hu et al., 2022), while RAG-specific designs regulate reliance on retrieved evidence (e.g., self-checking/controlled use and reducing over-reliance via preference alignment) (Asai et al., 2024; Jiang et al., 2025b). However, strong compression still poses a distinct challenge—whether compact retrieval representations can be consistently interpreted and exploited throughout decoding. Unlike embedding-based compression methods that mainly optimize representations or compression objectives (Cheng et al., 2024; Ge et al.; Rau et al., 2025; Pilchen et al., 2025), ArcAligner targets decode-time usability, strengthening how the generator integrates compressed retrieval information for grounded generation.

3 Methodology

We propose **ArcAligner**, a parameter-efficient alignment framework (Figure 2) that improves the usability of compressed embeddings by: (i) enforcing *layer-wise mandatory alignment* of context slots, and (ii) enabling *adaptive recursive alignment* whose depth is controlled by a learned *gate*. A three-stage training strategy progressively equips the model with reconstruction ability, task competence, and adaptive refinement.

3.1 Problem Setup

Given a query q , a retriever returns the top- k passages $\{p_i\}_{i=1}^k$ from a corpus \mathcal{P} . Instead of concatenating retrieved passages as text, we compress them into a short sequence of dense *compressed context embeddings*

$$E \in \mathbb{R}^{m \times d_r},$$

where m is the number of context slots, d_r denotes the embedding dimension of each compressed context embedding, and $|p_i|$ denotes the number of tokens in passage p_i , with $m \ll \sum_i |p_i|$. These embeddings are provided to the language model as a fixed set of designated *context slots*, forming a compact interface between retrieved evidence and the base model.

3.2 Context Slot Interface

Let the hidden size of the LLM be d . We map compressed context embeddings into the model space using a lightweight projector $W_\psi(\cdot)$, obtaining the *projected context-slot embeddings*

$$\tilde{E} = W_\psi(E) \in \mathbb{R}^{m \times d}, \quad (1)$$

where m is the number of context slots and $\tilde{E}_j \in \mathbb{R}^d$ denotes the embedding assigned to the j -th slot.

We construct an input sequence of length n and reserve a fixed set of context-slot positions $r = \{r_1, \dots, r_m\} \subset \{1, \dots, n\}$. Let $H^{(0,0)} \in \mathbb{R}^{n \times d}$ denote the initial token-wise hidden states fed into the first transformer layer, defined as

$$H_i^{(0,0)} = \begin{cases} \tilde{E}_j, & \text{if } i = r_j, \\ \text{Emb}(u_i), & \text{otherwise,} \end{cases} \quad (2)$$

where $\text{Emb}(\cdot)$ is the standard token embedding function and $\{u_i\}_{i=1}^n$ denotes the textual input sequence (e.g., the query and prompt tokens).

3.3 Selective LoRA on Context Slots

Inspired by multimodal models (Dong et al., 2024), we treat context as a special modality. Specifically, we design context slots and apply LoRA ((Hu et al., 2022)) updates only to context tokens while keeping the prompt tokens on the frozen base path.

Let $H^{(\ell,t)} \in \mathbb{R}^{n \times d}$ denote the hidden states at layer ℓ and recursive step t . At the block level, let $F_\theta^{(\ell)}(\cdot)$ denote the ℓ -th transformer layer with frozen base parameters θ , and let $\Delta F_\phi^{(\ell)}(\cdot)$ be the LoRA-induced residual.

We define a binary broadcast mask $M_r \in \{0, 1\}^{n \times 1}$, where $(M_r)_i = \mathbb{I}[i \in r]$. The resulting *selective-LoRA* layer is defined as

$$\mathcal{A}^{(\ell)}(H) = F_\theta^{(\ell)}(H) + M_r \odot \Delta F_\phi^{(\ell)}(H), \quad (3)$$

where \odot denotes element-wise multiplication broadcast along the hidden dimension.

3.4 Adaptive Recursive Alignment with Gate

We introduce an *Adaptive Recursive Aligner* that dynamically controls the refinement depth of context slots across transformer layers. A subset of layers is equipped with slot-wise gates, denoted by $\mathcal{L}_{\text{gate}}$, while all other layers perform a single refinement step. Let N denote the total number of transformer layers.

Mandatory refinement. At every layer $\ell \in \{0, \dots, N-1\}$, an initial refinement step is always applied:

$$H^{(\ell+1,0)} = \mathcal{A}^{(\ell)}(H^{(\ell,0)}). \quad (4)$$

Gate-controlled recursive refinement. At gated layers ($\ell \in \mathcal{L}_{\text{gate}}$), we allow up to T additional refinement steps indexed by $t = 1, \dots, T$. Let $H_r^{(\ell+1,t-1)} \in \mathbb{R}^{m \times d}$ denote the hidden states restricted to context-slot positions at step $t-1$. A lightweight gating network predicts a slot-wise refinement decision:

$$g^{(\ell,t)} = \sigma\left(\text{MLP}^{(\ell)}(H_r^{(\ell+1,t-1)})\right), \quad (5)$$

where $g^{(\ell,t)} \in [0, 1]^{m \times 1}$ and $\sigma(\cdot)$ denotes the sigmoid function.

Given the current state, we compute a candidate refinement by reapplying the same selective-LoRA:

$$\tilde{H}^{(\ell+1,t)} = \mathcal{A}^{(\ell)}(H^{(\ell+1,t-1)}). \quad (6)$$

We define $\text{stopgrad}(\cdot)$ as the stop-gradient operator, which blocks gradient propagation through

its argument. During training, we adopt a straight-through estimator (STE) (Bengio et al., 2013) to enable discrete refinement decisions while preserving gradient flow. Specifically, the hard gate $g_{\text{hard}}^{(\ell,t)} = \mathbb{I}[g^{(\ell,t)} \geq 0.5]$ is used in the forward pass, while gradients are backpropagated through the continuous probability:

$$g_{\text{STE}}^{(\ell,t)} = g^{(\ell,t)} + \text{stopgrad}\left(g_{\text{hard}}^{(\ell,t)} - g^{(\ell,t)}\right). \quad (7)$$

The context-slot update is then given by

$$H_r^{(\ell+1,t)} = H_r^{(\ell+1,t-1)} + g_{\text{STE}}^{(\ell,t)} \odot \left(\tilde{H}_r^{(\ell+1,t)} - H_r^{(\ell+1,t-1)}\right), \quad (8)$$

while non-slot positions remain fixed across recursive steps:

$$H_{\bar{r}}^{(\ell+1,t)} = H_{\bar{r}}^{(\ell+1,0)}. \quad (9)$$

At inference time, we replace $g_{\text{STE}}^{(\ell,t)}$ with the strictly binary gate $g_{\text{infer}}^{(\ell,t)} = g_{\text{hard}}^{(\ell,t)} \in \{0, 1\}^{m \times 1}$, so that only slots with $g_{\text{infer}}^{(\ell,t)} = 1$ are further refined, yielding an exact conditional update.

3.5 Three-Stage Training Strategy

We adopt a three-stage training strategy that progressively equips the model with (i) alignment to compressed embeddings, (ii) task grounding under compressed-slot inputs, and (iii) adaptive refinement via the gate.

Stage I: Reconstruction Alignment Pretraining.

We first warm up the model to interpret compressed embeddings by reconstructing the original text. Specifically, we optimize the projector parameters ψ in $W_\psi(\cdot)$ and the selective LoRA parameters ϕ , while keeping the base LLM frozen. Given an input where the designated slot positions R are filled with compressed embeddings \bar{E} , the model is trained to autoregressively reconstruct the context y using the negative log-likelihood (NLL) objective:

$$\mathcal{L}_{\text{rec}} = - \sum_{i=1}^{|y|} \log p_\theta(y_i | H^{(0,0)}, y_{<i}). \quad (10)$$

Stage II: Task-Grounded RAG Finetuning.

Starting from Stage I, we finetune the model on downstream RAG tasks so that it can answer queries by grounding on compressed slot inputs. During this stage, we continue updating ψ and ϕ but disable the gate, so that each layer performs

only the mandatory single-step refinement. The training target is the ground-truth answer and is optimized using the same NLL objective as in Stage I.

Stage III: Gate-Aware Adaptive Finetuning. In the final stage, we enable the gating mechanism and train the full adaptive recursive aligner. We jointly optimize the projector parameters ψ , the selective LoRA parameters ϕ , and the gate parameters. The forward pass follows § 3.4, using STE-based gating during training and hard binary gating at inference. The optimization in this stage still uses NLL loss. Through training in this stage, ArcAligner is trained more thoroughly and can learn slot refinement depth according to task requirements, thereby supporting more effective inference under strong compression.

4 Experiments

4.1 Implementation Details

Base Models. We use *Mistral-7B-Instruct-v0.2* (Jiang et al., 2023a) as the base language model in all main experiments. We use *SFR-Embedding-Mistral* (Meng et al., 2024) as the sentence encoder to encode passages into dense vectors. All parameters of the base model are frozen during training, except for the lightweight modules introduced by our method.

Passage Segmentation. We segment retrieved passages at the sentence level using spaCy (Honnibal et al., 2020). These sentence-based segments are used as the basic units for compression.

Training Data. Our training follows a three-stage training strategy. *Stage I (Pretraining)*. Pre-training data is built from a large-scale Wikipedia corpus. We follow standard preprocessing by segmenting the dump into fixed-length passages, then sample about 200k passages and train for one epoch. This stage learns a robust alignment between embeddings and the language model’s representation space. *Stage II & III (RAG Finetuning)*. Both stages are trained on the HotpotQA (Yang et al., 2018) dataset, using approximately 90K training samples. Stage II enables only the projector and LoRA parameters, while Stage III further enables the gate for adaptive refinement. Details can be found in Appendix C.

4.2 Datasets and Evaluation Metrics

Evaluation Datasets. We evaluate effectiveness and generalization on multi-hop and open-domain

QA benchmarks. For multi-hop QA, we use 2WikiMultiHopQA (Ho et al., 2020) and HotpotQA (Yang et al., 2018); for open-domain QA, we use NaturalQA (Kwiatkowski et al., 2019), PopQA_longtail (Mallen et al., 2023), TriviaQA* (Joshi et al., 2017), and WebQuestions (Berant et al., 2013), where *TriviaQA** denotes a subsampled evaluation set consisting of 500 randomly selected questions. For each dataset, we retrieve the top-20 passages with *Contriever* (Izacard et al., 2021) and rerank them with *RankZephyr* (Pradeep et al., 2023). The reranked top passage is used as the context. All datasets share the same retrieval and inference settings, with details in Appendix A.

Evaluation Metrics. We report Exact Match (**EM**), **F1**, and Accuracy (**Acc**). Following prior work (Asai et al., 2024; Mallen et al., 2023), we use *non-strict* EM, counting a prediction as correct if it contains the gold answer. F1 is computed as token-level overlap with the gold answer, complementing EM since longer outputs can increase EM but lower F1. We additionally report LLM-judged accuracy; details are in Appendix B.

4.3 Baselines

We compare against representative baselines: (1) **Naive**, which answers the question without any retrieval context; (2) **StandardRAG**, the classic retrieve-then-read setup that concatenates retrieved passages with the query; (3) **xRAG** (Cheng et al., 2024), which replaces text passages with projected dense retrieval embeddings for extreme compression; (4) **COCOM** (Rau et al., 2025), which injects learned context embeddings into the decoder; and (5) **LLMLingua-2** (Pan et al., 2024), which compresses retrieved contexts by retaining only informative tokens before generation.

4.4 Main Results

Table 1 and 2 report the main results. Some key findings are as follows.

(1) Compression vs. No Compression. StandardRAG achieves the highest EM and accuracy on most datasets, setting a performance upper bound by providing the full textual context. In contrast, embedding-based compression methods (e.g., xRAG and COCOM) show a notable drop, highlighting the limitations of current compression techniques. Our approach offers a trade-off and performs well on multi-hop and long-tail tasks.

(2) Language space vs. Embedding space.

Method	Comp. rate	2WikiMultiHopQA			HotpotQA			NaturalQA		
		EM	F1	Acc	EM	F1	Acc	EM	F1	Acc
w/o retrieval										
Naive [▽]	–	22.20	14.63	19.40	21.20	13.39	23.20	27.20	16.50	28.25
w/ retrieval										
StandardRAG [△]	–	31.60	18.55	22.60	37.40	19.72	36.60	43.49	23.56	44.57
xRAG	×128	48.80	8.66	22.00	<u>30.40</u>	6.00	27.80	44.13	3.39	34.04
COCOM	×4	26.80	29.38	27.20	25.20	<u>32.45</u>	31.18	36.59	<u>39.54</u>	41.50
COCOM	×16	27.80	<u>31.02</u>	<u>28.80</u>	24.40	31.77	<u>31.20</u>	35.18	41.04	41.94
LLMLingua-2	×3	<u>33.40</u>	15.78	23.20	31.40	16.41	29.40	<u>37.73</u>	18.98	36.95
ArcAligner (Ours)	×24	31.80	36.00	33.40	26.60	34.92	33.40	31.72	37.82	37.04

Table 1: EM/F1/Acc on 2WikiMultiHopQA, HotpotQA, and NaturalQA. Comp. rate denotes the context compression ratio. Best and second-best scores are highlighted in **bold** and underlined, respectively. *Italic* denote excluded reference settings: Naive[▽] (no-retrieval) and StandardRAG[△] (full-context).

Method	Comp. rate	PopQA_longtail			TriviaQA*			WebQuestions		
		EM	F1	Acc	EM	F1	Acc	EM	F1	Acc
w/o retrieval										
Naive [▽]	–	20.01	8.90	13.30	56.60	33.25	44.80	38.34	25.32	45.96
w/ retrieval										
StandardRAG [△]	–	45.96	23.10	33.95	69.40	35.63	60.20	39.67	23.96	45.23
xRAG	×128	<u>33.95</u>	4.94	28.95	69.80	11.53	55.80	55.86	5.14	<u>47.79</u>
COCOM	×4	24.59	25.15	24.37	60.80	<u>63.60</u>	<u>58.80</u>	42.72	36.43	47.44
COCOM	×16	24.16	<u>25.79</u>	24.59	58.00	61.91	58.40	37.84	<u>40.45</u>	48.77
LLMLingua-2	×3	40.17	15.66	31.09	<u>66.60</u>	30.62	51.80	37.75	21.04	42.77
ArcAligner (Ours)	×24	30.38	32.80	30.74	62.80	65.65	62.80	35.14	43.19	46.80

Table 2: EM/F1/Acc on PopQA_longtail, TriviaQA*, and WebQuestions. Comp. rate denotes the context compression ratio. Best and second-best scores are highlighted in **bold** and underlined, respectively. *Italic* denote excluded reference settings: Naive[▽] (no-retrieval) and StandardRAG[△] (full-context).

Method	HotpotQA			NaturalQA			TriviaQA*		
	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc
ArcAligner (Ours)	26.60	34.92	33.40	31.72	37.82	37.04	62.80	<u>65.65</u>	62.80
w/o Recursion	26.00	<u>34.48</u>	<u>32.60</u>	31.39	<u>37.64</u>	<u>36.90</u>	62.00	65.17	62.00
w/o Gate (Max Loop)	27.20	31.69	31.00	31.25	32.57	32.74	64.00	62.66	61.80
w/o LoRA & Recursion	25.00	32.02	32.00	31.58	37.50	36.76	<u>63.40</u>	66.20	<u>62.20</u>

Table 3: Ablation results on HotpotQA, NaturalQA, and TriviaQA*. We report non-strict EM, F1, and LLM-judged accuracy (Acc). **w/o Recursion** disables gate-controlled refinement loops at inference. **w/o Gate (Max Loop)** disables the gate while keeping a fixed number of refinement loops for all context slots. **w/o LoRA & Recursion** disables the selective-adaptation weights (LoRA) and refinement loops, but keeps the context slots.

Overall, language-based methods showed mixed results, suggesting they require unique and complex designs to handle different tasks. Embedding-based methods outperformed LLMLingua-2 overall and demonstrated stronger robustness across various tasks. Our ArcAligner exhibits even greater robustness and learn more task-oriented information.

(3) Shallow Alignment vs. Deep alignment

Shallow alignment methods such as xRAG exhibit high EM but low F1, resulting in excessively long outputs, poor task orientation, and even worse per-

formance. In contrast, deep alignment methods like COCOM and ArcAligner not only ensure information availability but also provide greater task orientation.

4.5 Ablation Study

We conduct ablations on two core components of ArcAligner, with results reported in Table 3.

Disabling refinement loops (*w/o Recursion*) consistently degrades performance, indicating that recursive refinement provides larger gains than a sin-

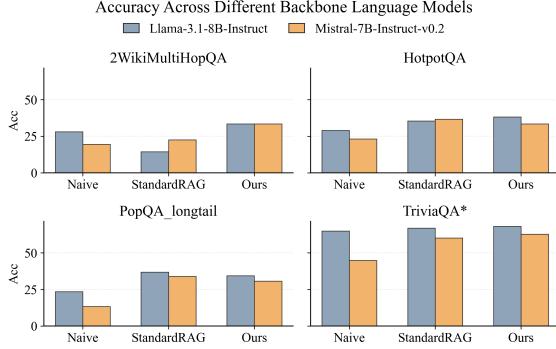


Figure 3: Accuracy across different backbone language models on four QA datasets. Grouped bars compare **Naive**, **StandardRAG**, and **ArcAligner(Ours)** under two backbones: **Llama-3.1-8B-Instruct** and **Mistral-7B-Instruct-v0.2**.

gle pass. Further removing the gating mechanism while keeping a fixed number of refinement loops for all context slots (*w/o Gate (Max Loop)*) leads to more pronounced drops on NaturalQA and TriviaQA*, suggesting that indiscriminate refinement is suboptimal and that the gate is crucial for selecting which context slots to refine.

Finally, removing selective adaptive weights while retaining projection-generated context slots (*w/o LoRA & Recursion*) performs worse than the full model, highlighting the importance of deep semantic alignment.

Overall, layer-wise selective adaptation and learned gated recursion are key to aligning compressed context embeddings.

4.6 Analysis Across Different Backbones

Figure 3 reports the accuracy of different methods under two backbone language models on four representative QA datasets. Across datasets, the relative performance ordering among *Naive*, *StandardRAG*, and *ArcAligner* remains largely consistent when switching between Llama-3.1 and Mistral-v0.2, although the absolute accuracy varies across backbones. In particular, *ArcAligner* exhibits comparable behavior under both backbone models and maintains competitive performance across datasets. These results suggest that the observed method-level trends are not specific to a particular backbone language model.

4.7 Reconstruction Ability After Pretraining

To investigate reconstruction pre-training, we compare the reconstruction perplexity (PPL) on the test set after the pre-training stage.

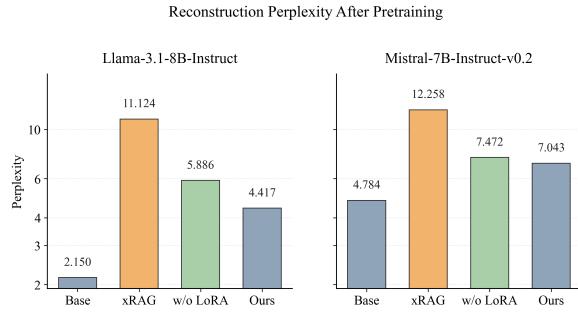


Figure 4: Reconstruction perplexity (PPL) after the pre-training stage, evaluated on a held-out pretraining test set with 10k samples. Results are reported for two backbone language models, **Llama-3.1-8B-Instruct** and **Mistral-7B-Instruct-v0.2**. Lower PPL indicates better reconstruction fidelity.

Figure 4 reports the results. *Base* represents the model directly reciting the context without compression, and can be considered as the upper limit of reconstruction performance. *xRAG* yields substantially higher PPL for both backbone language models, indicating that directly applying retrieval-based compression markedly reduces reconstruction fidelity during pre-training. Our model attains lower PPL than *xRAG* and *w/o LoRA*, showing that the proposed alignment design mitigates the degradation introduced by compression. Removing the deep alignment module (*w/o LoRA*) also lowers PPL relative to *xRAG*, but remains significantly worse than *Base*. This suggests that shallow projection alone is insufficient for the LLM to interpret compressed embeddings.

4.8 Analysis of Gate Behavior

To investigate the layer-by-layer behavior of gates, we compute the average number of refinement loops triggered per layer.

As shown in Figure 5, the gates exhibit a con-

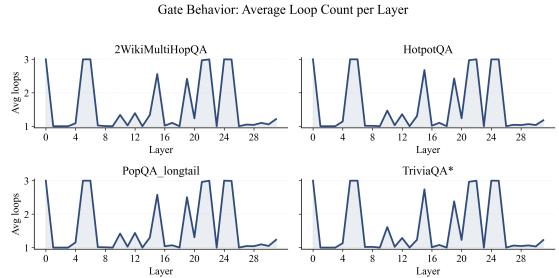


Figure 5: Gate behavior analysis across layers. The figure shows the token-level average number of refinement loops triggered by the gate at each layer evaluated on four datasets.

sistent pattern across all four datasets: most layers operate with a single forward pass, while a few layers selectively activate additional refinement loops. This suggests that the gate allocates extra computation at specific stages of the network, rather than uniformly increasing depth across layers. Notably, higher loop counts are concentrated in the early and late layers, whereas the middle layers remain relatively stable, approaching a single forward pass. We hypothesize that early layers are crucial for embedding alignment, while later layers support semantic fusion, which benefits QA. More interpretable mechanisms remain an important direction for further study.

4.9 4-way Error Category Analysis

To better understand where ArcAligner improves, we partition questions into four categories based on whether the *Naive* and *StandardRAG* succeed: **TT**, **TF**, **FT**, and **FF**. The results are shown in Figure 6. We observe several consistent trends.

First, in the **TT** category, where both the naive prompt and StandardRAG succeed, ArcAligner achieves strong performance across all datasets. This indicates that ArcAligner does not introduce harmful behavior on simpler or well-supported questions. In the **FT** category, ArcAligner consistently outperforms *xRAG*, suggesting that its compression preserves useful information better than comparable approaches.

Notably, ArcAligner shows clear advantages in the **TF** category, which typically corresponds to cases where retrieved passages are insufficient or

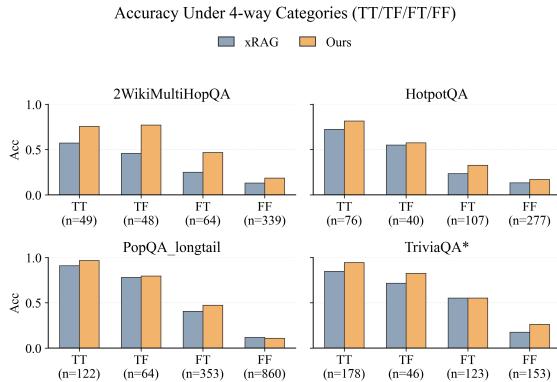


Figure 6: 4-way category analysis on four datasets. We partition questions into four groups based on whether *naive prompting* and *standard RAG* answer correctly: **TT**, **TF**, **FT**, and **FF**, where **T/F** denote correct/incorrect predictions. We report accuracy (*Acc*) of *xRAG* and ArcAligner within each category. Category size is shown under each label (e.g., **TT** with $n=41$).

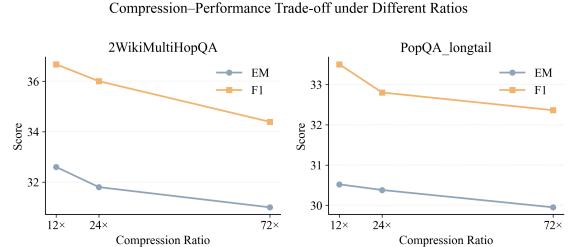


Figure 7: Performance under different compression ratios on 2WikiMultiHopQA and PopQA_longtail. EM/F1 scores are used as evaluation metrics.

contain misleading signals. Compared with *xRAG*, ArcAligner produces a much higher proportion of correct responses, indicating that it can better compensate for imperfect retrieval rather than merely relying on retrieved content. The **FF** category is the most challenging. ArcAligner still yields a small number of correct cases, which we attribute to the later training stages that encourage the LLM to attend more effectively to informative signals in the compressed context.

Overall, benefiting from adaptive recursion and three-stage training, ArcAligner improves performance and exhibits robustness for RAG tasks.

4.10 Effect of Compression Ratio

We further analyse the impact of different compression ratios on ArcAligner. For each ratio, we train a separate model with the same training settings, and vary only the sentence segmentation granularity that controls the ratio: $12\times$ is obtained by splitting each sentence into two segments, whereas $72\times$ is obtained by merging three consecutive sentences into one, with all other settings fixed.

As shown in Figure 7, at a higher compression ratio of $72\times$, performance decreased slightly, but not significantly. This indicates that the compressed representation retains the key evidence required for multi-hop inference. Meanwhile, we found some performance improvement as we reduce the compression ratio, suggesting that a smaller compression ratio can retain more key information and mitigate excessive information loss. This suggests that in practice, we need to flexibly adjust the compression ratio to achieve a trade-off between performance and efficiency.

5 Conclusion

In this paper, we propose ArcAligner, a parameter-efficient framework that enhances the usability of compressed context embeddings for RAG. By in-

troducing an adaptive recursive alignment mechanism with gating, ArcAligner ensures deep semantic alignment between compressed retrieval signals and the language model’s internal representations. Extensive experiments on knowledge-intensive QA benchmarks show that ArcAligner consistently outperforms existing compression baselines, especially in multi-hop and long-tail QA tasks, making it a valuable tool for efficient RAG systems.

Limitations

We discuss the limitations of ArcAligner as follows. Our current setup assumes a fixed context slot interface and a specific compression pipeline, and we do not systematically study how design choices such as passage segmentation, compression ratio, or slot budget affect the accuracy–efficiency trade-off. We also focus on a constrained retrieval setting and do not fully explore Top- K multi-document evidence, where complementary information across documents can be crucial; how to allocate slots and refinement budget across multiple retrieved passages remains open. Finally, ArcAligner relies on hard, slot-wise gating with a maximum recursion depth T at selected layers; while efficient in typical cases, the gate placement and T are tuned heuristically and may be suboptimal under strict compute/latency constraints, motivating future work on budget-aware gating or adaptive stopping.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Akari Asai, Zequi Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection.
- Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.
- Shaoshen Chen, Yangning Li, Zishan Xu, Yongqin Zeng, Shunlong Wu, Xinshuo Hu, Zifei Shan, Xin Su, Jiwei Tang, Yinghui Li, and 1 others. 2025. Dast: Context-aware compression in llms via dynamic allocation of soft tokens. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20544–20552.
- Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. xrag: Extreme context compression for retrieval-augmented generation with one token. *Advances in Neural Information Processing Systems*, 37:109487–109516.
- Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Xinting Huang, Dong Yu, and Zhicheng Dou. 2025. A silver bullet or a compromise for full attention? a comprehensive study of gist token-based context compression. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4861–4879.
- Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Xilin Wei, Songyang Zhang, Haodong Duan, Maosong Cao, and 1 others. 2024. Internlm-xcomposer2: Mastering free-form text-image composition and comprehension in vision-language large model. *arXiv preprint arXiv:2401.16420*.
- Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.
- Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

- Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pages 874–880.
- Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. *Mistral 7b. Preprint*, arXiv:2310.06825.
- Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376.
- Yi Jiang, Lei Shen, Lujie Niu, Sendong Zhao, Wenbo Su, and Bo Zheng. 2025a. Qagent: A modular search agent with interactive query understanding. *arXiv preprint arXiv:2510.08383*.
- Yi Jiang, Sendong Zhao, Jianbo Li, Haochun Wang, and Bing Qin. 2025b. *GainRAG: Preference alignment in retrieval-augmented generation through gain signal synthesis*. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10746–10757, Vienna, Austria. Association for Computational Linguistics.
- Yi Jiang, Sendong Zhao, Jianbo Li, Haochun Wang, Lizhe Zhang, Yan Liu, and Bing Qin. 2025c. Co-coa: Collaborative chain-of-agents for parametric-retrieved knowledge synergy. *arXiv preprint arXiv:2508.01696*.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.
- Jaehyun Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang Woo Lee, Minjoon Seo, Jung Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms. In *12th International Conference on Learning Representations, ICLR 2024*.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.
- Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.
- Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. *Compressing context to enhance inference efficiency of large language models*. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, Singapore. Association for Computational Linguistics.
- Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, and 1 others. 2025. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822.
- Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. Sfrembedding-mistral: enhance text retrieval with transfer learning. *Salesforce AI Research Blog*, 3:6.
- Alliot Nagle, Adway Girish, Marco Bondaschi, Michael Gastpar, Ashok Vardhan Makkuva, and Hyeji Kim. 2024. Fundamental limits of prompt compression: A rate-distortion framework for black-box language models. *Advances in Neural Information Processing Systems*, 37:94934–94970.
- Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, and 1 others. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *ACL (Findings)*.
- Hippolyte Pilchen, Edouard Grave, and Patrick Pérez. 2025. Arc-encoder: learning compressed text representations for large language models. *arXiv preprint arXiv:2510.20535*.

Ronak Pradeep, Sahel Sharifmoghaddam, and Jimmy Lin. 2023. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.

David Rau, Shuai Wang, Hervé Déjean, Stéphane Clinchant, and Jaap Kamps. 2025. Context embeddings for efficient answer generation in retrieval-augmented generation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, WSDM ’25, page 493–502, New York, NY, USA. Association for Computing Machinery.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengan Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

A Datasets

Here, we introduce in detail the datasets we used, which are seven datasets on four tasks.

2WikiMultiHopQA (Ho et al., 2020) and **HotpotQA** (Yang et al., 2018): Both are multi-hop question answering benchmarks constructed from Wikipedia articles. To control experimental cost, we follow prior work (Trivedi et al., 2023; Kim et al., 2024) and evaluate on the released subsampled splits, each containing 500 questions drawn from the original validation set.

NaturalQA (Kwiatkowski et al., 2019): A large-scale benchmark designed to support comprehen-

sive question answering. The questions are collected from real Google search queries, and the answers are annotated as text spans from corresponding Wikipedia articles by human annotators.

PopQA-longtail (Mallen et al., 2023): An open-domain QA dataset targeting long-tail factual knowledge. The questions are entity-centric and automatically generated from Wikidata knowledge triples using relation-specific templates. Entity popularity is quantified via Wikipedia page views, enabling controlled evaluation on low-popularity (long-tail) facts.

TriviaQA* (Joshi et al., 2017): A collection of open-domain trivia questions paired with answer annotations, originally sourced from online trivia websites. We use *TriviaQA** to denote a subsampled evaluation set of 500 randomly selected questions.

WebQuestions (Berant et al., 2013): A factoid question answering dataset constructed from real user queries issued to the Google Suggest API, where answers correspond to specific entities in Freebase.

Dataset statistics are summarized in Table 4

Task Type	Datasets	# Samples
Multi-HopQA	2WikiMultiHopQA	500
	HotpotQA	500
OpenQA	NaturalQA	3610
	PopQA_longtail	1399
	TriviaQA*	500
	WebQuestions	2032

Table 4: Description of tasks and evaluation datasets.

B Evaluation

Evaluation Metrics. We report three metrics: Exact Match (EM), F1, and Accuracy (Acc).

For EM and F1, we follow the standard evaluation protocol used in prior RAG work. Exact Match measures whether the model’s prediction contains the gold answer. Following Self-RAG (Asai et al., 2024) and When Not to Trust LMs (Mallen et al., 2023), we adopt a *non-strict* EM metric, where a prediction is considered correct if it includes the gold answer, rather than requiring an exact string match. F1 measures the token-level overlap between the predicted answer and the gold answer.

As noted in prior work, longer responses may artificially improve EM due to higher matching probability, while often reducing F1 due to the

Hyperparameter	Assignment
optimizer	AdamW
learning rate	2.0e-4
lr scheduler type	linear
warmup ratio	0.03
weight decay	0.0
LoRA r	128
LoRA alpha	32
LoRA dropout	0.05
max loops	3
loop layers	all
epochs	1
batch size	8
gradient accumulation steps	8
num GPUs	4
max train samples	200,000

Table 5: Hyperparameters for Stage I.

inclusion of irrelevant content. Therefore, EM and F1 provide complementary perspectives on answer quality.

In addition, we report Accuracy (Acc) based on large language model judgment. Specifically, we use **Qwen3-30B-A3B-Instruct-2507** (Yang et al., 2025) as an evaluator to assess whether the generated answer correctly addresses the question given the reference answer. The evaluator is prompted to produce a binary correctness judgment, which is then averaged over the dataset to compute Acc.

C Implementation Details

C.1 Hyperparameters

For all experiments, we adopt instruction-tuned variants of the base language models. All training and evaluation are conducted on a cluster of **4 NVIDIA H20-3e GPUs**.

Unless otherwise specified, all models are trained using the AdamW optimizer with linear learning rate scheduling. We report the detailed hyperparameter configurations for different training stages in Tables 5 and 6.

C.2 Prompts for Stage I

The list of instructions for paraphrase pretraining is shown in Table 7. They present the same meaning with natural language variance. Following prior work on embedding-based context compression, these paraphrase-style instructions are adapted from the instruction templates used in

Hyperparameter	Assignment
optimizer	AdamW
learning rate	2.0e-5
lr scheduler type	linear
warmup ratio	0.03
weight decay	0.0
LoRA r	128
LoRA alpha	32
LoRA dropout	0.05
max loops	3
loop layers	all
epochs	1
batch size	8
gradient accumulation steps	2
num GPUs	4
max train samples	90,447

Table 6: Hyperparameters for Stage II & III.

xRAG (Cheng et al., 2024).

C.3 Prompt for Stage II & III

During task fine-tuning, we represent the compressed background document using a short instruction line composed of repeated background placeholders. This line is inserted at a designated location in the prompt and marks the positions of the injected compressed segments:

*Refer to the background document: [B]
[B] ... [B]
Question: [Q]*

The number of background placeholders is set to $N = \max(1, |\text{sentences}|)$, and the instruction line is instantiated with N copies of [B].

D Algorithm

This section provides the detailed pseudocode of the ArcAligner forward computation, corresponding to the description in Section 3. The algorithm specifies how gated recursive updates are applied at selected layers and context-slot positions during both training and inference, as detailed in Algorithm 1.

E Case Studies

We present several representative cases to further analyze the behavior of our method (Figures 8–10).

Instruction intent	Template (with placeholders)
Equivalence statement	Background: $[B]$. This is equivalent to: $[T]$.
Direct rewrite	Rewrite the background in your own words: $[B] \rightarrow [T]$.
Restatement request	Provide a restatement of the background: $[B]$. Return: $[T]$.
Paraphrase-as-question	$[B]$ is a paraphrase of what? Answer with: $[T]$.
Two-formulation alignment	These two expressions convey the same meaning: (1) $[B]$ (2) $[T]$.
Minimal-output constraint	Restate $[B]$ using a single sentence. Output only $[T]$.

Table 7: Stage-I instruction templates. $[B]$ and $[T]$ denote the compressed background representation and the target textual reconstruction, respectively. We sample from multiple intent categories to diversify supervision while keeping a consistent I/O format.

Notation. We use traj to visualize the per-slot recursion depth determined by the gate. A token L corresponds to one recursion step, and the number of repeated L 's matches the loop count reported in loops . Thus, $L0.$ means the default single pass (loop count = 1), whereas LLL means three recursive passes (loop count = 3).

Algorithm 1 ArcAligner Forward

Require: Initial states $H^{(0,0)}$, blocks $\{\mathcal{A}^{(\ell)}\}_{\ell=0}^{N-1}$, gated layers $\mathcal{L}_{\text{gate}}$, max recursion T , mode $\in \{\text{train}, \text{infer}\}$

Ensure: Final states $H^{(N,0)}$

$H|_r$: context-slot positions; $H|\bar{r}$: non-slot positions

```

1: for  $\ell = 0$  to  $N - 1$  do
2:    $H^{(\ell+1,0)} \leftarrow \mathcal{A}^{(\ell)}(H^{(\ell,0)})$ 
3:   if  $\ell \in \mathcal{L}_{\text{gate}}$  then
4:     for  $t = 1$  to  $T$  do
5:        $g \leftarrow \sigma(\text{MLP}^{(\ell)}(H^{(\ell+1,t-1)}|_r))$ 
6:       if mode = train then
7:          $g \leftarrow g + \text{stopgrad}(\mathbb{I}[g \geq 0.5] - g)$ 
8:       else
9:          $g \leftarrow \mathbb{I}[g \geq 0.5]$ 
10:      end if
11:       $\tilde{H} \leftarrow \mathcal{A}^{(\ell)}(H^{(\ell+1,t-1)})$ 
12:       $H^{(\ell+1,t)}|_r \leftarrow H^{(\ell+1,t-1)}|_r + g \odot (\tilde{H}|_r - H^{(\ell+1,t-1)}|_r)$ 
13:       $H^{(\ell+1,t)}|\bar{r} \leftarrow H^{(\ell+1,0)}|\bar{r}$ 
14:    end for
15:     $H^{(\ell+1,0)} \leftarrow H^{(\ell+1,T)}$ 
16:  end if
17: end for
18: return  $H^{(N,0)}$ 

```

Question. In what country is Toronto Northwest?

Compressed Sentence Passages.

1. was redistributed between Davenport, Spadina, Trinity and York West ridings.
2. **Toronto Northwest was a federal electoral district represented in the House of Commons of Canada from 1925 to 1935.**
3. It was located in the city of Toronto in the province of Ontario.
4. This riding was created in 1924 from parts of Parkdale, Toronto North and York South ridings.
5. It consisted of the part of the city of Toronto north of Bloor Street, west of Bathurst St. and east of the Northern Division of the Canadian National Railway.

Answer. Canada.

Routing Behavior. At layer 31:

$$\text{traj} = [\text{L0.}, \text{LLL}, \text{L0.}, \text{L0.}, \text{L0.}]$$

The answer evidence appears in the **second sentence**, which is repeatedly updated via recursive routing (LLL).

Figure 8: Case 1: the answer evidence appears in the second sentence, which is repeatedly updated via recursive routing.

Question. What sport does Radik Zharparov play?

Compressed Sentence Passages.

1. **Radik Zharparov (born February 29, 1984) is a Kazakh ski jumper who has competed since 2003.**
2. At the 2006 Winter Olympics in Turin, he finished 11th in the team large hill and 26th in the individual normal hill events.
3. At the FIS Nordic World Ski Championships, Zharparov has finished 11th in team events three times and 24th in the individual normal hill events.
4. Zharparov's best individual World Cup finish was 11th in a large hill event in Finland in 2007.
5. His best individual career finish was second in an FIS Cup normal.

Answer. ski jumping.

Routing Behavior. At layer 31:

$$\text{traj} = [\text{LLL}, \text{L0.}, \text{L0.}, \text{L0.}, \text{L0.}]$$

The recursive routing (LLL, loop count = 3) is assigned to the first slot, which corresponds to the **first sentence** containing the explicit evidence.

Figure 9: Case 2: the answer evidence appears in the first sentence, which receives repeated recursive updates (LLL) at layer 31.

Question. What nationality is the director of film *Astronauts Gone Wild*?

Compressed Sentence Passages.

1. ***Astronauts Gone Wild* is a 2004 documentary video produced and directed by Bart Sibrel, a Nashville, Tennessee-based video maker.**
2. Sibrel made this video as a follow-up to his 2001 video *A Funny Thing Happened on the Way to the Moon*.
3. The title of the presentation is a wordplay on the *Girls Gone Wild* video series.
4. In *Astronauts Gone Wild*, Sibrel confronts nine Apollo ...

Answer. American.

Routing Behavior. At layer 14:

$$\text{traj} = [\text{LLL}, \text{L0.}, \text{L0.}, \text{L0.}]$$

The recursive routing (LLL, loop count = 3) is assigned to the first slot, which aligns with the **first sentence** describing the director and location cues used to infer nationality.

Figure 10: Case 3: the director evidence appears in the first sentence, which receives repeated recursive updates (LLL) at layer 14.