Accelerating Adaptive Retrieval Augmented Generation via Instruction-Driven Representation Reduction of Retrieval Overlaps

Jie Ou[1], **Jinyu Guo**[1*], **Shuaihong Jiang**[1], **Zhaokun Wang**[1],
Libo Qin[2], **Shunyu Yao**[3], **Wenhong Tian**[1]
[1] School of Information and Software Engineering,
University of Electronic Science and Technology of China
[2] Central South University
[3] Big data and artificial intelligent institute, China Telecom Research Institute

Abstract

Retrieval-augmented generation (RAG) has emerged as a pivotal method for expanding the knowledge of large language models. To handle complex queries more effectively, researchers developed Adaptive-RAG (A-RAG) to enhance the generated quality through multiple interactions with external knowledge bases. Despite its effectiveness, A-RAG exacerbates the pre-existing efficiency challenges inherent in RAG, which are attributable to its reliance on multiple iterations of generation. Existing A-RAG approaches process all retrieved contents from scratch. However, they ignore the situation where there is a significant overlap in the content of the retrieval results across rounds. The overlapping content is redundantly represented, which leads to a large proportion of repeated computations, thus affecting the overall efficiency. To address this issue, this paper introduces a model-agnostic approach that can be generally applied to A-RAG methods, which is dedicated to reducing the redundant representation process caused by the overlapping of retrieval results. Specifically, we use cache access and parallel generation to speed up the prefilling and decoding stages respectively. Additionally, we also propose an instruction-driven module to further guide the model to more effectively attend to each part of the content in a more suitable way for LLMs. Experiments show that our approach achieves 2.79 and 2.33 times significant acceleration on average for prefilling and decoding respectively while maintaining equal generation quality.

1 Introduction

Work smarter, not harder.

Allan F. Mogensen, 1930s

Large Language Models (LLMs) (e.g., LLaMA-2, PaLM, and GPT-4 (Touvron et al., 2023; Anil et al., 2023; Achiam et al., 2023)) have demonstrated remarkable capabilities across various natural language processing tasks. To address the growing demand for knowledge-intensive and factually grounded responses, Retrieval-Augmented Generation (RAG) has emerged as a promising paradigm to mitigate the inherent knowledge limitations of LLMs (Lewis et al., 2020; Borgeaud et al., 2022; Ram et al., 2023; Jiang et al., 2023; Asai et al.; Gao et al., 2023; Gupta et al., 2024). It enhances the performance by retrieving relevant information from external knowledge base to generate contextually appropriate and evidence-based responses.

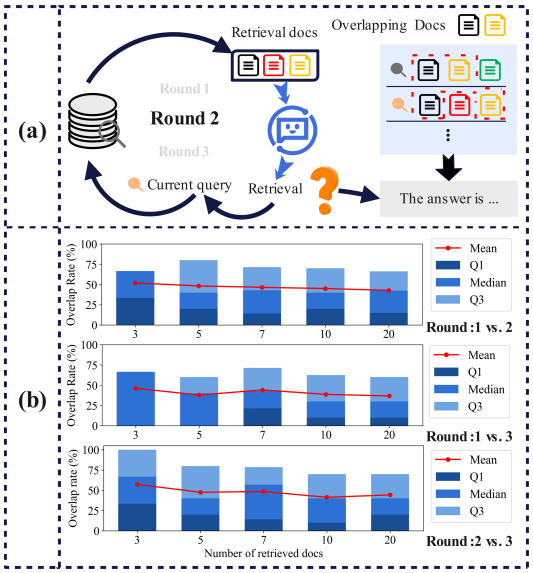


Figure 1: (a) The pipeline of A-RAG. (b) Analysis of document overlap between first and later retrievals (rounds 2-3) using 1000 2WikiMultihopQA samples, where Q1, Median, and Q3 represent the 25th, 50th, and 75th percentiles of document overlap ratios respectively.

Conventional RAG interacts with the external knowledge bases once and combines the retrieved documents with the query as input to LLMs, this can also be called single-round RAG. Nevertheless, single-round RAG encounters challenges in han-

*Corresponding author, Email: guojinyu@uestc.edu.cn
Our code: https://github.com/oujieww/idrfast

Findings of the Association for Computational Linguistics: ACL 2025, pages 26983–27000

dling complex user inquiries, as it is difficult to extract sufficient information by a single interaction. Consequently, an increasing number of researchers have begun to focus on and propose Adaptive-RAG (A-RAG) (Borgeaud et al., 2022; Ram et al., 2023; Mallen et al., 2023; Asai et al.; Trivedi et al., 2023; Jiang et al., 2023; Zhao et al., 2023; Yang et al., 2023b; Su et al., 2024; Zhang et al., 2024; Li et al.; Yao et al., 2024; Jeong et al., 2024; Wang et al., 2024a). Figure 1(a) illustrates the process of A-RAG, which dynamically determines whether to continue retrieval and adjusts the retrieval content based on the quality of the generated responses. Through iterative interactions with external knowledge bases, A-RAG obtains more comprehensive and valuable information to generate accurate and comprehensive answers.

Although A-RAG enhances the performance, it further exacerbates the inherent efficiency problems of the RAG due to external interactions and increased computation. Within its post-retrieval generation phase, existing methods process all retrieved contents from scratch. However, they overlooked a situation where, during a single A-RAG process, the high similarity among multiple rounds of queries results in significant overlap in the content of the retrieved results across these rounds, especially between adjacent rounds. Figure 1(b) shows the overlap ratio of documents between rounds with different settings for the number of retrieved documents. These overlapping contents are redundantly represented in each round, which leads to a large proportion of repeated computations, thereby affecting the overall efficiency.

To tackle this issue, we propose **I**nstruction-**D**riven **R**epresentation **R**eduction (**IDR**$_2$), a model-agnostic approach widely applicable to the A-RAG methods. It aims to improve the efficiency of A-RAG by eliminating repetitive representations of overlapping content and redundant autoregressive representation rounds. Concretely, we leverage representation reduction techniques in both the prefilling and decoding processes of the generation phase. During prefilling, we propose the Cross-Iteration Cache Sharing (CICS) module, which establishes a shared memory repository for document representations. It enables subsequent processing iterations to bypass repetitive computations through cached intermediate results, thereby reducing computational overhead for overlapping content. In addition, to further direct the model to more effectively attend to the various parts of the content after prefilling, we introduce the Instruction-driven Deduplication Guidance Reinforcement (IDGR) module. It leverages the instruction-following capabilities of LLMs to implement context-aware filtration, prioritizing semantically relevant cached information while suppressing redundant content through explicit linguistic guidance in a more suitable way for LLMs. In the decoding process, we propose a novel Information-Guided Parallel Generation (IGPG) module, which leverages the correlation between retrieved documents and the generated results. By integrating phrasal fragments as inputs at each autoregressive step, IGPG enables parallel generation. It reduces the autoregressive representation rounds to achieve acceleration.

We extensively evaluate our proposed method on multiple datasets. The experimental results show that our IDR$_2$ significantly reduces the representation process, which achieves 2.79 and 2.33 times acceleration for prefilling and decoding, consequently accelerating the entire A-RAG workflow by 2.0 times on average across various A-RAG approaches while maintaining the performance of generation. In summary, our contributions are mainly three-fold:

1. We propose IDR$_2$, an acceleration approach for A-RAG based on representation reduction techniques. It eliminates repetitive representations during the prefilling process and reduces autoregressive representation redundancy in the decoding phase, thereby speeding up the entire A-RAG workflow almost without performance loss.

2. We develop the Instruction-driven Deduplication Guidance Reinforcement module, which leverages the instruction-following capabilities of LLMs to further direct the model to more effectively attend to the various parts of the content after prefilling in a more suitable way for LLMs.

3. Experimental results demonstrate that our approach significantly enhances the efficiency of various A-RAG methods while exhibiting robust adaptability across diverse scenarios and various LLM scales.

2 Related Works

2.1 Adaptive-RAG

Generating satisfactory answers by single-round RAG remains challenging (Komeili, 2021; Zhu

et al., 2021; Liu et al., 2024; Jiang et al., 2023; Yang et al., 2023b; Ni et al., 2024). A-RAG was developed to address this limitation by enabling iterative interactions with external knowledge bases, leading to more accurate and complete answers through multiple retrieval rounds (Jiang et al., 2023; Yang et al., 2023b; Ni et al., 2024). Current A-RAG approaches can be classified into two categories: 1) Rule-based strategies, such as per-sentence iteration (Trivedi et al., 2023), sliding window tokens (Borgeaud et al., 2022; Ram et al., 2023), and contextual learning (Zhao et al., 2023; Zhang et al., 2024; Li et al.). 2) Self-perception strategies evaluate output confidence through internal states (Yao et al., 2024), LLM output layer (Jiang et al., 2023; Yang et al., 2023b; Su et al., 2024), or explicit language-level feedback (Asai et al.). In contrast to existing works that primarily focus on performance enhancement, to the best of our knowledge, we are the first to investigate the multi-turn document overlapping problem in A-RAG and improve efficiency through its resolution.

2.2 Inference Acceleration

Efficiency of LLM. Research on improving LLM efficiency can generally be classified into two categories: traditional optimization techniques (such as quantization, pruning, knowledge distillation, etc.) and LLM-specific optimizations which include: 1) Early Exiting (Teerapittayanon et al., 2016; Xin et al., 2020; Zhou et al., 2020; Kong et al., 2022; Yang et al.; Bae et al., 2023), uses prediction heads at different layers to enable early token exit when confidence thresholds are met. 2) Token Pruning (Goyal et al., 2020; Kim and Cho, 2021; Wang et al., 2021; Kim et al., 2022; Hou et al., 2022; Zhang et al., 2023b), retains only critical tokens based on importance ranking.3) Speculative Decoding (Chen et al., 2023; Leviathan et al., 2023; Spector and Re; Yang et al., 2023a; Zhang et al., 2023a; Kim et al., 2024; Miao et al., 2024), uses efficient smaller models to generate candidate tokens for batch verification by LLM.

Efficiency of RAG. Speculative retrieval (Zhang et al.) reduces retrieval overhead through local retrieval and verification mechanisms. Parallel document processing (Merth et al.) mitigates attention complexity by processing multiple documents independently rather than concatenating them. Document preprocessing (Lu et al., 2024) improves prefilling efficiency by preprocessing and storing document representation for the whole knowledge base. Small expert LMs generate initial document-specific drafts for LLM verification (Wang et al., 2024b). In contrast to prior work, our IDR_2 is a general framework. It enhances A-RAG efficiency by representation reduction with instruction-guided information extraction.

3 Methods

Figure 2 details the workflow of IDR_2, and we divide the internal process of each iteration round in A-RAG into three phases: retrieval, prefilling, and decoding. When processing a query, cached document representations in the CICS module are first verified. These representations are loaded and combined with uncached documents for prefilling, where the IDGR module guides LLMs to prioritize relevant content while filtering noise. Before each autoregressive step, matching subsequent phrase fragments are queried in the IGPG module to enable parallel generation. Ultimately, these approaches enhance the overall efficiency of A-RAG. Details of each module are given respectively in the remainder of this section.

3.1 Cross-Iterative Cache Sharing (CICS)

A-RAG (Zhao et al., 2023; Jiang et al., 2023; Yang et al., 2023b; Su et al., 2024; Zhang et al., 2024) relies on multiple rounds of retrieval-generation interactions to generate answers. We observe substantial document overlap between adjacent retrieval rounds. In light of this, duplicate representation for overlapping documents introduces computational redundancy, hampering the efficiency of A-RAG.

In order to avoid duplicate representation for overlapping documents, we propose the cross-iterative cache sharing (CICS) module. CICS initializes a cache space $\mathbb{C}$ to store Key-Value pairs corresponding to the documents retrieved in each round for each query q. Because A-RAG may modify the query at the end of each iteration, we denote the initial user query as q_0. At round t, the retrieval operation $D_t = \textbf{Retrieve}(q_t)$ returns a set of n documents $D_t = \{d_1^t, d_2^t, ..., d_n^t\}$ from the external knowledge base. n denotes the maximum number of documents that can be retrieved in each round. At round t, the LLM generates a sequence of tokens $A_t = \{a_t^1, a_t^2, ..., a_t^m\}$ by:

$$a_t^1, K_t, V_t = \textbf{LLM}_P(q_t, D_t, A_{<t}) \quad (1)$$

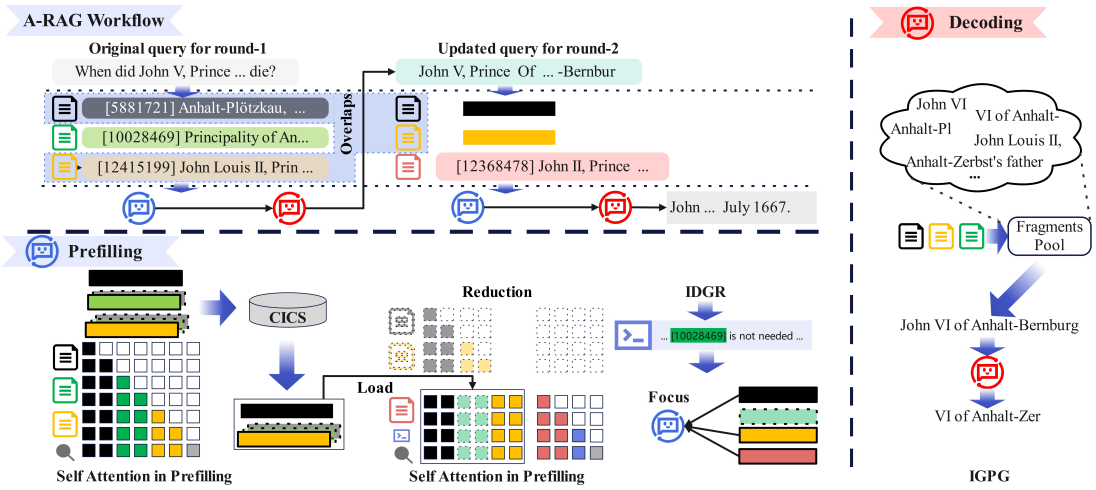


Figure 2: The pipeline of our IDR$_2$. The same color indicates the same document and representation.

$$
\begin{aligned}
a_t^i, k_t^i, v_t^i &= \textbf{LLM}_D(q_t, D_t, A_{<t}, a_t^{<i}, K_t, V_t) \\
K_t &= \textbf{Concat}(K_t, k_t^i) \\
V_t &= \textbf{Concat}(V_t, v_t^i)
\end{aligned} \tag{2}
$$

where the inference process of LLM is divided into two stages: prefilling and decoding, represented with $\textbf{LLM}_P$ and $\textbf{LLM}_D$ respectively. The K_t and V_t represent the Key-Value pairs for D_t and A_t. The a_t^i denotes the i^{th} generated token and m represents the number of generated tokens. The Eq.(2) denoted the autoregression process, which can only generate one token at each step. It needs to be executed through multiple steps to obtain the complete A_t. CICS stores the K_t, V_t from Eq.(1), which is the representation for D_t as shown in Figure 2.

After receiving retrieved results D_t, CICS extracts existing representation from the cache to avoid duplicate representation. Specifically, as the example shown in Figure 2, CICS directly loads the representation of documents #5881721 and #12415199, then integrates them with the text of document #12368478 for generation. This process can be formalized as:

$$
\begin{aligned}
K_t^o, V_t^o, D_t^o &= \textbf{Filter}(D_t, \mathbb{C}) \\
a_t^1, K_t, V_t &= \textbf{LLM}_P(q_t, D_t^n, A_{<t}, K_t^o, V_t^o)
\end{aligned} \tag{3}
$$

where D_t^o is the document set that has been processed in the previous round and appears again at the current round D_t. We can directly obtain its corresponding representation K_t^o and V_t^o from $\mathbb{C}$ without re-processing. For the new document set $D_t^n = D_t \setminus D_t^o$ at the current round, the prefilling is still required.

Eq.(3) optimizes the prefilling phase by reusing the cached representation of overlapping documents in CICS, avoiding redundant computation.

3.2 Instruction-driven Deduplication Guidance Reinforcement (IDGR)

The CICS provides efficient representation reuse across rounds. Notably, the Key-Value representation of each document d_i^t incorporates information from previously processed documents through self-attention. As an example illustrated in Figure 2, at round 1, the representation of document #12415199 contains information from documents #5881721 and #10028469 due to the self-attention mechanism.

The IDGR employs natural language instructions I_t in the prompt to guide the LLM in filtering redundant cached information. This module helps the model focus on relevant content for the current round, thereby ensuring high-quality generation. At each iteration, the instruction I_t is automatically generated based on two rules and is subsequently utilized during the prefilling phase, as:

$$
a_t^1, K_t, V_t = \textbf{LLM}_P(q_t, D_t^n, A_{<t}, R_t^o, I_t) \tag{4}
$$

where R_t^o is the K_t^o, V_t^o. As shown in Figure 2, the representation of document #10028469 is excluded from the computation. Furthermore, explicit instructions guide the LLM to ignore the information contained in the representation of document #12415199.

The instruction I_t is generated according to two rules: First, we use document identifiers to explicitly tell the LLM which documents are relevant and which are irrelevant for the current generation. This helps the LLM focus on pertinent context while filtering out redundant information. Second, we provide document relevance rankings to the LLM. In cases where we cannot directly adjust the input

order of documents based on their importance, explicit instructions guide the LLM to prioritize more relevant documents.

In practice, I_t can be implemented with simple natural language directives, leveraging the strong semantic understanding capabilities of LLMs. For instance, in the second round shown in Figure 2, the instruction is formulated as: "*#5881721 ... are related docs. #10028469 is unrelated. The relevance scores are ...*". The relevance scores can be obtained through either retriever rankings[1] or model-based scoring methods[2]. While scoring methods are not the focus of our study, they can be seamlessly integrated into our approach.

3.3 Information-Guided Parallel Generation (IGPG)

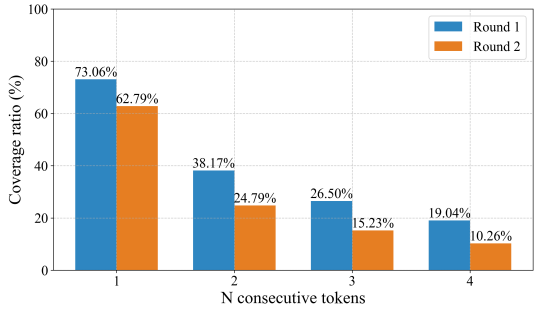


Figure 3: The x-axis represents the length of consecutive token combinations in the generated results. The y-axis represents the proportion of all combinations in the generated results that appear in the retrieved results. (LLaMA2-7B, 2WikiMultihopQA).

RAG differs from standard LLM applications in its access to extensive external context. For a given query q, standard LLM inference can only utilize q itself, whereas RAG leverages both q and retrieved documents D_t. Furthermore, in different iterations of A-RAG, the LLM-generated content maintains high relevance to the retrieved documents D_t, as illustrated in Figure 3. This relevance plays a crucial role in determining the current round's output. The LLM generates a significant amount of existing content. The autoregressive process of token-by-token generation primarily contributes to the low efficiency. In contrast to standard LLM generation, we have the information of D_t, which contains a substantial portion of the content that LLM must generate. Consequently, we can utilize the information from D_t to guide LLM in avoiding the independent autoregressive representation process for existing content. By constructing phrase fragments and verifying multiple tokens during autoregression, parallel generation can be achieved.

Different with traditional speculative decoding approaches, IGPG requires neither the construction of small language models nor training. During each RAG iteration, IGPG uses D_t to construct a approximate probabilistic language model $P(x_t|x_1, ..., x_{t-1}) \approx P(x_t|x_{t-N+1}, ..., x_{t-1})$, N is the number of recent tokens.

Our IGPG has two steps as speculative decoding approaches: draft generation and LLM parallel generation. In the draft generation, by iterating M steps, the module constructs a M-length draft token sequence $\hat{A}_{t,k} = \{\hat{a}^1_{t,k}, \hat{a}^2_{t,k}, ..., \hat{a}^M_{t,k}\}$, k denotes the k-th autoregression step. During LLM parallel generation, the LLM validates the draft token sequence through a forward pass by $P(\bar{a}^{i+M+1}_{t,k}|a^{\leq i}_{t,k-1}, \hat{a}^1_{t,k}, ..., \hat{a}^M_{t,k})$. The LLM examines whether each draft token aligns with the current output, as Eq.(5). When a draft token $\hat{a}^j_{t,k}$ fails validation, it is substituted with the LLM's prediction $\bar{a}^{j-1}_{t,k}$, and draft generation resumes from this point until the complete sequence A_t is produced.

$$a^j_{t,k} = \begin{cases} \hat{a}^j_t, & \text{if } \hat{a}^j_{t,k} = \bar{a}^{j-1}_{t,k}, j \geq 1 \\ \bar{a}^{j-1}_{t,k} & \text{otherwise \& stop} \end{cases} \quad (5)$$

where $\{\bar{a}^0_{t,k}, ..., \bar{a}^{i+M+1}_{t,k}\} = \textbf{LLM}_D(a^{\leq i}_{t,k-1}, \hat{A}_{t,k})$, $\bar{a}^0_{t,k}$ corresponds to $a^i_{t,k-1}$ autoregressive.

If at least one token in the $\hat{A}_{t,k}$ is validated by the LLM, combined with the token generated by the LLM itself, the total number of generated tokens is at least two, achieving a least $2\times$ speedup.

4 Experiments

4.1 Language Models

We employed the LLaMA2-7B/13B (Touvron et al., 2023) (L2-7B/13B) and the Vicuna-7B/13B (Chiang et al., 2023) (V-7B/13B). The Vicuna model is a chatbot fine-tuned on ShareGPT conversation data. It contains different knowledge from LLaMA and has an impact on A-RAG inference.

4.2 Downstream Task Datasets

We use the 2WikiMultihopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018), StrategyQA (Geva et al., 2021) and IIRC (Ferguson et al., 2020) as DRAGIN (Su et al., 2024). The datasets span multi-hop question answering, common sense reasoning, and reading comprehension, covering different

[1]https://github.com/Muennighoff/sgpt
[2]https://huggingface.co/BAAI/bge-reranker-base

Methods	LLMs	2WikiMultihopQA			HotpotQA			StrategyQA			IIRC		
		Pref. ↑	Deco. ↑	E2E ↑	Pref. ↑	Deco. ↑	E2E ↑	Pref. ↑	Deco. ↑	E2E ↑	Pref. ↑	Deco. ↑	E2E ↑
FLRAG+IDR$_2$	L2-7B	2.23×	2.37×	1.75×	2.27×	1.85×	1.53×	1.75×	1.49×	1.40×	3.08×	2.76×	2.10×
	L2-13B	2.12×	2.36×	1.64×	2.29×	2.00×	1.54×	1.76×	1.62×	1.40×	2.70×	2.65×	1.83×
	V-13B	2.05×	2.25×	1.60×	2.22×	1.97×	1.51×	1.65×	1.55×	1.31×	2.66×	2.72×	1.82×
FSRAG+IDR$_2$	L2-7B	2.04×	2.64×	2.32×	2.41×	1.88×	1.69×	2.36×	1.97×	1.79×	2.92×	2.88×	2.20×
	L2-13B	2.11×	2.47×	1.79×	2.16×	2.05×	1.59×	1.74×	1.59×	1.44×	2.16×	2.65×	1.74×
	V-13B	2.23×	2.31×	1.75×	2.15×	2.31×	1.66×	2.02×	1.78×	1.53×	2.35×	2.69×	1.92×
FLARE+IDR$_2$	L2-7B	2.81×	2.74×	2.60×	3.12×	**2.61**×	**2.31**×	2.71×	**2.27**×	1.77×	3.54×	2.97×	2.89×
	L2-13B	1.98×	2.24×	2.16×	2.42×	1.87×	1.80×	2.45×	1.71×	1.69×	3.58×	3.15×	3.01×
	V-13B	2.04×	2.14×	2.06×	2.27×	2.07×	1.98×	2.05×	1.93×	1.89×	2.70×	2.24×	2.22×
DRAGIN+IDR$_2$	L2-7B	3.34×	2.77×	2.52×	4.09×	2.08×	2.09×	**3.39**×	1.98×	**1.95**×	4.49×	2.58×	2.58×
	L2-13B	3.25×	**3.21**×	**2.61**×	3.97×	2.15×	2.01×	3.15×	1.98×	1.87×	4.25×	2.86×	2.65×
	V-13B	**3.57**×	2.81×	2.55×	**4.14**×	2.29×	2.15×	3.25×	1.88×	1.82×	**4.72**×	**4.00**×	**3.53**×

Table 1: The average speedup ratios of IDR$_2$ across different methods, models, and datasets. Pref. (prefilling) demonstrates acceleration achieved through CICS and IDGR. Deco. (decoding) shows speedup from IGPG. E2E (end-to-end) reflects the overall A-RAG acceleration, encompassing retrieval. (n = 3, 1× = baseline speed)

LLMs	Methods	2WQA		HQA		SQA	IIRC	
		EM ↑	F1 ↑	EM ↑	F1 ↑	Acc. ↑	EM ↑	F1 ↑
Llama2-7B	DRAGIN†	22.5	28.68	22.6	**33.02**	**65.10**	15.93	20.24
	DRAGIN+Ours	**25.4**	**33.17**	**22.9**	32.59	62.50	**16.04**	**20.24**
Llama2-13B	DRAGIN†	30.4	39.91	**31.6**	**42.60**	**66.10**	18.5	**22.59**
	DRAGIN+Ours	**34.4**	**41.50**	29.9	41.01	65.77	**18.55**	22.11
vicuna-13b	DRAGIN†	25.4	34.90	30.1	42.50	66.20	**23.90**	**28.11**
	DRAGIN+Ours	**28.9**	**36.77**	**31.7**	**42.58**	**68.00**	21.59	26.28

Table 2: The performance comparison on different datasets with different models. The † represents the reproduction.

types of reasoning tasks. For more detailed settings, please refer to the Appendix A.2.

4.3 Baselines

We selected four representative A-RAG methods as baselines. These methods are characterized by two key aspects: retrieval timing and query construction. Retrieval timing determines when to perform retrieval, while query construction decides what content to use for retrieval. These features directly impact both the quality and efficiency of A-RAG workflows. FL-RAG (Khandelwal et al., 2019; Borgeaud et al., 2022; Ram et al., 2023) retrieves every z tokens using previous tokens as queries, FS-RAG (Trivedi et al., 2023) performs retrieval after each sentence, FLARE (Jiang et al., 2023) triggers retrieval for uncertain tokens, using the last generated sentence as the query, and DRAGIN (Su et al., 2024) employs a dynamic approach based on content importance and uncertainty, utilizing attention distribution for query construction.

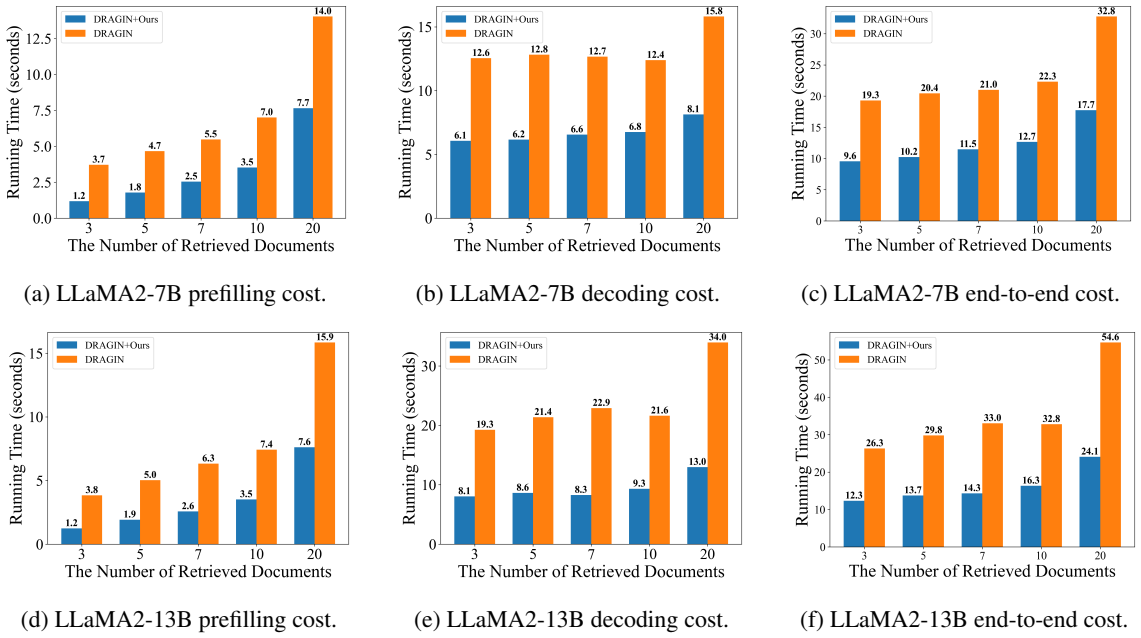
4.4 Implementation Details

For the retrieval modules, we select BM25 and follow DRAGIN (Su et al., 2024). Additionally, we also investigate replacing BM25 with SGPT (Muennighoff, 2022), a dense retrieval method. Our external knowledge is based on Wikipedia articles, which are divided into passages containing 100 tokens each. For the 7B and 13B models, we use four Nvidia 3090 GPUs and four H800 GPUs, respectively. n is the number of retrieved documents. Our IDR$_2$ based on the PyTorch (Paszke et al., 2019), details in the Appendix A.4.

4.5 Main Results

Table 1 shows the results of our approach applied to the different A-RAG methods, different models with scales, and different scenarios. Table 1 shows the acceleration ratios over baseline methods.

Overall, our method demonstrates end-to-end speed improvements ranging from 1.31× to 3.53×. During the prefilling phase, we observe acceleration factors from 1.75× to 4.72×. These results substantiate the efficacy of our proposed CICS and



(a) LLaMA2-7B prefilling cost. (b) LLaMA2-7B decoding cost. (c) LLaMA2-7B end-to-end cost.

(d) LLaMA2-13B prefilling cost. (e) LLaMA2-13B decoding cost. (f) LLaMA2-13B end-to-end cost.

Figure 4: The analysis of speedup for different numbers of retrieved documents.

IDGR mechanisms in effectively eliminating redundant representations. For the decoding phase, the IGPG technique achieves speedup ratios from 1.49× to 4.00× through the reduction of autoregressive inference iterations. A comparative analysis between the LLaMA2-7B and 13B architectures (underlining indicates better results) shows slightly better results for the 7B variant. We consider that this effect stems from the fact that smaller models require less additional overhead to store and load representations. Of the various A-RAG methods, the most significant performance improvement (4.72× for prefilling) is achieved when applying IDR_2 to DRAGIN. We attribute this to the query refinement mechanism of DRAGIN which improves query similarity, a property that effectively synergizes with our representation reduction approach.

Models	Methods	Pref. (s)↓	Deco. (s)↓	E2E (s)↓
L2-7B	DR.	3.71	12.55	19.31
	DR.+Ours	**1.18**	**6.07**	**9.56**
L2-13B	DR.	3.84	19.26	26.31
	DR.+Ours	**1.24**	**8.06**	**12.34**
V-13b	DR.	4.34	25.03	33.47
	DR.+Ours	**1.29**	**10.49**	**15.18**

Table 3: The runtime of IDR_2 on 2WikiMultihopQA dataset (n = 3), DR. denotes DRAGIN (Su et al., 2024).

The specific runtime of the different phases in the generation process are exhibited in Table 3. Our experiments reveal that even on high-end NVIDIA H800 GPUs, 13B-parameter models require 26.31-33.47 seconds to process a single request. The proposed method significantly reduces the overall latency to 9.56-15.18 seconds, demonstrating substantial performance improvements and enhanced practicality for real-world deployment.

Table 2 shows the performance of applying the IDR_2 to the DRAGIN method. The experimental results show that our approach effectively preserves the original performance while exhibiting strong adaptability and generalization across different models and tasks. Notably, our method also carries some enhancements on 2WikiMultiHopQA. We also provide performance results applied to other methods, in the Appendix A.10.

4.6 Ablation studies

4.6.1 Impact of Retrieval Size

We analyze the impact of varying numbers of retrieved documents on A-RAG, with results visualized in Figure 4. As the document count increases, the context length grows monotonically, resulting in progressively higher prefilling overhead proportion. Our approach maintains consistent speedup advantages across both prefilling and decoding.

4.6.2 Impact of Different Retrievers

Table 4 demonstrates the effectiveness of IDR_2 across different retrievers. Substituting the retriever with SGPT results in an EM score of 29.6 and a

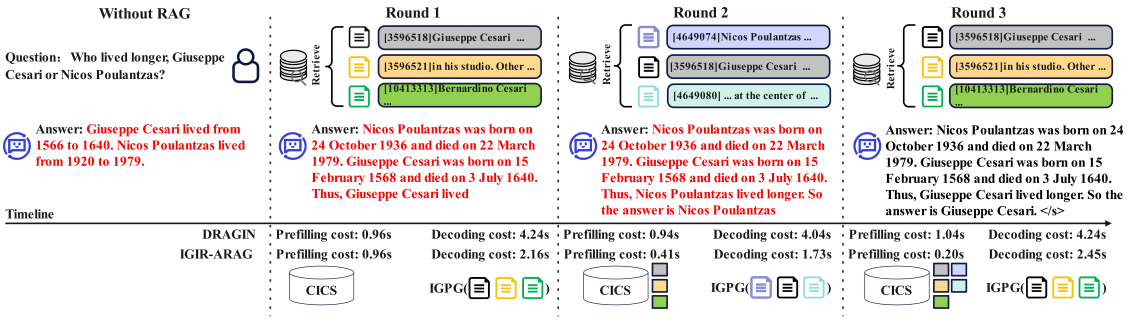


Figure 5: The detailed process of A-RAG based on a specific example, with LLaMA-13B.

Method	Retriever	EM ↑	E2E ↑
DRAGIN	BM25	30.4	1×
DRAGIN + IDR$_2$	BM25	34.4	2.61×
DRAGIN	SGPT	27.3	1×
DRAGIN +IDR$_2$	SGPT	29.6	2.73×

Table 4: Analysis of different retrievers on the 2WikiMultihopQA with LLaMA2-13B ($n = 3$).

speedup ratio of 2.73 ×. The consistent performance and acceleration above 2.6× across both frequency-based and semantic retrievers confirms the adaptability of IDR$_2$.

5 Analysis

5.1 Do these modules impact performance?

Table 5 shows how different modules affect the generation quality. The redundant information in the CICS module impacts the generation quality of LLM. This is evident in the performance of LLaMA2-7B, where EM dropped from 22.0 to 20.3, with LLaMA2-13B showing similar decreases. The introduction of IDGR not only alleviates the adverse effects of redundant information but also brings additional performance gains. The LLaMA2-7B achieves EM scores of 25.4, and LLaMA2-13B reaches 34.4. These results confirm that IDGR successfully maintains inference quality.

5.2 Specific Case Studies

Figure 5 shows three iterative rounds. In **Round 1** (empty CICS), baseline prefilling takes 0.96s. IGPG reduces decoding time from 4.24s to 2.16s. After hallucination detection, the process enters **Round 2**. IDR$_2$ reuses the cached representations of #359518 in CICS, saving 0.53s (prefilling). During the decoding process, IGPG builds fragment models, saving 2.31 seconds. For example, *Nicos Poulantzas* requires six token-level autoregression: *_N*, *icos*, *_P*, *oul*, *ant*, *zas*. After applying IGPG, only one autoregression is needed after the prefix "Nicos," is generated, achieving 4× local acceleration. Similarly, dates like *October 1979*, *September 1936*, *February 1568* achieve 4× acceleration by retrieving corresponding fragments from IGPG using month information. At **Round 3**, LLM further confirms *Giuseppe Cesari's* lifespan and generates the right answer. With all representations already in CICS, the prefilling time reduces by 0.804 seconds (about 80% of DRAGIN's original time), achieving 5× speedup. Total savings reach 1.334s (prefilling) and 6.16s (decoding), summing to 7.494s (48.47% reduction) over DRAGIN's 15.46s, achieving 1.95× end-to-end acceleration. A more detailed explanation can be viewed in the Appendix A.1.

CICS	IGPG	IDGR	LLaMA2-7B		LLaMA2-13B	
			EM ↑	F1 ↑	EM ↑	F1 ↑
-	-	-	22.5	28.68	30.4	39.91
✓	-	-	20.3	26.88	28.0	37.49
-	✓	-	22.4	28.51	30.4	40.02
✓	✓	-	20.2	26.78	28.3	37.62
✓	✓	✓	**25.4**	**33.17**	**34.4**	**41.50**

Table 5: Analysis of the impact of different modules on 2WikiMultihopQA based on DRAGIN.

6 Conclusion

This paper presents IDR$_2$, a model-agnostic approach that significantly improves the efficiency of A-RAG by addressing redundant representation processing across multiple iterations. To prevent redundant computation, this paper introduces the CICS module for caching document representations. The IDGR module then guides the generation process to focus on crucial information through the instruction-following capability of LLMs. Furthermore, the IGPG module leverages the corre-

lation between generated content and documents to enable parallel generation of existing content, reducing autoregressive rounds. Through extensive experiments, IDR_2 consistently demonstrates remarkable efficiency improvements, achieving up to 2.0× acceleration while maintaining generation quality. Our future work aims to further compress the KV cache while preserving its representation capabilities.

7 Limitations

We acknowledge the limitations of this paper. Although our method is a general approach, it requires the corresponding LLM to be open-source to apply the CICS and IGPG technologies in this paper. It should be noted that the method in this paper is not applicable to LLM APIs that only support text-based interfaces. Therefore, our future work also aims to develop more methods to overcome the limitations of similar scenarios.

8 Acknowledgments

This rescarch is supported by the Sichuan International Science and Technology Innovation Coopcration Project with ID 2024YFHZ0317, the Chengdu Scicnce and Technology Burcau Project with ID 2024-YF09-00041-SN, and the National Natural Science Foundation of China Project with ID W2433163.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Sangmin Bae, Jongwoo Ko, Hwanjun Song, and Se-Young Yun. 2023. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5910–5924.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

James Ferguson, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi. 2020. Iirc: A dataset of incomplete information reading comprehension questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1137–1147.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR.

Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. 2024. A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions. *arXiv preprint arXiv:2410.12837*.

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.

Le Hou, Richard Yuanzhe Pang, Tianyi Zhou, Yuexin Wu, Xinying Song, Xiaodan Song, and Denny Zhou. 2022. Token dropping for efficient bert pretraining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3774–3784.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7029–7043.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Gyuwan Kim and Kyunghyun Cho. 2021. Length-adaptive transformer: Train once with length drop, use anytime with search. In *Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2021*, pages 6501–6511. Association for Computational Linguistics (ACL).

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. 2024. Speculative decoding with big little decoder. *Advances in Neural Information Processing Systems*, 36.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794.

M Komeili. 2021. Internet-augmented dialogue generation. *arXiv preprint arXiv:2107.07566*.

Jun Kong, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. 2022. Accelerating inference for pretrained language models by unified multi-perspective early exiting. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4677–4686.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

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

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. In *The Twelfth International Conference on Learning Representations*.

Huanshuo Liu, Bo Chen, Menghui Zhu, Jianghao Lin, Jiarui Qin, Hao Zhang, Yang Yang, and Ruiming Tang. 2024. Retrieval-oriented knowledge for click-through rate prediction. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 1441–1451.

Songshuo Lu, Hua Wang, Yutian Rong, Zhi Chen, and Yaohua Tang. 2024. Turborag: Accelerating retrieval-augmented generation with precomputed kv caches for chunked text. *arXiv preprint arXiv:2410.07590*.

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.

Thomas Merth, Qichen Fu, Mohammad Rastegari, and Mahyar Najibi. Superposition prompting: Improving and accelerating retrieval-augmented generation. In *Forty-first International Conference on Machine Learning*.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. 2024. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 932–949.

Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904*.

Shiyu Ni, Keping Bi, Jiafeng Guo, and Xueqi Cheng. 2024. When do llms need retrieval augmentation? mitigating llms' overconfidence helps retrieval augmentation. *arXiv preprint arXiv:2402.11457*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Benjamin Frederick Spector and Christopher Re. Accelerating llm inference with staged speculative decoding. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*.

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

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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.

Hanrui Wang, Zhekai Zhang, and Song Han. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110. IEEE.

Ruobing Wang, Daren Zha, Shi Yu, Qingfei Zhao, Yuxuan Chen, Yixuan Wang, Shuo Wang, Yukun Yan, Zhenghao Liu, Xu Han, et al. 2024a. Retriever-and-memory: Towards adaptive note-enhanced retrieval-augmented generation. *arXiv preprint arXiv:2410.08821*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Zilong Wang, Zifeng Wang, Long Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, et al. 2024b. Speculative rag: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv:2407.08223*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 24824–24837.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023a. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*.

Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. Predictive pipelined decoding: A compute-latency trade-off for exact llm decoding. *Transactions on Machine Learning Research*.

Yuchen Yang, Houqiang Li, Yanfeng Wang, and Yu Wang. 2023b. Improving the reliability of large language models by leveraging uncertainty-aware in-context learning. *arXiv preprint arXiv:2310.04782*.

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*. Association for Computational Linguistics.

Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Weichuan Liu, Lei Hou, and Juanzi Li. 2024. Seakr: Self-aware knowledge retrieval for adaptive retrieval augmented generation. *arXiv preprint arXiv:2406.19215*.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2023a. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2023b. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Zhihao Zhang, Alan Zhu, Lijie Yang, Yihua Xu, Lanting Li, Phitchaya Mangpo Phothilimthana, and Zhihao Jia. Accelerating iterative retrieval-augmented language model serving with speculation. In *Forty-first International Conference on Machine Learning*.

Zihan Zhang, Meng Fang, and Ling Chen. 2024. Retrievalqa: Assessing adaptive retrieval-augmented generation for short-form open-domain question answering. *arXiv preprint arXiv:2402.16457*.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*.

A Example Appendix

A.1 More Detail for Case Study

Figure 6 demonstrates the detailed process of running the LLaMA2-13B model with both the original DRAGIN and our proposed IGIR-ARAG framework on the 2WikiMultihotQA dataset. Initially, the model receives the question *Who lived longer, Giuseppe Cesari or Nicos Poulantzas?*. Without accessing external knowledge, LLM generates the answer *Giuseppe Cesari lived from 1566 to 1640. Nicos Poulantzas lived from 1920 to 1979. Thus, Nicos Poulantzas lived longer.*, shown in the **Without RAG** phase. After hallucination detection indicates low confidence, the process enters the RAG phase, retrieving three documents numbered #3596518, #3596521, and #10413313.

In **Round 1**, with CICS initially empty, no prefilling acceleration occurs, taking 0.96 seconds. IGPG then models the three documents, reducing the Decoding time from 4.24 to 2.16 seconds through parallel decoding. After detecting hallucination in the generated answer, the process enters **Round 2**. Here, CICS reuses document #359518's representation, saving 0.53 seconds in prefilling. During decoding, IGPG builds fragment models for document reuse, saving 2.31 seconds through parallel decoding. For example, *Nicos Poulantzas* requires six token inferences: *_N*, *icos*, *_P*, *oul*, *ant*, *zas*. With IGPG modeling, only one inference is needed after the prefix "Nicos," achieving 4× local acceleration. Similarly, dates like *October 1979*, *September 1936*, *February 1568* achieve 4× acceleration by retrieving corresponding fragments from IGPG using month information.

In **Round 2**, despite accurately obtaining both individuals' lifespans, the comparison is incorrect, leading to **Round 3**. Here, LLM further confirms *Giuseppe Cesari's* lifespan and generates the right answer. With all documents already in CICS, the prefilling time reduces by 0.804 seconds (about 80% of DRAGIN's original time), achieving 5× speedup. Finally, LLM correctly compares the information and provides the right answer.

Throughout this case, the framework saves 1.334 seconds in prefilling and 6.16 seconds in Decoding, totaling 7.494 seconds saved. Compared to DRAGIN's original 15.46 seconds, this represents a 48.47% time reduction, achieving 1.95× speedup.

A.2 Dataset Setting

For 2WikiMultihopQA, we followed Wang et al.'s approach for generating reasoning chains and answers, while incorporating prompt templates from Trivedi et al. (2023) and Jiang et al. (2023). The HotpotQA implementation followed Trivedi et al. (2023)'s settings and prompt templates. Both datasets employ Exact Matching at the answer level and F1 scores at the token level. For common sense reasoning evaluation, we adopted the StrategyQA dataset (Geva et al., 2021), implementing Wei et al. (2022)'s methodology and prompt templates, along with insights from Jiang et al. (2023). The evaluation uses Exact Matching to compare generated *yes/no* answers with standard answers. For reading comprehension capabilities, we adopted the IIRC dataset (Ferguson et al., 2020) following Trivedi et al. (2023)'s setup and evaluation metrics similar to 2WikiMultihopQA.

For the prompt templates of different datasets, we use the configuration of the DRAGIN (Su et al., 2024). The number of test samples and the specific test sample ids used for each dataset are kept consistent with DRAGIN (Su et al., 2024). We use 8 shots for those datasets in prompt.

A.3 Impact of A-RAG Rounds

We analyzed the impact of our method on the rounds of A-RAG, as shown in Table 7. Our IDR_2 maintains comparable round counts (Ours: 2.34-4.77 vs. DRAGIN: 2.68-5.33), demonstrating that our method adheres to the original design principles of DRAGIN.

A.4 Description of the libraries used in the implementation

Our code is developed based on PyTorch (Paszke et al., 2019), with the data loading module constructed on the open source code provided by DRAGIN (Su et al., 2024). We used Elasticsearch-7.17.9 for BM25.

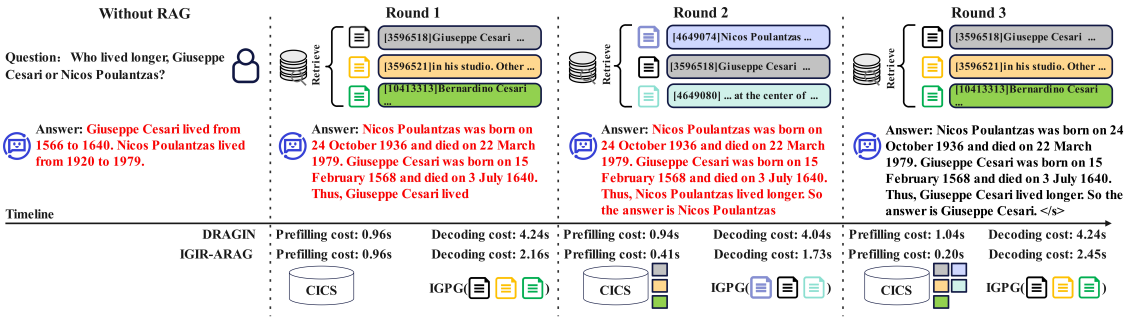


Figure 6: The detailed process of A-RAG based on a specific example, with LLaMA-13B.

Dataset statistics				
	2WikiMultihopQA	**HotpotQA**	**IIRC**	**StrategyQA**
Task	multi-hop QA	multi-hop QA	reading comprehension QA	commonsense QA
#Examples	1000	1000	954	1000

Evaluation settings				
	2WikiMultihopQA	**HotpotQA**	**IIRC**	**StrategyQA**
Metrics	EM, F1, Prec., Rec.	EM, F1, Prec., Rec.	EM, F1, Prec., Rec.	Accuracy

Table 6: Dataset statistics and experimental settings from DRAGIN (Su et al., 2024).

LLMs	Methods	2WMQA	HQA	SQA	IIRC
L2-7B	DR.	2.68	3.26	5.33	2.73
	DR.+Ours	2.34	2.77	4.77	2.08
L2-13B	DR.	2.74	3.69	4.66	3.06
	DR.+Ours	2.41	3.45	4.25	2.49
V-13B	DR.	3.02	3.47	4.41	3.28
	DR.+Ours	2.44	2.98	3.93	2.60

Table 7: The number of rounds of A-RAG.

A.5 Details of Prompt Template

The IDGR employs the prompt to guide the LLM in filtering redundant cached information, as shown below.

Step1: Learning from fewshots.

Prompt (Fewshots)

Question: When did the director of film Hypocrite (Film) die?
Answer: The film Hypocrite was directed by Miguel Morayta. Miguel Morayta died on 19 June 2013. So the answer is 19 June 2013.
February 1959. Ivania Martinich was born on 25 July 1995. Thus, Martin Hodge was born first. So the answer is Martin Hodge.
. . . . . .
Question: Which album was released earlier, Ngrtd or Everything I Love? Answer: Ngrtd was released on May 18, 2015 by Bomayé Music. Everything I Love was released on October 29, 1996. Thus, Everything I Love was released earlier. So the answer is Everything I Love.

Step2: Input the retrieved documents into the LLM.

Prompt (Retrieved documents)

Context:
[1672255] Christian August, Prince of Anhalt-Zerbst Christian August. . . . . .
[12419160] Frederick Augustus, Prince of Anhalt-Zerbst Frederick Augustus, Prince. . . . . .
[1672259] 1727 in Vechelde, Christian August married Johanna Elisabeth. . . . . .

Step3: Using IDGR to guide the model to focus on relevant content for the current round.

IDGR

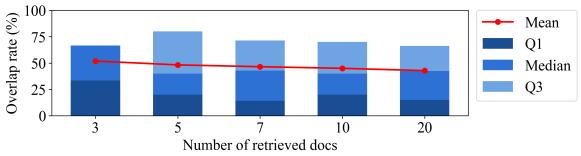
Document: [12368478] is related.
Document: [10028469] is unrelated.
The relevance scores are:
[12415199] Score: 3
[5881721] Score: 2
[12368478] Score: 1
Answer in the same format as before.

A.6 Results Calculation Methodology

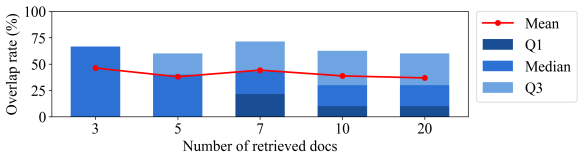
The calculation method of the speedup ratio presented in Table 8: first compute the individual speedup ratio for each sample, then calculate the arithmetic mean of all the ratios. The calculation method of the speed presented in Table 9 is calculated by taking the arithmetic mean of all sample speeds. Notably, the speedup ratio calculated by dividing the average speeds recorded in Table 9 differs from the result shown in Table 8, which is obtained by averaging individual sample speedup ratios.

A.7 Explanation of Overlap Rate

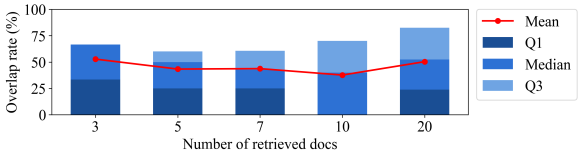
Figure 7 shows the overlap rate of the retrieved documents between different rounds. For example, as illustrated in Figure 7a, in the case of retrieving 3 documents, 25% of the samples have an overlap rate between 0% and 33%, 25% have an overlap rate between 33% and 66%, 25% have an overlap rate of 66%, and the remaining 25% have an overlap rate greater than 66%. The red dots represent the mean of the overlap rate in different numbers of retrieved documents. For example, in Figure 7b, when the number of the retrieved documents is 5, the mean overlap rate of the retrieved documents between the first and third rounds is 37.95%.



(a) Overlap rate comparison between round 1 and round 2.



(b) Overlap rate comparison between round 1 and round 3.



(c) Overlap rate comparison between round 1 and round 4.

Figure 7: The overlap rate comparison between different rounds.

A.8 The efficiency of IDR_2 on different devices

To verify whether IDR_2 has the same effect on different hardware devices, we tested the runtime of LLaMA-7B on the 2WikiMultihopQA dataset on Nvidia 3090 GPU and Nvidia H800 GPU respectively. As shown in Figure 9, we tested the runtime when setting different numbers of retrieved documents. The experimental results show that our IDR_2 framework achieves consistent performance improvements across different hardware devices.

A.9 Hyperparameters Analysis of M, N

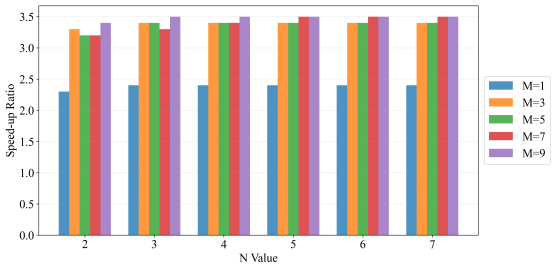


Figure 8: The speed-up ratios achieved with different combinations of M, N.

We conducted extensive experiments using LLaMA2-7B on the 2WikiMultihopQA dataset to evaluate the impact of different queue depths (M) and thread counts (N) on processing speed,

as shown in Figure.8. Based on these results, we chose N=5 and M=7 as our optimal configuration.

A.10 More Results

The results of acceleration for different models are listed in Table 8. The results of the runtime on different models are listed in Table 9. The results of performance using other methods are listed in Table 10. Table 11 shows the average number of documents cached per round and the corresponding memory requirements for different retrieval document formats. Table 12 adds an independent experimental result on the combination of CICS+IDGR.

Models	Methods	2WikiMultihopQA			HotpotQA			StrategyQA			IIRC		
		Pref.	Deco.	E2E	Pref.	Deco.	E2E	Pref.	Deco.	E2E	Pref.	Deco.	E2E
Llama2-7B	FLRAG	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FLRAG+Ours	2.23×	2.37×	1.75×	2.27×	1.85×	1.53×	1.75×	1.49×	1.40×	3.08×	2.76×	2.10×
	FSRAG	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FSRAG+Ours	2.04×	2.64×	2.32×	2.41×	1.88×	1.69×	2.36×	1.97×	1.79×	2.92×	2.88×	2.20×
	FLARE	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FLARE+Ours	2.81×	2.74×	2.60×	3.12×	2.61×	2.31×	2.71×	2.27×	1.77×	3.54×	2.97×	2.89×
	DRAGIN	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	DRAGIN+Ours	3.34×	2.77×	2.52×	4.09×	2.08×	2.09×	3.39×	1.98×	1.95×	4.49×	2.58×	2.58×
Llama2-13B	FLRAG	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FLRAG+Ours	2.12×	2.36×	1.64×	2.29×	2.00×	1.54×	1.76×	1.62×	1.40×	2.70×	2.65×	1.83×
	FSRAG	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FSRAG+Ours	2.11×	2.47×	1.79×	2.16×	2.05×	1.59×	1.74×	1.59×	1.44×	2.16×	2.65×	1.74×
	FLARE	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FLARE+Ours	1.98×	2.24×	2.16×	2.42×	1.87×	1.80×	2.45×	1.71×	1.69×	3.58×	3.15×	3.01
	DRAGIN	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	DRAGIN+Ours	3.25×	3.21×	2.61×	3.97×	2.15×	2.01×	3.15×	1.98×	1.87×	4.25×	2.86×	2.65×
Llama2-70B	FLRAG	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FLRAG+Ours	1.55×	2.24×	1.89×	1.53×	2.14×	1.76×	1.22×	1.64×	1.46×	1.75×	2.27×	1.92×
	FSRAG	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FSRAG+Ours	1.61×	2.60×	2.33×	1.68×	2.48×	2.27×	1.64×	2.32×	2.20×	1.82×	2.83×	2.62×
	FLARE	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FLARE+Ours	1.87×	2.09×	1.99×	2.33×	1.85×	1.81×	2.34×	1.69×	1.65×	3.68×	3.06×	2.98×
	DRAGIN	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	DRAGIN+Ours	3.02×	2.67×	2.23×	3.37×	2.16×	1.98×	2.99×	1.65×	1.64×	3.31×	2.79×	2.71×
Vicuna-7B-v1.5	FLRAG	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FLRAG+Ours	2.09×	2.21×	1.68×	2.29×	1.96×	1.61×	1.72×	1.61×	1.37×	2.81×	2.48×	1.84×
	FSRAG	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FSRAG+Ours	2.44×	2.92×	2.19×	2.11×	2.00×	1.78×	1.69×	1.64×	1.53×	2.59×	2.49×	2.16×
	FLARE	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FLARE+Ours	2.01×	2.09×	1.98×	2.11×	2.05×	1.93×	1.96×	1.88×	1.74×	2.65×	2.22×	2.19×
	DRAGIN	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	DRAGIN+Ours	3.84×	2.76×	2.54×	3.97×	1.98×	2.00×	3.46×	2.25×	2.17×	4.40×	2.39×	2.31×
Vicuna-13B-v1.5	FLRAG	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FLRAG+Ours	2.05×	2.25×	1.60×	2.22×	1.97×	1.51×	1.65×	1.55×	1.31×	2.66×	2.72×	1.82×
	FSRAG	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FSRAG+Ours	2.23×	2.31×	1.75×	2.15×	2.31×	1.66×	2.02×	1.78×	1.53×	2.35×	2.69×	1.92×
	FLARE	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	FLARE+Ours	2.04×	2.14×	2.06×	2.27×	2.07×	1.98×	2.05×	1.93×	1.89×	2.70×	2.24×	2.22×
	DRAGIN	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×	1×
	DRAGIN+Ours	3.57×	2.81×	2.55×	4.14×	2.29×	2.15×	3.25×	1.88×	1.82×	4.72×	4.00×	3.53×

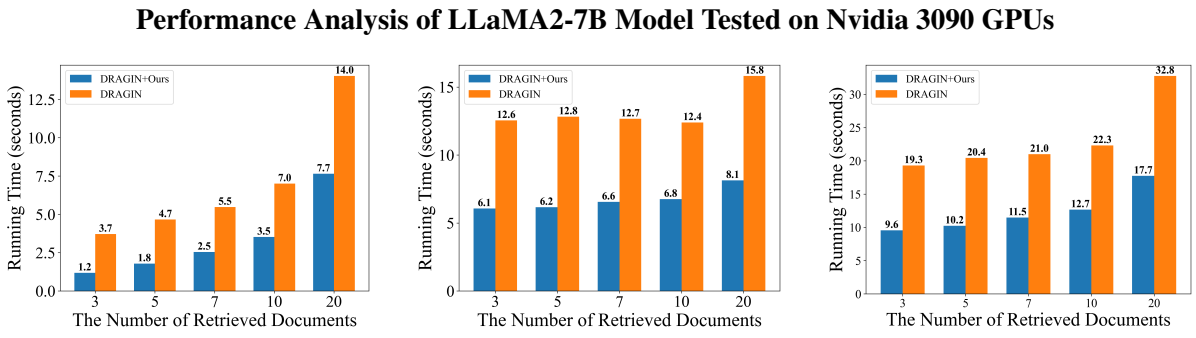
Table 8: The average speedup ratios before and after applying the IDR_2 framework.

Models	Methods	2WikiMultihopQA			HotpotQA			StrategyQA			IIRC		
		Pref.	Deco.	E2E	Pref.	Deco.	E2E	Pref.	Deco.	E2E	Pref.	Deco.	E2E
Llama2-7B	DRAGIN	3.71	12.55	19.31	5.50	21.61	31.83	6.81	28.99	41.79	4.73	25.31	33.75
	DRAGIN+Ours	1.18	6.07	9.56	1.49	12.58	18.26	2.19	18.22	25.71	1.18	12.29	16.20
Llama2-13B	DRAGIN	3.84	19.26	26.31	6.20	27.11	39.90	5.97	40.01	52.62	5.29	41.30	51.24
	DRAGIN+Ours	1.24	8.06	12.34	1.79	16.35	24.74	2.12	26.25	34.74	1.33	17.50	22.98
Vicuna-7B-v1.5	DRAGIN	4.56	16.40	24.07	5.90	16.83	27.66	5.11	19.20	28.11	5.63	33.63	43.38
	DRAGIN+Ours	1.30	7.62	11.61	1.59	9.58	15.60	1.69	11.35	16.59	1.47	17.71	23.08
Vicuna-13B-v1.5	DRAGIN	4.34	25.03	33.47	5.96	21.55	33.69	5.77	31.98	43.94	5.76	54.21	65.47
	DRAGIN+Ours	1.29	10.49	15.18	1.53	10.45	17.30	1.93	19.83	27.86	1.38	18.75	24.89

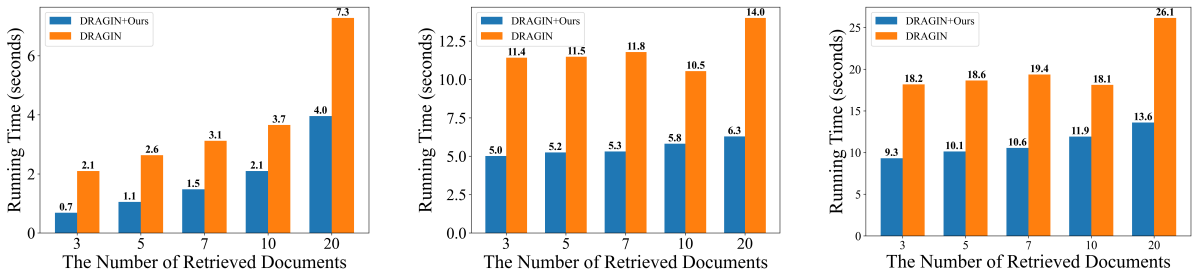
Table 9: The average runtime before and after applying the IDR_2 framework.

LLMs	Methods	2WQA		HQA		SQA	IIRC	
		EM ↑	F1 ↑	EM ↑	F1 ↑	Acc. ↑	EM ↑	F1 ↑
Llama2-7B	FLRAG†	14.10	19.49	15.10	24.55	63.50	14.68	18.83
	FLRAG+Ours	16.00	24.30	13.70	23.53	62.30	13.42	17.38
Llama2-13B	FLRAG†	21.90	32.63	21.80	33.43	66.70	15.62	18.93
	FLRAG+Ours	20.40	30.26	17.50	28.52	64.60	15.63	19.41
Vicuna-13b	FLRAG†	7.90	21.97	15.3	28.12	61.60	9.22	12.41
	FLRAG+Ours	6.80	12.83	14.20	25.48	61.99	8.39	10.20

Table 10: The performance comparison on different datasets with different models. The † represents the reproduction.



(a) LLaMA2-7B prefilling cost. (b) LLaMA2-7B decoding cost. (c) LLaMA2-7B end-to-end cost.



(d) LLaMA2-7B prefilling cost. (e) LLaMA2-7B decoding cost. (f) LLaMA2-7B end-to-end cost.

Figure 9: The speed comparison under different number of retrieved documents.

Methods	Number of retrieved documents		
	3	5	7
DRAGIN	3 0.24GB	5 0.41GB	7 0.57GB
DRAGIN+IDR$_2$	4.81 0.39GB	8.08 0.66GB	11.52 0.95GB

Table 11: Comparison of memory cost for LLaMA2-7B on 2WikiMultihopQA.

CICS	IGPG	IDGR	LLaMA2-7B	
			EM ↑	F1 ↑
-	-	-	22.5	28.68
✓	-	-	20.3	26.88
-	✓	-	22.4	28.51
✓	✓	-	20.2	26.78
✓	-	✓	**25.8**	**33.26**
✓	✓	✓	25.4	33.17

Table 12: Analysis of the impact of different modules on 2WikiMultihopQA based on DRAGIN.