[Experiment, Analysis, and Benchmark] XRAG: eXamining the Core - Benchmarking Foundational Components in Advanced Retrieval-Augmented Generation

Qili Zhang*, Qianren Mao†, Yangyifei Luo*, Yashuo Luo*, Hanwen Hao*,
Zhilong Cao*, Weifeng Jiang‡, Zhijun Chen*, Junnan Liu*,
Feng Yan*, Xiaolong Wang*, Jinlong Zhang*, Zhenting Huang*,
Zhixing Tan†, Jie Sun†, Bo Li*, Jianxin Li* and Philip S. Yu§
*Beihang University †Zhongguancun Laboratory ‡Nanyang Technological University §University of Illinois at Chicago
zhangqili@buaa.edu.cn, maoqr@zgclab.edu.cn, {luoyangyifei, luoyashuo, haohanwen}@buaa.edu.cn
zhilongcao66@gmail.com, weifeng001@e.ntu.edu.sg, zhijunchen@buaa.edu.cn, to.liujn@outlook.com
feng.yan@buaa.edu.cn, 3170153878@qq.com, zhangjinlong@buaa.edu.cn, zhentinghng@gmail.com
{zxtan, sunjie}@zgclab.edu.cn, 5481463@qq.com, lijx@buaa.edu.cn, psyu@uic.edu

***Abstract*—Retrieval-augmented generation (RAG) synergizes the retrieval of pertinent database with the generative capabilities of Large Language Models (LLMs), ensuring that the generated output is contextually relevant but also accurate and current. We introduce XRAG, an open-source, modular codebase that facilitates exhaustive evaluation of the performance of foundational components of advanced RAG modules. These components are categorized into four core phases: pre-retrieval, retrieval, post-retrieval, and generation. We systematically analyze them across reconfigured datasets, providing a comprehensive benchmark for their effectiveness. These components can be combined in different ways through an orchestrator, of which there are five types: sequential, conditional, iterative, parallel, and hybrid. Our work thoroughly evaluates the performance of advanced core components in RAG systems, offering actionable insights for optimizing RAG architectures in dataset-driven AI applications.**

***Index Terms*—Database, Retrieval-Augmented Generation, Evaluation, Benchmark, Data Management**

I. Introduction

Retrieval-Augmented Generation (RAG) [1]–[8] represents a pivotal strategy in Q&A tasks, demonstrating enhanced performance by delivering more informative and accurate answers compared to relying solely on large language models (LLMs). The efficacy of basic RAG systems [9], [10] is contingent depends on the seamless operation of four core components: pre-retrieval, retrieval, post-retrieval, and generation. The pre-retrieval stage indexes the corpus and reforms queries for efficient retrieval. The retrieval stage focuses on identifying and extracting documents relevant to a given query. The post-retrieval stage refines, summarizes, or compacts information to ensure contextual clarity. Finally, the generation stage employs the LLM to produce responses. In naive RAG systems, these stages are executed in a sequential manner. However, when dealing with complex retrieval scenarios, naive RAG often suffers from inefficiency and incomplete retrieval. Therefore, advanced RAG systems have optimized the retrieval process instead of sequential. Five RAG processes are as follows: sequential, conditional, iterative, parallel, and hybrid. These stages and processes critically influence output quality, highlighting the RAG framework's interdependence.

In real-world applications characterized by complex dataset structures, existing RAG frameworks exhibit significant limitations. Modular RAG toolkits like LangChain [12] and LlamaIndex [13] are typically cumbersome, making adaptation to new data challenging and validating or optimizing innovative methods inconvenient. Efficient RAG Frameworks like FastRAG [14], RALLE [15], AutoRAG [17], and LocalRQA [16] require users to independently reproduce published algorithms and offer limited component options, restricting the flexibility of RAG systems despite modular designs. In module evaluation tasks, FlashRAG [18] lacks uniformity in fundamental evaluation components, such as random seeds, generators, retrievers, and instructions, which hinders result comparability. RAGLAB [11], despite offering a fair experimental setup, lacks comprehensive evaluation strategies for assessing individual RAG components. Although ongoing efforts address these challenges through modular RAG processes (e.g., retrieval engines and generative agents), an implicit gap persists in the comparative performance evaluations of these advanced RAG modules within the overall RAG workflow. Comprehensive assessments of these modules are notably absent, making it difficult for researchers to evaluate their approaches in consistent experimental conditions.

To address the above issues, we concentrate on the core components of advanced RAG modules and the orchestrators of agentic RAG, conducting comprehensive experiments. We introduce XRAG [1], an open-source, modular codebase designed

[1]Code available at: https://github.com/DocAILab/XRAG.

TABLE I: Comparison of RAG Libraries. `Modular Design` (Mod.Dsgn) indicates toolkit modularity. `Fair Comparison` [11] (Fair.Comp) indicates evaluation by aligning key components like seeds, generators, retrievers, and instructions. `Unified Datasets` (Unif.Data) ensures unified dataset formats for retrieval and generation. `Modular Evaluation` (Mod.Eva) assesses RAG modular differences. ConR uses token-matching for evaluating retrieval. ConG uses token-matching for evaluating generation. CogL is based on LLM-based instructions for retrieval and generation evaluation. 'u' refers to No. of unified metrics.

RAG Library	Mod.Dsgn	Fair.Comp	Unif.Data	Mod.Eva	ConR	ConG	CogL
LangChain [12]	✔	✗	✗	✗	0	0	0
LlamaIndex [13]	✔	✗	✗	✔	6	0	7
FastRAG [14]	✔	✗	✗	✗	0	0	0
RALLE [15]	✔	✗	✗	✗	0	0	0
LocalRQA [16]	✗	✗	✔	✔	3	2	1
AutoRAG [17]	✔	✗	✔	✔	6	5	4
FlashRAG [18]	✔	✗	✔	✔	4	5	0
RAGLAB [11]	✔	✔	✔	✗	0	4	0
XRAG (ours)	✔	✔	✔	✔	$\mathbf{7}^u$	$\mathbf{10}^u$	$\mathbf{23}^u$

to evaluate foundational components of advanced RAG modules comprehensively. XRAG offers four principal capabilities:

Modular RAG Architecture: Fine-Grained Comparative Analysis. Extensive experiments are conducted on the advanced RAG modules in four stages: pre-retrieval, retrieval, post-retrieval, and generation. The core components cover three query rewriting strategies, six retrieval units, three post-processing techniques, and LLM generators from different vendors — OpenAI, Meta, and DeepSeek. XRAG provides an in-depth understanding of the capabilities of RAG components.

Agentic RAG Process: Multi-Orchestrator Integration and Workflow Design. The XRAG architecture incorporates five orchestrators: sequential, conditional, iterative, parallel, and hybrid. It provides a streamlined framework to flexibly orchestrating agentic RAG processes and integrate advanced RAG modules. Furthermore, to guide the development of RAG systems across various scenarios and data types, XRAG conducts performance evaluations of its orchestrators.

Unified Benchmark Datasets: Dual Assessment of Retrieval and Generation. To enhance the uniformity of datasets in RAG research, XRAG compiles and formats four prevalent benchmark datasets, preprocessing them into a unified format. This standardization enables concurrent assessment of both retrieval and generation, streamlining comparative evaluations across RAG modules and orchestrators.

Comprehensive Testing Methodologies: Multidimensional Evaluation Framework. To overcome the absence of a holistic evaluation system for RAG components, XRAG introduces an evaluation benchmark encompassing three perspectives thorough evaluation of retrieval and generation. . It comprises Conventional Retrieval Evaluation for retrieval-unit matching, Conventional Generation Evaluation for generation tests based on generative-token matching, and Cognitive LLM Evaluation for generation tests based on semantic understanding.

II. Related Work

A. RAG used for Agentic Data Management

In data management, benchmarks ensure fair performance evaluation by providing detailed insights beyond end-to-end metrics into components like query optimizers and retrieval models in RAG systems. While RAG systems are often viewed through the lens of natural language processing, their effectiveness is fundamentally governed by the quality, organization, and accessibility of the underlying knowledge corpus—a classic data management concern [19]. Existing methods in database systems for AI, such as SAGE [20], ARAG [21], and scRAG [22], often lack standardized datasets and metrics to specifically diagnose the retrieval component. Our work bridges this gap by introducing a benchmark framework that applies data management's rigorous evaluation principles to the RAG pipeline, enabling a systematic assessment of how data curation, indexing, and retrieval strategies ultimately impact the quality of generated answers. In certain database systems, such as HTAP databases [23], [24] and DB-GPT [25], XRAG can handle unstructured data and leverage RAG technology for retrieval and evaluation, enabling these database systems to be applied to scenarios such as generation and inference.

B. RAG Systems or Toolkits

Retrieval-Augmented Generation [1]–[8] has been widely adopted as an effective technique for mitigating hallucination issues in large language models across various complex generation and reasoning tasks. These tasks typically require retrieving critical information from large-scale databases to ensure accurate outcomes and logically sound reasoning processes. Toolkits like LangChain [12] and LlamaIndex [13], modularize the RAG process, increasing adaptability and broadening its applications. FastRAG [14] and RALLE [15] allow users to assemble RAG systems with core components, fostering more flexible implementations. AutoRAG [17] further supports users by identifying optimal RAG pipelines for custom data, facilitating bespoke RAG systems. LocalRQA [16] and RAGLAB [11] focus on RAG training, offering scripts for various component training. FlashRAG [18] and RAGLAB [11] advance algorithmic reproducibility in RAG systems by integrating numerous algorithms into a unified framework, supporting efficient replication of existing methods and promoting innovation in algorithm development. Despite their restricted scalability, undeveloped evaluation protocols, absent agentic RAG analysis, and non-standardized metrics, modular RAG frameworks pervade current engineering practice.

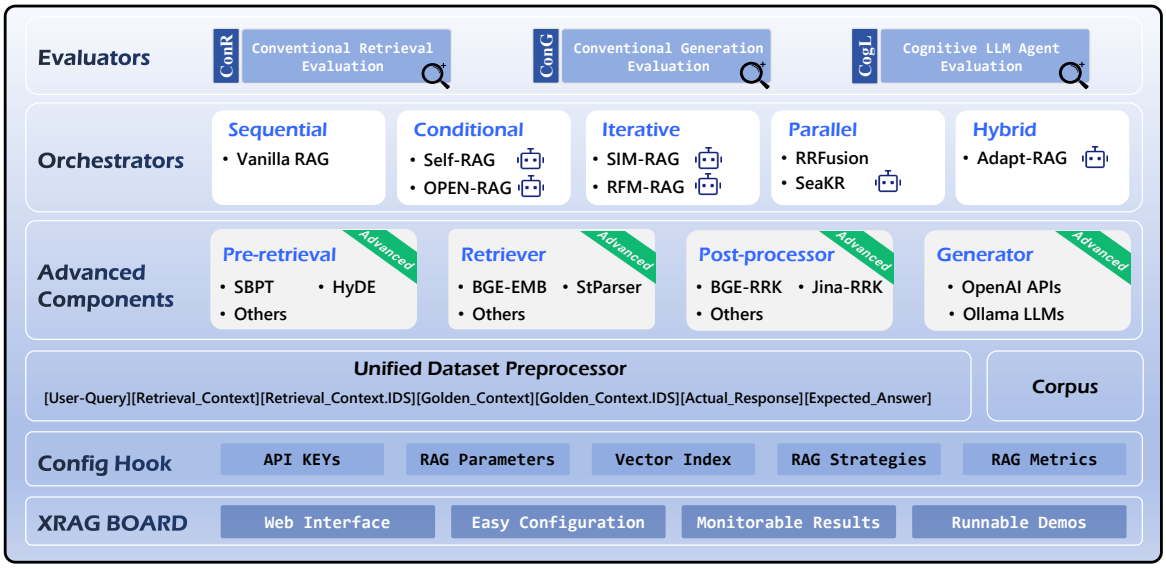


Fig. 1: Schematic overview of the XRAG framework. Icon "🤖" indicates that this is an agent-based method.

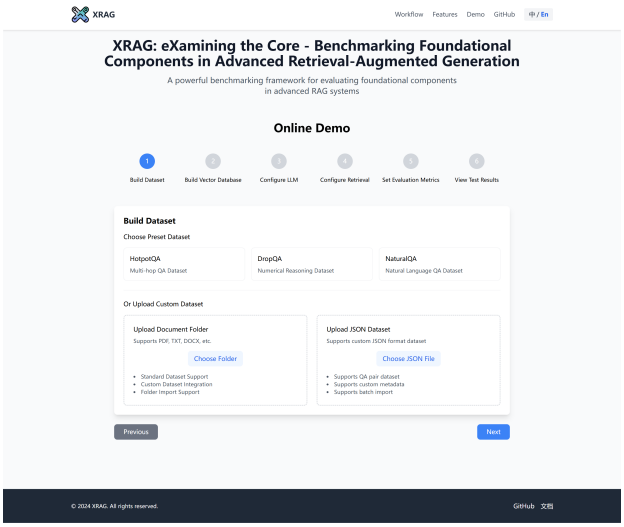


Fig. 2: A Screenshot of the Development Web UI of XRAG.

Meanwhile, researchers are progressively modularizing RAG workflows to address the dataset retrieval requirements in large language model generation tasks. Inspired by these modular RAG frameworks, we introduce XRAG to evaluate foundational components of advanced RAG modules comprehensively.

III. XRAG

Fig. 1 delineates the integrated modules and schematic structure of the XRAG framework. The framework is stratified into datasets and corpus, advanced components, orchestrator, and evaluators, integrated through XRAG's board and config hook [2]. The overall structure progresses from foundational to application-oriented components. Designed with a modular architecture, XRAG enables users to accomplish: preparation of normalized RAG datasets (Section III-C), assembly of the RAG components (Section III-A), orchestration of the RAG workflow (Section III-B) and evaluation of the RAG system's core components (Section III-D).

A. Basic & Advanced RAG Components

♣ **Pre-retrieval** Before retrieval, the pre-retrieval components leverage LLMs to refine user queries, enhancing the quality and relevance of the information retrieval process. Key methodologies include Step-back Prompting (`SBPT` [26]): Broaden's queries to enrich contextual grounding for answers, enhancing the contextual foundation for answer generation. Hypothetical Document Embedding (`HyDE` [27]): Transmutes the original query into a form that better aligns with the indexed documents, improving retrieval alignment and efficacy.

♠ **Retriever** We use two basic retrieval models as benchmarks: BGE-Large and JINA-Large, along with their open-source and consistently embedding models. For advanced retrieval strategies, we integrate the LlamaIndex to facilitate standard advanced methods. Reciprocal Rerank Fusion Retriever (`RRFusion` [28]) fuses indexes with a BM25-based retriever, capturing both semantic relations and keyword relevance. Both retrievers assign scores, enabling reciprocal reranking for node sorting without additional models or excessive computation. For dense retrieval, SentenceWindow Retriever (`StParser`) parses documents into single sentences per node, incorporating surrounding sentences for added context.

▼ **Post-processor** To improve retrieval accuracy and efficiency, XRAG uses post-processors like rerankers to refine nodes before returning them. We integrate the BGE reranker (BGE-RRK), which computes similarity scores using a Cross-Encoder model. Additionally, JINA-Reranker-V2 (JINA-RRK) supports high-accuracy multilingual document reranking.

[2]As shown in the Fig. 2, we have implemented the XRAG demo. These two parts are mainly functions within the demo and are not the focus of the experimental analysis.

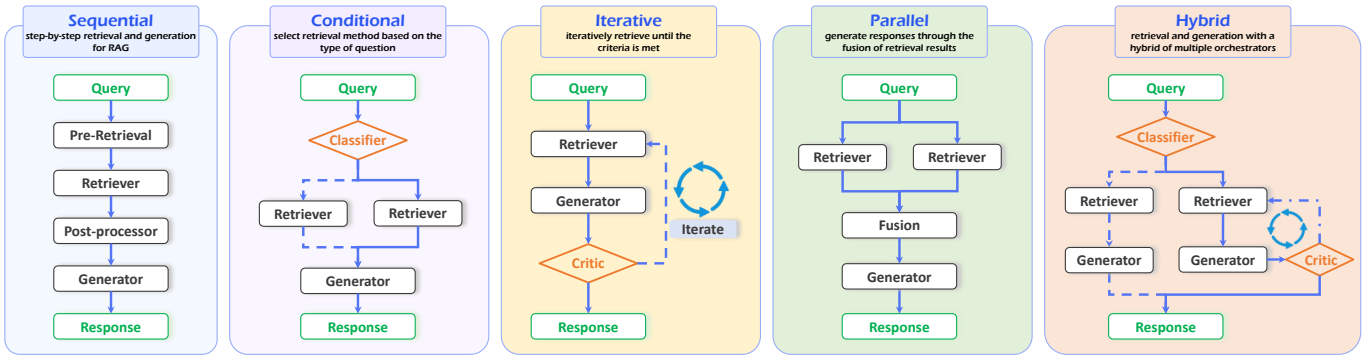


Fig. 3: Schematic of the Orchestrator's Logic in Coordinating Agentic RAG System Components.

▲ **Generator** The XRAG framework integrates various LLM generators, including those from HuggingFace Transformers APIs [3], ensuring compatibility with open-source LLMs. It also integrates the Ollama [4] framework, supporting the localized use of LLMs. In addition to open-source models, the generator module of XRAG includes support for closed-source LLM APIs. Thus, users can access diverse capabilities while retaining the option to use proprietary models.

B. Basic & Advanced RAG Orchestrators

With the advancement of RAG technology, modern agentic RAG frameworks are no longer confined to a linear pipeline structure. Instead, they employ more sophisticated orchestration methods to arrange fundamental RAG components, enhancing performance and adapting to diverse application scenarios. We use orchestrator modules to organize and manage the execution logic and workflow of RAG components. As illustrated in Fig. 3, the XRAG framework includes five types of orchestrators: sequential, conditional, iterative, parallel, and hybrid.

♣ **Sequential** The sequential orchestrator operates through the sequential orchestration of advanced components to perform retrieval, and is suited for simple application scenarios. Extending beyond Naive RAG, XRAG's sequential orchestrator leverages advanced modules to optimize retrieval performance while managing computational costs. With this orchestrator, the XRAG framework's workflow strictly follows a linear sequence of steps—"pre-retrieval→retrieval→post-processing→generation"—executed step-by-step to retrieve contents and generate answers.

♠ **Conditional** In the XRAG framework, we employ a conditional orchestrator to implement RAG methods that require conditional branching. This orchestrator selects the retrieval method according to the type of query, and dynamically adjusts the execution sequence and logic of RAG components based on specific conditions or rules. We have integrated Self-RAG [29] as a typical application of the conditional orchestrator, which employs self-reflection to determine whether retrieval is necessary. Additionally, Open-RAG [30], which performs targeted retrieval by first classifying queries, can also serve as an example of this conditional orchestration approach.

♥ **Iterative** The XRAG framework supports the use of an iterative orchestrator to implement RAG methods requiring multiple iterations. This orchestrator enables RAG components to execute repeatedly across rounds, progressively refining retrieval results and generated content until the criteria are met. As an advanced iterative orchestrator, SIM-RAG [31], which is capable of performing multiple retrieval steps and refining prompts during the iterative process, has been integrated into the XRAG framework. We have incorporated RFM-RAG [32] as another implementation of the iterative orchestrator. RFM-RAG dynamically constructs an evidence pool through iterative retrieval and formulates targeted queries until termination.

▼ **Parallel** To enable RAG methods requiring parallel logic, the XRAG framework supports the use of a parallel orchestrator. This orchestrator allows multiple RAG components to operate in logical parallelism, as opposed to the conditional orchestrator that relies on selection logic. As a classic parallel-orchestrated RAG technique, the retrieval logic of Reciprocal Rerank Fusion Retriever (`RRFusion` [28]) has been integrated into the XRAG framework as an orchestrator. Both retrievers in RRFusion assign scores in parallel, enabling reciprocal reranking for node sorting without additional models or excessive computation. The SeaKR [33] method, which performs parallel retrievals with multiple groups and selects the optimal retrieval for generation, has also been integrated into the XRAG framework.

▲ **Hybrid** The XRAG framework supports the use of a hybrid orchestrator to address complex RAG scenarios where a single orchestrator proves insufficient. The hybrid orchestrator can combine multiple orchestration logics to accommodate sophisticated RAG requirements. The framework integrates the Adapt-RAG [34] method as an instance of this hybrid orchestrator. Adapt-RAG incorporates both conditional and iterative orchestration logic, dynamically switching between direct and iterative retrieval depending on the complexity of the query, to achieve a flexible and efficient retrieval process.

C. Unified Benchmark Datasets & Corpus

We collect and preprocess four benchmark datasets for the XRAG framework, emphasising rigorous experimental

[3]https://github.com/huggingface/transformers
[4]https://github.com/ollama

TABLE II: Summary of Benchmark Datasets & Corpus. Original training and validation sets, are retained for potential future fine-tuning of RAG systems or other customized tasks, although XRAG does not currently support fine-tuning.

Datasets	Size			Corpus		Multi-hop	Constrained	Numerical	Set-logical
	Train	Validation	Test	Documents	Source				
HotpotQA (HQA)	86,830	8,680	968	508,826	Wikipedia	✔	✗	✔	✗
DropQA (DQA)	78,241	7,824	870	6,147	Wikipedia	✔	✗	✔	✔
NaturalQA (NQA)	100,093	10,010	1,112	49,815	Wikipedia	✔	✔	✔	✗
FinanceBench (FQA)	43,658	50	100	37,885	SEC Filings	✔	✗	✔	✔

validation of RAG systems. We develop a unified dataset structured to facilitate performance testing for both retrieval and generation modules, incorporating standardized formats:

```
User-Query || Retrieval-Context ||
Retrieval-Context.IDS || Golden-Context ||
Golden-Context.IDS || Actual-Response ||
Expected-Answer.
```

XRAG provides a unified architecture for retrieval and question-answering datasets [5], specifically HoppotQA [35], DropQA [36], NaturalQA [37] and FinanceBench [8]. The corpus used for indexing is derived from the metadata of these datasets' training, validation, and test sets. This methodology aligns with previous works [38] and supports the efficient deployment of vector databases. Table II shows that the HotpotQA corpus contains the highest document count, followed by NaturalQA, indicating that retrieval difficulty escalates with the number of documents required per query. The FinanceBench [8], [39] dataset consists of 37,885 financial documents covering 40 companies that are publicly traded in the USA. Moreover, these datasets address complex RAG question-answering scenarios, including Multi-hop Queries, Constrained Queries, Numerical Reasoning, and Logical Reasoning tasks. Multi-hop questions depend on iteratively retrieved documents and their interrelations to deduce the answer. Constrained Q&A generates each answer alongside a corresponding constraint or condition rather than providing a standalone response. Numerical reasoning involves performing arithmetic operations such as addition, subtraction, sorting, and counting. Set-logical reasoning addresses complex logical problems involving relationships among retrieved chunks.

We constructed retrievable documents using metadata from the original datasets, standardizing retrieval objects as document IDs for testing. Retrieval is successful if the retrieved chunk node corresponds to the annotated document ID. This method ensures consistency in retrieval labels and eliminates discrepancies caused by varying document chunking strategies. For the test set, we limit the number of samples to mitigate the high token cost associated with RAG and LLM evaluations. In HQA, NQA, and DQA, instead of evaluating the entire test set, we apply a sampling-based averaging method, reducing token usage while preserving reliability. For FQA, which has a smaller test set, we use all test data for evaluation.

The respective prompts of XRAG for the HQA, DQA, NQA and FQA datasets are listed below.

Prompt for XRAG suitable for HotpotQA and FinanceBench Datasets

Context information is below.
· · · · · · · · · {context_str} · · · · · · · · ·
Given the context information and no prior knowledge, answer the question: {query_str}
We have the opportunity to refine the original answer.
(only if needed) with some more context below.
· · · · · · · · · {context_msg} · · · · · · · · ·
Given the new context, refine the original answer to better.
Answer the question: query_str
If the context isn't proper, output the original answer again.
Original Answer: existing_answer

Prompt for XRAG suitable for DropQA and NaturalQA Datasets

Context information is below.
· · · · · · · · · {context_str} · · · · · · · · ·
Given the context information and no prior knowledge. Please provide a brief, shortest possible answer, ideally just one word for the following question:
Question: who has sold more records Oasis or Coldplay?
Expected Answer: Oasis
We have the opportunity to refine the original answer.
(only if needed) with some more context below.
· · · · · · · · · {context_msg} · · · · · · · · ·
Given the new context, refine the original answer to better.
Answer the question: query_str
If the context isn't proper, output the original answer again.
Original Answer: existing_answer

D. Evaluation Methods

To assess the quality of the RAG components, we integrate the Jury [40], a comprehensive package for the evaluation of NLG systems, with RAG community evaluation tools such as UpTrain[6] and DeepEval[7]. XRAG evaluators are pivotal in determining the effectiveness of both the retrieval and generation components. They are categorized into three groups: Conventional Retrieval Evaluation, Conventional Generation Evaluation, and Cognitive LLM Evaluation.

Conventional Retrieval Evaluation (ConR Evaluator). It supports six primary metrics: F1, Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP), along with Hit@1 and Hit@10. Additionally, it includes the DCG family of metrics, which assesses the effectiveness of ranking models by evaluating the quality of ordered results (`retrieval-context.ids`). This family comprises Discounted Cumulative Gain (DCG), Normalized Discounted Cumulative Gain (NDCG), and Ideal

[5]The dataset is CC BY-NC-SA 4.0 licensed (we maintained the same license agreement as the original dataset.), accessible via the URL https://huggingface.co/datasets/DocAILab/XRAG_Dataset. We also provide a documentation for the XRAG: https://docailab.github.io/XRAG/ on how to use the data and code.

[6]https://github.com/uptrain-ai/uptrain

[7]https://github.com/confident-ai/deepeval

Interpretation of Cognitive LLM Evaluation Metrics

- **Up-CRel** [Retrieval]: Uptrain-Context-Relevance
  - Pa. <Query, Retrieval-Context>
- **Up-CCns** [Retrieval]: Uptrain-Context-Conciseness
  - Pa. <Query, Golden-Context, Retrieval-Context>
- **Dp-ARel** [Generation]: DeepEval-Response-Relevancy
  - Pa. <Query, Actual-Response>
- **Up-RCmp** [Generation]: Uptrain-Response-Completeness
  - Pa. <Query, Actual-Response>
- **Up-RCnc** [Generation]: Uptrain-Response-Conciseness
  - Pa. <Query, Actual-Response>
- **Up-RRel** [Generation]: Uptrain-Response-Relevance
  - Pa. <Query, Actual-Response>
- **Up-RVal** [Generation]: Uptrain-Response-Valid
  - Pa. <Query, Actual-Response>
- **Up-RMch** [Generation]: Uptrain-Response-Matching
  - Pa. <Query, Actual-Response, Expected-Answer>
- **Dp-CPre** [Retrieval & Response]: DeepEval-Context-Precision
  - Pa. <Query, Actual-Response, Expected-Answer, Retrieval-Context>
- **Dp-CRec** [Retrieval & Response]: DeepEval-Context-Recall
  - Pa. <Query, Actual-Response, Expected-Answer, Retrieval-Context>
- **Dp-CRel** [Retrieval & Response]: DeepEval-Context-Relevance
  - Pa. <Query, Actual-Response, Retrieval-Context>
- **Up-RCns** [Retrieval & Response]: Uptrain-Context-Consistency
  - Pa. <Query, Actual-Response, Retrieval-Context>
- **Up-CUti** [Retrieval & Response]: Uptrain-Context-Utilization
  - Pa. <Query, Actual-Response, Retrieval-Context>
- **Up-FAcc** [Retrieval & Response]: Uptrain-Factual-Accuracy
  - Pa. <Query, Actual-Response, Retrieval-Context>
- **Dp-Fath** [Retrieval & Response]: DeepEval-Context-Faithfulness
  - Pa. <Query, Actual-Response, Retrieval-Context>
- **Dp-Hall** [Retrieval & Response]: DeepEval-Context-Hallucination
  - Pa. <Query, Actual-Response, Golden-Context>

Fig. 4: Interpretation of Cognitive LLM Evaluation Metrics

Discounted Cumulative Gain (IDCG). IDCG is the maximum DCG that can be obtained if the results are ideally ranked – arranged in descending order of their relevance (`golden-context.ids`).

Conventional Generation Evaluation (ConG Evaluator). These generative-token matching metrics can be classified into three broad categories. N-gram similarity metrics: ChrF [41], ChrF++ [42], METEOR [43], ROUGE F1 [44] (R1, R2, RL) focus on overlap in n-grams between generation (`actual-response`) and reference (`expected-answer`). Divergence-based metrics: MAUVE [45], Perplexity [46] measure content quality, diversity, and model learning by comparing the distribution between the generation and reference. Error-based accuracy metrics: Word Error Rate (WER) [47], Character Error Rate (CER) [47], assess the accuracy of `actual-response` by calculating the differences or errors when compared with the `expected-answer`.

Cognitive LLM Evaluation (CogL Evaluator). It is classified into three categories: Retrieval-oriented, Generation-oriented, and Combined Retrieval & Generation. Cognitive LLM Evaluation metrics, derived from UpTrain(prefixed with 'Up') and DeepEval(prefixed with 'Dp'), are classified based on the parameters unified by our XRAG framework. Response-oriented metrics involve response-related parameters including Response Relevance (Dp-ARel), Response Completeness (Up-RCmp) from DeepEval, Response Conciseness (Up-RCnc), Response Relevance (Up-RRel), and Response Validity (Up-RVal) and Response Matching (Up-RMch) from Uptrain. Retrieval-oriented metrics that lack response-related parameters and include retrieval-related parameters, assess context quality and consist of Context Relevance (Up-CRel) and Context Conciseness (Up-CCns), which come from UpTrain. Combined metrics evaluate the impact of retrieval on final responses and include Context Precision (Dp-CPre), Context Recall (Dp-CRec), Context Relevance (Dp-CRel), Response Consistency (Up-RCns), Context Utilization (Up-CUti), and Factual Accuracy (Up-FAcc), Faithfulness (Dp-Faith), and Hallucination (Dp-Hall). Detailed usage patterns of the CogL evaluation are illustrated in Fig. 4. Conventional rule-based metrics like Exact Match (EM) and ROUGE concentrate on n-gram matching, which sometimes falls short in capturing the comprehensive accuracy of language expressions; however, they remain fitting for tasks that necessitate distinct and singular responses. In contrast, metrics based on Cognitive LLMs evaluate the quality of generative language, which is particularly beneficial when the complexity of evaluative comprehension surpasses the capabilities of rule-based metrics.

XRAG evaluators have obvious advantages:

- **Evaluating with Multiple RAG Metrics in One Go**: The XRAG evaluator allows users to simultaneously assess various RAG-specific metrics. This capability streamlines the evaluation process, enabling comprehensive performance analysis without sequential evaluations.
- **Standardizes the Structure of Evaluation Metrics**: A unified data format simplifies the comparison between different RAG components in both retrieval and generation.
- **Character and Semantic $\times$ Retrieval and Generation**: It encompasses a 4-fold cross-dimensional analysis, including character-level matching tests and semantic-level understanding tests for both retrieval and generation.

IV. Experimental Setup

For document preprocessing, we used SentenceSplitter(chunk size: 128 tokens, overlap: 20 tokens) to segment documents into chunks and build a vector index. We adhered to LlamaIndex configurations for other RAG components, including the refine module for response synthesis. To ensure compatibility and efficiency, XRAG integrates Huggingface Transformers. All Q&A LLMs were set with Temperature $= 0$ to ensure experimental consistency [8]. Consequently, the evaluation metrics measure retrieval accuracy based on 3 retrieval nodes, fully encompassing the assumptions of most datasets that typically consider only one or two golden contextual nodes. The

[8]Nonetheless, using large models for generation or evaluation always results in minor variations, possibly due to randomness in MoE training or prompt interpretation biases [49].

TABLE III: Fixed and distinctive modules in Agentic RAG Orchestrators. SelfRAG-Llama2-7B and SIM-RAG-Llama3-2B are fine-tuned models used in the SelfRAG and SIM-RAG; T5-Large is a 770M-parameter model released by Google; and RRFusion employs the Reciprocal Rerank Fusion Retriever from the LlamaIndex toolkit.

Method	Fixed modules		Distinctive modules			
	Retrieval Model	Generative Model	Classifier	Critic	Fusion	Adapter
BGE-Large	BGE-Large-en-v1.5	deepseek-r1-distill-llama-70b	✗	✗	✗	✗
Self-RAG	BGE-Large-en-v1.5	deepseek-r1-distill-llama-70b	SelfRAG-Llama2-7B	✗	✗	✗
SIM-RAG	BGE-Large-en-v1.5	deepseek-r1-distill-llama-70b	✗	SIM-RAG-Llama3-2B	✗	✗
RRFusion	BGE-Large-en-v1.5	deepseek-r1-distill-llama-70b	✗	✗	RRFusion [28]	✗
Adapt-RAG	BGE-Large-en-v1.5	deepseek-r1-distill-llama-70b	✗	✗	✗	T5-Large [48]

TABLE IV: Upstream RAG **Retrieval Performance** using **Basic Retriever** of BGE-Large and JINA-Large. The conventional retrieval evaluation, denoted as ConR, is performed. The symbol $^\circ$ signifies metric values ranging from 0 to positive infinity, while all other values are within the $[0, 1]$ interval (%).

Dataset & Methods		Conventional Retrieval Evaluation (ConR Evaluator)							
		F1	MRR	Hit@1	Hit@10	MAP	NDCG	DCG$^\circ$	IDCG$^\circ$
HQA	JINA-Large	49.80	64.23	21.42	94.44	55.17	70.44	0.9823	1.1795
	BGE-Large	60.66	72.84	18.29	97.71	64.13	78.64	1.1644	1.3656
DQA	JINA-Large	20.88	24.42	06.90	34.47	24.42	26.76	0.3835	0.4482
	BGE-Large	22.58	27.10	07.12	42.30	27.10	29.58	0.4160	0.4849
NQA	JINA-Large	49.20	62.44	30.31	88.49	62.44	66.98	0.8843	1.0120
	BGE-Large	49.90	63.82	25.68	93.24	63.82	69.12	0.8799	1.0300
FQA	JINA-Large	9.65	40.75	38.00	43.00	40.75	41.39	0.5076	0.5268
	BGE-Large	6.05	26.83	23.00	29.00	26.83	27.45	0.3104	0.3278

generator model's context window—encompassing the query, prompt, retrieved context, and response content—is configured to 4096 tokens. To ensure a fair comparison when testing the orchestrators, we used identical modules where feasible, as detailed in Table III. Due to the high cost associated with token processing during LLM (OpenAI) reasoning and evaluation, Cognitive LLM Evaluation (CogL) metrics were included as a pilot study to demonstrate the XRAG framework's capability in semantic assessment. Experiments involving LLM generation were repeated three times, and standard deviations are reported in the tables of generation results.

Computational Resource Due to inherent design and size constraints, the deployment environments for each model are distinct. For the DQA dataset, testing was conducted on a single NVIDIA V100 GPU (32GB); the indexing phase (single-threaded) and the search-and-generation phase (four-threaded) required approximately one and two hours, respectively, with the BGE-Large index occupying 496 MB and the JINA-Large index occupying 391 MB of disk space. Similarly, on the NQA dataset using the same V100 hardware, the indexing and search phases took three and four hours, with index sizes of 5.4 GB for BGE-Large and 4.2 GB for JINA-Large. For the HQA dataset, an NVIDIA L20 GPU (48GB) was employed; both indexing and search processes were executed in a single-threaded manner, lasting four and eight hours, while the BGE-Large and JINA-Large indices utilized 18 GB and 14 GB of storage, respectively. Finally, the FQA dataset was evaluated on an NVIDIA RTX 4090 GPU (24GB), both single-threaded indexing and search/generation were completed in one hour each, with the BGE-Large and JINA-Large indices requiring 8.8 GB and 6.9 GB of disk space.

V. Experimental Results

A. Performance of modulor used for Retrieval

The retrieval performance varies sensibly across the four datasets, with the poorest quality observed on DQA and FQA in Table IV. Since DQA requires advanced discrete reasoning [50] and FQA demands paragraph understanding of financial documents, it presents a greater retrieval challenge.

i. Different datasets should focus on different testing metrics. The NDCG metric reflects that the basic retrieval model performs reasonably well in relevance and ranking accuracy on HotpotQA and NaturalQA datasets. Ignoring ranking accuracy, a high Hit@10 score (>0.8) indicates a substantial likelihood of retrieving semantically relevant document blocks. The Hit@1 metric measures the ability to return the correct answer as the top-ranked result, making it suitable for tasks with a single retrieval target (e.g., the golden single passage in the NaturalQA dataset). However, it is less effective for scenarios requiring multiple equally prioritized context targets, as seen in the HotpotQA dataset. Overall, the retrieval system performs well on HotpotQA and NaturalQA but faces challenges on DropQA, due to the complexity of the answer (Low Hit@1 scores reflect difficulty in retrieving the most relevant results.).

ii. To improve retrieval performance, priority should be given to retrieval and post-retrieval. Compared to the basic retriever in Table IV and the advanced retriever in Table V, the highlighted results indicate that adopting advanced retrieval modules, which integrate additional matching information, can enhance retrieval effectiveness. For example, RRFusion combines dense vectors with keyword retrieval, and the sentence window includes the context of chunk nodes in the retrieval objects. Specifically, it is observed that reranking methods can

TABLE V: Upstream RAG **Retrieval Performance** about advanced **Pre-retrieval (PR)**, **Advanced Retrieval (AR)** and **Post-processor (PP)**, where the BGE-Large model constitutes the fixed foundational retrieval model. Results underlined indicate superiority over Basic Retriever of BGE-Large, while **bold** results denote the best performance.

Dataset & Methods		Distinctive			Fixed	Conventional Retrieval Evaluation (ConR Evaluator)							
		PR	AR	PP	R	F1	MRR	Hit@1	Hit@10	MAP	NDCG	DCG°	IDCG°
HQA	SBPT	✔	✗	✗	BGE-Large	38.60	53.86	16.96	96.43	44.99	59.61	0.7174	0.8992
	HyDE	✔	✗	✗		53.07	67.39	17.86	**97.32**	58.32	72.44	0.9947	1.1753
	RRFusion	✗	✔	✗		59.99	76.92	12.24	96.94	66.25	79.72	1.0401	1.1192
	StParser	✗	✔	✗		42.92	59.54	14.02	97.20	49.82	65.17	0.8069	0.9921
	BGE-RRK	✗	✗	✔		**66.90**	**80.19**	**77.03**	97.30	**70.62**	**83.14**	**1.1354**	**1.2183**
	JINA-RRK	✗	✗	✔		64.68	76.89	76.92	96.43	68.83	79.60	1.1096	1.1733
DQA	SBPT	✔	✗	✗	BGE-Large	22.57	27.03	06.01	42.08	27.03	29.45	0.4149	0.4823
	HyDE	✔	✗	✗		22.74	26.40	06.81	41.48	26.40	28.89	0.4213	0.4906
	RRFusion	✗	✔	✗		26.36	29.31	14.29	19.05	29.31	30.18	0.3885	0.4135
	StParser	✗	✔	✗		27.70	30.34	05.13	**46.15**	30.34	31.28	0.4092	0.4355
	BGE-RRK	✗	✗	✔		31.72	34.27	28.46	38.88	34.27	35.44	0.4663	0.4990
	JINA-RRK	✗	✗	✔		**34.20**	**37.66**	**31.17**	44.16	**37.66**	**38.68**	**0.5097**	**0.5385**
NQA	SBPT	✔	✗	✗	BGE-Large	49.22	62.27	22.86	88.57	62.27	67.53	0.8737	1.0210
	HyDE	✔	✗	✗		50.12	62.94	41.03	94.87	62.94	68.69	0.8864	**1.0483**
	RRFusion	✗	✔	✗		**62.39**	**70.35**	42.86	**95.24**	**70.35**	**73.02**	**0.8949**	0.9700
	StParser	✗	✔	✗		59.79	68.42	28.95	94.74	68.42	71.29	0.8430	0.9239
	BGE-RRK	✗	✗	✔		59.14	68.26	50.91	76.36	68.26	71.20	0.8284	0.9113
	JINA-RRK	✗	✗	✔		58.97	66.67	**56.41**	79.49	66.67	68.68	0.8648	0.9215
FQA	SBPT	✔	✗	✗	BGE-Large	6.65	27.17	21.00	30.00	27.16	28.03	0.3367	0.3618
	HyDE	✔	✗	✗		4.75	21.00	18.00	23.00	21.00	21.58	0.2441	0.2602
	RRFusion	✗	✔	✗		**12.66**	18.00	17.00	19.00	18.00	18.26	0.1889	0.1963
	StParser	✗	✔	✗		7.83	12.12	9.00	12.00	12.00	12.12	0.1275	0.1275
	BGE-RRK	✗	✗	✔		8.86	35.83	35.00	39.00	35.84	36.71	0.4785	0.5033
	JINA-RRK	✗	✗	✔		10.20	**42.33**	**40.00**	**46.00**	**41.83**	**43.41**	**0.5524**	**0.5822**

markedly improve performance, particularly for challenging retrieval tasks such as DQA and FQA. For instance, both BGE-RRK and JINA-RRK outperform the basic retriever. This is because the reranker can evaluate the relevance of retrieved data, prioritizing content most likely to provide accurate and relevant answers. By allowing the LLM to focus on these top-ranked contexts during answer generation, it significantly enhances the accuracy and quality of responses.

B. Performance of modulor used for Generation

In examining the question-and-answer (Q&A) component, it is essential to collectively consider Tables VI, VII, VIII and XI. Table VI provides the test results when the human-annotated retrieval output is directly entered into the large model for Q&A purposes. Table VII and Table VIII illustrate the outcomes when the retriever employs the JINA-Large and BGE-Large models, respectively, to identify relevant sentences for the large model. Tables IX and X present the results of a Cognitive LLM evaluation conducted on the output generated by the XRAG system utilizing the BGE-Large retriever. In Table IX, the Dp-RMch metric scores below 0.5, while all other metrics exceed 0.8. This suggests that Dp-RMch is unsuitable for most non-critical scenarios, as it may misclassify responses that are semantically correct. The probability score P_{sc} in Table IX and Table X indicates that LLM testing depends on API requests, which may incur packet loss due to unstable requests, long processing times, timeout limits, or server-side errors. This reflects a particular weakness of LLM testing, even though the average request success rate reaches over 85%.

i. When the model parameter size is relatively small, directly supplying the large model with golden context may not yield optimal results for these datasets. Our analysis reveals that the experimental outcomes in Tables VII, and VIII generally surpass those in Table VI, with the exception of the results obtained using the DeepSeek-R1-70B model. The golden context typically comprises a single text passage (DQA and NQA) or two passages (HQA), which may provide inadequate information for the large model. Additionally, due to the limited reasoning capacity of small-scale LLMs, they may encounter difficulties in analyzing results from such concise texts, particularly for questions necessitating numerical computations and logical reasoning. Our retrieval model delivers the top k (top 3) results to the LLM, filtering and ranking context information to provide more comprehensive data. Some retrieved data, though not labeled as "golden context" for not being direct answers, are highly relevant to the query and contribute positively to the reasoning process. This observation implies that for any retrieval-augmented generation task, evaluating only the retrieval results may not accurately represent the overall RAG performance; concurrent evaluation of both retrieval and large model Q&A generation is imperative.

ii. Choosing the right type of LLM for a RAG task could be superior to merely increasing the model's parameters. When comparing with various LLMs, it was observed that the DeepSeek R1-7B and the Deepseek R1-70B inference model exhibited the poor performance. The high hallucination rate of the DeepSeek model series [51] is fundamentally at odds with the stringent accuracy demands of RAG scenarios. At Wikipedia

TABLE VI: Downstream RAG **Generation Performance** of LLMs, utilizing *Oracle* as the retrieval results. The conventional generation evaluation (ConG) is conducted. The top scores are emphasized in **bold**. The symbol $^\circ$ signifies metric values ranging from 0 to positive infinity, while all other values are within the $[0, 1]$ interval (%). The symbol $\downarrow$ denotes that lower metric values are preferable. The error value is the average of three trials, with a constant LLM Temperature of 0.

Datasets & Methods		Conventional Generation Evaluation (ConG Evaluator)								
Oracle: Golden-Context		ChrF	ChrF++	METEOR	R1	R2	RL	PPL$^\circ_\downarrow$	CER$^\circ_\downarrow$	WER$^\circ_\downarrow$
HQA	GPT-4o mini	21.75$_{\pm.024}$	20.01$_{\pm.012}$	21.62$_{\pm.008}$	12.93$_{\pm.007}$	05.32$_{\pm.003}$	12.71$_{\pm.005}$	111.98$_{\pm1.368}$	**11.08**$_{\pm.019}$	**09.75**$_{\pm.002}$
	llama3.1-8B	21.08$_{\pm.018}$	19.30$_{\pm.014}$	16.71$_{\pm.012}$	15.98$_{\pm.010}$	06.40$_{\pm.006}$	15.89$_{\pm.008}$	150.34$_{\pm1.600}$	14.92$_{\pm.017}$	12.15$_{\pm.009}$
	DeepSeek R1-7B	11.66$_{\pm.020}$	10.37$_{\pm.015}$	09.53$_{\pm.013}$	05.10$_{\pm.011}$	01.75$_{\pm.007}$	05.01$_{\pm.009}$	**69.22**$_{\pm1.450}$	23.39$_{\pm.021}$	18.12$_{\pm.008}$
	DeepSeek R1-70B	**41.23**$_{\pm.025}$	**40.06**$_{\pm.021}$	**45.26**$_{\pm.018}$	**32.22**$_{\pm.017}$	**21.09**$_{\pm.015}$	**32.00**$_{\pm.015}$	115.31$_{\pm1.950}$	13.00$_{\pm.014}$	09.80$_{\pm.006}$
DQA	GPT-4o mini	08.18$_{\pm.018}$	07.47$_{\pm.017}$	06.87$_{\pm.012}$	04.12$_{\pm.010}$	01.80$_{\pm.006}$	04.12$_{\pm.008}$	91.97$_{\pm1.400}$	**44.35**$_{\pm.017}$	**14.50**$_{\pm.009}$
	Llama3.1-8B	04.93$_{\pm.020}$	04.29$_{\pm.015}$	04.53$_{\pm.009}$	02.17$_{\pm.011}$	00.64$_{\pm.007}$	02.16$_{\pm.009}$	153.56$_{\pm1.712}$	76.50$_{\pm.021}$	23.63$_{\pm.011}$
	DeepSeek R1-7B	04.89$_{\pm.019}$	04.33$_{\pm.016}$	04.26$_{\pm.012}$	01.87$_{\pm.010}$	00.72$_{\pm.006}$	01.86$_{\pm.008}$	**49.51**$_{\pm1.350}$	90.94$_{\pm.020}$	28.58$_{\pm.007}$
	DeepSeek R1-70B	**17.08**$_{\pm.012}$	**18.40**$_{\pm.015}$	**26.18**$_{\pm.016}$	**14.36**$_{\pm.013}$	**04.80**$_{\pm.014}$	**14.28**$_{\pm.017}$	61.36$_{\pm1.552}$	63.34$_{\pm.013}$	19.43$_{\pm.005}$
NQA	GPT-4o mini	25.34$_{\pm.022}$	23.98$_{\pm.013}$	25.89$_{\pm.009}$	16.20$_{\pm.008}$	06.95$_{\pm.005}$	15.52$_{\pm.007}$	48.50$_{\pm1.239}$	**09.41**$_{\pm.020}$	**08.83**$_{\pm.006}$
	Llama3.1-8B	22.34$_{\pm.021}$	20.78$_{\pm.016}$	18.00$_{\pm.014}$	17.47$_{\pm.012}$	07.68$_{\pm.010}$	16.92$_{\pm.009}$	124.79$_{\pm1.807}$	14.19$_{\pm.018}$	11.10$_{\pm.013}$
	DeepSeek R1-7B	11.13$_{\pm.020}$	09.82$_{\pm.015}$	08.10$_{\pm.013}$	04.90$_{\pm.011}$	01.40$_{\pm.007}$	04.53$_{\pm.009}$	**40.97**$_{\pm1.331}$	20.71$_{\pm.021}$	16.93$_{\pm.008}$
	DeepSeek R1-70B	**28.71**$_{\pm.017}$	**27.72**$_{\pm.012}$	**33.64**$_{\pm.016}$	**18.82**$_{\pm.018}$	**10.31**$_{\pm.010}$	**18.24**$_{\pm.014}$	46.66$_{\pm1.564}$	14.79$_{\pm.019}$	13.06$_{\pm.007}$
FQA	GPT-4o mini	22.68$_{\pm.018}$	21.42$_{\pm.017}$	26.78$_{\pm.019}$	18.03$_{\pm.012}$	08.83$_{\pm.005}$	14.54$_{\pm.009}$	25.99$_{\pm1.279}$	74.47$_{\pm.970}$	49.81$_{\pm.482}$
	Llama3.1-8B	19.87$_{\pm.017}$	18.79$_{\pm.016}$	23.61$_{\pm.010}$	14.11$_{\pm.011}$	07.14$_{\pm.006}$	11.39$_{\pm.009}$	**18.23**$_{\pm1.017}$	57.31$_{\pm.538}$	46.67$_{\pm.412}$
	DeepSeek R1-7B	21.88$_{\pm.015}$	20.54$_{\pm.014}$	25.04$_{\pm.016}$	**25.04**$_{\pm.019}$	07.40$_{\pm.006}$	13.56$_{\pm.010}$	33.90$_{\pm1.170}$	**37.66**$_{\pm.329}$	30.79$_{\pm.272}$
	DeepSeek R1-70B	**23.34**$_{\pm.016}$	**22.13**$_{\pm.015}$	**29.10**$_{\pm.018}$	18.29$_{\pm.012}$	**08.97**$_{\pm.004}$	**15.12**$_{\pm.005}$	25.22$_{\pm1.064}$	39.88$_{\pm.314}$	**30.78**$_{\pm.371}$

TABLE VII: Downstream RAG **Generation Performance** of LLMs, utilizing *JINA-Large* as the retriever. Results underlined indicate superiority over RAG generation based on Oracle context, while **bold** results denote the best performance.

Datasets & Methods		Conventional Generation Evaluation (ConG Evaluator)								
Retriever: JINA-Large		ChrF	ChrF++	METEOR	R1	R2	RL	PPL$^\circ_\downarrow$	CER$^\circ_\downarrow$	WER$^\circ_\downarrow$
HQA	GPT-4o mini	26.65$_{\pm.030}$	25.20$_{\pm.025}$	29.81$_{\pm.020}$	16.87$_{\pm.015}$	08.82$_{\pm.010}$	16.72$_{\pm.005}$	95.72$_{\pm1.245}$	14.51$_{\pm.025}$	11.32$_{\pm.010}$
	Llama3.1-8B	**37.33**$_{\pm.010}$	**35.05**$_{\pm.018}$	**32.51**$_{\pm.022}$	**34.80**$_{\pm.013}$	**18.77**$_{\pm.012}$	**34.58**$_{\pm.007}$	173.88$_{\pm1.209}$	**09.50**$_{\pm.018}$	**07.83**$_{\pm.012}$
	DeepSeek R1-7B	19.28$_{\pm.022}$	18.18$_{\pm.018}$	22.62$_{\pm.012}$	10.90$_{\pm.008}$	04.97$_{\pm.004}$	11.04$_{\pm.009}$	77.40$_{\pm1.235}$	20.45$_{\pm.020}$	15.70$_{\pm.015}$
	DeepSeek R1-70B	23.81$_{\pm.025}$	22.55$_{\pm.016}$	27.99$_{\pm.018}$	15.32$_{\pm.010}$	08.10$_{\pm.006}$	15.17$_{\pm.011}$	**78.09**$_{\pm1.325}$	19.12$_{\pm.011}$	15.00$_{\pm.013}$
DQA	GPT-4o mini	08.80$_{\pm.023}$	08.36$_{\pm.018}$	09.66$_{\pm.012}$	04.80$_{\pm.008}$	02.00$_{\pm.001}$	04.80$_{\pm.006}$	55.93$_{\pm1.280}$	75.33$_{\pm.033}$	26.38$_{\pm.027}$
	Llama3.1-8B	**10.98**$_{\pm.028}$	**10.32**$_{\pm.022}$	**10.65**$_{\pm.017}$	**07.66**$_{\pm.012}$	**02.79**$_{\pm.006}$	**07.66**$_{\pm.009}$	164.35$_{\pm1.152}$	73.46$_{\pm.031}$	27.64$_{\pm.021}$
	DeepSeek R1-7B	07.17$_{\pm.029}$	06.92$_{\pm.018}$	08.44$_{\pm.013}$	03.78$_{\pm.007}$	01.16$_{\pm.005}$	03.77$_{\pm.008}$	60.70$_{\pm1.236}$	**69.78**$_{\pm.028}$	**23.20**$_{\pm.025}$
	DeepSeek R1-70B	07.75$_{\pm.026}$	07.82$_{\pm.017}$	09.52$_{\pm.019}$	04.66$_{\pm.014}$	01.48$_{\pm.008}$	04.64$_{\pm.012}$	**07.82**$_{\pm1.115}$	73.87$_{\pm.033}$	23.65$_{\pm.014}$
NQA	GPT-4o mini	29.37$_{\pm.024}$	28.29$_{\pm.022}$	32.61$_{\pm.018}$	19.49$_{\pm.014}$	11.12$_{\pm.009}$	19.90$_{\pm.007}$	53.16$_{\pm1.145}$	10.98$_{\pm.025}$	09.56$_{\pm.015}$
	Llama3.1-8B	**37.43**$_{\pm.020}$	**35.65**$_{\pm.028}$	**33.30**$_{\pm.022}$	**35.24**$_{\pm.017}$	**21.57**$_{\pm.014}$	**34.74**$_{\pm.009}$	170.36$_{\pm1.851}$	**10.82**$_{\pm.018}$	**08.82**$_{\pm.014}$
	DeepSeek R1-7B	21.16$_{\pm.015}$	20.10$_{\pm.020}$	22.50$_{\pm.016}$	12.93$_{\pm.009}$	06.29$_{\pm.004}$	12.43$_{\pm.008}$	46.83$_{\pm1.136}$	17.16$_{\pm.025}$	14.48$_{\pm.015}$
	DeepSeek R1-70B	24.20$_{\pm.023}$	23.17$_{\pm.015}$	27.39$_{\pm.010}$	15.57$_{\pm.010}$	08.08$_{\pm.013}$	14.99$_{\pm.009}$	**44.19**$_{\pm1.104}$	16.93$_{\pm.009}$	14.29$_{\pm.012}$
FQA	GPT-4o mini	**21.87**$_{\pm.020}$	**20.51**$_{\pm.017}$	**21.52**$_{\pm.019}$	**17.38**$_{\pm.020}$	**08.05**$_{\pm.011}$	**13.94**$_{\pm.014}$	41.23$_{\pm1.242}$	51.44$_{\pm.052}$	39.16$_{\pm.040}$
	Llama3.1-8B	17.33$_{\pm.017}$	16.22$_{\pm.018}$	16.25$_{\pm.017}$	14.16$_{\pm.016}$	06.69$_{\pm.008}$	11.74$_{\pm.010}$	39.35$_{\pm1.325}$	73.47$_{\pm.072}$	56.16$_{\pm.056}$
	DeepSeek R1-7B	18.88$_{\pm.017}$	17.61$_{\pm.019}$	17.86$_{\pm.018}$	14.33$_{\pm.012}$	06.03$_{\pm.008}$	11.39$_{\pm.010}$	34.59$_{\pm1.211}$	36.77$_{\pm.032}$	28.14$_{\pm.030}$
	DeepSeek R1-70B	21.06$_{\pm.023}$	19.63$_{\pm.015}$	20.90$_{\pm.010}$	16.36$_{\pm.010}$	07.21$_{\pm.013}$	12.71$_{\pm.009}$	**34.42**$_{\pm1.104}$	**29.01**$_{\pm.009}$	**23.04**$_{\pm.012}$

datasets, the most consistent performance was observed with the Llama3.1-8B model, which achieved optimal Q&A results when combined with either JINA-Large or BGE-Large retrieval outputs. Nonetheless, the perplexity (PPL) value of DeepSeek R1-7B and Deepseek R1-70B was the lowest. It is important to note that PPL and Q&A metrics assess different facets of the model: PPL emphasizes text fluency and generality, whereas ChrF and METEOR focus on the precise alignment between generated text and expected answers. While the DeepSeek series of models may not be closely aligned with expected answers in specific tasks, yet their strengths lie in superior semantic coherence and text fluency.

C. Performance of RAG Orchestrator

As shown in Table XI, RAG methods employing advanced orchestrators demonstrate substantial improvements in generation quality. In agentic RAG methodologies, SIM-RAG and Adapt-RAG demonstrate superior performance, whereas Self-RAG underperforms compared to the BGE retrieval results. RRFusion, as a traditional parallel retrieval method, exhibits a performance gap when compared to agentic RAG approaches such as Adapt-RAG and SIM-RAG.

i. Multi-hop question answering is more suitable for iterative orchestrators. In Table XI, SIM-RAG substantially outperforms other methods on the HQA dataset, which contains a high volume of multi-hop queries. For multi-hop question-answering tasks, an iterative orchestrator can progressively retrieve information from each hop of the multi-hop data through their cyclical process of optimizing retrieval results and generated content, until the retrieved content is sufficient to answer the question or no more information can be obtained.

ii. Conditional orchestrators often rely on heuristic decision logic, resulting in relatively poor transferability. In Table XI, Self-RAG demonstrates underwhelming per-

TABLE VIII: Downstream RAG **Generation Performance** of LLMs, utilizing *BGE-Large* as the retriever. Results underlined indicate superiority over RAG generation based on Oracle context, while **bold** results denote the best performance.

Datasets & Methods		Conventional Generation Evaluation (ConG Evaluator)								
Retriever: BGE-Large		ChrF	ChrF++	METEOR	R1	R2	RL	PPL$^{\circ}_{\downarrow}$	CER$^{\circ}_{\downarrow}$	WER$^{\circ}_{\downarrow}$
HQA	GPT-4o mini	28.23$_{\pm.030}$	26.85$_{\pm.025}$	33.07$_{\pm.020}$	17.96$_{\pm.015}$	09.51$_{\pm.010}$	17.86$_{\pm.005}$	90.51$_{\pm1.145}$	13.60$_{\pm.025}$	11.02$_{\pm.010}$
	Llama3.1-8B	**40.63**$_{\pm.010}$	**38.34**$_{\pm.018}$	**36.24**$_{\pm.022}$	**36.33**$_{\pm.013}$	**21.31**$_{\pm.012}$	**36.33**$_{\pm.007}$	181.71$_{\pm1.155}$	**10.40**$_{\pm.007}$	**08.10**$_{\pm.012}$
	DeepSeek R1-7B	20.02$_{\pm.022}$	18.76$_{\pm.018}$	23.71$_{\pm.012}$	11.40$_{\pm.008}$	05.19$_{\pm.004}$	11.33$_{\pm.009}$	81.34$_{\pm1.235}$	21.46$_{\pm.025}$	16.47$_{\pm.015}$
	DeepSeek R1-70B	26.06$_{\pm.023}$	24.78$_{\pm.012}$	31.08$_{\pm.015}$	17.22$_{\pm.010}$	08.95$_{\pm.017}$	17.17$_{\pm.010}$	**79.91**$_{\pm1.260}$	18.39$_{\pm.008}$	14.26$_{\pm.011}$
DQA	GPT-4o mini	08.89$_{\pm.025}$	08.48$_{\pm.018}$	10.38$_{\pm.012}$	05.04$_{\pm.008}$	02.11$_{\pm.004}$	05.03$_{\pm.006}$	**48.22**$_{\pm1.228}$	76.53$_{\pm.035}$	27.95$_{\pm.025}$
	Llama3.1-8B	**10.85**$_{\pm.028}$	**10.27**$_{\pm.022}$	**11.59**$_{\pm.017}$	**07.58**$_{\pm.012}$	**03.08**$_{\pm.006}$	**07.53**$_{\pm.009}$	148.43$_{\pm1.072}$	78.57$_{\pm.019}$	30.62$_{\pm.026}$
	DeepSeek R1-7B	07.49$_{\pm.029}$	07.20$_{\pm.018}$	08.63$_{\pm.013}$	03.18$_{\pm.007}$	00.93$_{\pm.005}$	03.16$_{\pm.008}$	56.18$_{\pm1.091}$	63.03$_{\pm.028}$	23.02$_{\pm.010}$
	DeepSeek R1-70B	08.67$_{\pm.021}$	08.66$_{\pm.014}$	10.58$_{\pm.017}$	04.99$_{\pm.013}$	01.62$_{\pm.016}$	04.98$_{\pm.017}$	65.68$_{\pm1.118}$	**61.99**$_{\pm.003}$	**20.22**$_{\pm.012}$
NQA	GPT-4o mini	29.37$_{\pm.024}$	28.29$_{\pm.022}$	32.61$_{\pm.018}$	19.49$_{\pm.014}$	11.12$_{\pm.009}$	18.99$_{\pm.007}$	53.16$_{\pm1.222}$	10.98$_{\pm.025}$	09.56$_{\pm.004}$
	Llama3.1-8B	**37.28**$_{\pm.020}$	**35.50**$_{\pm.028}$	**32.73**$_{\pm.022}$	**34.72**$_{\pm.017}$	**20.88**$_{\pm.014}$	**34.15**$_{\pm.009}$	158.86$_{\pm1.450}$	**10.60**$_{\pm.018}$	**08.84**$_{\pm.014}$
	DeepSeek R1-7B	23.51$_{\pm.015}$	22.50$_{\pm.020}$	26.39$_{\pm.016}$	14.48$_{\pm.009}$	07.64$_{\pm.004}$	14.18$_{\pm.008}$	50.35$_{\pm1.230}$	17.33$_{\pm.025}$	14.15$_{\pm.009}$
	DeepSeek R1-70B	25.52$_{\pm.023}$	24.47$_{\pm.018}$	29.35$_{\pm.019}$	16.41$_{\pm.017}$	08.27$_{\pm.016}$	15.78$_{\pm.010}$	**45.49**$_{\pm1.114}$	16.07$_{\pm.007}$	13.75$_{\pm.013}$
FQA	GPT-4o mini	**20.47**$_{\pm.019}$	**19.19**$_{\pm.017}$	**19.18**$_{\pm.020}$	**15.87**$_{\pm.015}$	**06.72**$_{\pm.011}$	**12.37**$_{\pm.013}$	**19.19**$_{\pm0.839}$	38.12$_{\pm.023}$	28.36$_{\pm.021}$
	Llama3.1-8B	15.55$_{\pm.020}$	14.51$_{\pm.028}$	14.89$_{\pm.022}$	12.11$_{\pm.017}$	04.74$_{\pm.014}$	09.22$_{\pm.009}$	30.67$_{\pm1.450}$	49.94$_{\pm.018}$	42.36$_{\pm.014}$
	DeepSeek R1-7B	19.04$_{\pm.015}$	17.67$_{\pm.020}$	16.48$_{\pm.016}$	13.44$_{\pm.009}$	05.52$_{\pm.004}$	10.88$_{\pm.008}$	38.17$_{\pm1.230}$	29.81$_{\pm.025}$	22.31$_{\pm.009}$
	DeepSeek R1-70B	19.82$_{\pm.015}$	18.47$_{\pm.016}$	19.14$_{\pm.012}$	14.04$_{\pm.012}$	05.91$_{\pm.007}$	11.20$_{\pm.010}$	32.78$_{\pm1.467}$	**26.33**$_{\pm.018}$	**20.20**$_{\pm.016}$

TABLE IX: Upstream & Downstream performance of the RAG framework using Cognitive LLM Evaluation focusing on Separate Retrieval and Generation Metrics. The second row indicates the mean success rate of API requests P_{sc} from three 100-entry samples, with green-highlighted results representing the average across 3-times sampling tests.

Datasets & Methods		CogL [Retrieval]		CogL [Generation]					
Retriever: BGE-Large		Up-CRel	Up-CCns	Dp-ARel	Dp-RCmp	Dp-RCnc	Dp-RRel	Dp-RVal	Dp-RMch
HQA	GPT-3.5 Turbo	0.7352$_{\pm0.012}$	0.9201$_{\pm0.023}$	0.9367$_{\pm0.034}$	0.7667$_{\pm0.045}$	0.9358$_{\pm0.056}$	0.7856$_{\pm0.067}$	0.9733$_{\pm0.078}$	0.1854$_{\pm0.089}$
		84.33$_{\pm1.234}$	89.67$_{\pm2.345}$	98.72$_{\pm1.456}$	98.72$_{\pm1.456}$	98.67$_{\pm1.678}$	99.00$_{\pm1.789}$	98.33$_{\pm1.890}$	98.67$_{\pm1.678}$
DQA	GPT-3.5 Turbo	0.6856$_{\pm0.121}$	0.3604$_{\pm0.142}$	0.9900$_{\pm0.263}$	0.8417$_{\pm0.084}$	0.9609$_{\pm0.105}$	0.8441$_{\pm0.226}$	0.9867$_{\pm0.147}$	0.2060$_{\pm0.068}$
		97.00$_{\pm1.176}$	94.33$_{\pm1.115}$	99.00$_{\pm1.303}$	93.31$_{\pm2.004}$	98.00$_{\pm1.223}$	98.33$_{\pm1.606}$	98.33$_{\pm1.200}$	89.33$_{\pm1.001}$
NQA	GPT-3.5 Turbo	0.8878$_{\pm0.031}$	0.4133$_{\pm0.052}$	0.9100$_{\pm0.073}$	0.8061$_{\pm0.094}$	0.8827$_{\pm0.015}$	0.8214$_{\pm0.036}$	0.9763$_{\pm0.057}$	0.4574$_{\pm0.078}$
		98.67$_{\pm1.678}$	94.33$_{\pm1.115}$	98.56$_{\pm1.121}$	98.33$_{\pm1.200}$	98.00$_{\pm1.505}$	98.33$_{\pm1.200}$	98.33$_{\pm1.200}$	94.33$_{\pm1.115}$

TABLE X: Upstream & Downstream performance of the RAG framework using Cognitive LLM Evaluation focusing on Combined Retrieval and Generation Metrics.

Datasets & Methods		CogL [Combined Retrieval & Generation]							
Retriever: BGE-Large		Dp-CPre	Dp-CRec	Dp-CRel	Up-RCns	Up-CUti	Up-FAcc	Dp-Fath	Dp-Hall
HQA	GPT-3.5 Turbo	0.9457$_{\pm0.022}$	0.6455$_{\pm0.034}$	0.6812$_{\pm0.046}$	0.8318$_{\pm0.058}$	0.6870$_{\pm0.070}$	0.7516$_{\pm0.082}$	0.9197$_{\pm0.094}$	0.5537$_{\pm0.106}$
		92.00$_{\pm1.345}$	99.67$_{\pm1.678}$	99.33$_{\pm1.456}$	99.67$_{\pm1.678}$	92.67$_{\pm1.789}$	92.00$_{\pm1.901}$	99.67$_{\pm1.678}$	99.33$_{\pm1.456}$
DQA	GPT-3.5 Turbo	0.8691$_{\pm0.112}$	0.6254$_{\pm0.134}$	0.8333$_{\pm0.156}$	0.7450$_{\pm0.178}$	0.6702$_{\pm0.190}$	0.5326$_{\pm0.202}$	0.7637$_{\pm0.214}$	0.7333$_{\pm0.226}$
		99.33$_{\pm1.456}$	97.00$_{\pm3.567}$	99.67$_{\pm1.678}$	99.67$_{\pm1.678}$	95.67$_{\pm1.901}$	98.33$_{\pm2.012}$	97.00$_{\pm2.123}$	95.27$_{\pm2.012}$
NQA	GPT-3.5 Turbo	0.9742$_{\pm0.033}$	0.8769$_{\pm0.045}$	0.9384$_{\pm0.057}$	0.8948$_{\pm0.069}$	0.6888$_{\pm0.081}$	0.8485$_{\pm0.093}$	0.9070$_{\pm0.105}$	0.3417$_{\pm0.117}$
		99.33$_{\pm1.456}$	99.33$_{\pm1.345}$	99.33$_{\pm1.456}$	98.67$_{\pm1.567}$	98.67$_{\pm1.678}$	97.33$_{\pm1.789}$	86.33$_{\pm1.901}$	95.27$_{\pm2.012}$

formance, which stems from the heuristic-based classifier within its conditional orchestrator. The classifier in conditional orchestrator often requires specific adjustments based on data characteristics and application scenarios, and fails to deliver effective classification when handling more complex datasets.

D. Latency and Token Consumption

As shown in Table XII, taking the vanilla RAG methods (e.g., BGE-Large and JINA-Large) as a reference, pre-retrieval methods such as SBPT (71k) and HyDE (280k) introduce only a slight increase of LLM token usage. While advanced retrieval methods (e.g., RRFusion and StParser) come at the cost of considerably greater retrieval latency (7119s for RRFusion, 1921s for StParser), they compensate by consuming fewer LLM tokens, thanks to heuristic optimizations applied during retrieval. Post-processors add some latency during the reranking stage. Orchestrators typically involve complex retrieval logic, including multiple retrieval steps and multiple rounds of LLM interactions, resulting in higher latency and greater token consumption. Considering the performance (as shown in Table XI and Table V) of retrieval modules alongside their latency and LLM token costs (as shown in Table XII), The rerank method is highly recommendable, as it consistently demonstrates stable performance gains across all datasets (e.g., Hit@1: +73.9% for BGE-RRK vs. BGE-Large on FQA) without incurring substantial increases in latency (+11.8% for BGE-RRK vs. BGE-Large) or token consumption (+1.2% for BGE-RRK vs. BGE-Large). In latency-insensitive scenarios, employing an orchestrator is also a favorable option, where a more sophisticated retrieval workflow can trade computational resources for enhanced retrieval and generation performance.

TABLE XI: Downstream RAG **Generation Performance** of various Orchestrators, utilizing *deepseek-r1-distill-llama-70b*, which has a 128K context window, as the generative model.

Datasets & Methods		Orchestrator	Conventional Generation Evaluation (ConG Evaluator)								
			ChrF	ChrF++	METEOR	R1	R2	RL	PPL$^{\circ}_{\downarrow}$	CER$^{\circ}_{\downarrow}$	WER$^{\circ}_{\downarrow}$
HQA	BGE-Large	Sequential	26.06$_{\pm.023}$	24.78$_{\pm.012}$	31.08$_{\pm.015}$	17.22$_{\pm.010}$	08.95$_{\pm.017}$	17.17$_{\pm.010}$	**79.91**$_{\pm1.260}$	18.39$_{\pm.008}$	14.26$_{\pm.011}$
	Self-RAG	Conditional	17.26$_{\pm.016}$	15.72$_{\pm.013}$	13.00$_{\pm.016}$	11.78$_{\pm.012}$	05.50$_{\pm.011}$	11.72$_{\pm.012}$	134.21$_{\pm1.853}$	**10.38**$_{\pm.012}$	**08.55**$_{\pm.011}$
	SIM-RAG	Iterative	**62.30**$_{\pm.018}$	**60.58**$_{\pm.019}$	**52.39**$_{\pm.020}$	**65.64**$_{\pm.015}$	**39.83**$_{\pm.017}$	**65.48**$_{\pm.009}$	267.62$_{\pm1.857}$	72.89$_{\pm.015}$	74.62$_{\pm.009}$
	RRFusion	Parallel	23.14$_{\pm.013}$	21.08$_{\pm.012}$	25.47$_{\pm.015}$	13.46$_{\pm.017}$	13.45$_{\pm.019}$	24.59$_{\pm.014}$	99.87$_{\pm1.516}$	18.39$_{\pm.009}$	14.26$_{\pm.017}$
	Adapt-RAG	Hybrid	31.20$_{\pm.018}$	29.91$_{\pm.012}$	33.56$_{\pm.016}$	23.43$_{\pm.018}$	14.10$_{\pm.016}$	23.17$_{\pm.012}$	106.73$_{\pm1.683}$	19.16$_{\pm.019}$	14.13$_{\pm.019}$
FQA	BGE-Large	Sequential	19.82$_{\pm.016}$	18.47$_{\pm.013}$	**19.14**$_{\pm.014}$	14.04$_{\pm.009}$	05.91$_{\pm.008}$	11.20$_{\pm.007}$	**32.78**$_{\pm0.614}$	26.33$_{\pm.018}$	20.20$_{\pm.015}$
	Self-RAG	Conditional	11.98$_{\pm.011}$	10.53$_{\pm.008}$	07.87$_{\pm.011}$	07.13$_{\pm.005}$	02.38$_{\pm.005}$	06.09$_{\pm.012}$	192.74$_{\pm1.476}$	05.40$_{\pm.011}$	04.69$_{\pm.011}$
	SIM-RAG	Iterative	**21.12**$_{\pm.015}$	**20.19**$_{\pm.015}$	15.81$_{\pm.017}$	**23.05**$_{\pm.013}$	**09.73**$_{\pm.014}$	**21.92**$_{\pm.007}$	242.16$_{\pm2.008}$	**00.95**$_{\pm.003}$	**01.10**$_{\pm.004}$
	RRFusion	Parallel	18.83$_{\pm.015}$	17.31$_{\pm.013}$	15.67$_{\pm.011}$	13.60$_{\pm.012}$	05.21$_{\pm.007}$	10.91$_{\pm.010}$	38.83$_{\pm0.536}$	35.38$_{\pm.020}$	26.84$_{\pm.017}$
	Adapt-RAG	Hybrid	19.97$_{\pm.014}$	18.43$_{\pm.013}$	18.07$_{\pm.011}$	14.57$_{\pm.012}$	05.86$_{\pm.008}$	12.16$_{\pm.009}$	54.08$_{\pm0.733}$	19.83$_{\pm.016}$	15.21$_{\pm.014}$

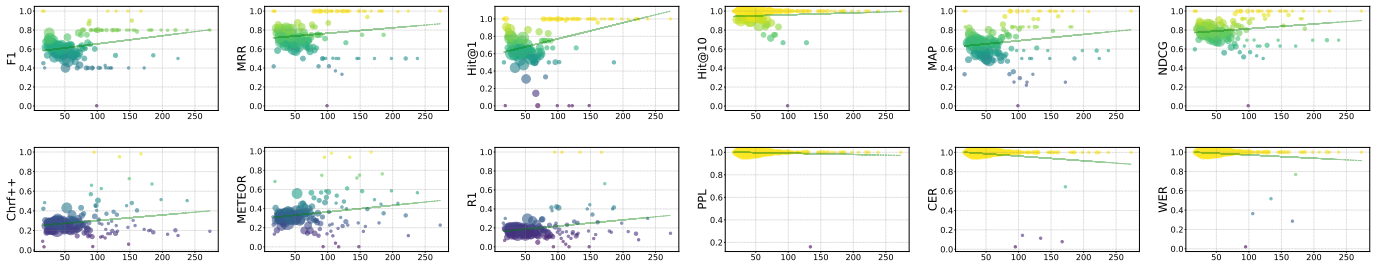


Fig. 5: Comparison of retrieval and generation effects as query length varies. Experiments conducted on the HQA dataset (Test).

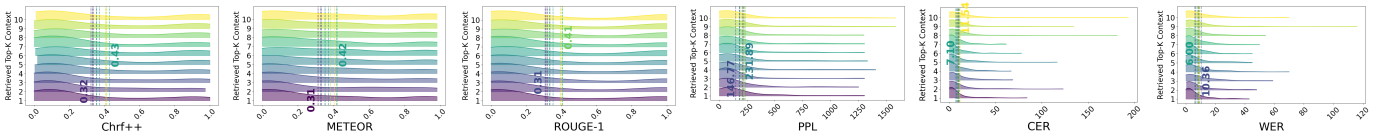


Fig. 6: Ridge plots of the impact of varying numbers of retrieved contexts on Q&A performance in the HQA dataset (Test). The chart shows maximum and minimum values.

TABLE XII: Latency and total token consumption summary of models on the full test set of the FQA dataset. All methods were evaluated under identical hardware (RTX 4090, 20 vCPU Intel(R) Xeon(R) Platinum 8470Q, 30GB RAM).

Methods	Latency(s)					Token(k)
	Index	Pre-retrieval	Retrieval	Post-processor	Generation	
BGE-Large	2087	✗	1641	✗	296	161
JINA-Large	2105	✗	1459	✗	256	165
SBPT	2087	183	1489	✗	301	232
HyDE	2087	492	1410	✗	288	441
RRFusion	2087	✗	7119	✗	256	98
StParser	2087	✗	1921	✗	159	63
BGE-RRK	2087	✗	1836	1232	235	163
JINA-RRK	2087	✗	2106	1387	253	165
Methods	**Latency(s)**					**Token(k)**
	Index	**Critic**	**Classifier**	**Retrieval**	**Generation**	
Adapt-RAG	2087	1215	1215	1215	3118	179
SIM-RAG	2087	59	✗	4085	5346	871
Self-RAG	2087	✗	42	1881	225	98

VI. Additional Discussion & Analysis

Q1: Whether longer queries have a greater impact? The illustration (as shown in Fig. 5) indicates a positive correlation between query length and the performance of several metrics, notably Hit@1 and METEOR, which demonstrate an ascending trajectory with increased query length. This correlation implies that the utilization of longer queries may augment the efficacy of retrieval and query-answering systems. Such enhancements are likely due to the enriched information that longer queries provide, facilitating a more precise interpretation of queries.

Q2:Whether more context has a greater impact? The illustration depicted in Fig. 6 demonstrates that the performance of query-answering systems remains relatively stable as the number of retrieval contexts increases. This observation suggests that augmenting the quantity of retrieval contexts may not significantly enhance model performance. Upon analyzing the mean value ranges, it is evident that for instance, in the ROUGE-1 metric, the mean value only moderately increases from 0.31 to 0.41 with the augmentation of retrieval contexts. This modest variation implies that the impact of increasing the number of contexts on performance improvement is limited. Furthermore, the observation of maximum values reaching 1.0 in multiple figures indicates that the key to enhancing performance may lie in improving the precision of retrieval. This involves identifying a few contexts most relevant to the query, rather than merely increasing the number of retrieved contexts. The ChrF++ and METEOR metrics exhibit relatively

TABLE XIII: Performance of different retrieval models on subsets of varying difficulty (Easy/Med./Hard) in HotpotQA, utilizing GPT-4o-mini as the generative model.

Difficulty & Models		Conventional Retrieval Evaluation (ConR Evaluator)			
		F1	Hit@1	Hit@10	MAP
Easy	BGE-Large	36.21	97.00	99.00	18.39
	JINA-Large	35.48	88.00	99.00	17.17
Med.	BGE-Large	32.38	86.00	100.00	16.68
	JINA-Large	29.30	73.00	96.00	14.56
Hard	BGE-Large	28.74	70.00	93.00	14.66
	JINA-Large	27.02	61.00	91.00	12.59

TABLE XIV: Performance of different Generation modules on subsets of varying difficulty (Easy/Med./Hard) in HotpotQA, utilizing BGE-Large as the generative model.

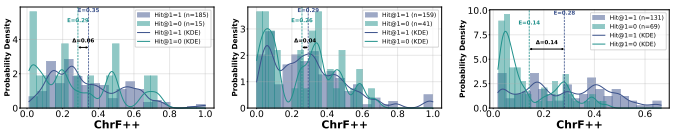
Difficulty & Models		Conventional Generation Evaluation (ConG Evaluator)			
		ChrF	METEOR	R1	CER$^{\circ}_{\downarrow}$
Easy	GPT-4o mini	35.56	43.36	25.02	11.27
	DeepSeek R1-70B	30.07	37.89	20.55	18.97
Med.	GPT-4o mini	32.53	39.70	20.79	14.70
	DeepSeek R1-70B	26.85	34.72	16.56	20.22
Hard	GPT-4o mini	25.42	28.52	16.25	13.16
	DeepSeek R1-70B	22.93	28.12	14.31	17.86

stable performance across varying numbers of retrieval contexts. This suggests that these metrics are not particularly sensitive to variations in the number of contexts. Consequently, this underscores the importance of focusing on enhancing the relevance and precision of retrieval results in the optimization of the retrieval module within RAG systems, rather than increasing the quantity of retrieved contexts.

Q3: Whether problem hard induce challenges in both retrieval and generation processes? As shown in Tables XIII and XIV, consistent with expectations, both retrieval and generative models achieve higher scores on easy datasets compared to harder datasets. From an evaluation perspective, more stringent metrics—such as F1 and Hit@1 for retrieval, and ChrF and METEOR for generation—are notably sensitive to dataset difficulty (e.g., On the Hit@1 metric, BGE-Large scores 97.00 on the easy dataset but only 70.00 on the hard one, a decline of approximately 28%). This sensitivity occurs because stricter metrics are more effective at detecting performance declines, which are inherent as question difficulty increases.

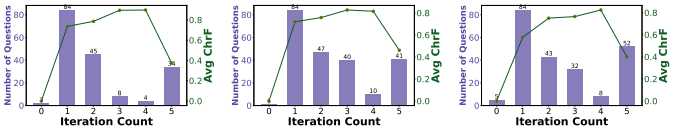
Q4: Whether superior retrieval results ensures superior generation? As shown in Figure 7,on the hard dataset, when the model hits at Hit@1, the mean of Chrf++ metric (0.28) is 0.14 higher than Chrf++ when the model non-hits (0.14). This indicates that successful retrieval substantially enhances answer quality for difficult questions. On the easy and medium datasets, the gap in answer quality between Hit@1 hits and non-hits is relatively small (0.06 for easy, 0.04 for medium). Moreover, when Hit@1 hits, there are a small number of samples with poor answer quality (chrF < 0.1), indicating that a good retrieval result does not necessarily yield a correct answer. This explains the phenomenon observed in Section V-B: "Retrieved context outperforms golden context".

Q5: Whether more iterations of RAG orchestrators lead to better generation results? As shown in Figure 8, the number of queries requiring 5 iterations was higher on the hard dataset (52) than on the easy (34) and medium (41) datasets, indicating that more challenging subsets require greater iteration counts. When the iteration limit has not been reached ($1 \leq$ count≤ 4), the quality of answer results improves markedly with increased iterations on the three subsets of HotpotQA dataset. This validates the suitability of iterative orchestration for multi-hop question answering tasks. For questions that are too difficult to answer by SIM-RAG, unsatisfactory answers may still be obtained after reaching the maximum iteration limit, which is why performance degrades significantly once the iteration count reaches 5.



(a) Easy (b) Medium (c) Hard

Fig. 7: The relationship between retrieval success and generation quality is examined for the subsets (easy, medium, hard) of HotpotQA, using GPT-4o-mini as the generative model.



(a) Easy (b) Medium (c) Hard

Fig. 8: The relationship between generation quality and SIM-RAG (Maximum iteration limit: 5) iterative rounds is examined for the subsets (easy, medium, hard) of HotpotQA.

VII. Conclusion

This work introduces XRAG, an open-source framework for benchmarking advanced RAG systems, which enables researchers to construct RAG systems more efficiently from datasets. It facilitates the selection of appropriate retrieval modules and logic based on data characteristics, and allows for the evaluation of retrieval performance.

In the future, we plan to enhance XRAG in the following aspects: i. Enable support for RAG component training to handle more complex data scenarios. ii. Integrate a wider range of benchmark datasets for evaluation, such as OpenQA [52], [53], long-form Q&A [54], [55], and multiple-choice Q&A [56], [57]. iii. Add the robustness evaluation related to dirty and malformed queries.

We encourage contributions from the open-source community to advance the XRAG framework. Our goal is to refine the XRAG framework by delivering a more efficient and dependable platform with comprehensive evaluation and development tools for RAG research community.

AI-Generated Content Acknowledgement

In the preparation of this paper, Deepseek-R1 was used exclusively for the purpose of polishing and refining the linguistic style of certain sentences to improve readability and fluency in section I, III, and VII. All core intellectual contributions, including but not limited to:

- Original Ideas and Novel Viewpoints: Any innovative concepts, hypotheses, and critical analysis are the product of the authors' own work.
- Experimental Data and Results: All code, figures, findings, and interpretations thereof were generated, designed, collected, and analyzed by the authors.
- Substantive Descriptions of the Work: The description of the methodology, experimental procedures, and discussion of the research implications were entirely authored by the research team.

References

[1] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, "Active retrieval augmented generation," in *EMNLP*. Association for Computational Linguistics, 2023, pp. 7969–7992. [Online]. Available: https://aclanthology.org/2023.emnlp-main.495

[2] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "REALM: retrieval-augmented language model pre-training," *CoRR*, vol. abs/2002.08909, 2020. [Online]. Available: https://arxiv.org/abs/2002.08909

[3] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, Q. Guo, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *CoRR*, vol. abs/2312.10997, 2023. [Online]. Available: https://arxiv.org/abs/2312.10997

[4] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre, "Improving language models by retrieving from trillions of tokens," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 2206–2240. [Online]. Available: https://proceedings.mlr.press/v162/borgeaud22a.html

[5] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," in *ICLR*. OpenReview.net, 2024. [Online]. Available: https://doi.org/10.18653/v1/2024.naacl-long.463

[6] A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, "Agentic retrieval-augmented generation: A survey on agentic rag," 2025. [Online]. Available: https://arxiv.org/abs/2501.09136

[7] H. Han, Y. Wang, H. Shomer, K. Guo, J. Ding, Y. Lei, M. Halappanavar, R. A. Rossi, S. Mukherjee, X. Tang, Q. He, Z. Hua, B. Long, T. Zhao, N. Shah, A. Javari, Y. Xia, and J. Tang, "Retrieval-augmented generation with graphs (graphrag)," *CoRR*, vol. abs/2501.00309, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2501.00309

[8] Q. Mao, Q. Zhang, H. Hao, Z. Han, R. Xu, W. Jiang, Q. Hu, Z. Chen, T. Zhou, B. Li, Y. Song, J. Dong, J. Li, and P. S. Yu, "Privacy-preserving federated embedding learning for localized retrieval-augmented generation," 2025. [Online]. Available: https://arxiv.org/abs/2504.19101

[9] W. Du, T. Luo, Z. Qiu, Z. Huang, Y. Shen, R. Cheng, Y. Guo, and J. Fu, "Stacking your transformers: A closer look at model growth for efficient llm pre-training," 2024. [Online]. Available: https://arxiv.org/abs/2405.15319

[10] S. Lu, H. Wang, Y. Rong, Z. Chen, and Y. Tang, "Turborag: Accelerating retrieval-augmented generation with precomputed kv caches for chunked text," 2024. [Online]. Available: https://arxiv.org/abs/2410.07590

[11] X. Zhang, Y. Song, Y. Wang, S. Tang, X. Li, Z. Zeng, Z. Wu, W. Ye, W. Xu, Y. Zhang *et al.*, "Raglab: A modular and research-oriented unified framework for retrieval-augmented generation," *arXiv preprint arXiv:2408.11381*, 2024. [Online]. Available: https://arxiv.org/abs/2408.11381

[12] H. Chase, "LangChain," Oct. 2022. [Online]. Available: https://github.com/langchain-ai/langchain

[13] J. Liu, "LlamaIndex," 11 2022. [Online]. Available: https://github.com/jerryjliu/llama_index

[14] P. Izsak, M. Berchansky, D. Fleischer, and R. Laperdon, "fastRAG: Efficient Retrieval Augmentation and Generation Framework," Feb. 2023. [Online]. Available: https://github.com/IntelLabs/fastrag

[15] Y. Hoshi, D. Miyashita, Y. Ng, K. Tatsuno, Y. Morioka, O. Torii, and J. Deguchi, "Ralle: A framework for developing and evaluating retrieval-augmented large language models," in *EMNLP (Demos)*. Association for Computational Linguistics, 2023, pp. 52–69. [Online]. Available: https://arxiv.org/abs/2308.10633

[16] X. Yu, Y. Lu, and Z. Yu, "Localrqa: From generating data to locally training, testing, and deploying retrieval-augmented QA systems," *CoRR*, vol. abs/2403.00982, 2024. [Online]. Available: https://arxiv.org/abs/2403.00982

[17] B. Kim and J. Kim, "AutoRAG," Oct. 2024. [Online]. Available: https://github.com/Marker-Inc-Korea/AutoRAG

[18] J. Jin, Y. Zhu, X. Yang, C. Zhang, and Z. Dou, "Flashrag: A modular toolkit for efficient retrieval-augmented generation research," *CoRR*, vol. abs/2405.13576, 2024. [Online]. Available: https://arxiv.org/abs/2405.13576

[19] A. Kim, M. Slot, D. G. Andersen, and A. Pavlo, "Anarchy in the database: A survey and evaluation of database management system extensibility," *Proc. VLDB Endow.*, vol. 18, no. 6, pp. 1962–1976, 2025. [Online]. Available: https://www.vldb.org/pvldb/vol18/p1962-kim.pdf

[20] J. Zhang, G. Li, and J. Su, "SAGE: A framework of precise retrieval for RAG," in *41st IEEE International Conference on Data Engineering, ICDE 2025, Hong Kong, May 19-23, 2025*. IEEE, 2025, pp. 1388–1401. [Online]. Available: https://doi.org/10.1109/ICDE65448.2025.00108

[21] Y. Xiao, J. Cao, Y. Jiang, and J. Zhou, "ARAG: analysis and retrieval augmented generation for comprehensive reasoning over socioeconomic data," in *41st IEEE International Conference on Data Engineering, ICDE 2025, Hong Kong, May 19-23, 2025*. IEEE, 2025, pp. 4624–4627. [Online]. Available: https://doi.org/10.1109/ICDE65448.2025.00368

[22] Y. Mao, Y. Zhu, Q. Liu, P. Liu, H. Yu, and Y. Gao, "scrag: an efficient retrieval augmented generation system for scrna-seq data analysis," in *41st IEEE International Conference on Data Engineering, ICDE 2025, Hong Kong, May 19-23, 2025*. IEEE, 2025, pp. 4560–4563. [Online]. Available: https://doi.org/10.1109/ICDE65448.2025.00352

[23] C. Zhang, G. Li, J. Zhang, X. Zhang, and J. Feng, "HTAP databases: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 6410–6429, 2024. [Online]. Available: https://doi.org/10.1109/TKDE.2024.3389693

[24] C. Zhang, G. Li, and T. Lv, "Hybench: A new benchmark for HTAP databases," *Proc. VLDB Endow.*, vol. 17, no. 5, pp. 939–951, 2024. [Online]. Available: https://www.vldb.org/pvldb/vol17/p939-zhang.pdf

[25] X. Zhou, Z. Sun, and G. Li, "DB-GPT: large language model meets database," *Data Sci. Eng.*, vol. 9, no. 1, pp. 102–111, 2024. [Online]. Available: https://doi.org/10.1007/s41019-023-00235-6

[26] H. S. Zheng, S. Mishra, X. Chen, H. Cheng, E. H. Chi, Q. V. Le, and D. Zhou, "Take a step back: Evoking reasoning via abstraction in large language models," in *ICLR*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=3bq3jsvcQ1

[27] L. Gao, X. Ma, J. Lin, and J. Callan, "Precise zero-shot dense retrieval without relevance labels," in *ACL (1)*. Association for Computational Linguistics, 2023, pp. 1762–1777. [Online]. Available: https://aclanthology.org/2023.acl-long.99

[28] G. V. Cormack, C. L. A. Clarke, and S. Büttcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *SIGIR*. ACM, 2009, pp. 758–759. [Online]. Available: https://plg.uwaterloo.ca/~gvcormac/cormacksigir09-rrf.pdf

[29] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," 2024.

[30] S. B. Islam, M. A. Rahman, K. Hossain, E. Hoque, S. Joty, and M. R. Parvez, "Open-rag: Enhanced retrieval-augmented reasoning with open-source large language models," *arXiv preprint arXiv:2410.01782*, 2024.

[31] D. Yang, L. Zeng, J. Rao, and Y. Zhang, "Knowing you don't know: Learning when to continue search in multi-round rag through self-practicing," in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025, pp. 1305–1315.

[32] L. Li, D. Shi, J. Zhou, X. Wei, M. Yang, S. Jin, and S. Yang, "Retrieval feedback memory enhancement large model retrieval generation method," *arXiv preprint arXiv:2508.17862*, 2025.

[33] Z. Yao, W. Qi, L. Pan, S. Cao, L. Hu, W. Liu, L. Hou, and J. Li, "Seakr: Self-aware knowledge retrieval for adaptive retrieval augmented generation," *arXiv preprint arXiv:2406.19215*, 2024.

[34] S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. C. Park, "Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity," *arXiv preprint arXiv:2403.14403*, 2024.

[35] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. [Online]. Available: https://aclanthology.org/D18-1259.pdf

[36] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner, "DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs," in *Proc. of NAACL*, 2019. [Online]. Available: https://aclanthology.org/N19-1246

[37] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. P. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov, "Natural questions: a benchmark for question answering research," *Trans. Assoc. Comput. Linguistics*, vol. 7, pp. 452–466, 2019. [Online]. Available: https://aclanthology.org/Q19-1026/

[38] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W. Yih, "REPLUG: retrieval-augmented black-box language models," in *NAACL-HLT*. Association for Computational Linguistics, 2024, pp. 8371–8384. [Online]. Available: https://doi.org/10.18653/v1/2024.naacl-long.463

[39] P. Islam, A. Kannappan, D. Kiela, R. Qian, N. Scherrer, and B. Vidgen, "Financebench: A new benchmark for financial question answering," 2023. [Online]. Available: https://arxiv.org/abs/2311.11944

[40] D. Cavusoglu, U. Sert, S. Sen, and S. Altinuc, "Jury: A comprehensive evaluation toolkit," 2023. [Online]. Available: https://arxiv.org/html/2310.02040v2

[41] M. Popovic, "chrf: character n-gram f-score for automatic MT evaluation," in *WMT@EMNLP*. The Association for Computer Linguistics, 2015, pp. 392–395. [Online]. Available: https://aclanthology.org/W15-3049

[42] ——, "chrf++: words helping character n-grams," in *WMT*. Association for Computational Linguistics, 2017, pp. 612–618. [Online]. Available: https://www.statmt.org/wmt17/pdf/WMT70.pdf

[43] S. Banerjee and A. Lavie, "METEOR: an automatic metric for MT evaluation with improved correlation with human judgments," in *IEEvaluation@ACL*. Association for Computational Linguistics, 2005, pp. 65–72. [Online]. Available: https://aclanthology.org/W05-0909

[44] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013/

[45] K. Pillutla, S. Swayamdipta, R. Zellers, J. Thickstun, S. Welleck, Y. Choi, and Z. Harchaoui, "MAUVE: measuring the gap between neural text and human text using divergence frontiers," in *NeurIPS*, 2021, pp. 4816–4828. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/260c2432a0eecc28ce03c10dadc078a4-Paper.pdf

[46] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker, "Perplexity—a measure of the difficulty of speech recognition tasks," *The Journal of the Acoustical Society of America*, vol. 62, no. S1, pp. S63–S63, 1977. [Online]. Available: https://pubs.aip.org/asa/jasa/article/62/S1/S63/642598/Perplexity-a-measure-of-the-difficulty-of-speech

[47] A. C. Morris, V. Maier, and P. D. Green, "From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition," in *INTERSPEECH*. ISCA, 2004, pp. 2765–2768. [Online]. Available: https://www.isca-archive.org/interspeech_2004/morris04_interspeech.pdf

[48] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[49] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy, "Challenges and applications of large language models," 2023. [Online]. Available: https://arxiv.org/abs/2307.10169

[50] Q. Ran, Y. Lin, P. Li, J. Zhou, and Z. Liu, "Numnet: Machine reading comprehension with numerical reasoning," in *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 2474–2484. [Online]. Available: https://aclanthology.org/D19-1251/

[51] M. A. Cusumano, "Deepseek inside: Origins, technology, and impact," *Commun. ACM*, vol. 68, no. 7, p. 18–22, Jun. 2025. [Online]. Available: https://doi.org/10.1145/3732796

[52] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, "When not to trust language models: Investigating effectiveness of parametric and non-parametric memories," in *ACL (1)*. Association for Computational Linguistics, 2023, pp. 9802–9822. [Online]. Available: https://aclanthology.org/2023.acl-long.546

[53] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension," in *ACL (1)*. Association for Computational Linguistics, 2017, pp. 1601–1611. [Online]. Available: https://aclanthology.org/P17-1147

[54] I. Stelmakh, Y. Luan, B. Dhingra, and M. Chang, "ASQA: factoid questions meet long-form answers," in *EMNLP*. Association for Computational Linguistics, 2022, pp. 8273–8288. [Online]. Available: https://aclanthology.org/2022.emnlp-main.566

[55] S. Min, K. Krishna, X. Lyu, M. Lewis, W. Yih, P. W. Koh, M. Iyyer, L. Zettlemoyer, and H. Hajishirzi, "Factscore: Fine-grained atomic evaluation of factual precision in long form text generation," in *EMNLP*. Association for Computational Linguistics, 2023, pp. 12 076–12 100. [Online]. Available: https://arxiv.org/abs/2305.14251

[56] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *ICLR*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/pdf?id=d7KBjmI3GmQ

[57] Z. Huang, A. Wu, J. Zhou, Y. Gu, Y. Zhao, and G. Cheng, "Clues before answers: Generation-enhanced multiple-choice QA," in *NAACL-HLT*. Association for Computational Linguistics, 2022, pp. 3272–3287. [Online]. Available: https://aclanthology.org/2022.naacl-main.239/