SenseSearch: Empowering Vision-Language Models with High-Resolution Agentic Search-Reasoning via Reinforcement Learning
Appendix

Yong Xien Chng[1,2*] Tao Hu[1,3*] Wenwen Tong[1*] Xueheng Li[1,3] Jiandong Chen[1] Haojia Yu[1]
Jiefan Lu[1] Hewei Guo[1] Hanming Deng[1] Chengjun Xie[3] Gao Huang[2✉] Lewei Lu[1✉]
[1]SenseTime Research [2]Tsinghua University [3]University of Science and Technology of China

1. Additional Details on Training Data

In this section, we provide more details on the training data used for both the cold-start supervised fine-tuning (SFT) phase and the reinforcement learning (RL) phase.

1.1. Cold-Start SFT Data

The Cold-start SFT dataset is constructed to equip the model with foundational capabilities in tool usage and reasoning. As visualized in the main text, we apply a rigorous filtering and synthesis pipeline to three primary data sources:

- **FVQA [10]:** Starting from the original training set of 4,849 samples, we identify "hard" samples where the base model frequently failed. After trajectory synthesis and validation, we retain **1,115** high-quality trajectories for SFT.
- **Pixel-Reasoner Corpus [8]:** We leverage the warm-start corpus containing 7.85k samples. Through our filtering process, we select approximately **2,000** samples that best demonstrate pixel-level reasoning capabilities.
- **Curated Expert Data:** To specifically enhance the model's proficiency in complex visual scene with multi-step tool invocations, we manually construct the training subset comprising **200** intricate reasoning trajectories.

This process yields a total of approximately 3,315 high-quality samples for the cold-start SFT phase.

1.2. Reinforcement Learning Data

For the RL phase, we utilize a larger and more diverse dataset to generalize the model's reasoning and tool-use policies. The RL training set comprises:

- **FVQA (Remaining):** The subset of the FVQA training set not used for SFT, consisting of **3,695** samples.
- **DeepEyes-4K [6]:** We utilize **4,000** samples from the DeepEyes-4K training set to reinforce high-resolution visual analysis.
- **Visual-Probe [6]:** We include the complete training set of **5,729** samples to support broad visual reasoning tasks.

2. Additional Details on Evaluation Data

In this section, we provide more details on the evaluation data used for agentic search and visual reasoning task.

2.1. Agentic Search

For agentic search, we mainly rely on our proposed HR-MMSearch, along with MMSearch [4], FVQA-test [10], Infoseek [1], SimpleVQA [2], LiveVQA [3] and MAT-Search [7]:

- **HR-MMSearch.** Existing benchmarks, such as FVQA-test [10] or MMSearch [4], typically rely on standard HD or lower-resolution images to test holistic scene understanding. However, this approach leaves a critical gap in evaluating an agent's ability to understand fine visual details. To fill this gap, we introduce HR-MMSearch, a benchmark designed to assess the fine-grained perception and search-reasoning capabilities of vision-language model (VLM) agents. As shown in Fig. 1, we construct the dataset through a four-phase pipeline: large-scale image crawling, filtering, human annotation, and rigorous quality checking. We begin by crawling candidate images exclusively from three reputable international news outlets—Reuters, the Associated Press (AP), and CNBC. By focusing on timely news photographs, we ensure that the images depict recent events unlikely to appear in existing VLM pre-training data. This design choice reduces the chance of models relying on memorized knowledge and instead encourages genuine use of external tools. After collection, we apply strict filtering to retain only 4K-resolution images from 2025, minimizing pre-training leakage risks while providing rich, fine-grained visual detail. During the human annotation phase, three annotators (all holding bachelor's degrees) independently assign each image to one of eight high-impact domains: Sports, Entertainment & Culture, Sci-

* Equal contribution. ✉ Corresponding authors.

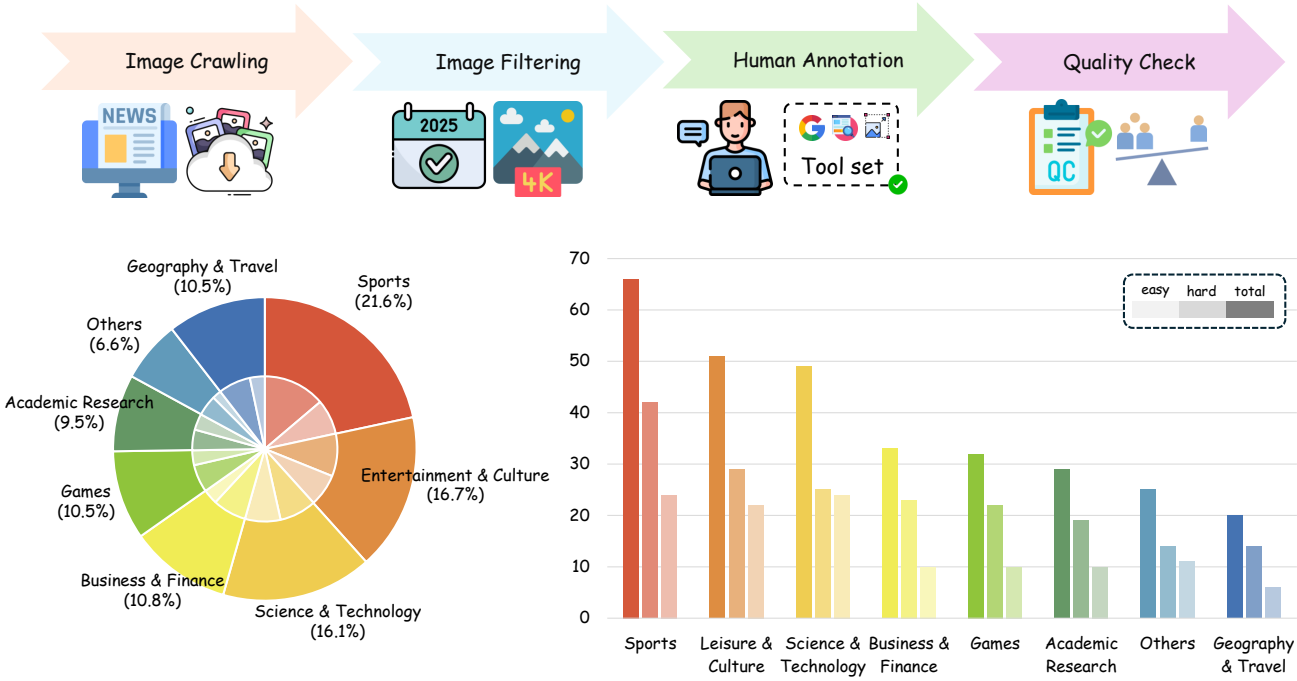

Figure 1. Overview of the proposed HR-MMSearch dataset. This figure details the methodology used to construct the dataset and presents the resulting distribution of categories and difficulties. The colorbar uses a gradation of dark, medium, and light to denote the number of total, hard, and easy samples, respectively.

ence & Technology, Business & Finance, Games, Academic Research, Geography & Travel, and Others. They then manually craft knowledge-intensive questions that target key visual subjects, especially small objects or text regions that occupy less than 5% of the image. Each question is designed so that solving it requires at least one of three multimodal tools: image search, text search, or image crop. Afterwards, three additional experts, each holding at least a master's degree, carefully cross-verify the resulting **305** image–question pairs to confirm the labels, assess quality, and ensure answer correctness. To define difficulty levels, we adopt a pass@8 evaluation protocol in which the agent generates eight independent rollouts for each question using the available tools. A question is considered solved if at least one rollout yields the correct answer. Using this agentic setup, we run Qwen2.5-VL-7B-Instruct as a representative agent to approximate question difficulty. Based on its performance, we categorize the 188 questions for which the model fails all eight rollouts as *Hard*. These failures typically arise on questions that require more complex reasoning or interaction, often involving three or more tool calls; notably, 17 of these questions require coordinated use of all three tools. The remaining 117 questions are classified as *Easy*, as the model produces at least one correct rollout. These easier questions generally require only one or two tool calls. As shown in the bottom-right panel of Fig. 1, the difficulty distribution is roughly consistent across all eight categories, with around 60% *Hard* samples and 40% *Easy* samples in each. Overall, HR-MMSearch offers a challenging and diverse benchmark for evaluating the capabilities of tool-augmented VLM agents in agentic search and fine-grained visual reasoning.

- **MMSearch.** We use MMSearch [4] to test whether models can retrieve up-to-date information or reason about obscure facts. The full dataset contains 300 manually collected examples across 14 subdomains, split into News and Knowledge sections. The News section covers events starting from August 2024 to avoid overlap with training data, while the Knowledge section focuses on rare facts that often challenge advanced models. Similar to MMSearch-R1 [10], we only use the **171** questions that include images and exclude text-only queries to focus on real-world information seeking with visual grounding.
- **FVQA-test.** We use the FVQA-test set [10] to ensure our evaluation spans both visual and textual domains. This benchmark includes **1,800** high-quality examples from three sources. The first 600 come from FVQA-auto-vc and are verified for accuracy and separated from training data. Another 600 are taken from the InfoSeek Hu-

man Split, with manually corrected answers to fix missing public labels. The final 600 were created by human annotators specifically for this benchmark to expand its coverage.

- **InfoSeek.** We evaluate real-world knowledge retrieval using the InfoSeek benchmark [1]. Its creators generated questions by converting Wikidata triples into natural-language questions using human-designed templates. These templates were developed for 300 relations and include placeholders for units and entity types to improve clarity. They removed unanswerable questions and balanced the dataset across entities to ensure quality. From the test split, we sample **2,000** instances to capture a diverse set of factual queries.
- **SimpleVQA.** SimpleVQA [2] focuses on factual accuracy and real-world usefulness. It combines two types of examples: image–question pairs from post-2023 VQA datasets and new pairs produced by experts using internet search results. All examples pass strict filters for difficulty and quality to ensure they test objective information. From the full benchmark, we use the **1,013** English examples to evaluate factual reasoning without language-related noise.
- **LiveVQA.** To measure performance on fast-changing news, we include LiveVQA [3]. This dataset draws content from major international outlets such as CNN and the BBC and spans 14 categories, including science and sports. It contains **3,602** pairs generated with GPT-4o, ranging from simple visual checks to complex questions requiring reasoning over accompanying text. This range allows us to test how well models combine visual and textual information in dynamic news environments.
- **MAT-Search.** We include MAT-Search [7] to evaluate agentic search and multimodal multi-hop reasoning. This benchmark contains **150** high-quality examples, each manually crafted and verified by human annotators. The questions vary in difficulty and require different depths of reasoning, with more complex queries involving additional inference steps and factual knowledge. These examples are designed to test a model's ability to handle composite problems, retrieve relevant external information, and use tools effectively, providing a focused evaluation of agentic multimodal reasoning.

2.2. Visual Reasoning

For visual reasoning, we mainly evaluate our models on V* Bench [11], HR-Bench [9] and MME-RealWorld [13]:

- **V* Bench.** V*-Bench [11] is designed to evaluate the detailed visual grounding and collaborative reasoning capabilities of VLMs, specifically addressing the need for visual search mechanisms. Sourced from the high-resolution SA-1B dataset, it features images with an average resolution of 2246×1582 that are visually crowded or contain small details. For this study, we utilize all **191** images from the benchmark. The dataset includes human-annotated tasks focused on attribute recognition and spatial relationship reasoning, which are deliberately crafted to be unsolvable without precise visual processing and the ability to focus on specific, often obscure, visual elements.
- **HR-Bench.** HR-Bench [9] serves as a specialized benchmark for assessing model performance on ultra-high-resolution inputs, challenging VLMs to overcome the information loss typically associated with image resizing. It evaluates perception capabilities across fine-grained single-instance and cross-instance tasks. To rigorously test the model's scalability and detail preservation, we employ two distinct splits from this dataset: the 4K resolution split and the 8K resolution split, containing **800** images each. This setup allows for a focused evaluation of the model's stability and accuracy when processing inputs with extreme pixel counts.
- **MME-RealWorld.** MME-RealWorld [13] is a large-scale, manually annotated benchmark targeting real-world scenarios that are perceptually challenging even for humans. It covers 43 subtasks across five key domains: OCR in the wild, remote sensing, diagrams and tables, video monitoring, and autonomous driving. The images feature high resolutions (average 2000×1500) and complex, clutter-heavy scenes requiring zooming and multi-step reasoning. In our experiments, we utilize **23,599** QA samples to evaluate the model's robustness in handling diverse, high-difficulty visual perception tasks in practical applications.

3. Additional Implementation Details

In this section, we provide more details on the reward model and search pipeline, the differences between the direct answer, RAG, and agentic workflows, and the evaluation metrics used throughout our training and evaluation processes.

3.1. Reward model

During training for RL, we utilize *Qwen2.5-VL-72B-Instruct* as the LLM-as-a-Judge for all experiments. The judge's response is generated using a greedy setting with a temperature of 0.0. The full prompt is given in Fig. 8.

3.2. Comparison between Direct Answer, RAG and Agentic Workflows

As described in Section 4.1 of the main paper, we evaluate our models against other baselines using three separate workflows, specifically Direct Answer, RAG, and Agentic Workflows. These workflows are distinguished primarily by the prompts used during inference, which control the tools available to the models. These different prompts are given in Fig. 9, Fig. 10 and Fig. 11.

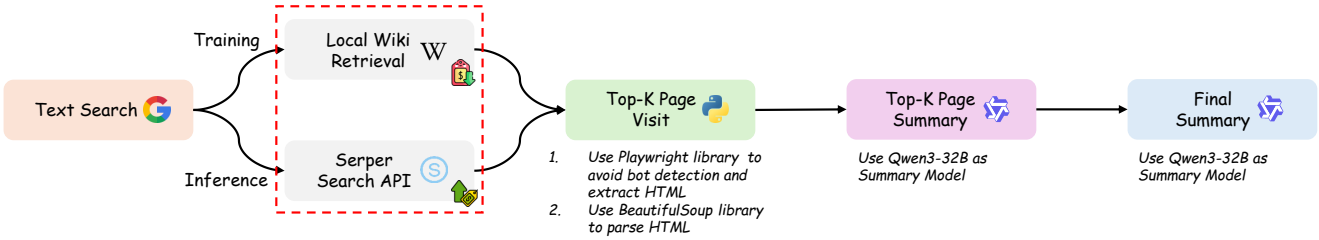

Figure 2. Overview of our Text Search Pipeline. The pipeline utilizes separate retrieval modes for training and inference. Local retrieval from a Wikipedia knowledge base is used during RL training to avoid the prohibitive cost of live web searches, while live web search (via Serper API) is used exclusively during inference. Crucially, the retrieved passages from both separate modes are uniformly processed by a Qwen3-32B summarizer before being passed to the main model.

3.3. Text search pipeline

The structure of our text search pipeline is illustrated in Fig. 2. Most aspects of the text-search setup are shared between training and inference. The main exception is that the text search method uses local retrieval during training and a live web search during inference. During RL training, we avoid the prohibitive cost of live web searches by using a locally hosted Wikipedia knowledge base built from the 20250901 dump file (*enwiki-20250901-pages-articles.xml.bz2*). Retrieval is performed with the E5-retriever [5], and the total number of returned passages is fixed at 5.

Unlike prior work, such as MMSearch-R1 [10], we do not use the Jina API to extract clean, LLM-friendly text. Instead, to mitigate bot detection, our system uses Playwright to fetch the HTML content. To keep the pipeline simple and efficient, we skip JavaScript rendering and parse the static HTML directly using Python's BeautifulSoup library. In contrast, during the inference phase, the text-search tool sends its queries through the Serper Search API. Crucially, in both the training and inference phases, the top 5 retrieved passages are first summarized individually by Qwen3-32B [12]. Following the individual summaries, a final, holistic summary of all 5 passages is then generated. This shared two-stage summarization process ensures that the model learns the core tool-use behaviors on data formatted identically to what it will encounter in a live setting. The complete prompts used for both page and final summarization are identical and are provided in Fig. 12.

This training design has clear benefits. It is fast and lightweight. The trade-off is that the system cannot read webpages that depend on JavaScript. While supporting these pages could improve performance, this limitation is acceptable because the model still learns the core tool-use behaviors. We observe that these behaviors transfer reliably to inference. Notably, this transfer succeeds even though the model never encounters real Serper Search outputs during its training.

3.4. Evaluation Metrics

We run all evaluations with a sampling temperature of 0.0 and choose different metrics based on the type of output in each domain. For agentic search tasks, we use an LLM-as-a-Judge setup because the answers are open-ended and require flexible semantic evaluation. In this setup, GPT-4o is used to score the Pass@1 accuracy by comparing the final response with the ground truth. For visual reasoning benchmarks, such as V* Bench [11], HR-Bench [9], and MME-RealWorld [13], we use Exact Match [6] because these datasets mainly contain closed-ended multiple-choice questions that require an exact string match with the correct option. To reduce variance caused by the sampling temperature in these strict visual tasks, we report Avg@8 accuracy for V Bench and HR-Bench by averaging the Exact Match score across eight independent attempts for each question. For the large-scale MME-RealWorld benchmark, which has less variance, we report Pass@1 accuracy.

4. Case Study

We present more SenseSearch inference cases in Fig. 3, Fig. 4 and Fig. 5.

5. Limitations

Despite the strong performance of SenseSearch, our error analysis reveals some limitations:

- **Vulnerability to Retrieval Noise.** As seen in Fig. 6, SenseSearch occasionally fails to distinguish between semantically similar but distinct attributes (e.g., conflating "based in" with "born in") within retrieved snippets. This suggests that the current reasoning module lacks sufficient robustness against distractor information in open-world search results, leading to hallucinated reasoning paths.
- **Ineffective Tool Usage.** In some scenarios requiring fine-grained visual extraction as shown in Fig. 7, SenseSearch-may fail to ground specific visual entities (e.g., "CHED

Regional Office 1") into the search query. Instead, it resorts to generic terms (e.g., "in this region"), resulting in the retrieval of irrelevant global statistics. This indicates a gap in cross-modal alignment during the tool parameter generation phase.

References

[1] Yang Chen, Hexiang Hu, Yi Luan, Haitian Sun, Soravit Changpinyo, Alan Ritter, and Ming-Wei Chang. Can pre-trained vision and language models answer visual information-seeking questions? In *EMNLP*, 2023. 1, 3

[2] Xianfu Cheng, Wei Zhang, Shiwei Zhang, Jian Yang, Xiangyuan Guan, Xianjie Wu, Xiang Li, Ge Zhang, Jiaheng Liu, Yuying Mai, et al. Simplevqa: Multimodal factuality evaluation for multimodal large language models. In *ICCV*, 2025. 1, 3

[3] Mingyang Fu, Yuyang Peng, Benlin Liu, Yao Wan, and Dongping Chen. Livevqa: Live visual knowledge seeking. *arXiv:2504.05288*, 2025. 1, 3

[4] Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Chaoyou Fu, Guanglu Song, Peng Gao, Yu Liu, Chunyuan Li, and Hongsheng Li. Mmsearch: Benchmarking the potential of large models as multi-modal search engines. In *ICLR*, 2025. 1, 2

[5] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. In *COLM*, 2025. 4

[6] Xin Lai, Junyi Li, Wei Li, Tao Liu, Tianjian Li, and Hengshuang Zhao. Mini-o3: Scaling up reasoning patterns and interaction turns for visual search. In *ICLR*, 2026. 1, 4

[7] Ziyu Liu, Yuhang Zang, Yushan Zou, Zijian Liang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual agentic reinforcement fine-tuning. In *ICCV*, 2025. 1, 3

[8] Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. In *NeurIPS*, 2025. 1

[9] Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *AAAI*, 2025. 3, 4

[10] Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-r1: Incentivizing lmms to search. *arXiv:2506.20670*, 2025. 1, 2, 4

[11] Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *CVPR*, 2024. 3, 4

[12] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, et al. Qwen3 technical report. *arXiv:2505.09388*, 2025. 4

[13] Yi-Fan Zhang, Huanyu Zhang, Haochen Tian, Chaoyou Fu, Shuangqing Zhang, Junfei Wu, Feng Li, Kun Wang, Qingsong Wen, Zhang Zhang, et al. Mme-realworld: Could your multimodal llm challenge high-resolution real-world scenarios that are difficult for humans? In *ICLR*, 2025. 3, 4

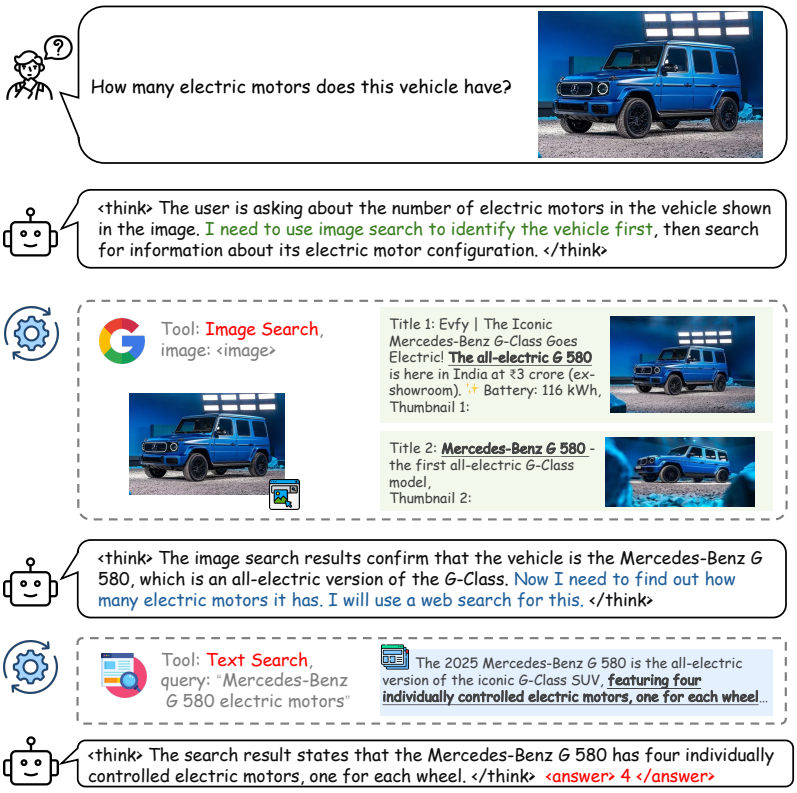

Figure 3. Case study 1 of SenseSearch.

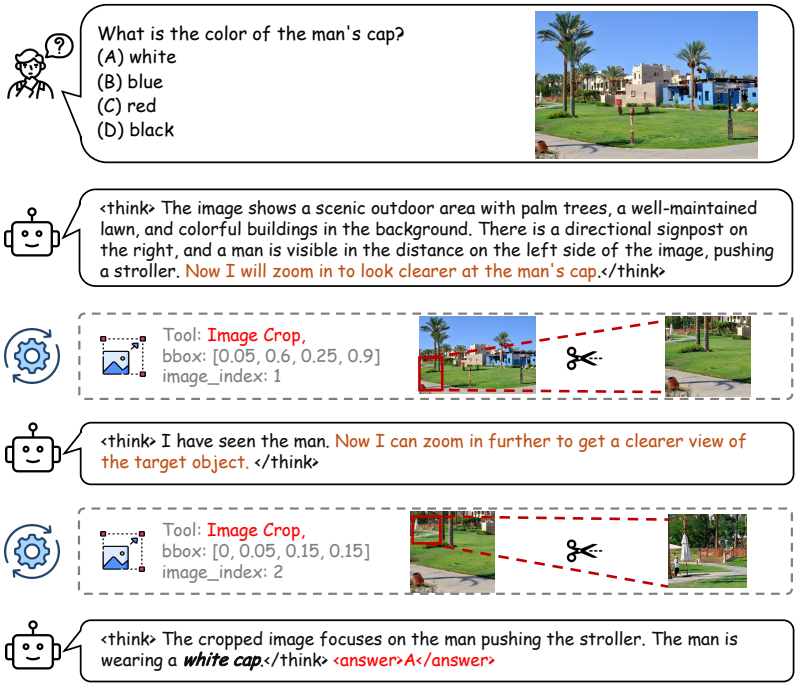

Figure 4. Case study 2 of SenseSearch.

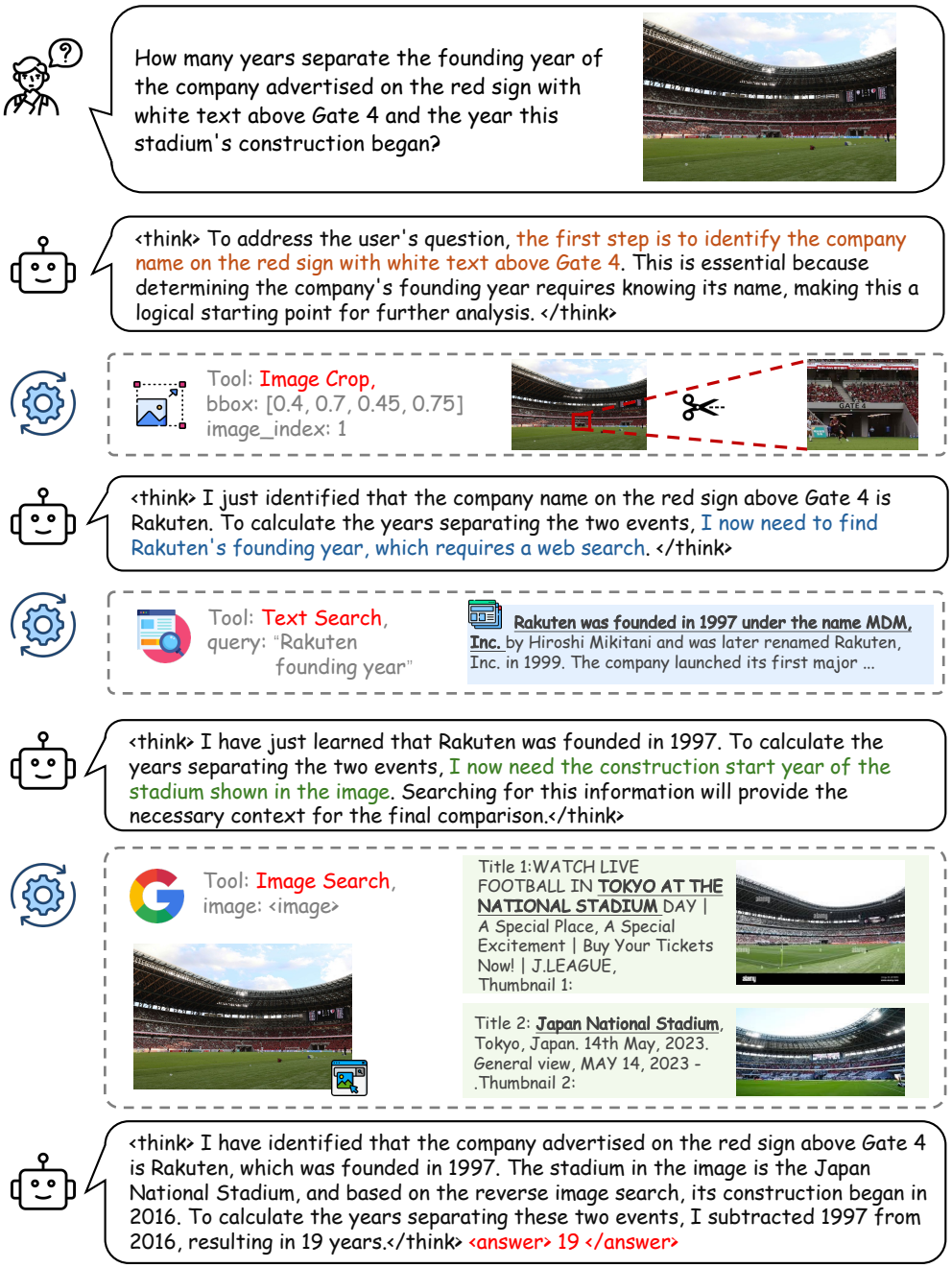

Figure 5. Case study 3 of SenseSearch.

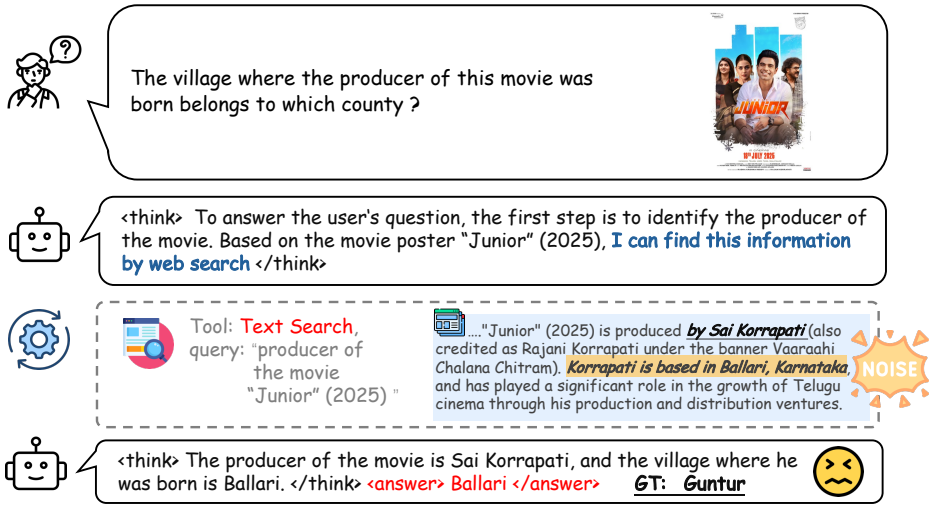

Figure 6. Case study 4 of SenseSearch.

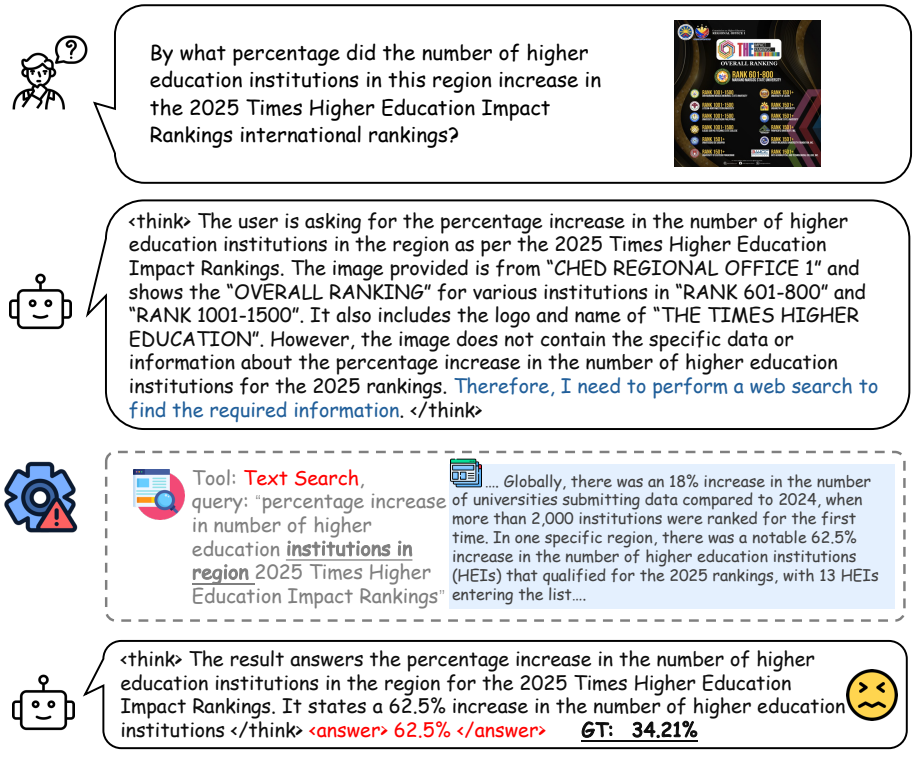

Figure 7. Case study 5 of SenseSearch.

System Message

You are an AI assistant tasked with evaluating the correctness of model responses based on an image, question, and ground truth answer. Your judgment should follow these principles:

1. Consider the image, question, and ground truth answer holistically before evaluating the model's response.
2. Your decision should be strictly Yes or No, based on whether the model's response is factually accurate and aligns with the ground truth answer.
3. If the model response is a more specific form of the ground truth answer, it is correct.
4. If the model response includes all key information but adds minor details, it is correct as long as the extra details are factually correct.
5. If the model response contradicts, modifies, or omits critical parts of the answer, it is incorrect.
6. For numerical values, ensure correctness even when presented in different units.
7. For names, check for first and last name correctness. If the middle name is extra but correct, consider it correct.
8. For yes/no questions, the response must exactly match "Yes" or "No" to be correct.
9. If the judgment can be made based solely on the text, you may choose to ignore the input image, as some images may be unfamiliar to you and could affect your judgment. Refer to the image only when necessary to minimize misjudgment.
10. If there are multiple candidate answers, you can also evaluate the model's response against all of them. If the response aligns with at least one candidate according to the rules above, it should be considered correct.
11. For multiple choice questions (A, B, C, D), be more lenient. If the model provides the correct letter choice, even with additional text or formatting, consider it correct.
12. If the model's answer contains the correct choice letter (A, B, C, or D) anywhere in the response, and it's clear this is the intended answer, mark it as correct.
13. Ignore formatting issues like extra parentheses, brackets, or minor text variations as long as the core answer is correct.

Your output must be in the following format:
<judge>Yes/No</judge>
<reason>Explanation of why the answer is correct or incorrect.</reason>

Prompt

 # Prompt:

Image, Question, and Model Response Evaluation
Question: {`question`}
Ground Truth Answer: {`ground_truth_answer`}
Model Response: {`model_response`}

Evaluation Instructions
Evaluate whether the Model Response is correct based on the Image, Question and Ground Truth Answer. Follow the predefined judgment rules and provide a clear Yes/No answer along with a justification.

Output Format
<judge>Yes/No</judge>
<reason>Detailed reasoning following the evaluation principles.</reason>

Figure 8. Full prompt used for *Qwen2.5-VL-72B-Instruct* as the LLM-as-a-Judge.

System Message

#Role
You are a step-by-step reasoning assistant.
Given a question, your task is to solve the problem **one substep at a time**.

Guiding Principles
At each turn, you must **either**:
1. Issue **one specific tool** enclosed in <tool_call> </tool_call> tags,
2. Or provide the **final answer** enclosed in <answer> </answer> tags.

All outputs **must begin with a thought** enclosed in <think> </think> tags, explaining your current reasoning and what to do next.

Output Format (strict):
Always start with <think>. Do not output the previous reasoning chain. Then, depending on the case, output one of the following:

1. If reasoning continues:
<think> Your current reasoning and next plan </think>
<tool_call> One precise, tool call to assist your reasoning </tool_call>

2. If ready to conclude:
<think> Summarize all reasoning and derive the answer </think>
<answer> Final answer </answer>

Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>

{"type": "function", "function": {"name": "web_search", "description": "Search the web for information you don't have or to verify facts.", "parameters": {"type": "object", "properties": {"query": {"type": "string", "description": "Query to search the web"}}, "required": ["query"]}}}
{"type": "function", "function": {"name": "crop_image", "description": "Crop the image based on the bounding box coordinates to zoom in on specific regions for detailed analysis.", "parameters": { "type": "object", "properties": { "bbox": { "type": "array", "items": { "type": "number" }, "minItems": 4, "maxItems": 4, "description": "Normalized bounding box [x1, y1, x2, y2], where 0.0 <= x1 < x2 <= 1.0 and 0.0 <= y1 < y2 <= 1.0. (x1,y1) is top-left corner, (x2,y2) is bottom-right corner." }, "image_index": { "type": "integer", "minimum": 1, "description": "Index of the image to crop: 1 for original input image, 2 for first cropped image, 3 for second cropped image, etc." }}, "required": ["bbox", "image_index"]}}}
{"type": "function", "function": {"name": "image_search", "description": "Reverse search the current image to get more information. This function does not accept any text queries or arguments.", "parameters": {"type": "object", "properties": {}}}}

</tools>

For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>

Prompt

 # Prompt:

`{image}`
`{question}`

Figure 9. Full prompt used during training and inference for the Agentic Workflow.

System Message

You are a helpful assistant.

Prompt

 # Prompt:

`{image}`
`{question}`

Figure 10. Full prompt used for the Direct Answer workflow.

System Message

#Role
You are a step-by-step reasoning assistant.
Given a question, your task is to solve the problem **one substep at a time**.

Guiding Principles
At each turn, you must **either**:
1. Issue **one specific tool** enclosed in <tool_call> </tool_call> tags,
2. Or provide the **final answer** enclosed in <answer> </answer> tags.

All outputs **must begin with a thought** enclosed in <think> </think> tags, explaining your current reasoning and what to do next.

Output Format (strict):
Always start with <think>. Do not output the previous reasoning chain. Then, depending on the case, output one of the following:

1. If reasoning continues:
<think> Your current reasoning and next plan </think>
<tool_call> One precise, tool call to assist your reasoning </tool_call>

2. If ready to conclude:
<think> Summarize all reasoning and derive the answer </think>
<answer> Final answer </answer>

Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>

{"type": "function", "function": {"name": "web_search", "description": "Search the web for information you don't have or to verify facts.", "parameters": {"type": "object", "properties": {"query": {"type": "string", "description": "Query to search the web"}}, "required": ["query"]}}}
{"type": "function", "function": {"name": "crop_image", "description": "Crop the image based on the bounding box coordinates to zoom in on specific regions for detailed analysis.", "parameters": { "type": "object", "properties": { "bbox": { "type": "array", "items": { "type": "number" }, "minItems": 4, "maxItems": 4, "description": "Normalized bounding box [x1, y1, x2, y2], where 0.0 <= x1 < x2 <= 1.0 and 0.0 <= y1 < y2 <= 1.0. (x1,y1) is top-left corner, (x2,y2) is bottom-right corner." }, "image_index": { "type": "integer", "minimum": 1, "description": "Index of the image to crop: 1 for original input image, 2 for first cropped image, 3 for second cropped image, etc." }}, "required": ["bbox", "image_index"]}}}

</tools>

For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>

Prompt

 # Prompt:

`{image}`
`{question}`

To help you answer the question, here are reverse image search results for the given image.

Reverse image search results:

`{image_search_results}`

Figure 11. Full prompt used for the RAG Workflow.

System Message

You are a helpful assistant. Your task is to summarize the main content of the given web page in no more than five sentences. Your summary should cover the overall key points of the page, not just parts related to the user's question.

If any part of the content is helpful for answering the user's question, be sure to include it clearly in the summary. Do not ignore relevant information, but also make sure the general structure and main ideas of the page are preserved. Your summary should be concise, factual, and informative.

Prompt

 # Prompt:

Webpage Content (first 30000 characters) is: `{content}`
Question: `{question}`

Figure 12. Full prompt used by Qwen3-32B to perform page summary and final summary.