Learning to Search: A Decision-Based Agent for Knowledge-Based Visual Question Answering

Anonymous ACL submission

Abstract

Knowledge-based visual question answering (KB-VQA) requires vision-language models to understand images and use external knowledge, especially for rare entities and long-tail facts. Most existing retrieval-augmented generation (RAG) methods adopt a fixed pipeline that sequentially retrieves information, filters it, and then produces an answer. Such a design makes it difficult to adapt to diverse question types. Moreover, it separates retrieval from reasoning, making it hard for the model to decide when to search, how to refine queries, or when to stop. As a result, the retrieved evidence is often poorly aligned with the question. To address these limitations, we reformulate KB-VQA as a search-agent problem and model the solving process as a multi-step decision-making procedure. At each step, the agent selects one of four actions-Answer, Image Retrieval, Text Retrieval, and Caption-based on its current information state. We further design an automated pipeline to collect multi-step trajectories that record the agent's reasoning process, tool usage, and intermediate decisions. These trajectories are then used as supervision for fine-tuning. Experiments on InfoSeek and E-VQA demonstrate that our method achieves state-of-the-art performance, consistently outperforming prior baselines and confirming the effectiveness of our framework.

1 Introduction

In recent years, Vision–Language Models (VLMs) have achieved remarkable progress on multimodal tasks such as Visual Question Answering (VQA), demonstrating strong capabilities in cross-modal understanding and generation in recent benchmarks(Tanaka et al., 2025; Xiao et al., 2024). However, when questions involve fine-grained entities or long-tail encyclopedic knowledge, relying solely on the knowledge internalized in model parameters is often insufficient to support accurate reasoning, leading to an overall degradation in performance(Long et al., 2025). This has led to the emergence of research in knowledge-based visual question answering (KB-VQA), which requires not only understanding visual content but also dynamically acquiring relevant information from external knowledge sources.

Retrieval-Augmented Generation (RAG) has become a dominant paradigm in KB-VQA. These methods retrieve candidate knowledge from large external corpora, such as Wikipedia, and inject the retrieved content into the model context to expand its knowledge coverage(Choi et al., 2025). Recent studies have explored various directions, including multimodal query construction, cross-modal retrieval, and post-retrieval filtering, which together have led to noticeable improvements in KB-VQA performance(Hong et al., 2025; Deng et al., 2025).

Despite these advances, most existing approaches still follow a modular and static pipeline design. This design implicitly assumes that all questions should follow the same retrieval and reasoning process. However, KB-VQA questions differ significantly in their dependence on external knowledge. While some questions can be answered directly without external retrieval, others require reflective retrieval to gather sufficient evidence. As a result, a single retrieval step with paragraph-level reranking is often insufficient. By treating retrieval and reasoning as a fixed pipeline, existing methods struggle to explicitly track and exploit the current information state during inference, making it difficult to decide when to stop retrieval or how to adapt queries based on accumulated evidence, which often results in poor semantic alignment between retrieved content and the question.

Based on these observations, we argue that retrieval and reasoning in KB-VQA should be viewed as a dynamic decision-making process instead of a fixed pipeline. We therefore reformulate KB-VQA under a *search agent* paradigm, where the

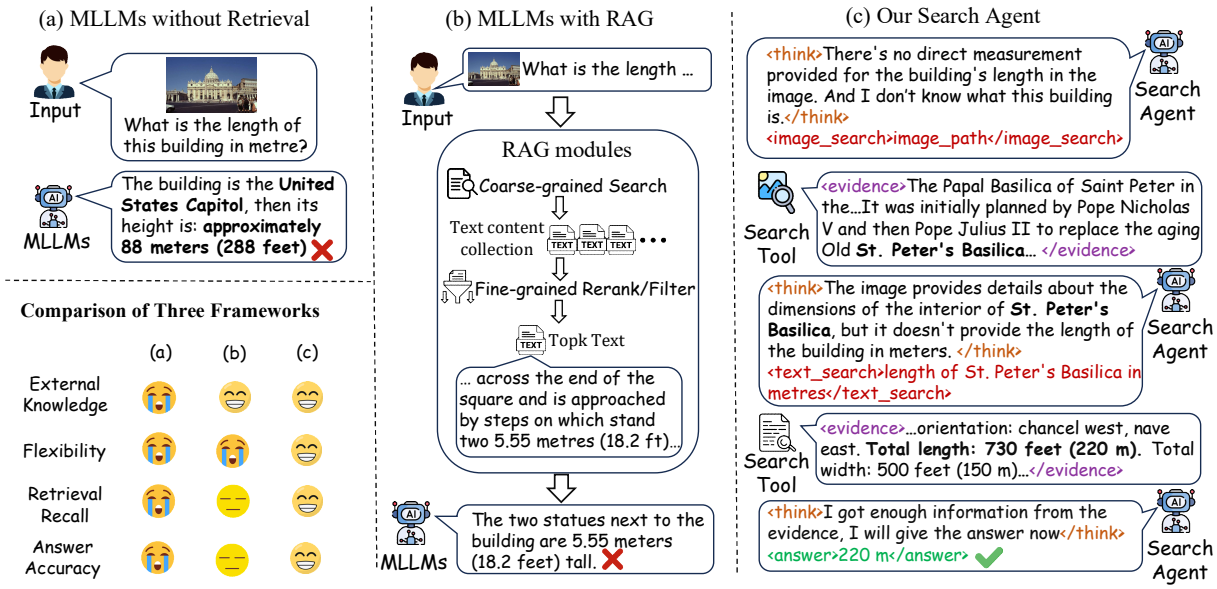

Figure 1: **Comparison of three paradigms for KB-VQA: MLLMs without retrieval, MLLMs with RAG, and our search agent. (a) MLLMs without Retrieval** lack access to external knowledge and often fail on long-tail queries. **(b) MLLMs with RAG** introduce external context but may suffer from noisy or mismatched retrieval due to fixed pipelines. **(c) Our Search Agent** formulates KB-VQA as a multi-step decision process, enabling adaptive tool usage and structured evidence accumulation.

model no longer passively consumes predefined retrieval results. Instead, it acts as an agent that autonomously decides whether to use tools, how to use them, and when to terminate the search, conditioned on its current information state. Specifically, we formalize the entire problem-solving process as a multi-step decision sequence with a structured multimodal action space: **Answer** for generating the final answer; **Image Retrieval** for performing image-based similarity search and returning top-ranked matched images together with their associated documents as candidate evidence; **Text Retrieval** for retrieving relevant textual passages given a textual query; and **Caption** for converting visual content into task-oriented descriptive text to facilitate evidence retrieval and answer generation. During inference, the agent continuously analyzes the current multimodal information state and selects the next action at each step until evidence is gathered to support a final answer. To equip the model with both multimodal understanding and effective tool-selection abilities, we further design an automated multi-stage framework for constructing reasoning–retrieval trajectories. Based on this framework, we generate standardized trajectory data covering diverse KB-VQA scenarios, including different action types, retrieval depths, and evidence accumulation patterns. Each trajectory annotates the complete decision path from the initial input to the final answer. Using these trajectories, we perform supervised fine-tuning (SFT) to teach the model to select appropriate actions under different information states. Experiments on two long-tail KB-VQA benchmarks, InfoSeek(Chen et al., 2023) and E-VQA(Mensink et al., 2023), demonstrate that our method outperforms existing approaches in both accuracy and reasoning stability. Our main contributions can be summarized as follows:

- We study knowledge-based visual question answering from an agent perspective, identify the fundamental limitations of modular RAG pipelines in terms of dynamic planning and state awareness, and propose a more flexible Search Agent framework.It allows the model to decide when to retrieve, how to retrieve, and when to stop.

- We construct multi-step decision trajectory data on InfoSeek and E-VQA, covering diverse evidence acquisition patterns and train the model through supervised fine-tuning to jointly learn reasoning and tool usage.

- Our approach achieves significant improvements over prior methods on two mainstream KB-VQA benchmarks, InfoSeek and E-VQA, demonstrating the effectiveness of dynamic planning-based retrieval agents.

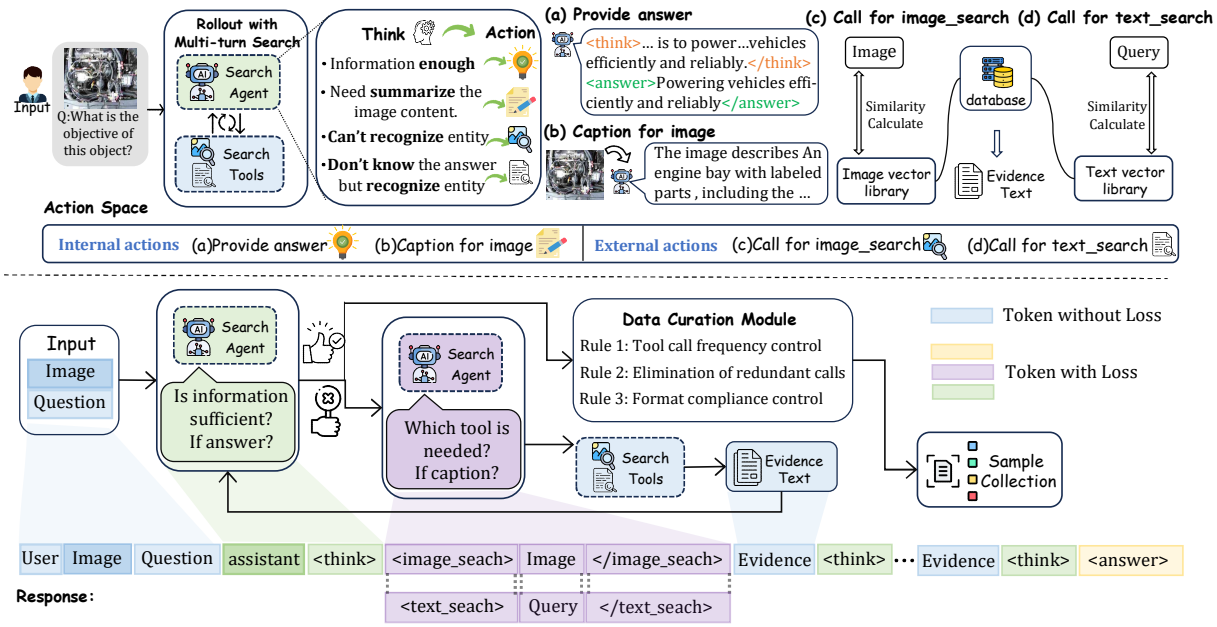

Figure 2: **Overview of the proposed DBAgent framework and trajectory-based training pipeline. Top:** Inference-time decision-making, where the agent dynamically selects among multiple actions (e.g., image retrieval, text retrieval, captioning, and answering) based on the current information state. **Below:** Construction of multi-step reasoning trajectories, data curation rules, and linearization into structured token sequences for SFT.

2 Related Work

2.1 Supervised Fine-Tuning

Supervised fine-tuning (SFT) remains a central paradigm for adapting large language models to instruction following, structured output generation, and complex multi-step reasoning. Recent studies show that supervising models with explicit reasoning trajectories or intermediate decision states significantly improves reasoning stability and error localization, particularly in long-horizon tasks(Du et al., 2024; Sullivan et al., 2025; Yu et al., 2025).

Beyond static reasoning traces, recent work has explored trajectory-style supervision for learning tool use and environment interaction. Agent-oriented SFT frameworks show that models can acquire structured reasoning–action patterns when trained on annotated interaction trajectories, enabling effective tool invocation, planning, and feedback integration(Chen et al., 2025; Liu et al., 2025). These results indicate that modern SFT, when combined with trajectory-level supervision, provides a strong foundation for learning structured reasoning and tool-augmented problem solving.

2.2 Search Agents

Recent work views search-augmented problem solving as a step-by-step process: models first generate search queries based on their current reasoning, then retrieve relevant information from external sources, and finally refine or filter the retrieved results to produce better answers(He et al., 2025; Li et al., 2025; Wu et al., 2025).

Most existing search-agent frameworks are designed for open-world information seeking, where the main challenge lies in generating effective queries under noisy environments(Luo et al., 2025; Li et al., 2024; Zheng et al., 2025). In contrast, KB-VQA assumes access to a curated knowledge base and focuses on how to select and use tools based on the current information state. Moreover, KB-VQA further requires grounding the question to the correct visual entity and deciding whether additional evidence is necessary, making tool selection, stopping, and evidence integration the key bottlenecks beyond query formulation. Recent multimodal agent and planning work studies when to invoke tools and how to optimize multi-step retrieval policies, but they are generally evaluated in open-world or general-purpose settings, leaving the KB-VQA-specific decision boundary underexplored.(Wang et al., 2024)

2.3 KB-VQA Retrieval-Augmented Methods

KB-VQA targets questions whose answers depend on long-tail, fine-grained, and often encyclopedic knowledge that is not directly observable in the image. Recent studies therefore treat KB-VQA as

a multimodal RAG problem(Zhang et al., 2025; Su et al., 2024): the system retrieves candidate evidence (typically from large-scale resources such as Wikipedia articles and associated images) and conditions the VLM on the retrieved content to improve coverage on underrepresented queries. Several recent works improve KB-VQA by enhancing multimodal retrieval or post-retrieval filtering. Nevertheless, they still rely on a static workflow, where retrieval is fixed and independent of the model's reasoning process.

However, most KB-VQA systems still rely on fixed and modular pipelines(Yan and Xie, 2024; Cocchi et al., 2024). Retrieval is usually performed in a predefined manner, followed by filtering and answer generation. This design assumes that all questions follow the same reasoning process, which is often not true. This motivates us to move beyond fixed pipelines and propose an agent-based formulation of KB-VQA.

3 Methods

We propose a search-agent framework for knowledge-intensive KB-VQA. The core idea is to reformulate KB-VQA from a fixed *retrieve–then-generate* pipeline into a multi-step decision-making process. Instead of following a predefined workflow, the model dynamically decides whether to retrieve external information, which tool to use, how to form queries, and when to stop searching, based on the current information state.

As a result, the depth of search adapt to the complexity of the question, avoiding unnecessary tool calls or premature termination. We first present the formal formulation in Section 3.1, then introduce the inference-time execution and action semantics in Section 3.2, and finally describe trajectory construction and supervised fine-tuning in Section 3.3.

3.1 Problem Formulation

Given an input image I and a natural language question q, the goal of KB-VQA is to produce the answer y. Instead of modeling this process as a single-step mapping, we formulate it as a sequential decision-making problem. The model interacts with an external knowledge environment over multiple steps, gradually collecting evidence and refining its understanding. At step t, the model maintains a state s_t, which represents all the information it has observed so far:

$$s_t(I, q, e_1, \ldots, e_{t-1}), \tag{1}$$

where e_i denotes the evidence returned by tool calls in previous steps. Based on the current state s_t, the model selects an action a_t from a predefined action space $\mathcal{A}$. If the selected action involves an external tool, the environment returns new evidence e_t, which is appended to the state:

$$s_{t+1} = (s_t, a_t, e_t). \tag{2}$$

This process continues until the model outputs an answer or reaches a maximum step budget B. The resulting sequence forms a complete search trajectory:

$$\tau = (I, q, a_1, o_1, a_2, o_2, \ldots, a_T), \tag{3}$$

where o_t denotes the environment observation at step t.

3.2 Dynamic Search-Agent Framework

Overall Framework. As shown in Figure 2, our inference follows an iterative *reason–act–observe–update* loop. The detailed inference procedure is summarized in Algorithm 1 in Appendix D. The reasoning process is explicitly generated between the `<think>` and `</think>` tags. Our framework allows the model to dynamically decide: (1) whether additional information is needed, (2) what type of information to acquire, (3) which tool to use, and (4) when to stop and generate the final answer. This adaptive mechanism enables the model to adjust its search depth based on question difficulty, producing reasoning trajectories with matched complexity.

Action Selection. Based on its reasoning, the model selects one action from a predefined action space. Importantly, this action space is designed to reflect the intrinsic challenges of KB-VQA. For example, in many cases, the question does not explicitly mention the target entity, and the model must first infer the entity from the image before it can issue a reliable textual query. As a result, the actions are not independent, but form a collaborative toolset.

Formally, the action space is defined as:

$$\mathcal{A} = \left\{a^{\text{ans}}, a^{\text{text}}, a^{\text{img}}, a^{\text{cap}}\right\}. \tag{4}$$

Each action corresponds to a specific form of interaction with the external knowledge environment:

- **Answer** (a^{ans}): If the search agent determines that the accumulated evidence is sufficient, it outputs the final answer enclosed within the `<answer>` and `</answer>` tags.

- **Text Search** (a^{text}): The search agent generates a natural-language query, enclosed within the <text_search> and </text_search> tags, to retrieve relevant textual passages from the knowledge base.

- **Image Search** (a^{img}): The search agent queries the visual retrieval module using the <image_search> and </image_search> tags, which returns visually similar images together with their associated textual descriptions.

- **Caption** (a^{cap}): When the current multimodal representation is insufficient to form a reliable retrieval query—for example, when the target entity cannot be confidently identified—the model may generate a task-oriented caption enclosed within the <caption> and </caption> tags. This caption abstracts the visual content into a textual form and serves as an intermediate semantic representation for subsequent query rewriting and retrieval.

State Update. When a tool action is selected, the retrieved results are wrapped into structured evidence and enclosed within the <evidence> and </evidence> tags. This evidence is then appended to the state to form s_{t+1}. The caption action does not introduce new external information and therefore does not update the observable state. This design keeps the semantics of state transitions clear: only interactions with the environment produce new information.

3.3 Trajectory Construction and Training

Although modern multimodal models are strong at multi-step reasoning, instruction following, and cross-modal generation, these abilities are usually learned as isolated skills rather than as a unified decision policy. In KB-VQA, the model must not only answer questions, but also repeatedly assess its information state and decide whether to retrieve, how to retrieve, and when to stop. We observe that without explicit training on this decision process, models struggle to coordinate these abilities into stable and controllable behaviors.

3.3.1 Failure-Aware Trajectory Branching

We design a failure-aware branching mechanism based on common failure patterns in KB-VQA. For each original image–question–answer sample, we first test whether the model can answer correctly without using any external tools. Such samples are treated as *parametrically sufficient* and used to construct zero-retrieval trajectories.

If the model fails, we further categorize the failure into two types: (1) the entity is recognized but factual knowledge is missing, and (2) the entity cannot be reliably identified. In the first case, the primary bottleneck is the lack of encyclopedic knowledge, and therefore text retrieval is required to access the relevant information. In the second case, the model lacks a clear semantic anchor, so visual retrieval and semantic reconstruction are required. For the second type, we introduce an explicit intermediate step where the model generates a task-oriented caption enclosed within <caption> and </caption>. This caption acts as a semantic bridge from vision to language and enables accurate query reformulation.

3.3.2 Difficulty Modeling

Questions in KB-VQA differ significantly in search depth, tool coordination, and reasoning structure. This forms a distribution with varying levels of **decision complexity** across different problems. To capture this structure, we partition the dataset into multiple difficulty subsets:

$$\mathcal{D} = \bigcup_{k=1}^{K} \mathcal{D}_k. \tag{5}$$

We characterize the difficulty of KB-VQA problems along three key dimensions: (1) search depth, (2) tool usage structure, and (3) intermediate reasoning patterns. Based on these factors, we group trajectories into easy, medium, and hard subsets. From each subset, we sample a fixed number of trajectories for training.

3.3.3 Training Template and SFT Objective

To ensure the model learns decision behaviors rather than memorizing retrieved content, we use a unified instruction template to regulate reasoning, action selection, and output formats in a structured and consistent manner. At each step, the model must choose exactly one action and explicitly state its reasoning between the <think> and </think> tags.

If the current information is sufficient, the model outputs the final answer enclosed within <answer> and </answer>. Otherwise, it selects either <text_search> or <image_search>. When needed, it can first generate a <caption> as an intermediate representation.

During training, each trajectory is linearized into a token sequence:

$$(u, a_1, o_1, \ldots, a_n), \tag{6}$$

where u is the initial instruction, a_i is the action token, and o_i is the environment observation enclosed within the <evidence> and </evidence> tags for each step. We mask all observation tokens and apply supervision only to decision-related tokens. The loss is defined as:

$$\mathcal{L} = -\sum_j \log p_\theta(t_j \mid t_{<j}) \cdot I(t_j \in \mathcal{Y}), \tag{7}$$

where $\mathcal{Y}$ denotes the set of decision tokens.

This shifts supervision from predicting the final answer to learning the sequence of decisions that lead to the answer. Although trained via supervised fine-tuning, DBAgent learns a decision policy rather than following a fixed pipeline, allowing it to adapt its retrieval strategy to unseen decision states during inference.

4 Experiment

4.1 Experimental Setup

Datasets. We evaluate our method on two knowledge-intensive visual question answering benchmarks: InfoSeek (Chen et al., 2023) and Encyclopedic-VQA (E-VQA) (Mensink et al., 2023). Both datasets require models to answer image-based questions by leveraging external encyclopedic knowledge. InfoSeek consists of approximately 1.3M image–question–answer triplets associated with around 11K distinct Wikipedia pages. Following prior work, the validation set is further categorized into two subsets, namely *Unseen-Entity* and *Unseen-Question*, which evaluate the model's generalization ability to novel entities and novel question formulations in practice. Encyclopedic-VQA (E-VQA) contains over 221K question–answer pairs, each linked to up to five images and covering approximately 16.7K fine-grained entities corresponding to Wikipedia pages. Following standard practice, we report results on the official test set.

Baselines. We compare DBAgent with a diverse set of baselines that cover different modeling paradigms in our experiments. These include multimodal large language models that directly answer questions, retrieval-augmented generation models, and models that incorporate explicit reasoning mechanisms.

Evaluation Metrics. We evaluate both answer quality and retrieval quality, following the official protocols of each dataset. For InfoSeek, we use Exact Match (EM) as the primary metric. A prediction is considered correct if it exactly matches any of the ground-truth answers. For E-VQA, we adopt the BERT-based Matching (BEM) score(Bulian et al., 2022), which measures the semantic similarity between predicted answers and ground-truth answers. To assess retrieval performance, we report a hit-based retrieval accuracy, which measures whether the ground-truth Wikipedia article is successfully covered by the retrieved results at any turn. This metric reflects how effectively the retrieval module provides relevant external evidence for downstream reasoning and answer generation.

4.2 Implementation Details

Knowledge Base and Retrieval Tools. We construct an external knowledge base from Wikipedia and support both image-level and text-level retrieval. For image retrieval, we follow an EchoSight-style indexing strategy by encoding all Wikipedia images into a dense vector space and maintaining a mapping from each image to its source article(Yan and Xie, 2024). Given a query image, the system retrieves the most similar results and returns the associated Wikipedia article as candidate evidence. In all experiments, we use the top-k results with $k = 1$. For text retrieval, we segment each Wikipedia article into multiple textual sections and encode all sections using BGE embeddings. Retrieval is performed at the section level, and the top-k most relevant sections are returned as textual evidence, with $k = 3$. We use Qwen2.5-VL-7B-Instruct as the backbone model for both training and inference in all experiments.

Training Setup. We fine-tune Qwen2.5VL-7B using supervised fine-tuning on our trajectory-based training datasets. The trajectories are grouped into three difficulty levels, namely easy, medium, and hard, according to question complexity and the required retrieval depth, and are sampled with a balanced ratio of 1:1:1. They are constructed from the InfoSeek and E-VQA benchmarks, yielding approximately 160K samples from each dataset. To prevent the model from memorizing retrieved content, we apply a loss-masking strategy that excludes tool-returned evidence tokens from supervision. Additional hyperparameter choices and implementation details are provided in the supple-

Method	LLM	Retriever	E-VQA		InfoSeek		
			Single-Hop	All	Unseen-Q	Unseen-E	All
Zero-shot MLLMs							
BLIP-2	Flan-T5$_{XL}$	–	12.6	12.4	12.7	12.3	12.5
InstructBLIP	Flan-T5$_{XL}$	–	11.9	12.0	8.9	7.4	8.1
LLaVA-v1.5	Vicuna-7B	–	16.3	16.9	9.6	9.4	9.5
GPT-4V	GPT-4V	–	26.9	28.1	15.0	14.3	14.6
Qwen2.5-VL-7B (Base)	Qwen2.5-VL-7B (Base)	–	21.7	20.3	22.8	24.1	23.7
Classical Retrieval-Augmented Models							
DPR	Multi-passage BERT	CLIP ViT-B/32	29.1	–	–	–	12.4
RORA-VLM	LLaVA-v1.5-7B	CLIP ViT-L/14	–	20.3	25.1	27.3	–
EchoSight	Mistral-7B/LLaMA-3-8B	EVA-CLIP-8B	19.4	–	–	–	27.7
Wiki-LLaVA	LLaVA-v1.5-7B	CLIP ViT-L/14 + Contriever	17.7	20.3	30.1	27.8	28.9
Retrieval-Augmented Models with Reasoning / RL							
ReflectVA	LLaMA-3.1-8B	EVA-CLIP-8B	28.0	29.2	40.4	39.8	40.1
ReflectiVA	Qwen2.5-VL-7B	EVA-CLIP-8B	36.8	36.8	43.5	44.3	43.9
VLM-PRF	Qwen2.5-VL-7B	EVA-CLIP-8B	37.1	36.0	43.3	42.7	42.8
VLM-PRF	InternVL3-8B	EVA-CLIP-8B	40.1	39.2	43.5	42.1	42.5
Ours: Agent-based Search							
DBAgent (w/o SFT)	Qwen2.5-VL-7B	EVA-CLIP-8B + bge-m3	23.6	23.9	24.9	24.2	24.4
DBAgent (SFT: InfoSeek)	Qwen2.5-VL-7B	EVA-CLIP-8B + bge-m3	45.1	44.3	**46.5**	**51.0**	**49.9**
DBAgent (SFT: E-VQA)	Qwen2.5-VL-7B	EVA-CLIP-8B + bge-m3	**46.5**	**45.8**	43.1	50.2	48.4
DBAgent (SFT: Mixed)	Qwen2.5-VL-7B	EVA-CLIP-8B + bge-m3	46.0	45.2	43.6	50.4	48.7

Table 1: Main results on E-VQA and InfoSeek. We report VQA accuracy (%) under different LLMs and retrievers. Our agent-based methods are highlighted in light blue.

Trajectory Type	Sample Proportion	Retrieval Recall	Answer Accuracy
A	5.4	–	69.7
I→A	25.7	65.9	56.0
T→A	36.1	81.6	49.5
I→T→A	15.7	55.2	43.5
T→T→A	17.1	70.3	41.1

Table 2: Trajectory-level analysis on the test set.

Model	Generator	E-VQA	InfoSeek		
		Single-Hop	Un-Q	Un-E	All
Qwen2.5-VL-3B	Qwen2.5-3B	72.1	47.0	43.0	44.9
Qwen2.5-VL-7B	Qwen2.5-7B	78.3	41.6	41.3	41.4
ReflectiVA	Qwen2.5-VL-3B	72.9	53.4	53.9	53.7
Wiki-LLaVA	LLaVA-v1.5-7B	38.5	52.7	50.3	51.5
ReflectiVA	LLaVA-MORE-8B	75.2	57.8	57.4	57.6
ReflectiVA	Qwen2.5-VL-7B	71.3	56.0	56.0	56.0
ReAG	Qwen2.5-VL-7B	81.5	60.7	58.9	59.7
DBAgent (SFT: InfoSeek)	Qwen2.5-VL-7B	80.2	60.0	61.4	61.1
DBAgent (SFT: E-VQA)	Qwen2.5-VL-7B	79.4	59.6	60.0	59.9
DBAgent (SFT: Mixed)	Qwen2.5-VL-7B	80.9	60.2	62.7	62.1

Table 3: VQA accuracy scores on Encyclopedic-VQA and InfoSeek with oracle Wikipedia pages.

mentary material. All data construction procedures are conducted exclusively on the standard training split and do not involve any information from the evaluation sets.

4.3 Main Results and Analyses

Overall Results on Encyclopedic-VQA and InfoSeek We evaluate our method on two representative KB-VQA benchmarks in the experiments. The results are shown in Table 1. We train our models DBAgent under three data settings: InfoSeek, E-VQA, and a mixture of the two. Across all settings, DBAgent achieves the best performance under all settings. Specifically, it reaches 45.2% on E-VQA (All), outperforming the strongest baseline by about 6 points. On InfoSeek (All), DBAgent achieves 48.7%, which is also substantially higher than existing methods. Compared with zero-shot MLLMs and conventional RAG-based approaches, our method shows substantial improvements on both datasets. Moreover, DBAgent maintains consistent advantages on the Unseen-Q and Unseen-E splits of InfoSeek, achieving 43.6% and 50.4%. These results indicate that the DBAgent not only improves overall answer accuracy, but also generalizes better to unseen question forms and previously unseen entities.

Trajectory Distribution and Difficulty Stratification We analyze the distribution of different reasoning trajectories, as shown in Table 2. Here, I, T, and A denote image retrieval, text retrieval, and answer generation, respectively. As the trajectory length increases, retrieval recall improves (e.g., 70.3% for T→T→A), but answer accuracy drops (down to 41.1%). This reflects that longer trajectories usually correspond to more difficult questions

Method	E-VQA (All)	InfoSeek (All)
No Retrieval	20.3	23.7
Forced Image Retrieval	22.5	27.7
Caption-based RAG	24.7	30.5
DBAgent (SFT: Mixed)	45.2	48.7

Table 4: Ablation study on different retrieval and decision strategies. We report the overall accuracy on E-VQA and InfoSeek.

in practice. These results indicate that using a fixed-depth retrieval strategy is not suitable for KB-VQA. Instead, adaptively adjusting the search depth better matches the task structure.

Oracle Analysis with Ground-Truth Wikipedia Pages Under the oracle setting, where the ground-truth Wikipedia pages are directly provided, DBAgent still maintains a clear advantage as shown in Table 3. This suggests that our gains are not solely due to improved retrieval, but also come from more effective evidence selection and evidence-grounded reasoning.

4.4 Ablation Studies and Behavioral Analysis

Ablation on Retrieval and Decision Strategies As shown in Table 4, simply adding retrieval does not always lead to better performance. In particular, two fixed retrieval variants show only marginal gains: (i) performing image retrieval for every sample and using the Top-1 matched document, and (ii) first generating a caption and then conducting text retrieval with Top-3 passages.These results suggest that DBAgent's advantage lies in deciding when and how to retrieve evidence.

Impact of Knowledge Base Size We further analyze the impact of the knowledge base size. As shown in Figure 3,when the size increases from 10k to 100k, the performance of all methods drops. In contrast, DBAgent shows a much smoother degradation, dropping from 63.3% to **48.7%**. This indicates that our method is more robust in high-noise scenarios, further supporting that decision-based search is more suitable for KB-VQA task.

Impact of Different Top-k Settings As shown in Table 5, a smaller Top-k limits retrieval recall, whereas a larger Top-k not only introduces more retrieval noise but also substantially increases the context length, diluting critical information and leading to a clear performance trade-off. For example, on InfoSeek, a moderate Top-k setting achieves

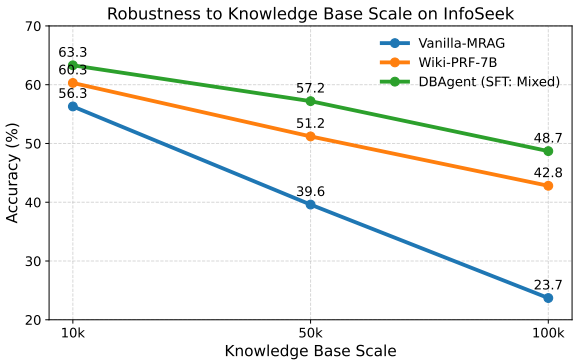

Figure 3: ablation studies on the scale of knowledge base on InfoSeek.

Text k	Image k			Text k	Image k		
	1	2	3		1	2	3
1	40.4	40.2	39.8	1	44.4	43.9	42.6
3	45.2	45.1	43.7	3	48.7	48.2	46.6
5	45.2	44.9	40.6	5	49.1	47.5	46.3
(a) E-VQA				(b) InfoSeek			

Table 5: Impact of retrieval top-k for text and image retrieval. We report overall accuracy (%) on E-VQA and InfoSeek.

48.7%, while further increasing k does not lead to continuous improvements. Our method is more stable under different Top-k combinations. In particular, it performs best when the text Top-k is 3 and the image Top-k is 1, showing that multi-step decision-making can effectively mitigate noise accumulation.

5 Conclusion

We present DBAgent, a decision-based search agent framework for knowledge-based visual question answering (KB-VQA). Instead of following a fixed retrieval-then-generation pipeline, DBAgent formulates KB-VQA as a multi-step decision-making process, where the model dynamically selects among different actions based on its current information state. To enable effective learning of such behaviors, we further propose an automated trajectory construction framework and build a large-scale, high-quality training dataset that explicitly records reasoning steps, tool usage, and intermediate decisions. This dataset provides structured supervision for training decision-aware agents. Extensive experiments on InfoSeek and E-VQA demonstrate that DBAgent consistently outperforms existing retrieval-augmented baselines, confirming the effectiveness and feasibility of our framework.

6 Limitations

Datasets and Generalization Our experiments are conducted on two KB-VQA benchmarks, InfoSeek and E-VQA. These datasets contain many entity-based questions and long-tail knowledge. However, they are curated benchmarks with relatively clean retrieval environments. In real-world settings, retrieved evidence may be noisier, incomplete, or inconsistent. How well the agent performs in such open-world environments requires further study.

Trajectory Construction Our method relies on automatically generated multi-step trajectories for training. This allows scalable data construction without manual annotation. However, the trajectory quality depends on the prompting strategy and the base model used during generation. Errors in these trajectories may influence the learned reasoning and tool-selection behavior.

Action Space Design Our agent operates with a predefined set of actions, including answer generation, image retrieval, text retrieval, and captioning. These actions cover common KB-VQA scenarios. However, real applications may require interaction with additional tools such as knowledge graphs or external APIs.

Inference Efficiency Our framework performs iterative reasoning and retrieval during inference. This step-by-step process helps the model gather information adaptively. However, it may increase inference latency compared with single-pass retrieval pipelines.

References

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Moitinho de Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haim ing Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, and 260 others. 2023. Gpt-4 technical report.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *ArXiv*, abs/2502.13923.

Jannis Bulian, Christian Buck, Wojciech Gajewski, Benjamin Boerschinger, and Tal Schuster. 2022. Tomayto, tomahto. beyond token-level answer equivalence for question answering evaluation. In *Conference on Empirical Methods in Natural Language Processing*.

Davide Caffagni, Federico Cocchi, Nicholas Moratelli, Sara Sarto, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. 2024. Wiki-llava: Hierarchical retrieval-augmented generation for multimodal llms. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1818–1826.

Jack Xiaoyu Chen, Fazhong Liu, Na Liu, Yu Luo, Erqu Qin, Harry Zheng, Tian Dong, Haojin Zhu, Yan Meng, and Xiao Wang. 2025. Step-wise adaptive integration of supervised fine-tuning and reinforcement learning for task-specific llms. *ArXiv*, abs/2505.13026.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multilinguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Annual Meeting of the Association for Computational Linguistics*.

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? *ArXiv*, abs/2302.11713.

Changin Choi, Wonseok Lee, Jungmin Ko, and Wonjong Rhee. 2025. Multimodal iterative rag for knowledge-intensive visual question answering. *ArXiv*, abs/2509.00798.

Federico Cocchi, Nicholas Moratelli, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. 2024. Augmenting multimodal llms with self-reflective tokens for knowledge-based visual question answering. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9199–9209.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Albert Li, Pascale Fung, and Steven C. H. Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. *ArXiv*, abs/2305.06500.

Jiaqi Deng, Kaize Shi, Zonghan Wu, Huan Huo, Dingxian Wang, and Guandong Xu. 2025. Enabling collaborative parametric knowledge calibration for retrieval-augmented vision question answering.

Yuwei Du, Jie Feng, Jie Zhao, and Yong Li. 2024. Trajagent: An llm-agent framework for trajectory modeling via large-and-small model collaboration.

Yichen He, Guanhua Huang, Peiyuan Feng, Yuan Lin, Yuchen Zhang, Hang Li, and Weinan E. 2025. Pasa: An llm agent for comprehensive academic paper search. In *Annual Meeting of the Association for Computational Linguistics*.

Yuyang Hong, Jiaqi Gu, Qi Yang, Lubin Fan, Yue Wu, Ying Wang, Kun Ding, Shiming Xiang, and Jieping Ye. 2025. Knowledge-based visual question answer with multimodal processing, retrieval and filtering. *ArXiv*, abs/2510.14605.

Paul Lerner, Olivier Ferret, and Camille Guinaudeau. 2024. Cross-modal retrieval for knowledge-based visual question answering. In *European Conference on Information Retrieval*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning*.

Shiyu Li, Yang Tang, Yifan Wang, Peiming Li, and Xi Chen. 2025. Reseek: A self-correcting framework for search agents with instructive rewards. *ArXiv*, abs/2510.00568.

Yangning Li, Yinghui Li, Xinyu Wang, Yong Jiang, Zhen Zhang, Xinran Zheng, Hui Wang, Hai-Tao Zheng, Pengjun Xie, Philip S. Yu, Fei Huang, and Jingren Zhou. 2024. Benchmarking multimodal retrieval augmented generation with dynamic vqa dataset and self-adaptive planning agent. *ArXiv*, abs/2411.02937.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023. Improved baselines with visual instruction tuning. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26286–26296.

Mingyang Liu, Gabriele Farina, and Asuman E. Ozdaglar. 2025. Uft: Unifying supervised and reinforcement fine-tuning. *ArXiv*, abs/2505.16984.

Xinwei Long, Zhiyuan Ma, Ermo Hua, Kaiyan Zhang, Biqing Qi, and Bowen Zhou. 2025. Retrieval-augmented visual question answering via built-in autoregressive search engines. *ArXiv*, abs/2502.16641.

Haohao Luo, Jiayi Kuang, Wei Liu, Ying Shen, Jian Luan, and Yang Deng. 2025. Browsing like human: A multimodal web agent with experiential fast-and-slow thinking. In *Annual Meeting of the Association for Computational Linguistics*.

Thomas Mensink, Jasper R. R. Uijlings, Lluís Castrejón, Arushi Goel, Felipe Cadar, Howard Zhou, Fei Sha, Andre F. de Araújo, and Vittorio Ferrari. 2023. Encyclopedic vqa: Visual questions about detailed properties of fine-grained categories. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3090–3101.

Jingyuan Qi, Zhiyang Xu, Rulin Shao, Yang Chen, dingnan jin, Yu Cheng, Qifan Wang, and Lifu Huang. 2024. Rora-vlm: Robust retrieval-augmented vision language models. *ArXiv*, abs/2410.08876.

Xin Su, Man Luo, Kris W Pan, Tien Pei Chou, Vasudev Lal, and Phillip Howard. 2024. Sk-vqa: Synthetic knowledge generation at scale for training context-augmented multimodal llms. *ArXiv*, abs/2406.19593.

Michael Sullivan, Mareike Hartmann, and Alexander Koller. 2025. Procedural environment generation for tool-use agents. *ArXiv*, abs/2506.11045.

Quan Sun, Yuxin Fang, Ledell Yu Wu, Xinlong Wang, and Yue Cao. 2023. Eva-clip: Improved training techniques for clip at scale. *ArXiv*, abs/2303.15389.

Ryota Tanaka, Taichi Iki, Taku Hasegawa, Kyosuke Nishida, Kuniko Saito, and Jun Suzuki. 2025. Vdocrag: Retrieval-augmented generation over visually-rich documents. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24827–24837.

Chenyu Wang, Weixin Luo, Qianyu Chen, Haonan Mai, Jindi Guo, Sixun Dong, Xi Xuan, Zhengxin Li, Lin Ma, and Shenghua Gao. 2024. Mllm-tool: A multimodal large language model for tool agent learning. *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 6678–6687.

Feijie Wu, Wei Zhu, Yuxiang Zhang, Soumya Chatterjee, Jiarong Zhu, Fan Mo, Rodin Luo, and Jing Gao. 2025. Portool: Tool-use llm training with rewarded tree. *ArXiv*, abs/2510.26020.

Junbin Xiao, Nanxin Huang, Hangyu Qin, Dongyang Li, Yicong Li, Fengbin Zhu, Zhulin Tao, Jianxing Yu, Liang Lin, Tat-Seng Chua, and Angela Yao. 2024. Videoqa in the era of llms: An empirical study. *International Journal of Computer Vision*, 133:3970 – 3993.

Yibin Yan and Weidi Xie. 2024. Echosight: Advancing visual-language models with wiki knowledge. In *Conference on Empirical Methods in Natural Language Processing*.

Bin Yu, Hang Yuan, Haotian Li, Xueyin Xu, Yuliang Wei, Bailing Wang, Weizhen Qi, and Kai Chen. 2025. Long-short chain-of-thought mixture supervised fine-tuning eliciting efficient reasoning in large language models. *ArXiv*, abs/2505.03469.

Zhengxuan Zhang, Yin Wu, Yuyu Luo, and Nan Tang. 2025. Fine-grained knowledge structuring and retrieval for visual question answering.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *Conference on Empirical Methods in Natural Language Processing*.

A Datasets

We evaluate our method on two widely used knowledge-based visual question answering benchmarks, InfoSeek and E-VQA. Both datasets are designed to test whether a model can go beyond the

visible content of an image and incorporate external knowledge. However, they differ significantly in their construction principles, knowledge types, and reasoning requirements. This diversity allows us to comprehensively examine the decision-making and search behaviors of our agent.

InfoSeek. InfoSeek is a large-scale benchmark specifically designed for knowledge-intensive visual question answering. Each example consists of an image, a natural language question, and a set of acceptable answers. Unlike traditional VQA datasets that focus on object recognition or simple visual attributes, InfoSeek emphasizes long-tail entities and factual knowledge that is rarely memorized by vision-language models.

A key characteristic of InfoSeek is that most questions cannot be answered solely based on the image. Instead, the image serves as a visual anchor that points to a specific entity, such as a landmark, an animal species, a historical artifact, or a cultural object. The question then asks for factual attributes of that entity, such as its origin, function, scientific classification, or historical background. Answering such questions typically requires consulting an external knowledge source.

InfoSeek also exhibits strong diversity in question types. Some questions can be resolved with a single retrieval step, while others require multiple rounds of refinement because the initial query is too ambiguous or incomplete. This makes InfoSeek particularly suitable for evaluating whether a model can decide when to retrieve, how to formulate queries, and how to revise them based on newly obtained evidence.

E-VQA. E-VQA is another benchmark designed for entity-centric visual question answering. Each question is explicitly tied to a visual entity in the image, and the answer usually involves factual or encyclopedic knowledge about that entity. Compared to InfoSeek, E-VQA focuses more on fine-grained attributes, such as taxonomic categories, functional properties, or specific biographical facts.

A notable feature of E-VQA is that many questions assume the model can correctly identify the entity from the image. However, recognizing the entity alone is often insufficient to answer the question. The model must also retrieve or recall the corresponding factual information. This makes E-VQA a good testbed for separating two distinct challenges: visual grounding and knowledge acquisition.

In addition, E-VQA includes a wide range of domains, such as animals, plants, monuments, tools, and artworks. This diversity further increases the difficulty of relying on parametric knowledge alone, especially for long-tail concepts.

Comparison and Motivation. Although both datasets require external knowledge, they differ in how that knowledge is triggered. In InfoSeek, the model often needs to discover what the image refers to before it can even formulate a meaningful query. In E-VQA, the model can often identify the entity but still lacks the specific attribute required by the question.

These differences make the two datasets complementary. InfoSeek stresses the need for adaptive entity discovery and query formulation, while E-VQA emphasizes fine-grained factual lookup after visual grounding. Together, they form a challenging testbed for dynamic, multi-step reasoning.

Importantly, neither dataset can be reliably solved with a fixed retrieval-then-answering pipeline. Some questions require no retrieval at all, some require a single retrieval step, and others require multiple refinements. This variability motivates our formulation of KB-VQA as a decision-making process rather than a static pipeline. Our search-agent framework is designed to explicitly model this variability, enabling the agent to decide when to retrieve, which modality to use, and how to refine its queries.

B Baselines

We compare our method with a broad set of baselines that represent different paradigms for multimodal reasoning and retrieval in KB-VQA. These baselines can be roughly grouped into three categories: zero-shot multimodal large language models, classical retrieval-augmented models with fixed pipelines, and retrieval-augmented models with explicit reasoning or reflection mechanisms. Together, they cover the most common design choices adopted in existing systems.

B.1 Zero-shot Multimodal Large Language Models.

We include several strong pretrained multimodal models that directly generate answers from images and questions without explicit access to external knowledge sources, including BLIP-2, InstructBLIP, LLaVA, and GPT-4V.

BLIP-2 and InstructBLIP.(Li et al., 2023; Dai et al., 2023) These models connect a frozen large language model with a visual encoder via a lightweight query transformer. This design enables flexible multimodal generation while preserving the generalization ability of large language models. However, factual knowledge is implicitly stored in the model parameters, which makes these models sensitive to the coverage of the pretraining data. As a result, they often struggle with long-tail entities and rare facts.

LLaVA-v1.5.(Liu et al., 2023) LLaVA follows a similar philosophy but is trained with large-scale multimodal instruction tuning. This improves its robustness and instruction-following ability, especially for conversational and compositional queries. Nevertheless, it still relies on parametric memory for factual information and does not explicitly interact with external knowledge bases.

GPT-4V.(Achiam et al., 2023) GPT-4V is a strong proprietary vision-language model with broad general knowledge and strong reasoning ability. It serves as an important reference point for zero-shot performance. Despite its strength, its access to knowledge remains implicit, and it does not expose explicit retrieval or evidence-grounding behaviors.

Qwen2.5-VL.(Bai et al., 2025) Qwen2.5-VL is a recent open-source multimodal large language model that supports strong visual understanding and instruction following. It integrates a vision encoder with a large language model and is trained with large-scale multimodal instruction data. This design enables the model to handle a wide range of multimodal tasks, including visual question answering and image-grounded reasoning. However, similar to other zero-shot multimodal models, factual knowledge is mainly stored in model parameters. Without explicit retrieval mechanisms, the model may still struggle with questions that require external knowledge or long-tail factual information.

B.2 Classical Retrieval-Augmented Models.

DPR.(Lerner et al., 2024) DPR is a dense passage retrieval framework originally developed for open-domain question answering. It uses a dual-encoder architecture to embed queries and passages into a shared vector space. In our setting, DPR serves as a representative text retriever that retrieves relevant passages based on a fixed query formulation.

RORA-VLM.(Qi et al., 2024) RORA-VLM integrates retrieval into the multimodal pipeline by combining a vision-language model with a CLIP-based image retriever. Retrieved visual content is used to augment the generation process, allowing the model to access additional visual context beyond the input image.

EchoSight.(Yan and Xie, 2024) EchoSight is a multimodal retrieval-augmented framework that retrieves visually or semantically related images and associated textual descriptions using CLIP-style encoders. The retrieved information is concatenated with the original input and passed to a language model for answer generation.

Wiki-LLaVA.(Caffagni et al., 2024) Wiki-LLaVA augments a vision-language model with external Wikipedia knowledge. It retrieves relevant articles or passages using a CLIP-based retriever and fuses the retrieved content into the language model input.

B.3 Retrieval-Augmented Models with Reasoning.

ReflectiVA.(Cocchi et al., 2024) ReflectiVA introduces a reflection-based reasoning strategy that encourages the model to generate intermediate reasoning traces and revise its answer if inconsistencies are detected. This design improves robustness and reduces certain types of hallucinations.

VL-MPRF.(Hong et al., 2025) VL-MPRF incorporates a multi-perspective reasoning framework that aggregates retrieved evidence from different views. It emphasizes structured reasoning over multiple knowledge snippets and integrates the reasoning process into the language model through prompt-based decomposition.

Unified Discussion. Although the above methods differ in architecture and training strategies, they share a common assumption: the structure of the reasoning and retrieval process is largely predefined. Retrieval is either always performed or implicitly triggered by prompt templates. The model is not explicitly trained to decide whether retrieval is necessary, which modality to use, or when to stop.

Moreover, query formulation is often treated as a static transformation rather than a learnable decision. When retrieval fails or returns insufficient

information, most systems either hallucinate or passively consume the returned evidence. In practice, KB-VQA exhibits diverse difficulty levels. Some questions can be answered directly from the image, while others require one or multiple rounds of refinement. A fixed pipeline cannot naturally express this diversity.

C Retriever

EVA-CLIP.(Sun et al., 2023) EVA-CLIP is a large-scale vision-language representation model built upon the CLIP framework. It improves visual representation quality by scaling both the vision encoder and the pretraining data. The model is trained with extensive image-text pairs and adopts stronger visual backbones, which leads to improved cross-modal alignment between images and text. In our framework, EVA-CLIP is used as the image retriever to retrieve visually or semantically related images from the external knowledge base. The retrieved images provide additional visual context that can support subsequent reasoning and answer generation.

BGE-M3.(Chen et al., 2024) BGE-M3 is a multilingual dense retrieval model designed for general-purpose information retrieval. It produces high-quality text embeddings that support both semantic matching and cross-lingual retrieval. Compared with earlier dense retrievers, BGE-M3 demonstrates strong performance across various retrieval benchmarks and supports flexible query formulations. In our framework, BGE-M3 is used as the text retriever to retrieve relevant textual evidence from the knowledge base. The retrieved passages provide complementary factual information that may not be directly inferred from the visual content alone.

D DBAgent Inference Algorithm

This appendix provides the detailed inference procedure of DBAgent, corresponding to the decision process described in Section 3.2. During inference, the agent iteratively generates reasoning tokens and selects actions based on the current information state. Depending on the generated action tag, the system invokes different retrieval tools and appends the returned evidence to the state. The process continues until the agent outputs the final answer or reaches the maximum turn budget.

Algorithm 1 Inference procedure of DBAgent with multi-turn tool interaction.

Require: image x, question q, policy model π_θ, text retriever $\mathcal{R}_{\text{text}}$, image retriever $\mathcal{R}_{\text{img}}$, maximum turn budget B

Ensure: final answer y

Initialize output $y \leftarrow \emptyset$, evidence buffer $\mathcal{E} \leftarrow \emptyset$

Initialize turn counter $b \leftarrow 0$
while $b < B$ **do**
  Initialize one-turn output $y_b \leftarrow \emptyset$
  while True **do**
    Sample next token $z \sim \pi_\theta(\cdot \mid x, q, \mathcal{E}, y, y_b)$
    Append z to y_b
    if z is a closing tag or $\langle eos \rangle$ **then**
      break
    end if
  end while
  $y \leftarrow y \| y_b$
  if <text_search> is detected in y_b **then**
    Parse query u from y_b
    Retrieve text evidence $e \leftarrow \mathcal{R}_{\text{text}}(u)$
    Insert <evidence>e</evidence> into $\mathcal{E}$
  else if <image_search> is detected in y_b **then**
    Parse image input q_{img} from y_b; if missing, set $q_{\text{img}} \leftarrow x$
    Retrieve visual evidence $e \leftarrow \mathcal{R}_{\text{img}}(q_{\text{img}})$
    Insert <evidence>e</evidence> into $\mathcal{E}$
  else if <answer> is detected in y_b **then**
    Parse final answer span $y^\star$
    return $y^\star$
  else
    (optional) append a short reflection message
  end if
  $b \leftarrow b + 1$
end while
return the last valid answer span if any =0

E Training Details

Training setup. We fine-tune our search-agent model using full-parameter supervised fine-tuning on the constructed multi-stage trajectories. All experiments are conducted on 8 NVIDIA A800 GPUs, each with 80GB memory. To support long multi-turn contexts and reduce memory consumption, we enable bfloat16 precision, gradient checkpointing, and ZeRO-3 optimization.

Optimization and scheduling. We adopt Qwen2.5-VL-7B-Instruct as the backbone model. The per-device training batch size is set to 1, and we use gradient accumulation with 4 steps, resulting in a global batch size of 32. We use a cosine learning rate scheduler with an initial learning rate of 7×10^{-6} and a warmup ratio of 0.06. Weight decay is set to 0.

Sequence length and training duration. To accommodate long reasoning trajectories, we set the maximum sequence length to 16,384 tokens. The model is trained for 3 epochs over approximately 200k trajectories. We randomly hold out 2% of the training data as a validation set and perform evaluation every 2000 steps. We report the best-performing checkpoint based on validation accuracy. All models are trained using the same optimization settings across datasets to ensure fair comparison.

Reproducibility. All hyperparameters, prompts, and trajectory construction rules are provided in the supplementary material. Our training setup follows standard practices in large-scale multimodal instruction tuning and can be reproduced with commonly available GPU clusters.

F Extended Analysis: Relationship Between Retrieval and Answer Correctness

To better understand the behavior of our search-agent framework, we analyze the relationship between retrieval correctness and final answer correctness. Table 6 summarizes the distribution of samples across four possible cases.

Correct retrieval and correct answer. This is the most common case (70.4%). It corresponds to ideal trajectories where the agent selects the appropriate tool, retrieves relevant evidence, and integrates it correctly into its reasoning.

Correct retrieval but wrong answer. This case accounts for 29.6% of the samples. Although the retrieved evidence already contains the required information, the model still fails to answer correctly. This suggests that errors are not always caused by retrieval failures, but often by weaknesses in reasoning or evidence utilization.

Incorrect retrieval but correct answer. We find that 38.6% of the samples can still be answered correctly even when retrieval is incorrect. This indicates that some questions can be solved using the model's parametric knowledge or visual cues, without relying on external evidence. These cases show that retrieval should not be treated as a mandatory step.

Table 6: Relationship between retrieval correctness and answer correctness. We report the proportion (%) of samples in each case.

	Answer Correct	**Answer Wrong**
Retrieval Correct	70.4	29.6
Retrieval Incorrect	11.4	88.6

Incorrect retrieval and wrong answer. In 71.4% of the cases with incorrect retrieval, the final answer is also wrong, confirming that retrieval quality remains crucial for knowledge-intensive questions.

Implications. Overall, retrieval and answer correctness are strongly correlated, but not perfectly aligned. Correct retrieval is helpful but not sufficient, and not all questions strictly depend on retrieval. This highlights a limitation of fixed retrieval-augmented pipelines, which assume that retrieval is always necessary and that a single retrieval step is sufficient.

In contrast, our framework treats KB-VQA as a sequential decision-making process, allowing the agent to decide when to retrieve, which tool to use, how to refine queries, and when to stop. This leads to more flexible and efficient trajectories that better match the diverse difficulty levels of real-world KB-VQA problems.

G Prompt Design

This subsection describes the prompts used in our search-agent framework. They cover two settings: (1) the main agent prompt used for inference and training, and (2) a set of multi-stage prompts used to construct high-quality trajectories. Our goal is not only to teach the model to answer questions, but also to teach it when external knowledge is needed, which tool to use, how to write better queries, and when to stop.

Unlike retrieval-augmented generation pipelines that follow a fixed order of steps, we treat KB-VQA as a multi-step decision process. At each step, the model reasons about the current state, selects an action, and updates its decision after observing

new information. The prompts are designed to expose these intermediate decisions in a standardized, machine-readable format.

Search-agent prompt for inference and training. The core prompt follows a simple think–act pattern. At each turn, the model first writes a short reasoning trace inside `<think>`, then outputs exactly one action tag. This structure encourages the model to reflect when it receives new information, instead of retrieving by default.

Answer action. The model outputs `<answer>` when the current image and available evidence are sufficient. The answer must be short and contain only the final answer span.

Text-retrieval action. The model outputs `<text_search>` when it cannot answer yet but can identify what the question refers to in the image. The query must be concise and specific. It should combine an entity name (or a short visual description) with the attribute asked by the question, so the retriever can fetch the missing knowledge efficiently.

Image-retrieval action. The model outputs `<image_search>` when it cannot answer and also cannot reliably identify the key entity from the image. To keep a stable tool interface, the content of this tag is fixed to the placeholder token `image_path`.

Optional caption for query refinement. If the model has already performed image retrieval and still needs text retrieval, it may output an optional `<caption>` block before the final action. The caption must describe only what is visible, without speculation. Its role is to provide concrete visual keywords that help form a better text query.

Formatting constraints. We enforce strict formatting rules. Each turn must start with `<think>` and end with exactly one action tag. Only the allowed tags may appear, and no trailing text is permitted. This makes the reasoning, the decision, and the tool usage explicit and easy to supervise.

Multi-stage trajectory construction. To teach dynamic decision-making, we construct trajectories with multiple stages. Each stage uses two prompts: an answering prompt and a judging prompt. The answering prompt produces an answer given the currently available information. The judging prompt then checks correctness (offline) and, when the answer is wrong, decides what to do next and produces a rewritten query or a tool choice. This design allows us to build trajectories with different structures, including direct answering, single-step retrieval, and multi-step refinement.

Stage 1: initial reasoning and routing. Stage 1 uses only the image and the question. The answering prompt requests a reasoning trace, the main entity, and a tentative answer. The judging prompt verifies the answer. If it is wrong, the judge determines whether the failure is caused by incorrect entity grounding or missing knowledge. If the entity is wrong, it routes to image retrieval. If the entity is correct but knowledge is missing, it routes to text retrieval. In both cases, it outputs the next action in a standardized format.

Stage 2: tool-based re-answering and query rewriting. Stage 2 follows the routing decision from Stage 1. In the image-based branch, the model receives text evidence obtained through image retrieval and answers again using the image plus evidence. If the answer is still unsupported, the judging prompt produces a detailed visual caption and a rewritten text query, where the caption is restricted to visible content only. In the text-based branch, the model answers using retrieved text evidence. If the answer remains unsupported, the judge explains what information is missing and outputs a new text query that is meaningfully different from the previous one.

Stage 3: final re-answering with new evidence. Stage 3 handles harder cases where Stage 2 is still insufficient. The model is given the full history of the trajectory together with newly retrieved text evidence from an updated query. The goal is to treat the new evidence as the primary factual source and produce a final answer. The Stage-3 answering prompt follows the same contract as Stage 2: a short `<think>` block focusing on what matters in the image and how the new evidence supports the conclusion, followed by a final `<answer>` block. No other tags are allowed.

Offline judging for data construction. For trajectory construction, we also use judging prompts to label whether the answer at each stage is correct under a semantic match rule (case-insensitive, minor wording differences allowed). When an answer is wrong, the judge outputs the next action choice and a rewritten query in the required format. To keep the trace realistic, the judge is instructed not

to mention gold answers or evaluation language in its explanations.

Design principles. We follow a few simple principles. First, decisions are explicit: the model must state its reasoning and select actions directly. Second, the output format is tightly constrained, which makes trajectories easy to parse and suitable for supervised learning. Third, answering and judging are separated, so we can supervise both solving and decision-making. Fourth, failed attempts are treated as useful signal: instead of discarding them, we turn them into refinement steps that teach better tool use. Finally, the framework is fully multimodal, because the image is considered at every stage rather than only at the beginning.

Search-Agent Prompt (Inference & Training)

Task. Answer the question. Always reason in `<think>...</think>` when you get new information (image or `<evidence>...</evidence>`).

Choose EXACTLY ONE action per turn:

Action 1: Answer.
`<answer> ... </answer>`
Use this action if the current image/evidence is sufficient to answer exactly.

Action 2: Text Retrieval.
`<text_search> QUERY </text_search>`
Use this action if you cannot answer yet but you can identify what the question refers to in the image.
QUERY rule:
- QUERY = (entity name or short visual description) + (asked attribute)
- Keep it concise and specific.

Action 3: Image Retrieval.
`<image_search>image_path</image_search>`
Use this action if you cannot answer and you cannot identify the key entity from the image.

IMPORTANT:
- The content inside `<image_search>` MUST be exactly `image_path` (fixed placeholder).
- Do NOT change it.

Caption (optional):
If you already used `<image_search>` and still need `<text_search>`, output an optional caption to help write a better query.
`<caption> ... </caption>`
- Caption must be ONE short visible-only sentence.
- `<caption>` may appear only before the final action tag.

Format rules:
- Output must start with `<think>`.
- Only use tags: `<think>`, `<answer>`, `<text_search>`, `<image_search>`, `<caption>`.
- End with exactly one action tag: `<answer>` or `<text_search>` or `<image_search>`.
- `<caption>` may appear only before the final action tag.
- After `</answer>`, `</text_search>`, or `</image_search>`, output nothing.

Answer format:
- `<answer>` must contain ONLY the final answer span.
- Keep it very short: prefer 1–4 words (or a single number/date/unit if required).
- Do NOT write a full sentence.
- Do NOT add explanations, punctuation, or prefixes.

Stage-1 Multimodal VQA Prompt

You are a multimodal question answering model. The user will provide one image and one question about that image.

Your task consists of three steps:

Step 1: Reasoning.
In `<think>...</think>`, write your detailed reasoning.

- Describe what you see in the image.
- Explain how the visual content relates to the question.
- Show the logical steps used to derive the answer.
- Do not include the final answer here.

Step 2: Entity Identification.
In `<entity>...</entity>`, output the main entity in the image.

- Use a short noun phrase.
- Output exactly one entity.

Step 3: Final Answer.
In `<answer>...</answer>`, output the final short answer only.

Strict Output Format (Mandatory):
`<think>. . . </think>`
`<entity>. . . </entity>`
`<answer>. . . </answer>`

- Each tag must appear exactly once and be properly closed.
- Do not output any other tags.
- Do not output anything before `<think>` or after `</answer>`.
- Do not write "unknown" in `<answer>`.

Example Output Format

```
<think>
I first look at the image. It shows a football
player wearing a light blue and white striped
jersey with the number 10. He is on a football
pitch with a ball at his feet. This jersey
pattern is characteristic of the Argentina
national team, so the answer is Argentina
national team.
</think>
<entity>Lionel Messi</entity>
<answer>Argentina national team</answer>
```

Stage-1 Judgment Prompt

You are a strong language model used to **JUDGE and POST-PROCESS** Stage-1 multimodal QA outputs.
For each sample, you will receive the following fields:

- [question]: the question about the image
- [stage1_think]: the reasoning text generated in Stage-1
- [stage1_entity]: the entity predicted in Stage-1
- [stage1_answer]: the answer predicted in Stage1
- [gold_answer]: the ground-truth answer
- [gold_entity]: the ground-truth entity (e.g., Wikipedia title)

Interpreting gold_answer:

- gold_answer denotes a **set** of acceptable answers, where elements are separated by "|".
- stage1_answer is considered **CORRECT** if it semantically matches *any* element in this set.

Your task consists of three steps:

Step 1: Answer Judgment.
Decide whether stage1_answer is **CORRECT** compared to gold_answer.

- Use semantic matching.
- Ignore case, punctuation, and minor wording differences.

Step 2: Correct Case.
If the answer is correct, output exactly:
[correct] and **nothing else**.

Step 3: Wrong Case.
If the answer is wrong, further judge whether stage1_entity matches gold_entity.

Case (a): Entity is Wrong.

- Use **IMAGE RETRIEVAL**.
- Output exactly:

```
[wrong]
<image_search> image_path </image_search>
<choose> ... </choose>
```

where:

- <image_search> must contain only the fixed placeholder image_path.
- <choose> briefly explains why stage1_entity does not match gold_entity.

Case (b): Entity is Correct but Knowledge is Missing.

- Use **TEXT RETRIEVAL**.
- Output exactly:

```
[wrong]
<text_search> ... </text_search>
<choose> ... </choose>
```

where:

- <text_search> contains exactly **one concise natural-language query**.
- The query must mention the entity and the missing information.
- Do not include reasoning sentences inside <text_search>.
- <choose> briefly explains why text retrieval is chosen.

Strict Output Format (Mandatory):

- Always start with [correct] or [wrong].
- For [correct], output only [correct].
- For [wrong], output exactly one of the following two options.

Option 1: Text Retrieval

```
[wrong]
<text_search> ... </text_search>
<choose> ... </choose>
```

Option 2: Image Retrieval

```
[wrong]
<image_search> image_path </image_search>
<choose> ... </choose>
```

- Never output both <text_search> and <image_search>.
- Do not output any other tags, JSON, or explanations outside the required format.

Stage-2 Image-Based Answering Prompt

You are a multimodal question answering model for a two-stage pipeline.
You will receive:

- The original image (image_path)
- A question (question)
- A short history from stage1_output that includes:
  - <choose>...</choose>: why the Image Retriever was chosen
  - <image_search>...</image_search> or <img_search>...</img_search>: the image path used for retrieval
- Retrieved text evidence in <evidence>...</evidence>

Your goal
Use the original image together with the retrieved text evidence as additional knowledge, then answer the question again more accurately.

Output ONLY two tags:

Tag 1:
<think> ... </think>

- Briefly describe what matters in the image.
- Use the evidence to fill in the missing knowledge and complete the reasoning.
- Do NOT put the final answer here.

Tag 2:
<answer> ... </answer>

- Output the final short answer ONLY.
- No explanation in <answer>.

Strict format:

- Output tags in this order exactly:

```
<think> ... </think>
<answer> ... </answer>
```

- Each tag must appear exactly once and be properly closed.
- Do NOT output any other tags.
- Do NOT output anything before <think> or after </answer>.
- Must always output a concrete answer in <answer>.
- Even if the information is incomplete or uncertain, infer and provide the single most likely answer.

Stage-2 Image-Based Judge and Rewrite Prompt

You are a multimodal model used to **JUDGE and REWRITE** queries after a Stage-2 image-based answering attempt.
You will receive:

- ONE image (image_path)
- [question]: the question
- [stage1_output]: the Stage-1 routing output that contains <choose> and <image_search> (history)
- [evidence]: text retrieved via image retrieval
- [stage2_answer]: the Stage-2 answer produced after seeing evidence
- [gold_answer]: ground-truth answer (FOR EVALUATION ONLY)

Interpreting gold_answer:

- gold_answer denotes a **set** of acceptable answers, where elements are separated by "|".
- stage2_answer is CORRECT if it semantically matches any element of this set.

CRITICAL RULE (trajectory realism):

- In real inference, the model does NOT know gold_answer.
- If you output [wrong] and rewrite a query, your <think> MUST be based ONLY on:
  - question
  - image content
  - evidence
  - whether stage2_answer is supported
- You MUST NOT mention gold_answer, "ground truth", or any comparison to it inside <think>.

Task 1 (offline evaluation):
Decide whether stage2_answer is CORRECT compared to gold_answer (semantic match; ignore case, punctuation, and minor wording differences).

Task 2 (output):
If CORRECT, output exactly:
[correct]
and NOTHING else.

If WRONG, output exactly:
[wrong]
<caption> ... </caption>
<think> ... </think>
<text_search> ... </text_search>
where:

<caption>

- A detailed, faithful description of the image relevant to the question.
- Mention main objects, scene, and visible context.
- Do NOT speculate beyond what is visible.

<think>

- Explain what information is missing or unclear.
- Focus on why the current evidence is insufficient.
- Do NOT mention gold_answer or "ground truth".

<text_search>

- ONE concise natural-language query.
- Do NOT include reasoning sentences.
- Incorporate key terms from the caption and the question.
- Must be meaningfully different from the previous retrieval intent.

Output constraints (VERY IMPORTANT):

- Always start with [correct] or [wrong].
- For [correct], output ONLY [correct].
- For [wrong], output ONLY:

```
[wrong]
<caption> ... </caption>
<think> ... </think>
<text_search> ... </text_search>
```

- Do not output any other tags, JSON, explanations, or trailing text.

Stage-2 Text-Based Answering Prompt

You are a multimodal question answering model.
The user will send you:

- ONE image (image_path)
- ONE question (question)
- A short history from stage1_output that includes:
  - <choose>...</choose>: why text retrieval was chosen
  - <text_search>...</text_search>: the rewritten query used for retrieval
- Retrieved text evidence in <evidence>...</evidence>

Your goal
Use the image, the question, and the retrieved evidence as additional knowledge, then answer the question again.

You must output TWO tags:

Tag 1:
<think> ... </think>

- Briefly describe what you see in the image (only what matters).

- Use the evidence to correct or complete missing knowledge.
- Do NOT put the final answer here.

Tag 2:
`<answer> ... </answer>`

- Output the final short answer ONLY.
- No explanation in `<answer>`.

Strict output format (VERY IMPORTANT):

- Output tags in the following order exactly:

```
<think> ... </think>
<answer> ... </answer>
```

- Each tag must appear exactly once and be properly closed.
- Do NOT output any other tags.
- Do NOT output anything before `<think>` or after `</answer>`.
- Even if the information is incomplete or uncertain, infer and provide the single most likely answer.

Example:

```
<think>
The image shows a conger eel-like fish.
The evidence states conger eels belong to
the family Congridae, so the closest upper
taxonomy asked by the question should be that
family.
</think>
<answer>
Congridae
</answer>
```

Stage-2 Text-Based Judge and Rewrite Prompt

You are a language model used to **JUDGE and REWRITE** queries after a Stage-2 text-based answering attempt.
You will receive:

- `[question]`: the question
- `[stage1_output]`: the Stage-1 routing output that contains `<choose>` and `<text_search>`
- `[evidence]`: retrieved text evidence
- `[stage2_answer]`: the Stage-2 answer produced after seeing evidence
- `[gold_answer]`: ground-truth answer (FOR EVALUATION ONLY)

Interpreting `gold_answer`:

- `gold_answer` denotes a **set** of acceptable answers, where elements are separated by "|".
- `stage2_answer` is CORRECT if it semantically matches any element of this set.

CRITICAL RULE (trajectory realism):

- In real inference, the model does NOT know `gold_answer`.
- If you output `[wrong]` and rewrite a query, your `<think>` MUST be based ONLY on:
  - the question
  - the evidence
  - whether `stage2_answer` is supported
- You MUST NOT mention `gold_answer`, "ground truth", or any comparison to it inside `<think>`.

Task 1 (offline evaluation):
Decide whether `stage2_answer` is CORRECT compared to `gold_answer` (semantic match; ignore case, punctuation, and minor wording differences).

Task 2 (output):
If CORRECT, output exactly:
`[correct]`
and NOTHING else.

If WRONG, output exactly:
`[wrong]`
`<think> ... </think>`
`<text_search> ... </text_search>`

where:

`<think>`

- Explain what is missing, unclear, or mismatched.
- Explain why a new query is needed.
- Do NOT mention `gold_answer`.

`<text_search>`

- ONE concise natural-language query.
- Do NOT include reasoning sentences.
- Must be meaningfully different from the previous query.

Output constraints (VERY IMPORTANT):

- Always start with `[correct]` or `[wrong]`.
- For `[correct]`, output ONLY `[correct]`.
- For `[wrong]`, output ONLY:

```
[wrong]
<think> ... </think>
<text_search> ... </text_search>
```

- Do not output any other tags, JSON, explanations, or trailing text.

Stage-3 Image-Based Answering Prompt

You are a multimodal question answering model (Stage-3 answering).
You will receive:

- ONE image (`image_path`)
- ONE question
- Stage-1 routing history in `stage1_output`:
  - `<choose>...</choose>` and
  - `<image_search>...</image_search>` (image retrieval was chosen)
- Stage-2 context (history):
  - `[stage2_evidence]`: text retrieved via image retrieval
  - `[stage2_answer]`: the Stage-2 answer

- [stage2_judge] or [stage2_new_caption / new_text_search_query]:
  - a caption of the image
  - a rewritten text retrieval query

- Stage-3 new retrieved evidence:
  - [stage3_new_evidence]: text retrieved via the rewritten query

Your goal
Use the image, the question, the Stage-2 background evidence, and the Stage-3 new evidence as the most relevant fresh knowledge, then answer the question again.

You MUST output TWO tags in this exact order:
<think> ... </think>
<answer> ... </answer>

<think>

- Briefly describe what you see in the image (only what matters).
- Use stage3_new_evidence as the primary factual source.
- You may use stage2_evidence as supplementary background.
- Do NOT mention gold/ground truth or any evaluation artifacts.

<answer>

- Output the final short answer ONLY.
- No extra explanation inside <answer>.

Strict output format (VERY IMPORTANT):

- Output tags in the following order exactly:

<think> ... </think>
<answer> ... </answer>

- Each tag must appear exactly once and be properly closed.
- Do NOT output any other tags.
- Do NOT output anything before <think> or after </answer>.
- Even if uncertain, output the single most likely answer.

Stage-3 Judge Prompt (Image Branch)

You are a multimodal model used to **JUDGE** Stage-3 answers and, if needed, produce a **HARD-LEARNING forced reasoning trace**.
You will receive:

- ONE image (image_path)
- [question]
- [stage1_output]: contains <image_search> and <choose>
- [evidence_stage2]: text retrieved via image retrieval
- [stage2_answer]
- [stage2_judge]: may contain
  - <caption>...</caption>
  - <think>...</think>
  - <text_search>...</text_search>
- [stage2_new_caption]: image caption used to rewrite the query (if present)
- [new_text_search_query]: rewritten text query (if present)
- [stage3_text_search_query]: the query actually used for Stage-3 retrieval (if present)
- [stage3_new_evidence]: NEW evidence retrieved using the rewritten query
- [stage3_answer]: Stage-3 answer
- [gold_answer]: acceptable answers separated by "|"

TASK 1 (OFFLINE JUDGMENT):
Decide whether stage3_answer is CORRECT compared to gold_answer.

CRITICAL CORRECTNESS RULES:

- gold_answer is a SET of acceptable answers separated by "|".
- stage3_answer is CORRECT if it semantically matches ANY ONE option.
- Ignore case, punctuation, whitespace, and minor wording differences.
- Allow standard normalization (e.g., units, rounding).

OUTPUT FORMAT CONSTRAINTS (VERY IMPORTANT):

- For [correct], output ONLY [correct].
- For [wrong], output ONLY [wrong].
- No extra text, no additional tags, no JSON.

Stage-3 Text-Based Answering Prompt

You are a multimodal question answering model.
The user will send you:

- ONE image (image_path)
- ONE question (question)

History (trajectory):

- Stage-1 output (stage1_output) that may include:
  - <choose>...</choose>: why a retriever was chosen
  - <text_search>...</text_search>: the rewritten query used for Stage-2 retrieval
- Stage-2 attempt summary:
  - stage2_answer: the answer produced at Stage-2
  - stage2_new_think: why information was still insufficient (if any)
  - new_text_search_query: the rewritten query for Stage-3 retrieval (if any)

New retrieval for Stage-3:

- stage3_text_search_query: the query used to retrieve new evidence
- stage3_new_evidence in <evidence>...</evidence>

Your goal:
Use the image, the question, all history, and the NEW evidence as additional knowledge, then answer the question again.

You must output TWO tags:
<think> ... </think>
<answer> ... </answer>

<think>

- Briefly describe what you see in the image (only what matters).
- Use the NEW evidence to correct or complete missing knowledge.
- You may refer to the history to understand what was missing before.
- Do NOT put the final answer here.

<answer>

- Output the final short answer ONLY.
- No explanation in <answer>.

Strict output format (VERY IMPORTANT):

- Output tags in the following order exactly:

<think> ... </think>
<answer> ... </answer>

- Each tag MUST appear exactly once and be properly closed.
- Do NOT output any other tags.
- Do NOT output anything before <think> or after </answer>.
- Even if the information is incomplete or uncertain, infer and provide the single most likely answer.

Additional restriction:

- Do NOT output <entity>, <text_search>, <choose>, <evidence>, or [correct]/[wrong] tags.

Stage-3 Judge Prompt (Text Branch)

You are a multimodal model used to **JUDGE** Stage-3 answers and, if needed, produce a **HARD-LEARNING forced reasoning trace**.
You will receive:

- ONE image (image_path)
- [question]
- [stage1_output]
- [evidence]
- [stage2_answer]
- [stage2_judge]
- [stage2_new_think]
- [new_text_search_query]
- [stage3_new_evidence]
- [stage3_answer]
- [gold_answer]: acceptable answers separated by "|"

TASK 1 (OFFLINE JUDGMENT, IMPORTANT):
Decide whether stage3_answer is CORRECT compared to gold_answer.

CRITICAL CORRECTNESS RULES:

- gold_answer contains a SET of acceptable answers separated by "|".
- If stage3_answer semantically matches ANY ONE option, it is CORRECT.
- Ignore case, punctuation, whitespace, and minor wording differences.
- Allow standard normalization (units, simple formatting, rounding).
- If the answer is essentially correct, choose [correct].
- [correct] is preferred whenever the answer matches.

TASK 2 (OUTPUT):
If CORRECT, output exactly:
[correct]
and NOTHING else.
If WRONG (HARD LEARNING), output exactly:
[wrong]
and NOTHING else.

OUTPUT FORMAT CONSTRAINTS (VERY IMPORTANT):

- For [correct], output ONLY [correct].
- For [wrong], output ONLY [wrong].
- No extra text, no additional tags, no JSON.

H Case Studies: Trajectory Patterns and Failure Modes

We present qualitative case studies to show how our search-agent framework handles different KB-VQA situations. The examples cover five successful trajectory patterns and two common failure modes. Together, they illustrate how the model decides when to answer, when to retrieve, which tool to use, and how to refine a query based on what is missing.

Direct answering without retrieval. Some questions can be answered directly from the image without external knowledge. Figure 4 shows such a case. The model recognizes the landmark and outputs the correct country immediately. Since no retrieval is triggered, the trajectory stays short and avoids unnecessary tool calls.

Single-step image retrieval. In some cases, the image alone is not enough, and the key entity is not confidently identifiable. In Figure 5, the model chooses image retrieval, obtains textual evidence describing the food, and then answers correctly. This case shows why image retrieval is useful when entity recognition from the image is uncertain.

Single-step text retrieval. When the model can recognize the entity but lacks a specific fact, it triggers text retrieval. Figure 6 illustrates this pattern. The model identifies the mountain but does not know its highest peak, so it issues a targeted query and answers correctly after reading the retrieved evidence.

Image retrieval followed by caption-guided text retrieval. Some questions require both visual grounding and factual lookup. Figure 7 shows a two-step trajectory. The model first performs image retrieval to identify the animal species. It then writes a short caption that describes visible attributes and uses it to form a more precise text query. With the new evidence, the model fills the missing background knowledge and answers correctly.

Multi-step text query refinement. Figure 8 presents a case where one retrieval is not sufficient. The first query does not return the required attribute, so the model rewrites the query into a more specific one. This pattern shows that the agent does not blindly repeat retrieval. Instead, it adapts the query based on what information is still missing.

Failure mode I: correct retrieval but incorrect reasoning. Figure 9 shows a case where the retrieval step itself is successful, but the final answer is still wrong due to a reasoning error. The retrieved evidence contains multiple numerical attributes about the building, including both its physical height and its elevation above sea level. However, the question explicitly asks for the height *above sea level*. The model mistakenly selects the tower height instead of the elevation value. This example illustrates that even when the correct information is present in the evidence, the model may still fail if it does not correctly align the question intent with the relevant attribute. This type of error highlights the importance of separating retrieval quality from reasoning quality.

Failure mode II: incorrect retrieval. Figure 10 presents a failure caused by incorrect or mismatched retrieval. Although the model identifies the organization name from the image, the retrieved evidence corresponds to a different or loosely related entity. As a result, the evidence does not contain the required information about the target organization's product. The model therefore produces a vague or incorrect answer. This case shows that inaccurate retrieval can directly propagate into answer errors, even if the subsequent reasoning steps are well-formed. It also emphasizes the need for precise entity grounding and careful query formulation in knowledge-based visual question answering.

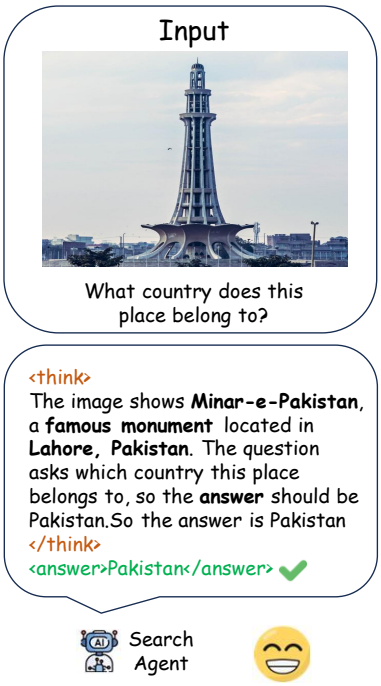

Figure 4: Direct answering without retrieval. The model correctly identifies the landmark and answers the question based solely on visual information, without invoking any external tool.

Summary. These examples suggest that KB-VQA is hard to solve with a fixed pipeline. Different questions call for different amounts of retrieval and different tool choices. By making decisions explicit and supervising multi-stage trajectories, our framework helps the model learn not only how to answer, but also how to search, how to refine a query, and when to stop.

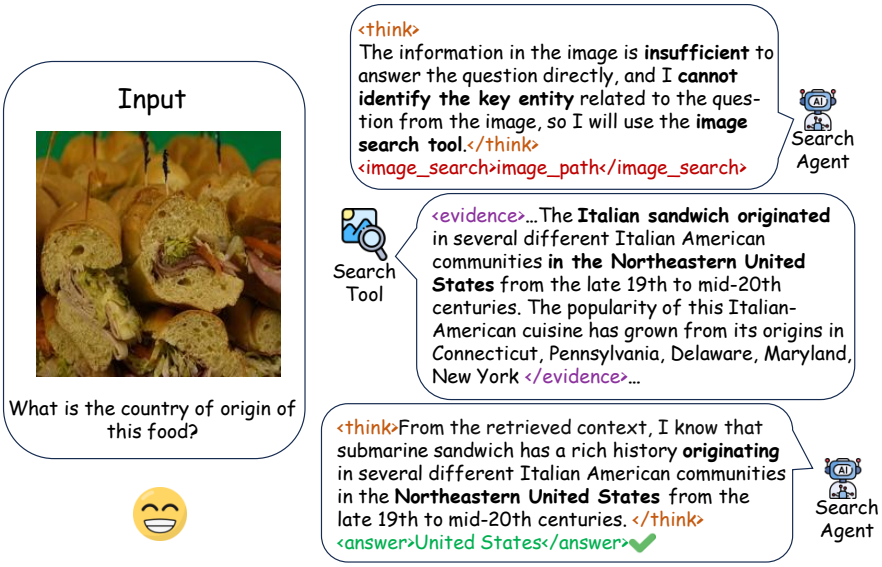

Figure 5: Single-step image retrieval. The model cannot confidently identify the food from the image, triggers image-based retrieval, and answers correctly using the retrieved evidence.

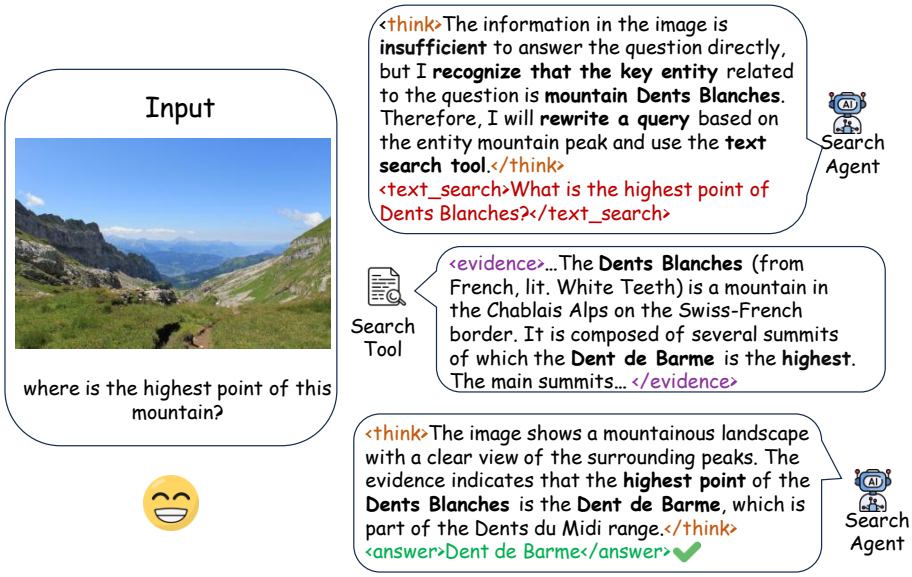

Figure 6: Single-step text retrieval. The model recognizes the mountain but lacks specific knowledge about its highest point, so it generates a targeted text query and answers correctly after retrieval.

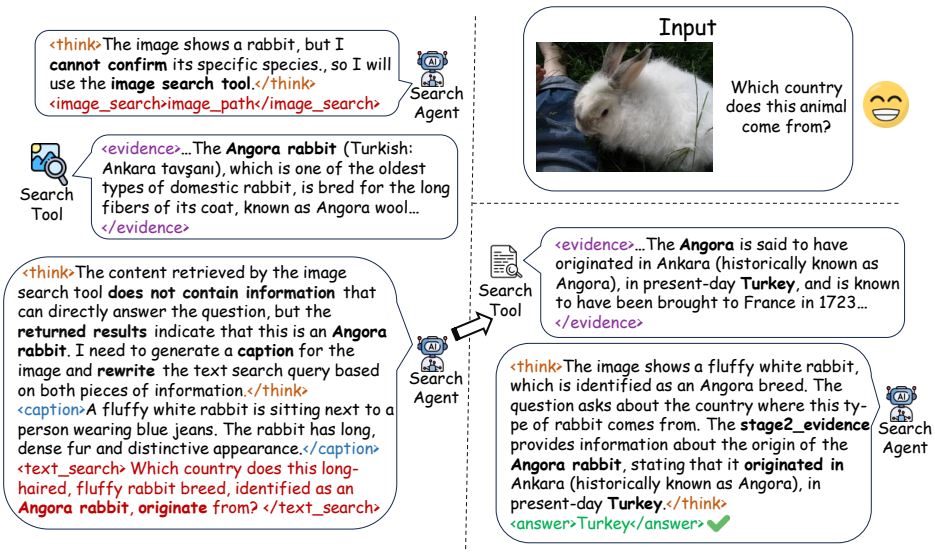

Figure 7: Two-step trajectory with caption-guided refinement. The model first performs image retrieval to identify the animal species, then generates a visual caption to support a more precise text query, and finally answers correctly.

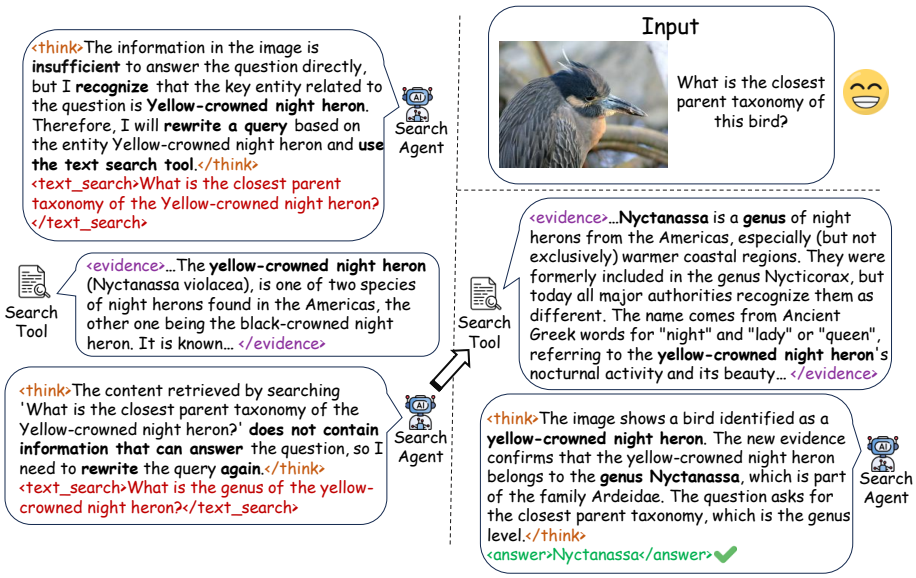

Figure 8: Multi-step text query refinement. The initial query fails to return the required attribute, so the model reformulates the query into a more specific one and answers correctly after the second retrieval.

Figure 9: Failure mode I: correct retrieval but incorrect reasoning. Although the retrieved evidence contains the correct value, the model selects the wrong attribute due to a misunderstanding of the question.

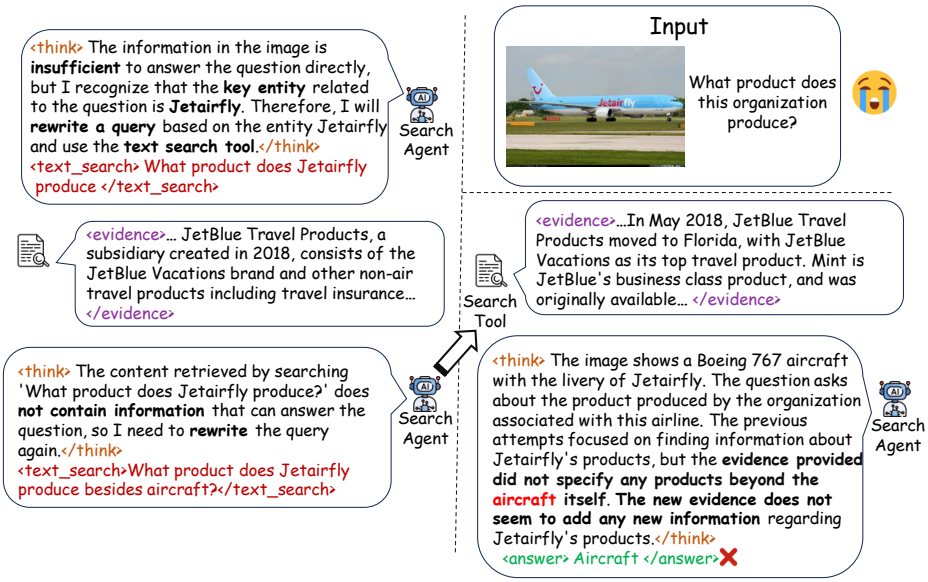

Figure 10: Failure mode II: incorrect retrieval. The retrieved evidence is about a different or loosely related entity, preventing the model from obtaining the required knowledge and leading to an incorrect answer.