Corrective Retrieval Augmented Generation

- Large Language Models (LLMs) inherently hallucinate due to reliance on parametric knowledge.
- Retrieval-Augmented Generation (RAG) is a practical complement but is vulnerable to irrelevant retrieved documents.
- **Corrective Retrieval Augmented Generation (CRAG)** is proposed to enhance generation robustness.
- CRAG includes a **lightweight retrieval evaluator** that assesses retrieved document quality for a query, triggering different knowledge retrieval actions based on a confidence degree.
- To overcome limitations of static corpora, CRAG utilizes **large-scale web searches** to augment retrieval results.
- A **decompose-then-recompose algorithm** is designed to focus on key information and filter irrelevant content from retrieved documents.
- CRAG is plug-and-play, seamlessly integrating with various RAG-based approaches.
- Experiments show **CRAG significantly improves the performance of RAG-based methods across different generation tasks.**

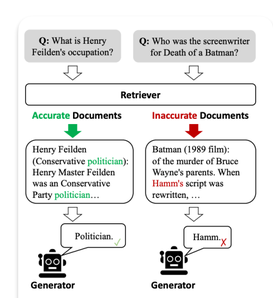
Figure 1: The examples show that a low-quality retriever is prone to introducing a substantial amount of irrelevant information, impeding the generators from acquiring accurate knowledge and potentially misleading them.

LLM Hallucinations: Large Language Models (LLMs) exhibit impressive abilities but inevitably suffer from **hallucinations** due to **factual errors** and their inability **to guarantee accuracy solely from parametric knowledge.**

RAG Limitations:Retrieval-Augmented Generation (RAG) integrates external knowledge but its effectiveness relies heavily on the relevance and accuracy of retrieved documents. Low-quality retrievals can introduce irrelevant information, mislead generators, and cause hallucinations.

Problem with Current RAG:Most conventional RAG approaches indiscriminately incorporate retrieved documents, regardless of relevance. They also treat entire documents as reference knowledge, even though much of the text is often non-essential.

★**Introducing CRAG:**This paper proposes : Corrective Retrieval-Augmented Generation (CRAG)** to self-correct retriever results and improve document utilization for augmentation, specifically addressing scenarios where the retriever returns inaccurate results.

★ Key points of CRAG:

* **Lightweight Retrieval Evaluator:** Assesses the overall quality of retrieved documents for a query, providing a confidence degree (Correct, Incorrect, Ambiguous).

- **Dynamic Knowledge Retrieval:** For "Incorrect" or "Ambiguous" cases, CRAG integrates **large-scale web searches** as an extension to static corpora, broadening the scope and diversity of retrieved information.

* **Decompose-then-Recompose Algorithm:** Refines retrieved information by selectively focusing on key insights and eliminating redundant/non-essential contexts, optimizing data utilization.

* **Plug-and-Play & Effectiveness:** CRAG is designed to be plug-and-play, demonstrated by its implementation into existing RAG frameworks (RAG, Self-RAG). Experiments show it significantly improves RAG-based approach performance.

Overview

CRAG

CRAG aims to **correct** and refine retrieved information to enhance the quality of generated output

Core process of CRAG

Evaluator -> Action Trigger -> refinement

CRAG
1. Initial Retrieval: Documents are fetched by another retriever, such as Contriever.
2. Retrieval Evaluator (T5-large based): This component scores the retrieved documents based on their relevance to the query.
3. Action Trigger: Based on these scores, the system determines whether a document is 'Correct', 'Incorrect', or 'Ambiguous', and then initiates an appropriate corrective action accordingly (e.g., use the document as is, discard and re-retrieve, or perform further verification).
4. Generation: An LLM generates the final answer based on the corrected/selected documents.

Overview of CRAG

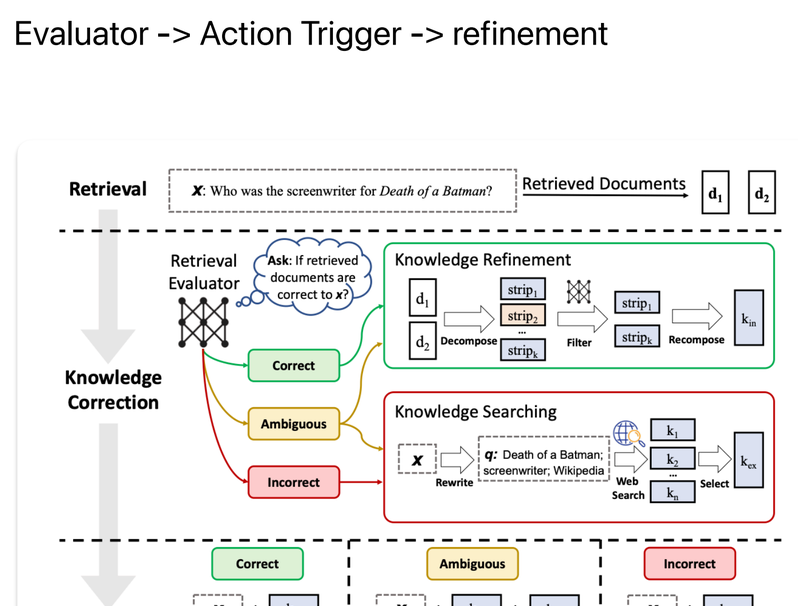

Figure 2: An overview of the proposed CRAG at inference. A retrieval evaluator is constructed to evaluate the relevance of the retrieved documents to the input, and estimate a confidence degree based on which different knowledge retrieval actions of {Correct, Incorrect, Ambiguous} can be triggered.

```
Algorithm 1: CRAG Inference
Require : E (Retrieval Evaluator), W (Query Rewriter), G (Generator)
Input   : x (Input question), D = {d1, d2, ..., dk} (Retrieved documents)
Output  : y (Generated response)
1  score_i = E evaluates the relevance of each pair (x, d_i), d_i ∈ D
2  Confidence = Calculate and give a final judgment based on {score1, score2, ...score_k}
   // Confidence has 3 optional values: [CORRECT], [INCORRECT] or [AMBIGUOUS]
3  if Confidence == [CORRECT] then
4      Internal_Knowledge = Knowledge_Refine(x, D)
5      k = Internal_Knowledge
6  else if Confidence == [INCORRECT] then
7      External_Knowledge = Web_Search(W Rewrites x for searching)
8      k = External_Knowledge
9  else if Confidence == [AMBIGUOUS] then
10     Internal_Knowledge = Knowledge_Refine(x, D)
11     External_Knowledge = Web_Search(W Rewrites x for searching)
12     k = Internal_Knowledge + External_Knowledge
13 end
14 G predicts y given x and k
```

Retrieval Evaluator

CRAG designs corrective strategies to enhance generation robustness during inference. For an input query and retrieved documents , a **lightweight retrieval evaluation estimates relevance score**

Usefulness" Determination
Based on this relevance score, the system determines whether "this search result is useful or not.

Determination = threshold
If the relevance score S is less than the threshold T(S<T), then the system deems the search result "not useful.

To train retrieval evaluator:

Base model : T5-large

Data Used for fineTuning:

- Positive Samples: High-quality, relevant passages obtained from the PopQA dataset through the "golden subject wiki title" for each question. These served as the "relevance signals."
- Negative Samples: Documents randomly sampled from the retrieval results that were, in fact, irrelevant to the input query.

Scoring

Quantified into three confidence degrees - correct, incorrect and ambiguous

three types of actions are designed and triggered accordingly where the upper and lower thresholds are set.

Correct :
If the confidence score is higher than the upper threshold, the retrieved document is identified as Correct
-> implies that no corrective action is needed, and the document can be used as is.

Incorrect:
Incorrect if below the lower threshold. Otherwise, a more soft and intermediate action,

judgment triggers a specific, often more drastic, corrective action (e.g., discarding the document, re-retrieving information from scratch).

Ambiguous:

Ambiguous is executed. Each retrieved document is conducted individually and integrated eventually.

triggers a "soft and intermediate action" (e.g., further verification, re-ranking, or using a different generation strategy), as the document is neither clearly correct nor clearly incorrect.

To evaluate the overall performance of CRAG

2. LLMs Evaluated:
- **Public LLMs:**
- LLaMA2-7B, 13B
- Alpaca-7B, 13B (instruction-tuned)
- CoVE65B (introduces iterative engineering)

- **Proprietary LLMs (trained with proprietary data):**
- LLaMA2-c13B (chat version)
- ChatGPT
- Perplexity.ai (InstructGPT-based production search system)

3. Retrieval Methods/Baselines:
- **Baselines without retrieval:** Standard LLMs as listed above.
- **Standard RAG:**
- LLaMA2-7B, 13B
- Alpaca-7B, 13B
- LLaMA2-7B instruction-tuned in Self-RAG
- The standard RAG approach involves prepending top retrieved documents to the query before the LM generates output. Google Search API is used for retrieval.

- **Advanced RAG:**
- **SAIL (Luo et al., 2023):** Instruction-tuned an LM on Alpaca data with retrieved documents inserted before instructions.
- **Self-RAG (Asai et al., 2024):** Tuned LLaMA2 on instruction-tuning data with reflection tokens labeled by GPT-4.
- **Retrieval-augmented baselines trained with private data:**
- Ret-ChatGPT
- Ret-LLaMA-chat
- Perplexity.ai

4. Evaluation Metrics:
- **Accuracy:** For PopQA, Pub, and ARC datasets.
- **FactScore:** For the Bio dataset.

Method	PopQA (Accuracy)	Bio (FactScore)	Pub (Accuracy)	ARC (Accuracy)
LMs trained with propriety data				
LLaMA2-c$_{13B}$	20.0	55.9	49.4	38.4
Ret-LLaMA2-c$_{13B}$	51.8	79.9	52.1	37.9
ChatGPT	29.3	71.8	70.1	**75.3**
Ret-ChatGPT	50.8	-	54.7	**75.3**
Perplexity.ai	-	71.2	-	-
Baselines without retrieval				
LLaMA2$_{7B}$	14.7	44.5	34.2	21.8
Alpaca$_{7B}$	23.6	45.8	49.8	45.0
LLaMA2$_{13B}$	14.7	53.4	29.4	29.4
Alpaca$_{13B}$	24.4	50.2	55.5	54.9
CoVE$_{65B}$	-	71.2	-	-
Baselines with retrieval				
LLaMA2$_{7B}$	38.2	78.0	30.0	48.0
Alpaca$_{7B}$	46.7	76.6	40.2	48.0
SAIL	-	-	69.2	48.4
LLaMA2$_{13B}$	45.7	77.5	30.2	26.0
Alpaca$_{13B}$	46.1	77.7	51.1	57.6
LLaMA2-hf-7b				
RAG	50.5	44.9	48.9	43.4
CRAG	54.9	47.7	59.5	53.7
Self-RAG*	29.0	32.2	0.7	23.9
Self-CRAG	49.0	69.1	0.6	27.9
SelfRAG-LLaMA2-7b				
RAG	52.8	59.2	39.0	53.2
CRAG	59.8	74.1	**75.6**	68.6
Self-RAG	54.9	81.2	72.4	67.3
Self-CRAG	**61.8**	**86.2**	74.8	67.2

Results :
- CRAG consistently and significantly outperforms standard RAG across all four datasets

- **SelfRAG-LLaMA2-7b base:** Improves PopQA by 7.0%, Bio (FactScore) by 14.9%, PubHealth by 36.6%, and Arc-Challenge by 15.4%.
- **LLaMA2-hf-7b base:** Improves PopQA by 4.4%, Bio (FactScore) by 2.8%, and Arc-Challenge by 10.3%.

- **Self-CRAG vs. Self-RAG:** Self-CRAG also demonstrates substantial improvements over the state-of-the-art Self-RAG.
- **LLaMA2-hf-7b base:** Improves PopQA by 20.0%, Bio (FactScore) by 36.9%, and Arc-Challenge by 4.0%.
- **SelfRAG-LLaMA2-7b base:** Improves PopQA by 6.9%, Bio (FactScore) by 5.0%, and PubHealth by 2.4%.

- **RAG (Retrieval-Augmented Generation):** External information retrieval + LLM answer generation.
- **CRAG (Contextualized Retrieval Augmented Generation):** In RAG, it evaluates the quality of the retrieved information and adjusts the LLM's answer generation strategy accordingly (smarter utilization of information).
- **Self-RAG:** In RAG, it trains the LLM to "reflect" and judge the necessity of retrieval and the quality of the answer itself (proactive role of the LLM).
- **Self-CRAG:** Aims for the most optimized answer generation by combining CRAG's precise information management with Self-RAG's self-reflection capabilities.

Results

	LLaMA2-hf-7b	SelfRAG-LLaMA2-7b
CRAG	54.9	59.8
w/o. Correct	53.2	58.3
w/o. Incorrect	54.4	59.5
w/o. Ambiguous	54.0	59.0
Self-CRAG	49.0	61.8
w/o. Correct	43.6	59.6
w/o. Incorrect	47.7	60.8
w/o. Ambiguous	48.1	61.5

	LLaMA2-hf-7b	SelfRAG-LLaMA2-7b
CRAG	54.9	59.8
w/o. refinement	49.8	54.2
w/o. rewriting	51.7	56.2
w/o. selection	50.9	58.6
Self-CRAG	49.0	61.8
w/o. refinement	35.9	52.2
w/o. rewriting	37.2	58.4
w/o. selection	24.9	57.9

	LLaMA2-hf-7b	SelfRAG-LLaMA2-7b
PopQA		
CRAG	54.9	59.8
RAG	50.5	52.8
RAG w. web	52.2	53.8
Self-CRAG	49.0	61.8
Self-RAG	29.0	54.9
Self-RAG w. web	24.9	57.9

Table 5: Comparison results between CRAG, Self-CRAG and RAG, Self-RAG with the same input in terms of accuracy.

Figure 3: The generation performance of Self-RAG and Self-CRAG given different retrieval performance on the PopQA dataset with SelfRAG-LLaMA-7b. The lower horizontal line demonstrates the performance of the generator without retrieval.

★Process

User Input

→ Initial Retrieval

→ **Retrieval Evaluator (and optional Web Search)**

→ **Decompose (of retrieved documents)**

→ **Filtering/Scoring (of decomposed strips)**

→ **Recompose (of relevant strips)**

→ LLM Generation

Step 1: User Input

Step 2 : Initial Retrieval
:Based on the user's query, relevant documents are retrieved from an external database (e.g., a vector DB-FAISS, Pinecone, Vewaviate, Milvus, Chroma..).

Step 3: Retrieval Evaluator
The quality of the **entire retrieved documents** (relevant/irrelevant) is assessed.
If the quality of the retrieved documents is deemed low, CRAG can trigger a web search to find better information => corrective part

Step 4: Decompose
- The **documents** obtained from the initial retrieval or web search are broken down into smaller "knowledge strips."

Step 5: Filtering and Scoring
- Each "knowledge strip" is then **scored for relevance**, and strips with low relevance to the user's query are **filtered out**.
- *A Retrieval Evaluator might be involved here again, or a separate module might perform this role.*

Step 6: Recompose
- The filtered, i.e., highly relevant, "knowledge strips" are **combined again to form the "internal knowledge"** that the LLM will utilize.

Step 7: LLM Generation

```
# Conceptual Pseudocode for Self-CRAG

# Define hypothetical functions and classes

class LLM:
    def generate(self, prompt: str, reflection_tokens=False) -> str:
        """
        The LLM generates text.
        If reflection_tokens=True, it can include Self-RAG-like reflection tokens.
        """
        pass # Actual LLM API call logic

    def assess_document_quality(self, query: str, document: str) -> str:
        """
        Acts as CRAG's Retrieval Assessor. Evaluates document quality as 'High', 'Medium', 'Low', etc.
        """
        pass # LLM-based document quality assessment logic

    def self_reflect(self, prompt: str) -> dict:
        """
        Generates Self-RAG's self-reflection tokens.
        E.g., {'is_relevant': True, 'is_supportive': False, 'is_factual': True}
        """
        pass # LLM-based reflection token generation logic

class Retriever:
    def retrieve_documents(self, query: str, top_k: int = 5) -> list[str]:
        """
        Retrieves relevant documents based on the query.
        """
        pass # Vector DB query, web search API call, etc.

def run_self_crag(user_query: str, llm: LLM, retriever: Retriever) -> str:
    """
    Pseudocode representing the overall workflow of Self-CRAG.
    """

    print(f"User Query: {user_query}")

    # 1. Self-RAG's initial judgment: Is retrieval needed?
    # In practice, the LLM makes this decision after seeing the initial question, simplified here.
    # Process where LLM analyzes the question and generates Self-RAG Reflection Tokens (initial)
    initial_reflection = llm.self_reflect(f"Does the question '{user_query}' require external knowledge retrieval?")
    if not initial_reflection.get('requires_retrieval', True): # Hypothetical token
        print("Initial reflection: No retrieval needed. Generating directly...")
        return llm.generate(f"Question: {user_query}")

    print("Initial reflection: Retrieval seems necessary. Proceeding with retrieval...")

    # 2. Document Retrieval
    retrieved_documents = retriever.retrieve_documents(user_query)
    if not retrieved_documents:
        print("No documents retrieved. Generating based on LLM's internal knowledge.")
        return llm.generate(f"Question: {user_query}")

    final_answer = ""
    for doc in retrieved_documents:
        print(f"\nProcessing retrieved document: {doc[:100]}...") # Print only a portion of the document

        # 3. CRAG's Document Quality Assessment (Retrieval Assessor)
        doc_quality_assessment = llm.assess_document_quality(user_query, doc)
        print(f"CRAG Document Quality: {doc_quality_assessment}")

        # 4. Self-RAG's Reflection on Retrieval Results
        retrieval_reflection = llm.self_reflect(f"Query: {user_query}\nDocument: {doc}\nIs this document relevant and supportive?")
        print(f"Self-RAG Retrieval Reflection: {retrieval_reflection}")

        # 5. Determine Generation Strategy by combining CRAG and Self-RAG judgments (Adaptive Generation)
        generation_strategy = "internal_knowledge" # Default strategy

        if doc_quality_assessment == "High" and retrieval_reflection.get('is_relevant') and retrieval_reflection.get('is_supportive'):
            generation_strategy = "fully_utilize_document"
            print("Strategy: High quality document, fully utilize.")
            prompt = f"Based on the following document, answer the question:\nDocument: {doc}\nQuestion: {user_query}"
            generated_text = llm.generate(prompt, reflection_tokens=True) # Generate including reflection tokens
        elif doc_quality_assessment == "Medium" and retrieval_reflection.get('is_relevant'):
            generation_strategy = "combine_with_internal"
            print("Strategy: Medium quality document, combine with internal knowledge.")
            prompt = f"Considering the following document, answer the question. You may also use your internal knowledge:\nDocument: {doc}\nQuestion: {user_query}"
            generated_text = llm.generate(prompt, reflection_tokens=True)
        else: # Low quality, or not relevant/supportive based on Self-RAG reflection
            generation_strategy = "re_evaluate_or_skip"
            print("Strategy: Low quality/irrelevant document, re-evaluate or skip.")
            # In practice, this could involve further searching, trying other documents, or generating with LLM's internal knowledge only.
            prompt = f"You received a document that was not very helpful. Please try to answer the question based on your general knowledge if possible, or indicate if more information is needed:\nQuestion: {user_query}"
            generated_text = llm.generate(prompt, reflection_tokens=True)

        # 6. Self-RAG's Final Reflection on the Generated Answer
        final_generation_reflection = llm.self_reflect(f"Question: {user_query}\nGenerated Answer: {generated_text}\nIs this answer factual and complete?")
        print(f"Self-RAG Final Generation Reflection: {final_generation_reflection}")

        if final_generation_reflection.get('is_factual') and final_generation_reflection.get('is_complete'):
            final_answer = generated_text
            print("Final answer deemed factual and complete. Stopping.")
            break # Stop if a satisfactory answer is obtained
        else:
            print("Answer not yet satisfactory. Trying next document or refining...")
            # In practice, this would involve logic to search again, try different strategies, or refine the answer.

    if not final_answer:
        print("Could not find a satisfactory answer using retrieved documents. Generating best effort answer.")
        final_answer = llm.generate(f"Please provide the best possible answer to: {user_query}")

    return final_answer

# --- Usage Example (Not actually executable) ---
# Create hypothetical LLM and Retriever objects (in reality, these would be API calls to OpenAI, HuggingFace, etc.)
# my_llm = LLM()
# my_retriever = Retriever()

# answer = run_self_crag("What is the capital of France?", my_llm, my_retriever)
# print(f"\nFinal Self-CRAG Answer: {answer}")
```

Conclusion

The CRAG aims to address the issue of LLM producing hallucinations or factually incorrect statements
The "decompose-recompse" algorithm plays a crucial role

Decomposing : broken down into smaller, more granular pieces of information, referred to as "knowledge strips
Filtering : A retrieval evaluator then assesses the relevance of each individual knowledge strip to the user's query. Irrelevant strips are filtered out, ensuring that only the most pertinent information is retained.
Recomposing: The relevant knowledge strips are then concatenated to form a refined set of "internal knowledge". This newly assembled information is what is ultimately used to generate the response

1) RAG (**Retrieval-Augmented Generatio**)

Objective: This technology helps LLMs (Large Language Models) answer questions not by solely relying on their pre-trained knowledge, but by searching for external documents (information) and generating more accurate and factual responses based on that retrieved information.

Objective: This technology helps LLMs (Large Language Models) answer questions not by solely relying on their pre-trained knowledge, but by searching for external documents (information) and generating more accurate and factual responses based on that retrieved information.

Type to enter text

Process:
1. **Retrieval:** Documents relevant to the user's query are searched from an external database (e.g., web search engines, Wikipedia, academic papers, etc.).
2. **Generation:** The retrieved document content and the user's query are fed together into the LLM to generate the answer.

Advantages: Reduces hallucination, allows for the incorporation of up-to-date information.
Disadvantages: The quality of the generated answer can vary significantly depending on the quality of the retrieved documents. Problems can arise if irrelevant documents are retrieved or if the retrieved documents themselves contain errors.

Decompose-then-Recompose
evaluates the overall quality of retrieved documents and can even trigger web searches to find better information if the initial retrieval results are suboptimal.

Decompose-then-Recompose"

Decompose:
- Break down a complex and challenging large problem into smaller, more manageable, and independent parts (sub-problems or components).
- Each sub-problem is much simpler and clearer than the original problem, making it easier to solve or understand individually.
- During this process, it's crucial to identify the core and peripheral aspects of the problem, and to understand their interdependencies.

ReCompose:
- Individually solve or optimize each of the decomposed sub-problems or components.
- Subsequently, combine these individually solved parts to construct a complete solution for the original large problem.
- At this stage, it's important to consider how each part interacts and contributes to the overall system, integrating them harmoniously.

2. CRAG (Contextualized Retrieval Augmented Generation)

CRAG is a proposed method to improve upon the shortcomings of RAG. Specifically, it focuses on more intelligently utilizing and managing the quality of retrieved documents.
Core Idea: It evaluates how useful retrieved documents are (e.g., informativeness, relevance) and adjusts the LLM's answer generation strategy based on that assessment.
Process (Additions to RAG):
1. **Retrieval:** Documents are retrieved in the same way as with RAG.
2. **Retrieval Assessor:** The "quality" of the retrieved documents is evaluated. For example, it assesses how relevant the document is to the query, whether it contains sufficient information, etc.
3. **Adaptive Generation:** The LLM changes its answer generation strategy based on the evaluation results:
   - **High-quality documents:** Actively utilizes the retrieved documents to generate the answer.
   - **Low-quality documents:** Does not rely on the retrieved documents, instead using the LLM's own knowledge, or even refusing to generate an answer at all.
   - **Medium-quality documents:** Refers to the retrieved documents but combines them with the LLM's own knowledge, or may attempt further searches.
4.

Advantages: Enhances the accuracy of LLM answers by increasing the reliability of the retrieved information. It can filter out "useless information" or process it differently.
Analogy: If RAG is like "referring to all search results for now," CRAG is like "assessing the reliability of search results, picking only the trustworthy ones, or even adopting a completely different strategy."

3. Self-RAG

Self-RAG is an RAG model trained to allow the LLM itself to perform "reflection" or "evaluation" on its retrieval and generation processes.
Core Idea: As the LLM generates an answer, it simultaneously makes **meta-judgments (reflection)** such as "Does this answer require retrieval?", "Is this search result helpful?", or "Is this answer factual?".
Process:
1. **Question Analysis and Retrieval Decision:** The LLM analyzes the query and autonomously decides, "This question requires external retrieval."
2. **Retrieval:** Based on its decision, it retrieves documents.
3. **Search Result Evaluation and Answer Generation (Self-Reflection & Generation):** The LLM generates an answer based on the retrieved documents, but at the same time, it is trained to generate specific "tokens" (special markers within the prompt), such as:
   - [Is_Relevant]: Is the search result relevant to the question?
   - [Is_Supportive]: Does the search result support the answer?
   - [Is_Factually_Correct]: Is the generated answer factually accurate?

Through these "reflection tokens," the LLM provides feedback to itself during the generation process, thereby improving quality.
4.

Advantages: The LLM can actively control the retrieval process and autonomously evaluate and improve the quality of its generated answers.
Disadvantages: The LLM must be specifically trained (instruction-tuned) to understand and generate these "reflection tokens." This means it's difficult to directly apply it to existing, general LLMs.

4. Self-CRAG

As the name suggests, Self-CRAG is a combination of CRAG and Self-RAG.
Objective: To create synergy by adding CRAG's ability to "more meticulously manage the quality of retrieved information" to Self-RAG's ability to "self-evaluate and reflect."
Core Idea: It uses Self-RAG's self-reflection mechanism, but like CRAG, it fine-tunes the answer generation strategy based on the quality of the retrieved documents.
Process:
1. **Self-RAG's Retrieval and Reflection:** Similar to Self-RAG, the LLM autonomously decides the necessity of retrieval and generates reflection tokens after retrieval.
2. **CRAG's Quality Evaluation and Adaptive Generation:** CRAG's "search result evaluation" logic is added here. Beyond simply "relevant/not relevant," it more precisely evaluates the informativeness, reliability, etc., of the retrieved documents, and then fine-tunes the answer generation strategy based on that evaluation.
   - For example, even if Self-RAG determines "this search result is relevant," Self-CRAG further evaluates the "depth" or "accuracy" of that search result in a CRAG-like manner. Instead of just using it, it reacts more intelligently, for instance, by thinking "this information is insufficient, so I need to search more" or "this information is excellent, so it's enough on its own."
3.

Advantages: By combining Self-RAG's proactivity with CRAG's meticulous information management capabilities, it aims for the most powerful and accurate answer generation.
Analogy: If Self-RAG is like "asking myself if I did well," Self-CRAG is like "asking myself if I did well, and at the same time, more meticulously checking how trustworthy the materials I referred to were."

```
+---------------------+
|  1. User Input      |
|  "What is the       |
|  capital of France?"|
+----------+----------+
           |
           v
+----------+----------+
| 2. Call run_self_crag |
|     Function        |
| (Passes User Query) |
+----------+----------+
           |
           v
+------------------------------------------------------------------------+
| 3. Execute run_self_crag Function Internal Logic (Complex Self-CRAG & CRAG Logic) |
| +--------------------------------------------------------------------+ |
| | 3.1. LLM's Initial Self-Judgment (Start of Self-RAG)               | |
| |    - "Does this question require external retrieval?"              | |
| |      (e.g., 'requires_retrieval': True)                            | |
| +--------------------------------------------------------------------+ |
|         | (If deemed necessary)                                        |
|         v                                                              |
| +--------------------------------------------------------------------+ |
| | 3.2. Document Retrieval                                            | |
| |    - Retriever searches for relevant documents from external databases | |
| |      (e.g., documents about "Paris")                               | |
| +--------------------------------------------------------------------+ |
|         | (List of retrieved documents)                                |
|         v                                                              |
| +--------------------------------------------------------------------+ |
| | 3.3. Iterative Processing for Each Retrieved Document              | |
| |   +------------------------------------------------------------+   | |
| |   | 3.3.1. CRAG's Document Quality Assessment (Retrieval Assessor) | | |
| |   |    - LLM assesses document informativeness, relevance, reliability | | |
| |   |      (e.g., 'High', 'Medium', 'Low')                       |   | |
| |   +------------------------------------------------------------+   | |
| |             |                                                      | |
| |             v                                                      | |
| |   +------------------------------------------------------------+   | |
| |   | 3.3.2. Self-RAG's Reflection on Retrieval Results          |   | |
| |   |    - LLM self-judges "Is this document relevant and helpful?" | | |
| |   |      (e.g., 'is_relevant': True, 'is_supportive': True)    |   | |
| |   +------------------------------------------------------------+   | |
| |             |                                                      | |
| |             v                                                      | |
| |   +------------------------------------------------------------+   | |
| |   | 3.3.3. Adaptive Generation Strategy Decision based on CRAG/Self-RAG Judgments | | |
| |   |    - (e.g., 'High' quality + 'relevant/supportive' -> fully utilize document) | | |
| |   |    - (e.g., 'Medium' quality + 'relevant' -> combine with internal knowledge) | | |
| |   |    - (e.g., 'Low' quality or 'irrelevant' -> re-evaluate/skip/internal knowledge only) | | |
| |   +------------------------------------------------------------+   | |
| |             |                                                      | |
| |             v                                                      | |
| |   +------------------------------------------------------------+   | |
| |   | 3.3.4. Answer Generation                                   |   | |
| |   |    - LLM generates answer according to the decided strategy |  | |
| |   +------------------------------------------------------------+   | |
| |             |                                                      | |
| |             v                                                      | |
| |   +------------------------------------------------------------+   | |
| |   | 3.3.5. Self-RAG's Final Reflection on Generated Answer     |   | |
| |   |    - LLM self-judges "Is the generated answer factual and complete?" | | |
| |   |      (e.g., 'is_factual': True, 'is_complete': True)       |   | |
| |   +------------------------------------------------------------+   | |
| |             | (If satisfactory, end iteration)                     | |
| |             v                                                      | |
| | <--- (If more documents or not satisfactory, proceed to next document or try another strategy) ---+ |
| +--------------------------------------------------------------------+ |
+------------------------------------------------------------------------+
           |
           v
+----------+----------+
| 4. Return Final Answer |
| (Optimal Answer Selected) |
+----------+----------+
           |
           v
+---------------------+
|  5. Present Answer  |
|     to User         |
|  "The capital of    |
|  France is Paris."  |
+---------------------+
```