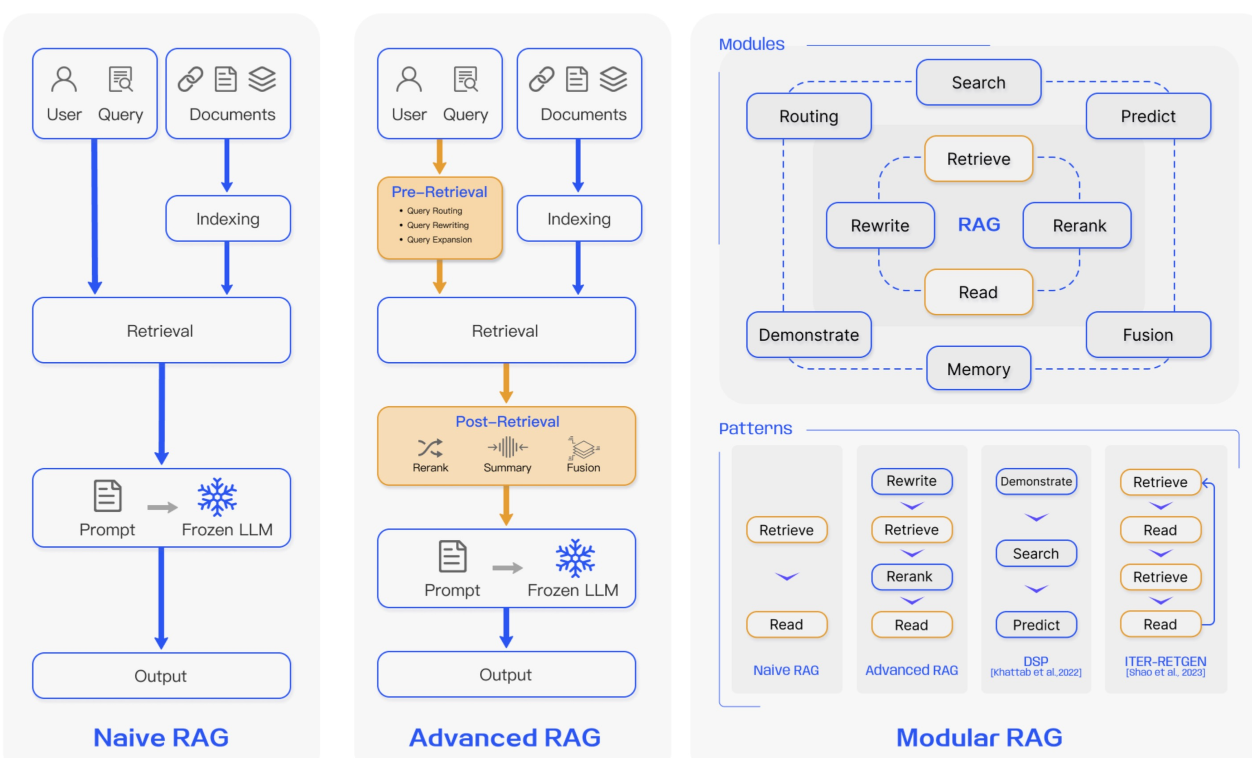
Revisiting the Solution of Meta KDD Cup 2024: CRAG

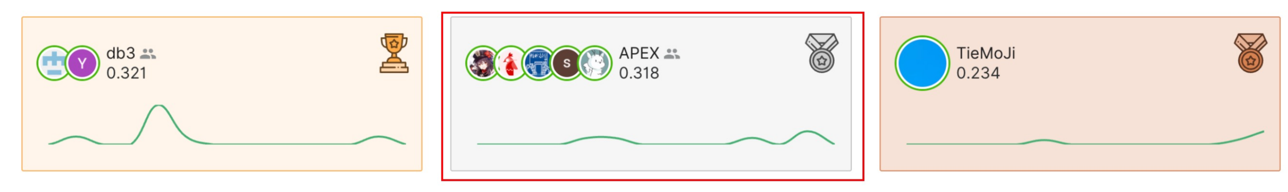
Jie Ouyang, Yucong Luo, Mingyue Cheng, Daoyu Wang, Shuo Yu, Qi Liu, Enhong Chen

State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China, Hefei, China

Introduction

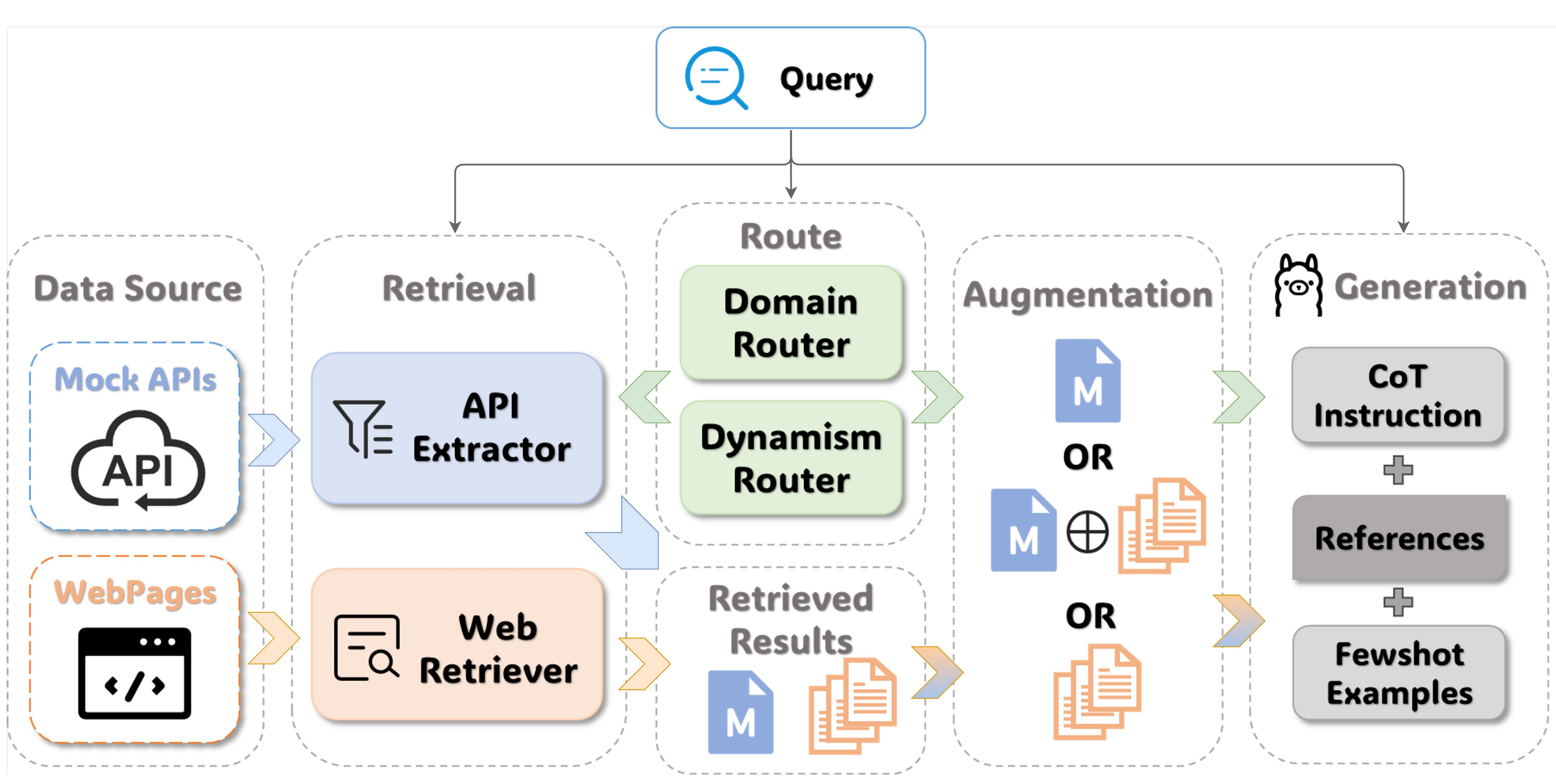
Meta introduced the CRAG with QA pairs, web pages, and Mock APIs to simulate Knowledge Graph (KG) search, hosting the KDD CUP 2024 Challenge to address these issues. We achieved a ranking of **2nd out of 384** teams in task2, task3 and over all ranking in the automatic evaluation.





RAG (Retrieval-Augmented Generation) enhances LLM with external knowledge. It evolves from Naive to Advanced to Modular stages:
1. **Naive RAG**: Uses "Retrieve-Read" framework, facing retrieval precision and content hallucination issues during generation.
2. **Advanced RAG**: Improves retrieval quality with pre- and post-retrieval strategies, focusing on essential information selection.
3. **Modular RAG**: Integrates specialized modules and new patterns for enhanced retrieval relevance and task performance, promoting adaptability and efficiency across tasks.

Methodology



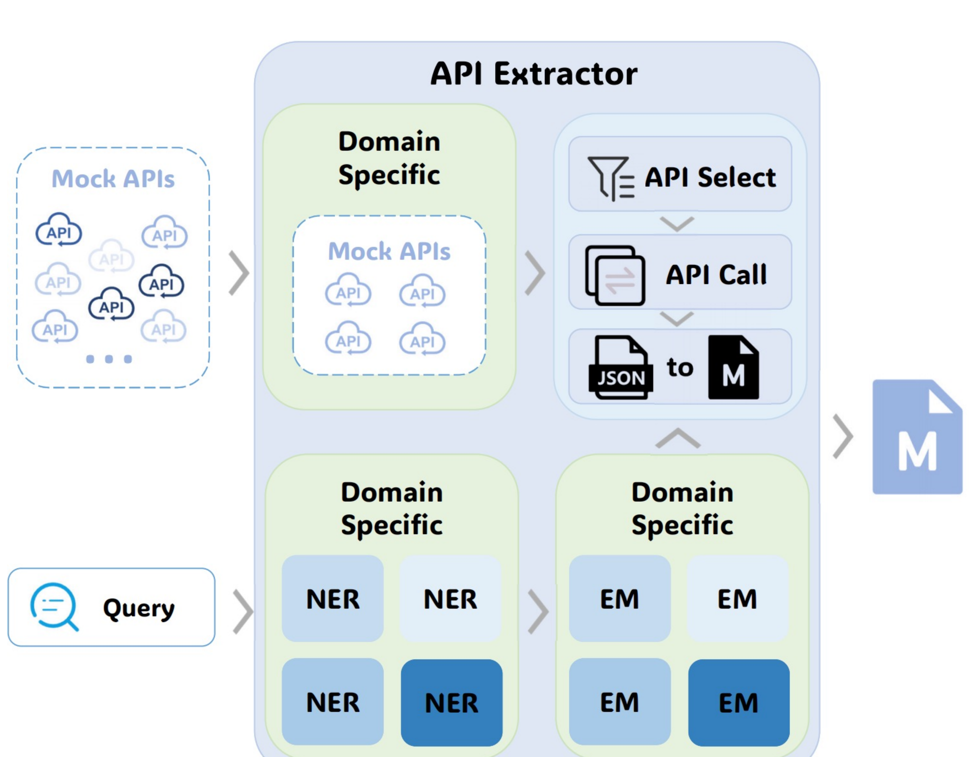
Framework of our solution

Stage	Knowledge Acquisition and Parsing	Knowledge Storage and Retrieval	Inference Based on LLM
Phase 1a	BoilerPy for web page parsing without KG	All-MiniLM as embedding model	Basic RAG instructions
Phase 1b	Newspaper3k for web page parsing, formatting KG-retrieved info as text with web page	BGE as embedding model, BGE-Rerank for reranking	CoT inference with intermediate reasoning in outputs; Agent inference
Phase 2 (Final Solution)	**Newspaper3k** for web page parsing, **manually refining KG info** separately from web page	**BGE** as embedding model, **BGE-Rerank** for reranking, adding KG info after web page retrieval	**CoT inference** with **few-shot examples**

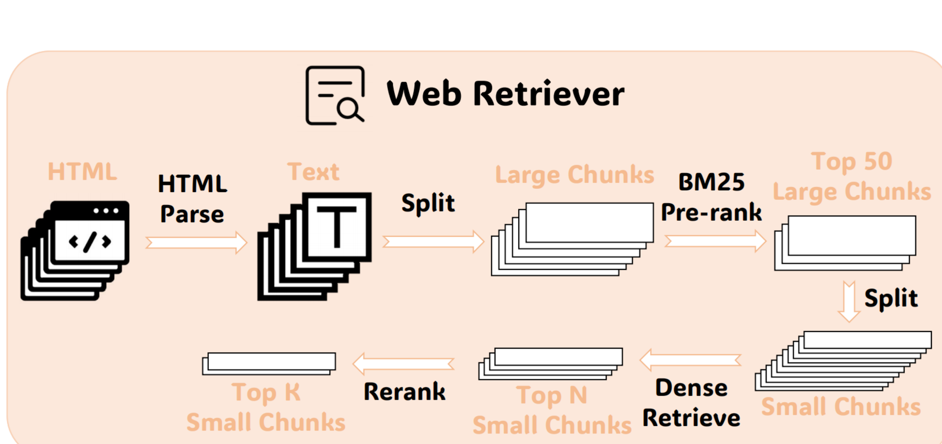
Progression of Stages in the Competition

Interesting finding: Despite **agents** being effective for reasoning, **time constraints and uncontrolled iteration cycles** led us to choose **CoT** with **few-shot examples** for in-context learning.

Key Components



We use an **embedding model for web page** retrieval and the KG API with **NER** to identify entities and access domain-specific APIs, then combine results with web references.



For task 3, out of 50 candidate web pages, **BM25** first retrieves the top 50 large chunks, then **Dense retrieve** selects the top k small chunks.

- **CoT Instruction**: Created a Chain of Thought (CoT) prompt to guide the LLM in structured responses based on web references.
- **In-context Learner**: Developed adaptive few-shot examples to enhance the LLM's accuracy in handling factual errors across domains.

Experiments

Table 3: Overall Preformance of our solutions on all 3 Tasks.

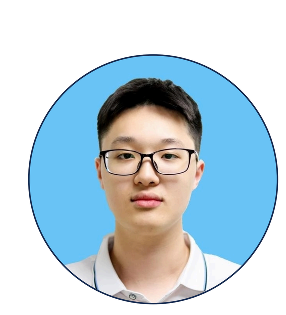
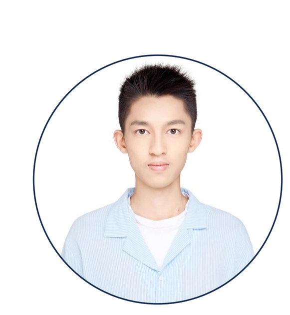
	Score(%)	Accuracy(%)	Hallucination(%)	Missing(%)
LLM Only	-7.29	28.01	35.30	36.69
RAG Baseline	-6.78	34.79	41.58	**23.63**
Task 1	11.82	29.98	18.16	51.86
Task 2	31.22	46.75	**15.54**	37.71
Task 3	**31.66**	**48.21**	16.56	35.23

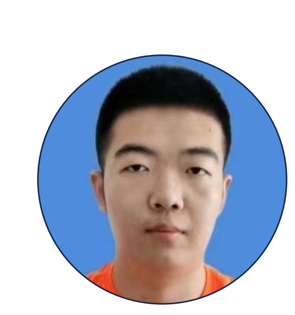
Table 4: Ablation Study for Prompt Construction on Task 2.

	Score(%)	Accuracy(%)	Hallucination(%)	Missing(%)
w/o Fewshot&CoT	25.53	52.08	26.55	**21.37**
w/o Fewshot	27.13	51.35	24.22	24.43
w/o CoT	28.52	**53.32**	24.80	21.88
Task 2	**31.22**	46.75	**15.54**	37.71

Code

Code : https://github.com/USTCAGI/CRAG-in-KDD-Cup2024



Yucong Luo   Jie Ouyang   Daoyu Wang   Shuo Yu

Supervisor: Mingyue Cheng, Qi Liu, Enhong Chen