QA-Dragon: Query-Aware Dynamic RAG System for Knowledge-Intensive Visual Question Answering

Zhuohang Jiang*
The Hong Kong Polytechnic
University
Hong Kong SAR, China
zhuohang.jiang@connect.polyu.hk

Pangjing Wu*
The Hong Kong Polytechnic
University
Hong Kong SAR, China
pang-jing.wu@connect.polyu.hk

Xu Yuan*
The Hong Kong Polytechnic
University
Hong Kong SAR, China
xander.yuan@connect.polyu.hk

Wenqi Fan†
The Hong Kong Polytechnic
University
Hong Kong SAR, China
wenqi.fan@polyu.edu.hk

Qing Li†
The Hong Kong Polytechnic
University
Hong Kong SAR, China
csqli@comp.polyu.edu.hk

Abstract

Retrieval-Augmented Generation (RAG) has been introduced to mitigate hallucinations in Multimodal Large Language Models (MLLMs) by incorporating external knowledge into the generation process, and it has become a widely adopted approach for knowledge-intensive Visual Question Answering (VQA). However, existing RAG methods typically retrieve from either text or images in isolation, limiting their ability to address complex queries that require multi-hop reasoning or up-to-date factual knowledge. To address this limitation, we propose **QA-Dragon**, a **Q**uery-**A**ware **D**ynamic **RAG** System for Kn**o**wledge-I**n**tensive VQA. Specifically, QA-Dragon introduces a domain router to identify the query's subject domain for domain-specific reasoning, along with a search router that dynamically selects optimal retrieval strategies. By orchestrating both text and image search agents in a hybrid setup, our system supports multimodal, multi-turn, and multi-hop reasoning, enabling it to tackle complex VQA tasks effectively. We evaluate our QA-Dragon on the Meta CRAG-MM Challenge at KDD Cup 2025, where it significantly enhances the reasoning performance of base models under challenging scenarios. Our framework achieves substantial improvements in both answer accuracy and knowledge overlap scores, outperforming baselines by 5.06% on the single-source task, 6.35% on the multi-source task, and 5.03% on the multi-turn task. The source code for our system is released in https://github.com/jzzzzh/QA-Dragon.

CCS Concepts

• **Information systems → Information retrieval**.

---

*Authors contributed equally to this research.

†Prof. Qing Li and Prof. Wenqi Fan are the advisors of the team.

Keywords

Retrieval-Augmented Generation, Multimodal Large Language Model, Visual Question Answering

1 Introduction

Powered by advanced Large Language Models (LLMs) [3, 4, 12, 15] and sophisticated visual perception modules, Multimodal Large Language Models (MLLMs) [8–10, 17] have demonstrated strong understanding and reasoning capabilities across a wide range of vision-language tasks, such as Visual Question Answering (VQA). Despite these advancements, MLLMs still face significant challenges when addressing queries that require long-tail or up-to-date knowledge and multi-hop reasoning, often generating hallucinated or inaccurate responses [5, 13]. These issues stem from the scarcity of relevant knowledge in MLLMs' training corpus and the inherent difficulty of memorizing low-frequency facts [2]. Retrieval-Augmented Generation (RAG) [7] has recently emerged as a promising solution, promoting MLLMs by incorporating external information to complement their internal knowledge. However, multimodal RAG (MM-RAG) still faces significant challenges, such as interpreting complex queries, selecting appropriate retrieval tools, refining relevant information, and enabling effective multi-turn interactions.

To address these challenges, we introduce a Query-Aware Dynamic RAG System for Knowledge-Intensive VQA (**QA-Dragon**), specifically designed to address cross-domain, knowledge-based, and multi-hop reasoning VQA tasks. To be specific, QA-Dragon incorporates three specialized reasoning branches and a series of modular components, including a Pre-Answer Module, Search Router, Tool Router, Image & Text Retrieval Agents, a Multimodal Reranker, and a Post-Answer Module. These components combine to dynamically select the optimal retrieval strategy based on domain-specific query characteristics. Specifically, the Search Router interprets the intent of a given query. It dispatches it to the appropriate retrieval

pathway, while the Tool Router further refines the execution by selecting between image-based and text-based retrieval agents according to the query modality. To ensure the relevance and quality of the retrieved information, the multimodal reranker uses a coarse-to-fine refinement process that grounds final responses in high-quality evidence.

By supporting multimodal, multi-turn, and multi-hop reasoning, QA-Dragon can address the complexity of real-world VQA scenarios. Empirical results demonstrate that our framework yields substantial improvements in both answer accuracy and knowledge overlap, outperforming strong baselines by 5.06% on the single-source task, 6.35% on the multi-source task, and 5.03% on the multi-turn task.

2 Competition Description

To highlight the challenges of real-world VQA tasks, Meta and AIcrowd launched the Meta CRAG-MM Challenge 2025 [16], an official KDD Cup competition focused on comprehensive RAG for multimodal, multi-turn question answering (QA) over images captured in the wild. This benchmark combines egocentric photos from Ray-Ban Meta smart glasses with factual QA pairs, a mock image-based knowledge base, web search APIs, and rigorous truthfulness scoring, offering the first large-scale testbed for end-to-end MM-RAG systems.

2.1 Dataset

The **CRAG-MM** release comprises three coordinated resources: an *Image Set*, a *QA Set*, and *Retrieval Database*.

- **Image Set** contains 5,000 RGB images, of which 3,000 are first-person "egocentric" shots from smart-glasses, while the rest are ordinary photos scraped from the open web. These images span 14 topical domains (e.g., Books, Food, Shopping, Vehicles) and intentionally include low-quality or cluttered views that emulate real-world wearable scenarios.
- **QA Set** includes two complementary partitions:
  (1) **Single-turn**: More than 3.88k independent QA pairs (1.94k validation + 1.94k public-test).
  (2) **Multi-turn**: 1,173 dialog sessions (586 validation + 587 public-test) comprising 2–6 interleaved questions per image.
- **Retrieval Database** is composed of two controlled sources designed to equalize access across teams:
  (1) An *image–KG mock API* that returns visually similar images and structured metadata (title, brand, price, etc.) keyed on the query image.
  (2) A *textual web-search mock API* that yields URL, title, snippet, timestamp, and full HTML for up to 50 pages per query, interleaving hard negatives to mimic real-world noise.

Together, these ingredients create a realistic yet reproducible sandbox for studying hallucination-free answer generation in wearable contexts.

2.2 Tasks

The competition defines three progressively harder tasks, including **Single-source Augmentation**, **Multi-source Augmentation** and **Multi-turn QA** tasks.

- **Task #1 – Single-source Augmentation**: Given *only* the image-KG API, the system must retrieve structured facts linked to the image and produce a grounded answer.
- **Task #2 – Multi-source Augmentation**: Web-search API results are added, requiring the system to fuse heterogeneous evidence, filter noise, and justify answers drawn from both the image KG and the web.
- **Task #3 – Multi-turn QA**: Dialog sessions of 2–6 turns probe contextual understanding, answer consistency, and the ability to decide when the current image is still relevant versus when text-only reasoning suffices.

3 Methodology

To address the challenge of grounded and trustworthy answering for real-world multimodal queries, we propose **QA-Dragon**, a framework that integrates domain-aware reasoning, adaptive retrieval, and trustworthiness verification. QA-Dragon decomposes the problem into three branches with multiple processes: 1) a **Pre-Answer Module**, which performs domain classification and generates an initial reasoning trace and answer using a domain-specific Chain-of-Thought (D-CoT) agent; 2) a **Search Router**, which inspects the reasoning trace to determine whether additional external evidence is required, and if so, selects between retrieval-augmented generation or answer verification; 3) a **Tool Router**, which decides whether to invoke a 4) **Image Search Agent** or a 5) **Text Search Agent** to search for useful information, 6) a **Coarse-to-Fine Multimodal Reranker**, which selects the most relevant information to augment answers, and 7) a **Post-Answer Module**, which consolidates retrieved evidence with the initial hypothesis to generate a final, verifiable response. By explicitly modeling domain context and evidence needs, QA-Dragon improves factual consistency, supports low-latency fallback through direct output, and mitigates hallucination via adaptive verification and reranking strategies.

3.1 Pre-Answer Module

3.1.1 Domain-Router. Real-world multimodal queries are highly diverse, spanning topics such as animals, food, vehicles, and complex data visualizations. Treating all queries uniformly risks suboptimal reasoning due to domain mismatch and lack of specialized context. To address this, we introduce a domain router, which identifies the semantic domain of each query–image pair and enables domain-specific reasoning strategies tailored to the multimodal queries.

The domain router predicts the semantic domain of the input (x, q) pair, where x is an image and q is the corresponding textual query, allowing the system to invoke specialized reasoning agents with in-domain examples and tailored prompts. Specifically, we leverage BLIP-2 [8], an efficient vision–language model, to bridge image understanding and language reasoning through a lightweight query transformer. It can jointly encode visual and textual signals, enabling robust multimodal classification.
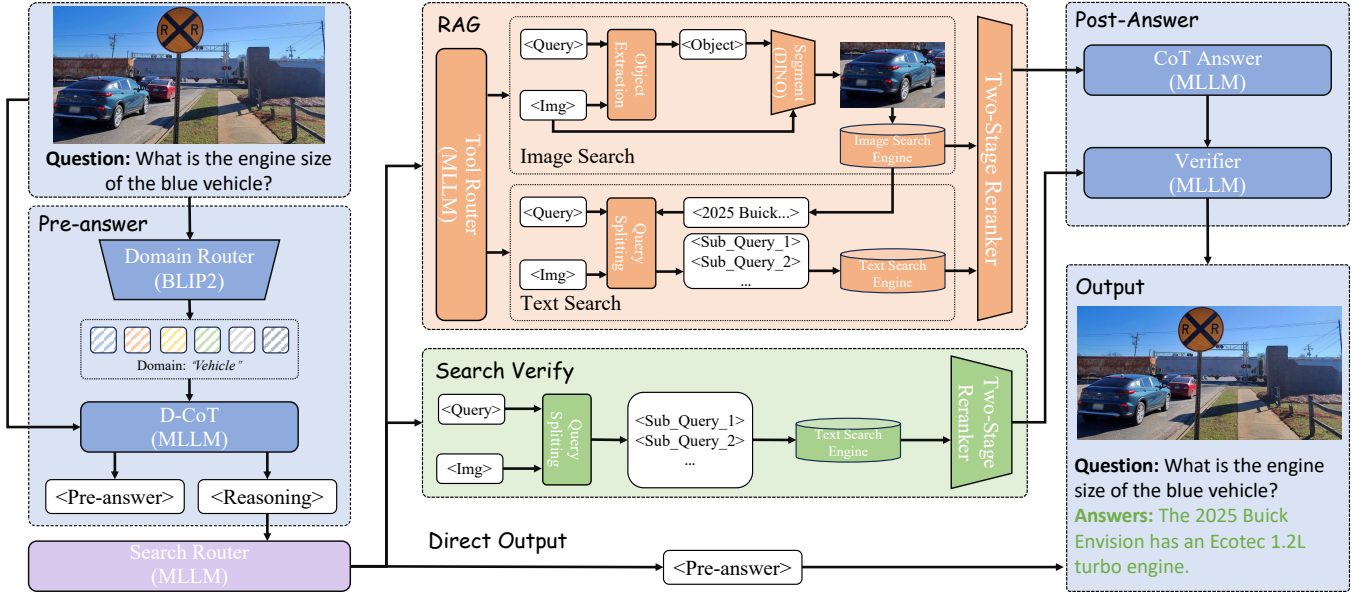
Figure 1: An overview of the QA-Dragon framework. Given a multimodal query–image pair, the system first processes the input through a *Domain Router* and D-CoT to produce a draft answer and reasoning trace. A *Search Router* then classifies the query into one of three branches: *a) Direct Output*, which emits the answer immediately; *b) Search Verify*, which retrieves external evidence for verification; and *c) RAG*, which performs retrieval via tool routing, multimodal search, and reranking. The resulting evidence is passed to the *Post-Answer Module* for final verification and answer refinement.

The BLIP-2 is finetuned using the domain annotations available in the competition dataset. The predicted domain label d is then used to dispatch the query to the corresponding D-CoT process, which performs in-domain reasoning under customized prompting templates.

3.1.2 D-CoT. While LLMs exhibit strong generalization, they often lack awareness of their knowledge boundaries [5, 13]. To mitigate this, we introduce the D-CoT, which performs a structured pre-answer reasoning process to identify what the model confidently knows and where it lacks sufficient information. This early introspective step enables the system to decide whether further evidence retrieval is needed.

The D-CoT module prompts the MLLM to generate step-by-step and domain-aware reasoning grounded in visual content and the user query. It is accomplished by composing a domain-specific prompt with a small number of curated in-domain few-shot examples, forming an input sequence. Specifically, we prompt MLLM to generate a provisional answer and an explicit reasoning trace. To ensure domain-relevant and interpretable reasoning, each prompt includes explicit behavioral constraints. The model must 1) identify the exact object referenced by the query, 2) reason over image and contextual clues step-by-step, and 3) explicitly signal uncertainty when necessary information is missing, as illustrated in the Appendix. For queries that are out of domain, we have designed a set of standard prompts to generate temporary answers.

This instruction reveals the model's internal logic and provides a transparent basis for downstream routing decisions. The resulting reasoning trace is then passed to the search router, determining

whether the provisional answer is sufficient or if further retrieval or verification steps should be invoked.
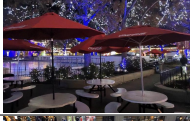
3.2 Search Router

Not all queries require the same depth of post-processing. Some queries can be resolved directly based on image-grounded information, such as extracting OCR text or calculating visual formulas. Others may require external evidence or verification to ensure factual accuracy. To make these decisions adaptively, we introduce the Search Router, a key component that selects the most suitable execution path based on the pre-answer and reasoning trace from the D-CoT.

The search router is to determine whether to directly output the preliminary answer, perform factual verification using external sources, or invoke a retrieval-augmented generation process to synthesize a new, grounded response. This routing mechanism allows the system to balance efficiency and reliability, ensuring that each query receives just the right amount of computational effort for trustworthy results.

The router takes as input the original query and image, along with the draft answer, reasoning trace, and predicted domain from the D-CoT module. Instead of relying solely on the language model's self-assessment, the router is implemented as a lightweight classifier based on LLaMA-3.2-11B-Vision, which is trained on features derived from the reasoning trace and answer content. These features include:

Table 1: Execution Path Selection based on CoT Output and Router Cues.

Branch	Trigger	Processing Modules	Image	Query
Direct Output	D-CoT succeeds, and the query is classified as self-contained (*e.g.*, arithmetic, OCR)	–		*"What is written on these umbrellas?"*
Search Verify	D-CoT succeeds, but external evidence is recommended for verification	Text Toolchain → Fusion & RAG Engine → Verifier		*"In which year did the car on the right begin production?"*
RAG-Augment	D-CoT fails (*e.g.*, "I don't know") or Router detects open-world cues (*e.g.*, entities, numbers)	Visual Toolchain → Text Toolchain → Fusion & RAG Engine → Verifier		*"Who founded this cafe?"*

- **Answerability Flags:** Whether the reasoning contains uncertain phrases like *"I don't know"* or falls into known failure patterns.
- **Answer Task Heuristics:** Whether the answer is a number, an OCR string, or a named object, indicating that the information is directly observable from the image.
- **Uncertainty Patterns:** Whether the reasoning exhibits speculative or fallback logic that may require further verification.

Based on these indicators, the router assigns each query to one of three execution paths: **Direct Output**, when the answer can be confidently derived from the image alone; Search **Verify**, when the draft answer lacks certainty and requires evidence checking; and **RAG**, when the query depends on external knowledge not present in the image or reasoning trace.

3.3 Tool Router

To efficiently retrieve missing information, our framework must determine not only whether to search, but also what kind of search to perform. Queries differ in the type of evidence required: some lack object identity (*e.g., "What model is this?"*), while others require factual attributes not visible in the image (*e.g., "What is the price of this car?"*). Motivated by this need for precision in retrieval, we introduce the tool router, a lightweight decision module that selects the appropriate retrieval modality, including **image search**, **text search**, **both**, or **neither**, based on the reasoning trace obtained from the pre-answer module.

Operating after the D-CoT module, the tool router ultilizes the LLaMA-3.2-11B-Vision model to evaluate the reasoning trace to assess the model's knowledge boundaries. If the object mentioned in the query has not been identified with a specific name, it will initiate an image search to retrieve visually similar items and infer the object's identity. In contrast, if the LLM has already identified the object but lacks necessary facts that are not inferable from the image, such as specifications, statistics, or pricing, it triggers a text search to collect relevant information from the web. Queries that involve analytical reasoning tasks, such as mathematical calculations or language translation, are typically self-contained and do not require retrieval. The key decision prompt is illustrated in Appendix A.

3.4 Image Search Agent

The image search agent is responsible for grounding visual entities relevant to the query. It combines **Multimodal Object Extraction**, **Segmentation**, **Multi-image Search Engine**, and **Entity Selection Module** to prepare visual evidence for answer augmentation.

3.4.1 Multimodal Object Extraction. To identify the visual entities relevant to a user query, we employ the LLM to extract a set of grounded object candidates from the input image and question.

Given an image and its associated query, the model predicts a list of salient and visually present objects that are likely to be referenced or needed for downstream reasoning. Each extracted object is described using concise, coarse-grained terms, such as "car," "book," or "brand", and is constrained to represent tangible items only, explicitly excluding abstract concepts and actions. To ensure generalization across domains, the object names are normalized into category-level descriptions and are limited to short, interpretable phrases of up to three words. This process yields a structured object set that supports subsequent steps in query grounding, answer generation, and retrieval conditioning.

From the extracted candidates, a secondary selection step identifies the most relevant object to the query. This object is chosen based on its visual presence, contextual alignment with the query, and positional cues in the image. When multiple similar instances exist, the model appends a distinguishing attribute (*e.g.*, "red car," "white car") to ensure precise reference. The result is a single object label that serves as the anchor for following image segmentation.

3.4.2 Segmentation. In the VQA process, users' questions often focus on a specific object in an image rather than the entire image. To achieve more granular retrieval, we introduce a refined search mechanism based on image segmentation. Specifically, we extract key object words from the image during the multimodal object extraction stage, use grounding DINO [11] to locate and crop these targets, and generate corresponding local image regions for subsequent retrieval. For images containing multiple objects with the same name, our system also supports the simultaneous extraction and processing of multiple related regions to more comprehensively cover the user's query intent.

3.4.3 Multi-Image Search Engine. To meet the needs of single-image or multi-image retrieval, we designed and implemented a customised image retrieval engine that can automatically adapt to multiple image regions generated during the image segmentation stage. The engine performs searches based on the *"crag-mm-2025/image-search-index-validation"*[16] index and uses the *"clip-vit-large-patch14-336"*[14] model to calculate the semantic embedding similarity between the query and candidate image results. Finally, we fuse multiple retrieval results and reorder them based on similarity scores to obtain image content more closely related to the query semantics, thereby providing high-quality visual support for multimodal QA.

3.4.4 Entity Selection. When external image retrieval is performed, it is essential to ensure that the retrieved candidates visually correspond to the actual object depicted in the image. Without this verification, downstream reasoning may rely on mismatched entities, leading to hallucinated or irrelevant answers. To address this challenge, we introduce an entity selection step, which verifies whether any retrieved entity is visually present and consistent with the object shown in the query image.

3.5 Text Search Agent

The text search agent is responsible for retrieving external textual evidence relevant to a given query–image pair. It is composed of three sequential components: the **Query Rephrase**, the **Fusion Search** and the **Text Search Engine** to handle multi-hop reasoning, entity disambiguation, and weakly grounded queries.

3.5.1 Query Rephrasing Module. Since search engines rely on exact keyword matching, they struggle with multi-hop reasoning and ambiguous queries. To address this, we decompose each complex query into multiple clear, context-complete sub-queries and replace ambiguous references to improve retrieval accuracy.

Specifically, we first analyze the original query and its reasoning chain. For example, given the multi-hop question *"Can the 2023 model with a 1.3L AWD engine of this vehicle get from Washington, DC to Baltimore on 5 gallons of gas?"*, the system generates sub-queries like: *1) "What is the fuel efficiency (mpg) of the 2023 1.3L AWD model of [vehicle name]?"*, and *2) "What is the driving distance from Washington, DC to Baltimore?"*.

To further improve retrieval precision, a sub-query enhancement module combining textual cues with entities extracted from associated images resolves ambiguous references (*e.g.,* replacing *"this vehicle"* with the '*2023 Chevrolet Trailblazer"*) and supplements missing context, producing semantically clear and independent sub-queries. These refined sub-queries are then issued to the search engine to improve the relevance and recall of the retrieved contents.

3.5.2 Fusion Search. The fusion search serves as a bridge between the image and text retrieval pipelines by leveraging grounded object information extracted during the image search stage to improve the quality and precision of text-based evidence retrieval. This is particularly useful when the original user query lacks a specific object identity or when disambiguation is required to generate a meaningful web search query.

After identifying the most relevant object in the image and verifying it through the entity selection, the fusion search process
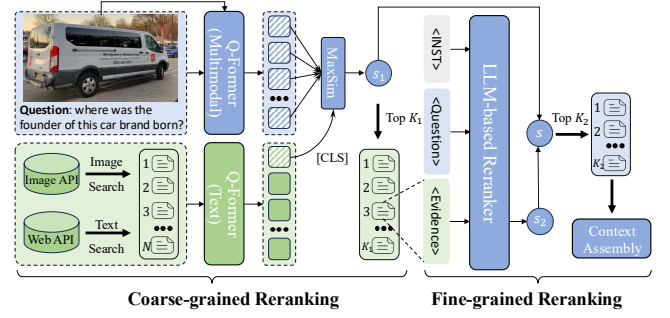

Figure 2: The pipeline of multimodal reranker.

constructs object-aware textual queries by combining the user's original question with the name of the verified object. This enables the system to issue more focused and semantically grounded web search queries, such as transforming a vague question like *"What's the price of this?"* into a more informative query like *"Price of red sports car BMW M4."*

3.5.3 Text Search Engine. To accommodate the retrieval needs of single or multiple sub-queries, we have built a custom text retrieval engine that can adapt to multiple sub-queries automatically. The engine performs query operations in the *"crag-mm-2025/web-search-index-validation"*[16] index and uses the *"bge-large-en-v1.5"* model[19] to calculate the similarity between the search results and the query's semantic embeddings. Finally, we fuse multiple retrieval results and reorder them based on similarity to obtain more relevant document content, thereby supporting subsequent responses.

3.6 Coarse-to-fine Multimodal Reranker

To efficiently use evidence retrieved from image and text search agents and filter query-relevant information, we employ a two-stage coarse-to-fine multimodal reranker that performs **Evidence Chunking**, **Multimodal Reranking**, and **Context Assembly**.

3.6.1 Evidence Chunking. Let $\mathcal{D}_{\text{text}}$ and $\mathcal{D}_{\text{img}}$ denote the textual and image-based retrieval sets, respectively. The initial evidence set is defined as $\mathcal{D}_{\text{all}} = \mathcal{D}_{\text{text}} \cup \mathcal{D}_{\text{img}}$, where $d \in \mathcal{D}_{\text{all}}$ is a paragraph from retrieved passages or an attribute block from image metadata. For each paragraph d, we first segment it into manageable pieces using a hierarchical chunking function $f_{\text{chunk}}(d)$, which takes into account the document structure (e.g., titles, paragraphs) as well as fixed-length spans. Given an attribute block, we construct a sentence for each attribute using the template: "The <key> of <entity> is <value>", where <entity> refers to the entity name from the metadata. By concatenating these sentences, we form a paragraph corresponding to the attribute block. This generated paragraph is then processed using $f_{\text{chunk}}(d)$. Finally, the chunked evidence set can be obtained by

$$C = \bigcup_{d \in \mathcal{D}_{\text{all}}} f_{\text{chunk}}(d), \tag{1}$$

where $c_i \in C$ is a candidate evidence.

3.6.2 Multimodal Reranking. We perform cascaded reranking for C in two stages, as depicted in Figure 2.

Coarse-grained Reranking: To ensure the relevance with both the question and image, a vision-language model f_{qformer} based on Q-FORMER [8] is first used to filter evidence at a coarse granularity. Given a question-image pair (q, x), Q-FORMER encodes it into a set of query tokens $\{z_q^1, ..., z_q^{N_q}\}$. On the candidate side, c_i is encoded into a single [CLS] token z_t by Q-FORMER's text encoder. Then, the relevance score for each evidence is computed via the following similarity function:

$$s_{\text{coarse}}^{(i)} = \max_{j=1,...,N_q} \left(\text{sim} \left(z_q^j, z_t \right) \right), \tag{2}$$

which measures the maximum pairwise similarity between the query token embedding of (q, x) and the [CLS] token embedding of c_i. Subsequently, the top K_1 candidates with the highest scores are selected. However, any candidate with scores below a predefined threshold τ_{coarse} is discarded.

Fine-grained Reranking: To more accurately evaluate similarity, we employ an LLM-based reranker (QWEN3-RERANKER-0.6B [20]) for point-wise reranking within a single context. Specifically, each evidence from the previous step is combined with the question and a well-designed instruction to form an input context, which is then fed into QWEN3-RERANKER f_{qwen} to compute a relevance score, *i.e.*,

$$s_{\text{fine}}^{(i)} = f_{\text{qwen}}(q, c_i | \text{instruction}). \tag{3}$$

In this step, we select the top-K_2 chunks with the highest cumulative score, *i.e.*, $s^{(i)} = s_{\text{fine}}^{(i)} \times s_{\text{coarse}}^{(i)}$. Similar to the first stage, only chunks with a score above $\tau_{\text{fine}} \times \tau_{\text{coarse}}$ are retained.

3.6.3 Context Assembly. For each query, we reorganize the selected chunks into a final evidence string as:

$$\mathcal{E} = \text{Join} \left(\{\text{Sort}(c_i)\}_{i=1}^{K_2} \right), \tag{4}$$

which serves as the input to the post answer generator and verifier.

This coarse-to-fine multimodal reranking mechanism filters noisy or redundant retrievals out while retaining semantically grounded, query-relevant evidence from both retrieval sources.

3.7 Post Answer Generator & Verifier

Given the question-image pair (q, x), and the reranked evidence context $\mathcal{E}$ (as constructed in Section 3.6), we employ a CoT-based generator followed by an answer verifier to produce the final answer.

3.7.1 CoT-based Answer Generation. For complex multimodal reasoning tasks, the capability of the MLLM remains limited, as demonstrated in Table 2. To enhance its reasoning performance, we employ CoT prompting techniques [18] and ICL examples [6], guiding the model to articulate intermediate steps during problem-solving. The MLLM is prompted to produce a detailed reasoning process and a concise final answer for each query. The reasoning involves identifying supporting evidence from $\mathcal{E}$ relevant to the query, while the final answer addresses the query directly. These intermediate reasoning steps also support answer verification in the subsequent stage. Refer to the Appendix for detailed templates.

3.7.2 Answer Verifier. To reduce hallucinations and ensure factual precision, we implement a dual-verification mechanism to assess answer reliability. First, we explore a lightweight white-box verifier that leverages the token probabilities of the generated response to quantify uncertainty [1]. Let $H = \{h_1, h_2, ..., h_T\}$ denote the hidden states at the final decoder layer for each output token from the CoT-based Answer Generator. We compute two statistical metrics: *Minimum Token Probability* and *Normalized Token Probability*, *i.e.*, $s_{\text{min}} = \min\{t_1, ..., t_T\}$ and $s_{\text{mean}} = \frac{1}{T} \sum_{i=1}^{T} t_i$. Here, t_i denotes the token probability, while T is the number of tokens. Then, both s_{min} and s_{mean} are passed to a thresholding function that determines whether to accept or reject the generated answer. An answer is accepted only if $s_{\text{min}} \geq \tau_{\text{min}}$ and $s_{\text{mean}} \geq \tau_{\text{mean}}$; otherwise, the system abstains or defaults to a fallback response (e.g., "I don't know").

Moreover, we design an MLLM-based verifier to assess whether the reasoning produced by the CoT-based Answer Generator is logically sound and supported by the retrieved evidence. Specifically, the MLLM is provided with the question-image pair (q, x), the evidence context $\mathcal{E}$, and the reasoning response, and is tasked with classifying the reasoning as either "Correct" or "Incorrect." The detailed prompt used for this step is provided in the Appendix.

Finally, we integrate the white-box and MLLM-based verifiers to yield high-certainty answers. This dual-verification approach ensures that final answers are grounded in the retrieved context and internally coherent and filtered by confidence, thereby reducing hallucinations in open-ended generation.

4 Experiments

In this section, we report our main results and conduct an ablation study to demonstrate the effectiveness of our key components.

4.1 Experimental Setup

According to the rules set by the organizers, we conduct our experiment on a single Nvidia L40s with 48GB of memory. The network connection is turned off during the inference process, meaning we can only leverage the knowledge from the given database. It also requires that each query be addressed within 10 seconds and labeled as failed after that.

In the model setup, we use *Llama3.2-11B-Vision*[1] for all our MLLMs. The search API supports both text search and image search, which employs *"BAAI/bge-large-en-v1.5"* and *"openai/clip-vit-large-patch14-336"* for text and image embedding generation, respectively.

In our experiments, we compare our method with the following baselines. (1) **LLM Only** prompts the model to answer without retrieval; (2) **CoT** adds step-by-step reasoning. (3) **Direct RAG** retrieves references using the original query and image, and generates answers based solely on the retrieved content. Furthermore, during the development process, we conducted an in-depth analysis of the model outputs and rapidly iterated through four versions of the model:

- **QA-Dragon**[†]: Utilize the domain router and tool router to process the image search and text search.
- **QA-Dragon**[‡]: Fuse the image search result into the text search during the search process.

---

[1] https://huggingface.co/meta-llama/Llama-3.2-11B-Vision

Table 2: Overall Performance of Our Framework across Single-Source and Multi-Source Tasks.

Method	Components					Single-Source			Multi-Source		
	DR	TR	FS	SR	FT	Accuracy (↑)	Overlap (↑)	Elapse (↓)	Accuracy (↑)	Overlap (↑)	Elapse (↓)
LLM Only	–	–	–	–	–	15.79	18.98	0.71	15.79	18.98	0.71
CoT	–	–	–	–	–	16.25	20.07	1.78	16.25	20.07	1.78
Direct RAG	–	–	–	–	–	14.40	22.67	1.62	16.87	27.15	1.95
QA-Dragon[†]	✓	✓	✗	✗	✗	16.77	31.18	3.23	18.99	34.24	3.27
QA-Dragon[‡]	✓	✓	✓	✗	✗	17.44	33.57	3.32	22.08	42.62	3.83
QA-Dragon	✓	✓	✓	✓	✗	21.31	41.09	4.15	23.22	41.77	4.79
QA-Dragon★	✓	✓	✓	✓	✓	20.79	41.42	4.85	22.39	41.65	5.97

DR: Domain Router; TR: Tool Router; FS: Fusion Search; SR: Search Router; FT: Finetuning.

Table 3: Overall Performance of Our Framework on Multi-Turn Task.

Method	Accuracy (↑)	Overlap (↑)	Elapse (↓)
LLM Only	19.75	19.72	0.71
CoT	19.47	20.27	1.78
Direct RAG	20.95	33.74	1.95
QA-Dragon	24.78	48.26	7.00
QA-Dragon★	23.94	47.49	7.80

Table 4: Ablation Study for Key Components in Our QA-Dragon Framework.

Method	Single-Source			Multi-Source		
	Acc.	Overlap	Elapse	Acc.	Overlap	Elapse
QA-Dragon	21.31	41.09	4.15	23.22	41.77	4.79
w/o Domain Router	19.04	41.07	4.19	21.25	41.85	5.07
w/o Tool Router	18.32	41.49	4.02	19.97	42.32	4.25
w/o Query Splitting	–	–	–	18.32	41.70	4.02
w/o Reranking	20.90	41.30	4.15	22.14	41.19	5.07

Query Splitting is not applied under the single-source scenario.

- **QA-Dragon:** Propose a search router to categorize VQA queries into three search strategies.
- **QA-Dragon★:** Integrated a finetuning module to enhance the reasoning capability.

4.1.1 Metrics. In line with the MM-CRAG Benchmark, we conduct a model-based automatic evaluation for our experiment. The automatic evaluation employs rule-based **Overlap** matching and prompt *GPT-4o-mini* as an assessment to check the **Accuracy** of the answer. Moreover, we report the average second **Elapse** of each query since efficiency is a critical aspect required by the organizers. The final score of each method is computed as the average score across all examples in the evaluation set.
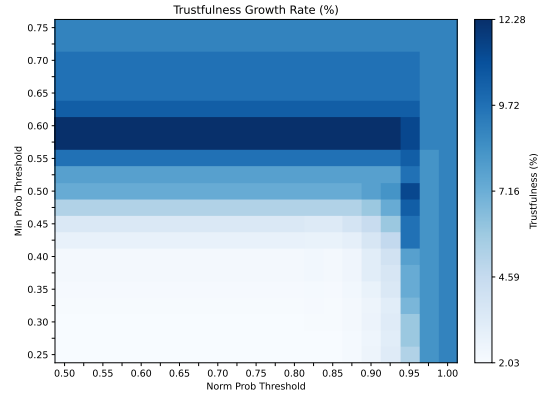
Figure 3: The Trustfulness Growth Rate (%) of QA-Dragon.

4.2 Overall Performance

Table 2 and Table 3 report the performance of our method and baselines. Comparing our solutions to the LLM and RAG baselines, we observe significant advantages in performance across all three tasks. Our method showcases notable improvements in accuracy and information retention. Specifically: **QA-Dragon and QA-Dragon★ Achieves the Best Overall Performance**. QA-Dragon outperforms all baselines and ablations in both single-source (21.31%), multi-source (23.22%) and multi-turn (24.78%) settings. The QA-Dragon★ achieves the highest overlap scores (41.42% and 41.65%), indicating better alignment with ground-truth supporting evidence. **Baselines Fall Short**. The LLM Only, CoT, and Direct RAG baselines significantly underperform in accuracy and overlap. **Trade-off between Performance and Efficiency**. The most powerful variant, QA-Dragon★, achieves the best performance but incurs the highest latency (5.97s).

4.3 Ablation Study

To assess the contribution of each core component in our framework, we conduct a series of ablation experiments. Table 4 reports the results in terms of answer accuracy, entity overlap with ground truth, and response latency, to evaluate both correctness, efficiency, and alignment with human-annotated outputs.

Domain Router. Removing the domain router leads to a noticeable drop in accuracy for both single-source (from 21.31% to 19.04%)
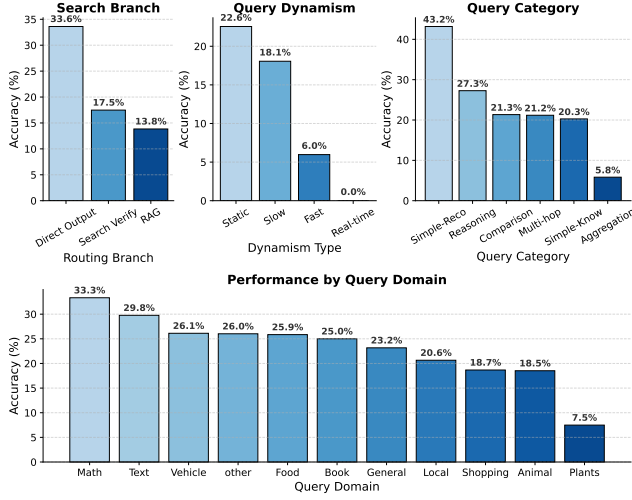
Figure 4: Accuracy over different query taxonomies on the single-source task.

Figure 5: Accuracy over different query taxonomies on the multi-source task.

and multi-source settings (from 23.22% to 21.25%). It demonstrates the importance of domain-aware query routing in selecting appropriate retrieval strategies, especially when grounding queries in specific content domains.

Tool Router. Disabling the tool router results in further performance degradation, particularly in the single-source setting (from 21.31% to 18.32%). Interestingly, overlap remains high (41.49% and 42.32%), suggesting that while relevant content is still retrieved, the absence of precise tool selection limits the model's ability to convert evidence into correct answers.

Query Splitting. Eliminating query splitting, which decomposes complex questions into interpretable sub-goals, significantly degrades the performance. This result validates the effectiveness of breaking down complex queries into multiple logical sub-queries to enable more rigorous textual retrieval.

Two-stage Reranking. We further test the effect of removing two-stage reranking in the retrieval pipeline. Accuracy slightly decreases (from 21.31% to 20.90% in single-source and from 23.22% to 22.14% in multi-source), and latency increases by nearly 0.3s in the multi-source case. This indicates that reranking retrieved candidates meaningfully improves answer selection without compromising efficiency.

Verifier. Figure 3 demonstrates the trustworthy performance of the QA-Dragon model on the Multi-Source task. We observe that the Answer Verifier significantly enhances trustworthy performance by 12.28%. Furthermore, to select suitable hyperparameters, we perform an exhaustive grid search over $[0.25, 0.75]$ for τ_{min} and $[0.50, 1.00]$ for τ_{norm}, optimizing for the truthfulness score. Based on the results, we set $\tau_{min} = 0.60$ and $\tau_{norm} = 0.90$, under which an optimal balance between accuracy and trustworthiness can be maintained.

5 Perspectives

To better understand the behavior and limitations of our QA-Dragon framework, we conduct a detailed evaluation across four query
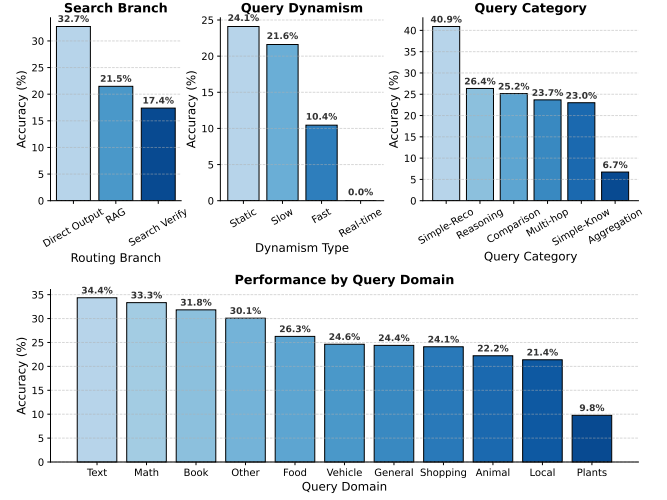
taxonomies: search branch, query dynamism, query category, and query domain, as shown in Figure 4 and 5, respectively.

Across both tasks, we observe several consistent patterns. Queries routed to the *Direct Output* branch achieve the highest accuracy (33.6% for single-source and 32.7% for multi-source), significantly outperforming *Search Verify* and *RAG Augment*, highlighting the effectiveness of the *Search Router* in identifying queries that can be resolved through image-grounded or self-contained reasoning directly. Accuracy decreases steadily as *query dynamism* increases, with real-time queries yielding no correct answers. This illustrates the challenge of temporally sensitive reasoning and emphasizes the need for improved temporal grounding in future systems. In terms of query category, *Simple Recognition* achieves the highest performance (43.2% and 40.9%), while *Aggregation* is the most difficult (5.8% and 6.7%), likely due to its multi-hop and cross-source nature. Among query domains, *Math* and *Text* lead in accuracy, benefiting from structured formats and OCR-friendly content. Conversely, the *Plant* domain shows the lowest performance (7.5% and 9.8%), reflecting the difficulty of fine-grained visual discrimination in biologically similar classes.

6 Conclusion

In this work, we proposed the **QA-Dragon**, a query-aware dynamic RAG system tailored for complex and knowledge-intensive VQA tasks. Unlike the traditional RAG system, which retrieves the text and image separately, the QA-Dragon integrates a domain router, a search router, and orchestrates hybrid retrieval strategies via dedicated text and image search agents. Our system supports multimodal, multi-turn, and multi-hop reasoning through orchestrating both text and image search agents in a hybrid setup. Extensive evaluations on the Meta CRAG-MM Challenge (KDD Cup 2025) demonstrate that our system significantly improves answer accuracy, surpassing other strong baselines.

References

[1] Dylan Bouchard and Mohit Singh Chauhan. 2025. Uncertainty Quantification for Language Models: A Suite of Black-Box, White-Box, LLM Judge, and Ensemble Scorers. *arXiv preprint arXiv:2504.19254* (2025).

[2] Yang Chen, Hexiang Hu, Yi Luan, Haitian Sun, Soravit Changpinyo, Alan Ritter, and Ming-Wei Chang. 2023. Can Pre-trained Vision and Language Models Answer Visual Information-Seeking Questions?. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 14948–14968.

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

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research* 24, 240 (2023), 1–113.

[5] Federico Cocchi, Nicholas Moratelli, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. 2025. Augmenting multimodal llms with self-reflective tokens for knowledge-based visual question answering. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 9199–9209.

[6] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A Survey on In-context Learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 1107–1128.

[7] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6491–6501.

[8] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*. PMLR, 19730–19742.

[9] Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. 2024. Vila: On pre-training for visual language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 26689–26699.

[10] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in neural information processing systems*, Vol. 36. 34892–34916.

[11] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. 2023. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499* (2023).

[12] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

[13] Jielin Qiu, Andrea Madotto, Zhaojiang Lin, Paul A Crook, Yifan Ethan Xu, Xin Luna Dong, Christos Faloutsos, Lei Li, Babak Damavandi, and Seungwhan Moon. 2024. Snapntell: Enhancing entity-centric visual question answering with retrieval augmented multimodal llm. *arXiv preprint arXiv:2403.04735* (2024).

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PmLR, 8748–8763.

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

[16] Jiaqi Wang, Xiao Yang, Kai Sun, Parth Suresh, Sanat Sharma, Adam Czyzewski, Derek Andersen, Surya Appini, Arkav Banerjee, Sajal Choudhary, Shervin Ghasemlou, Ziqiang Guan, Akil Iyer, Haidar Khan, Lingkun Kong, Roy Luo, Tiffany Ma, Zhen Qiao, David Tran, Wenfang Xu, Skyler Yeatman, Chen Zhou, Gunveer Gujral, Yinglong Xia, Shane Moon, Nicolas Scheffer, Nirav Shah, Eun Chang, Yue Liu, Florian Metze, Tammy Stark, Zhaleh Feizollahi, Andrea Jessee, Mangesh Pujari, Ahmed Aly, Babak Damavandi, Rakesh Wanga, Anuj Kumar, Rohit Patel, Wen tau Yih, and Xin Luna Dong. 2025. CRAG-MM: Multimodal Multi-turn Comprehensive RAG Benchmark. arXiv:2510.26160 [cs.CV] https://arxiv.org/abs/2510.26160

[17] P Wang, S Bai, S Tan, S Wang, Z Fan, J Bai, K Chen, X Liu, J Wang, W Ge, et al. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024. *URL https://arxiv. org/abs/2409.12191* (2024).

[18] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[19] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597 [cs.CL]

[20] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176* (2025).

A Prompts

Evaluator Prompt

System Prompt:
You are a visual assistant tasked with addressing the user's query for the image based on your inherent knowledge.

General Reasoning Guidelines:

(1) Generate step-by-step reasoning to address the query using evidence from the image and your knowledge. Limit your reasoning to no more than 5 concise steps, with each step written as a single sentence. Stop reasoning once you have enough information to answer the query or when you determine that necessary information is lacking.

(2) In your reasoning, identify the exact object that the query is about by its exact name (*e.g.*, car model, food name, brand name, species name, etc.). If no clear object matches the query, you may refer to textual clues visible in the image if available.

(3) If the query involves multiple objects or relationships, dedicate one reasoning step to each object or relationship, and then summarize the result in a final step. For example:

  (a) The exact name of the object in the image that the query is about is <specific_object_name>.

  (b) Next, the exact name of the object related to the first one is <specific_object_name>."

  (c) ...

(4) If you cannot determine the necessary information from the image or the query, explicitly state: "I cannot determine the <what> that the query is about."

(5) Do not suggest that the user to refer to external sources.

(6) Always begin your reasoning with: "1. The exact name of the object that the query "<query>" is about is <specific_object_name>." and make your final reasoning concise.

`<Doamin-aware ICL cases>`

Output Format:

(1) The exact name of the object in the image that the query is about is <specific_object_name>.

(2) Then, I ...

(3) ...

`{"Reasoning": "<summary_reasoning_string>"}`

Image Object List Prompt

System Prompt:
You are an expert AI system for object detection and identification. Your task is to recognize and list high-level object categories shown in an image that are relevant to a given question. Return only structured results in JSON format.

User Prompt:
Identify and list up to `{self.object_num}` major distinct objects in the image that are visually present and relevant to the question: "query".

Only include tangible, visible items (e.g., "car", "brand", "clothing", "book", "device", "food", "building").

Do not include abstract concepts (e.g., "emotion", "relationship") or actions (e.g., "running", "shopping").

Use general categories, not specific names: "BMW" to "car", "ZARA" to "clothing brand", "iPhone 13" to "smartphone", "Coca-Cola" to "drink"

Each object name should be short with no more than 3 words.

If unsure of the exact identity, use the closest general category (e.g., "electronic device", "building", "plant").

Return the result in the following JSON format strictly:

`{"Object_List": ["<object_name_string>", "<object_name_string>", ...]}`

Image Object Selection Prompt

System Prompt: You are an AI assistant to select one object from a list of objects, which is most relevant to the object queried by the question. Only return structured results in JSON format.

User Prompt: Given the image and list of objects detected in the image: `{object_list}` and the question: "`{query}`" select the one object in the object list that the question is about.

If the query includes position-related words, give priority to objects at or near the position.

Give a short sentence about the reason to choose the object and return the final selected object in this format:

`{"Object": "<object_name_string>"}`

Evaluator Prompt

System Prompt: You are an action-selector. Given three inputs: (1) the user's query, (2) any prior reasoning text, and (3) the image, decide in a single turn which retrieval tool or tools must run before the answer is generated. You may choose image_search, text_search, both, or neither.

Tools: Tool #1 image_search

[Description] Retrieve visually similar images via embeddings to identify an object whose specific name is still unknown.

[Use when] The object in the picture is not known or known only by a generic label (e.g., "car", "jacket", "statue") instead of a specific name/model/species in previous reasoning.

[Input] The input image.

[Output] Text snapshots (top-k) from Wikipedia or Amazon that show visually similar objects.

Tool #2 text_search

[Description] Issue a refined natural-language web query to fetch textual facts about an object whose specific name is already known.

[Use when] The query requires additional information not available in the image.

[Input] A text query constructed from the user's question + the known identity.

[Output] Text snippets (top-k) from relevant websites.

Decision logic

(1) Do you or the reasoning text already know the object's specific identity (proper noun or model name)?
   - Yes: set 'need_image_search' to false.
   - No: set 'need_image_search' to true.

(2) Does the query need additional information that is not visible in the image (specifications, history, statistics, price, etc.)?
   - Yes: set 'need_text_search' to true.
   - No: set 'need_text_search' to false.

(3) If it is about addressing some scientific calculation queries like math, physics, etc., or language translation, set both flags to false.

(4) If the object is a "book", a "logo-bearing packaged goods", or "plant", set 'need_image_search' to false. Both flags may be true; when so, run image_search first, then text_search.

Produce exactly one sentence to conduct the decision logic. This is the only non-JSON text allowed.

Immediately after the sentence, output a single valid JSON object: Decision logic: <concise explanation>

Tool calling decision: "need_image_search": <true/false>, "need_text_search": <true/false>

Do not output anything else.

Here are some examples:

`<Tool Using ICL cases>`

Post Answer Generation Prompt

System Prompt: You are a helpful assistant who truthfully answers user questions about the provided image. If you are not sure about the answer, please say 'I don't know.'.

User Prompt: Answer the given question based on the provided image and your own knowledge. Please think step by step and give a response containing the following parts:

- reason: the information from the image or your own knowledge that leads to the answer, which should be clear and concise within 2-3 sentences. If you are not sure about the answer, please say 'I don't know.'.
- answer: your final answer in a concise format within one sentence. The answer should include critical information from the reason to support your answer. When referring to a specific object in the image, please use the name of the object, rather than 'this', 'that', or 'it'. If the reason is 'I don't know, ' please also say 'I don't know' in the answer.

-Examples-

`<ICL examples>`

-Real Data-

Question: `<question>`

Your Output:

Answer Verifier Prompt

System Prompt: You are a helpful assistant who evaluates whether the agent's answer to the user's image query is reasonable based on the evidence.

User Prompt: Given an image query, the retrieved evidence, and the agent's candidate answer, assess the correctness of the answer by following these guidelines:

(1) Unsupported Answer. If the answer is not supported by the image and evidence, please respond the follows:
   Reason: Briefly (1–2 sentences) explain why the answer lacks sufficient support.
   Response: Incorrect Answer

(2) Contradicted Evidence. If there is conflicting information in the evidence, please respond the follows:
   Reason: Briefly (1–2 sentences) state the specific contradictory evidence.
   Response: Incorrect Answer

(3) Unclear or Incomplete Answer. If the answer is vague or fails to fully address the question, please respond the follows:
   Reason: Briefly (1–2 sentences) explain why the answer is unclear or incomplete.
   Response: Incorrect Answer

(4) Correct Answer. If the answer is fully supported by the image and evidence, please respond the follows:
   Reason: Briefly (1–2 sentences) state why the answer is correct.
   Response: Correct Answer

-Real Input-
Query: <quesion>
Evidence: <evidence>
Answer: <answer>

Your Output: