BeaconChart: Verifier-Guided Corrective Instruction for Chart Understanding

Hoyeon Kim [*1 2] **Seungjin Han** [*3 2] **David R. Mortensen** [2]

Abstract

Chart question answering remains challenging because small errors in evidence extraction and visual grounding can easily propagate into incorrect answers. Although recent methods improve chart understanding, most still rely on one-pass answer generation without explicitly verifying whether the extracted evidence is accurate, and verifier feedback rarely persists beyond the current example. We propose **BeaconChart**, a verifier-guided multi-agent framework that improves chart understanding by accumulating verifier feedback as corrective instruction. Rather than directly trusting an initial prediction, BeaconChart first plans the required evidence, performs chart-aware extraction, and verifies whether the predicted answer is correctly grounded in the chart. When verification fails, it converts the feedback into reusable guidance that improves future inference on similar chart failures. Experiments on ChartQA and ChartX show that BeaconChart consistently outperforms both general-purpose multimodal large language models and recent chart-specialized baselines. [‡]

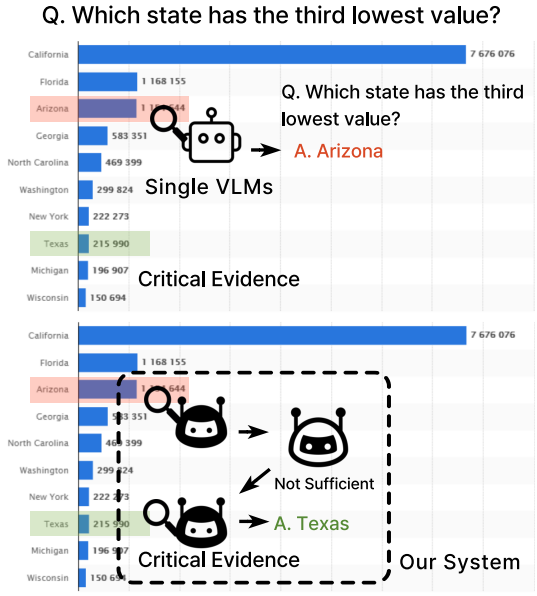

Figure 1. While one-pass chart QA with a single VLM is vulnerable to early perception errors that can lead to misleading evidence and an incorrect answer, our feedback-driven multi-agent framework recovers critical evidence through iterative refinement and produces the correct answer.

1. Introduction

Charts are a fundamental tool for communicating quantitative information, enabling people to rapidly perceive trends, comparisons, distributions, and relationships in data (Franconeri et al., 2021; Ware, 2019; Borkin et al., 2013; Keim, 2002). Their widespread use across science, industry, education, and public communication has made accurate chart interpretation an important problem for multimodal intelligence (Wu et al., 2021; Huang et al., 2024). In particular, chart question answering requires a model to move beyond coarse visual recognition and instead identify relevant evidence, recover precise values, and perform grounded reasoning over chart-specific structure (Masry et al., 2022). This challenge was initially addressed through task-specific Chart Question Answering formulations and specialized models (Kahou et al., 2017; Kafle et al., 2018; Methani et al., 2020; Masry et al., 2022), but is now increasingly revisited in light of the rapid progress of multi-modal large language models (Liu et al., 2023; Masry et al., 2024; Goswami et al., 2025; Huang et al., 2025).

Despite this progress, chart question answering remains fundamentally challenging because it requires substantially more than generic visual recognition. To answer a chart question correctly, a model must identify the relevant chart structure, interpret visual elements such as axes, legends, and marks, extract precise values from the appropriate re-

[*]Equal contribution [‡]This work was done while Hoyeon Kim and Seungjin Han were visiting Carnegie Mellon University. [1]Department of Economics, Sogang University, Seoul, Republic of Korea [2]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA [3]Department of Applied Artificial Intelligence, Sungkyunkwan University, Republic of Korea. Correspondence to: David R. Mortensen <dmortens@cs.cmu.edu>.

ICML 2026 Workshop on Scalable Learning and Optimization for Efficient Multimodal AI Agents, Seoul, Korea. Copyright 2026 by the author(s).

gions, and then perform reasoning that remains faithful to the visual evidence throughout the entire inference process (Masry et al., 2022; Liu et al., 2023; Moured et al., 2024). These steps are tightly coupled, so even a small upstream error, such as attending to the wrong region, misreading a value, or misinterpreting a structural cue, can easily propagate through the inference process and ultimately corrupt the final answer, as illustrated in Figure 1. However, most existing approaches still treat chart question answering primarily as a direct answer-generation problem, with limited mechanisms for explicitly checking whether extracted evidence is accurate and for carrying forward verifier feedback beyond the current example (Masry et al., 2022; Xia et al., 2025).

Another limitation of existing approaches is that verifier feedback is rarely preserved and reused for recurring chart-specific failures (Tang et al., 2025; Vogel et al., 2025). In chart question answering, errors are often systematic rather than isolated: models may repeatedly confuse stacked segments in bar charts, overlook small slices in pie charts, or misalign values in multi-axis charts. Nevertheless, most methods still process each example independently (Wu et al., 2024; Bo et al., 2024), even when similar correction patterns have already appeared in prior cases. Consequently, they fail to accumulate reusable guidance from past mistakes or to transfer such guidance to structurally similar inputs (Jiang et al., 2025). This instance-bound behavior makes refinement inherently myopic, limiting the model to correcting only the current prediction rather than improving its future inference behavior. These observations motivate a mechanism that converts verifier feedback into chart-specific corrective instruction, enabling reusable guidance across similar failures.

To address these limitations, we propose **BeaconChart**, a verifier-guided multi-agent framework for chart question answering that converts verifier feedback into reusable corrective instruction. BeaconChart is motivated by the observation that chart QA failures are often recurring rather than isolated, making verifier feedback valuable not only for correcting the current prediction but also for guiding future inference on structurally similar cases. Concretely, BeaconChart first plans what evidence is required by the question, then uses a chart-specialized perception agent to extract the relevant evidence and produce an answer candidate, and finally applies an independent verifier to check whether the extracted evidence and predicted answer are correctly grounded in the chart. When verification fails, BeaconChart does not discard the feedback after the current example; instead, it abstracts the feedback into chart-specific corrective instructions, stores them as memory, and reuses them to guide subsequent inference. In this way, BeaconChart improves chart understanding not only through instance-level refinement, but also by providing chart-aware corrective guidance that becomes increasingly specialized to recurring failure patterns over time.

Our main contributions are as follows:

- **BeaconChart** reformulates chart QA as a coordinated multi-agent process that identifies, extracts, and validates critical evidence, instead of directly producing a single-pass, non-correctable answer.
- An independent verifier detects unsupported or incorrect predictions by checking whether the extracted evidence and answer are correctly grounded in the chart, enabling inference-time refinement rather than blindly trusting a single-pass output.
- **BeaconChart** converts verifier feedback into reusable corrective instruction, allowing guidance triggered by failed evidence extraction to be reused as specialized cues for future similar chart cases rather than remaining a one-off correction.

2. Related Work

Chart Datasets Chart Question Answering studies answering natural-language questions grounded in charts and data visualizations (Masry et al., 2022; Methani et al., 2020). Early benchmarks such as FigureQA (Kahou et al., 2017) and DVQA (Kafle et al., 2018) focused on relatively simplified visual reasoning or restricted chart settings, while later datasets including PlotQA (Methani et al., 2020) and ChartQA (Masry et al., 2022) introduced more realistic charts, larger scale, and more numerically demanding questions. More recently, ChartX (Xia et al., 2025) further broadened the evaluation space by covering diverse chart types, topics, and reasoning tasks, making it a more challenging benchmark for assessing robustness under varied chart structures and question demands.

Chart Modeling Approaches To tackle these increasingly realistic and demanding benchmarks, chart-specialized models have focused on strengthening chart-aware representation learning and reasoning. Early chart-focused models, such as UniChart (Masry et al., 2023) and MatCha (Liu et al., 2023), improved chart understanding through stronger structural modeling, chart-specific pretraining, element extraction, and numerical reasoning. Building on this line, later multimodal approaches, including ChartInstruct (Masry et al., 2024), TinyChart (Zhang et al., 2024), ChartGemma (Masry et al., 2025), ChartVLM (Xia et al., 2025), and VProChart (Huang et al., 2025), further extended chart understanding through instruction tuning, efficient chart-oriented perception, visual perception alignment, and stronger downstream reasoning performance. Despite these advances, chart question answering remains fragile under complex chart structures and diverse reasoning demands (Xu

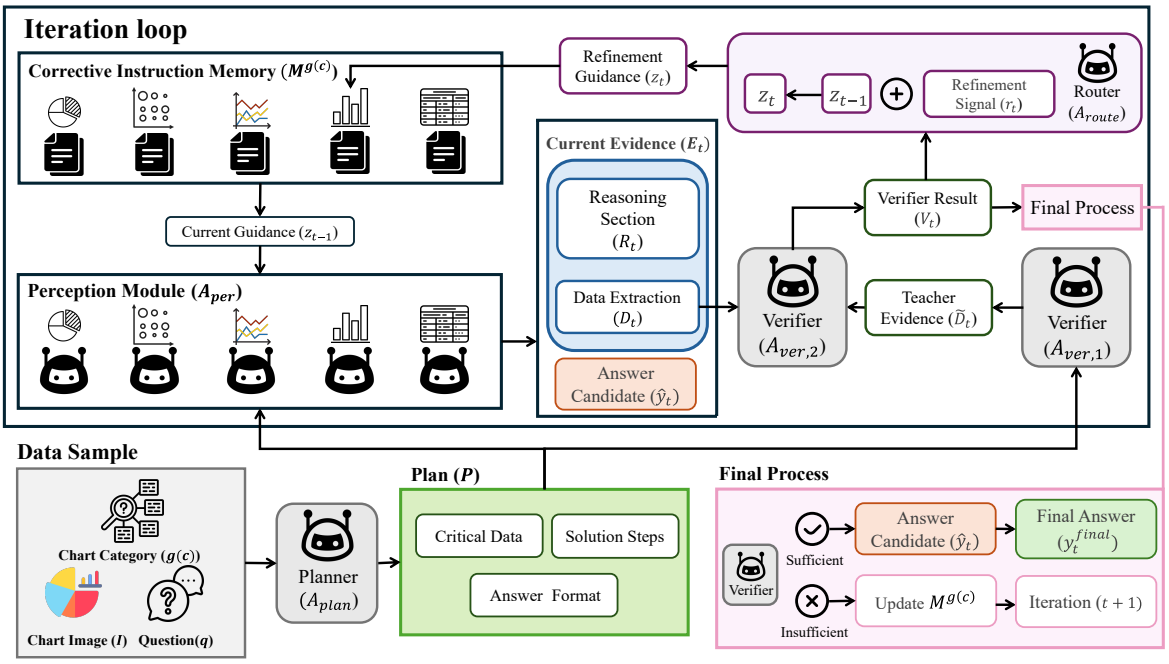

Figure 2. Overview of **BeaconChart**. The planner produces a chart-aware plan, the chart-type-specific perception module extracts evidence and predicts an answer under reusable corrective guidance, and the verifier independently checks whether the extracted evidence is sufficient. The equality symbol indicates a sufficient evidence match, while an insufficient decision triggers iterative refinement. Check marks denote the perception agents activated for the current chart type.

et al., 2023; Tang et al., 2023; Xia et al., 2025). Existing methods have improved chart-specific perception and reasoning, but they still largely treat each example as an isolated prediction problem. As a result, failures in visual grounding, value extraction, or answer justification are typically handled only within the current example, limiting their ability to improve future predictions on structurally similar cases. This limitation motivates refinement frameworks that can not only identify unreliable evidence or reasoning during inference, but also use the resulting feedback to better guide future predictions on similar cases.

Multi-Agent Refinement Recent refinement frameworks increasingly decompose inference into coordinated roles such as planning, checking, reflection, and memory management. Methods such as Self-MAP (Deng et al., 2024), MAPLE (Bai et al., 2025b), FRAME (Yu et al., 2025), and SAGE (Liang et al., 2025) show that role-specialized agents can improve robustness by enabling intermediate verification, iterative correction, and memory-aware refinement across reasoning steps. Complementing this line, feedback-driven refinement methods further show that generated feedback can serve as an effective signal for improving outputs during inference. Learning to Refine with Fine-Grained Natural Language Feedback (Wadhwa et al., 2024) and Verification and Refinement of Natural Language Explanations (Quan et al., 2024) demonstrate that verification and natural-language feedback can be used to revise intermediate reasoning or final predictions, while reflective methods such as Reflexion (Shinn et al., 2023) further suggest the value of carrying forward lessons from prior failures. However, these approaches are mostly designed for general reasoning or agentic settings, and they do not explicitly focus on improving perception reliability in chart understanding, where errors in visual grounding and value extraction directly determine the final answer. Our method builds on this line by using an independent verifier to diagnose perception failures and convert the resulting feedback into corrective instruction that improves chart understanding.

3. BeaconChart

3.1. Problem Formulation

As illustrated in Figure 2, BeaconChart follows a verifier-guided iterative pipeline for chart question answering. The framework first plans what evidence is required to answer a question, then performs chart-specialized perception to extract the relevant evidence and produce an answer candidate, and finally applies an independent verifier to check whether the target evidence required for the question is sufficiently supported and grounded in the chart. When verification fails, the resulting failure signal is converted into corrective instruction for subsequent refinement.

Given a chart image I and a natural-language question q, the goal is to predict the answer y. We consider a chart-

aware inference setting in which each input is associated with a chart-type signal c, which is mapped to a broader chart category $g(c)$ for chart-specific inference. BeaconChart formulates chart question answering as an iterative evidence-refinement process over at most T iterations. The framework consists of three VLM-based modules: a planner A_{plan}, a chart-specialized perception agent $A_{\text{per}}^{g(c)}$, and an independent verifier A_{ver}, together with a chart-category-specific corrective instruction memory $M^{g(c)}$.

For an input (I, q), the planner first produces a plan P that specifies what evidence to extract and how to solve the question. At iteration t, the perception agent takes P, $g(c)$, and the current refinement guidance z_{t-1} as input, and produces textual evidence E_t together with an answer candidate $\hat{y}_t$. The verifier then checks whether the target-critical evidence required by P is sufficiently supported by the current perception output and the chart image. At the first iteration, z_0 is initialized from the corrective instruction memory of the current chart category. If verification fails, the resulting failure is used to derive refinement guidance for the next iteration and, when appropriate, to update the memory for future problems of the same chart category. The process continues until an answer is verified or the maximum number of iterations is reached. The final prediction is the first verified answer candidate, if one exists; otherwise, the answer candidate from the final iteration is returned.

3.2. Stage 1: Chart-Aware Planning

The first stage determines what evidence is required and how the question should be solved before answer generation. The planner agent A_{plan} produces a compact plan

$$P = A_{\text{plan}}(I, q, g(c)), \tag{1}$$

which specifies three elements: (i) the reasoning steps required to answer the question, (ii) the critical data fields that must be extracted, and (iii) the expected answer format. For example, for a question such as *"How much higher is the population in 2020 than in 2010?"*, the plan may specify the following steps: *locate the population chart and identify the yearly categories*, *extract the values for 2010 and 2020*, and *compute the difference between the two values*. It may also identify the critical data fields as *2010 population* and *2020 population*, with the expected answer format being a numeric value representing the difference.

3.3. Stage 2: Chart Understanding

Conditioned on the chart category $g(c)$, the plan P, and optional refinement guidance, the perception agent $A_{\text{per}}^{g(c)}$ performs chart understanding to extract textual evidence and generate an answer candidate:

$$(E_t, \hat{y}_t) = A_{\text{per}}^{g(c)}(I, q, P, z_{t-1}), \quad t = 1, \ldots, T. \tag{2}$$

The agent $A_{\text{per}}^{g(c)}$ is specialized to the structural characteristics of the chart family, such as how values are arranged, how marks correspond to legends or axes, and which visual cues are most relevant for the current chart type. At the first iteration, the agent operates without problem-specific refinement feedback. When the verifier rejects the current prediction, the resulting refinement guidance is passed to the next iteration and appended to the perception prompt to encourage more reliable chart understanding and evidence extraction.

We represent the extracted evidence at iteration t as

$$E_t = (D_t, R_t), \tag{3}$$

where D_t denotes the Data Extraction section and R_t denotes the Reasoning section generated by the perception agent. The Data Extraction section contains chart-relevant observations, such as category values, legend associations, and axis-aligned comparisons, while the Reasoning section contains intermediate reasoning steps that connect the extracted observations to the current answer candidate $\hat{y}_t$. This explicit textual evidence representation makes the intermediate inference process interpretable and provides a concrete object for downstream verification.

3.4. Stage 3: Independent Verification

The third stage evaluates whether the current perception output contains the target evidence required to answer the question. Rather than directly trusting the extracted evidence from the perception agent, BeaconChart uses a two-step verification procedure that independently re-reads the chart and then compares the independently extracted values with the current perception result.

First, an independent verifier re-extracts the critical values specified by the planner from the chart image. Let $K(P)$ denote the set of critical data fields identified by the plan P. The first verifier reads the chart directly using only the chart image, question, and the required critical keys, and produces an independent extraction

$$\tilde{D}_t = A_{\text{ver},1}(I, q, K(P)), \tag{4}$$

where $\tilde{D}_t$ consists of compact key–value observations for the required critical data fields. This step does not use the perception agent's extracted values, and therefore serves as an independent teacher-style reference.

Second, a comparison verifier checks whether the perception agent's extracted evidence is semantically consistent with the independently extracted reference:

$$\mathcal{V}_t = A_{\text{ver},2}(\tilde{D}_t, D_t, q). \tag{5}$$

where D_t denotes the Data Extraction section of the current perception output. The comparison focuses on the target

values required by the question rather than requiring exact string matches between keys. In particular, the verifier allows semantically equivalent matches, such as minor lexical variation in key names or small numeric deviations within a tolerance margin, while marking mismatches when required target values are missing, uncertain, or inconsistent.

Based on this comparison, the verifier produces a binary sufficiency decision $s_t \in \{0, 1\}$. The prediction is accepted only when all target-critical values required for the question are supported by semantically consistent evidence in both the independent extraction $\tilde{D}_t$ and the perception extraction D_t. Otherwise, the prediction is treated as insufficiently supported.

3.5. Stage 4: Verifier-Guided Refinement

When verification fails, BeaconChart enters a verifier-guided refinement stage that transforms the failure signal into guidance for the next perception step. The goal of this stage is not to directly revise the current prediction, but to produce corrective instruction that can improve subsequent chart understanding and, when appropriate, remain reusable across future problems.

Let $\mathcal{V}_t$ denote the verifier output at iteration t, and let $M^{g(c)}$ denote the corrective instruction memory associated with chart category $g(c)$. Given the planning output P, the current perception output E_t, the verifier result $\mathcal{V}_t$, and the candidate corrective instructions in $M^{g(c)}$, a routing module determines how the current failure should be handled:

$$\begin{aligned} a_t &= A_{\text{route}}(P, E_t, \mathcal{V}_t, M^{g(c)}), \\ a_t &\in \{\texttt{Reuse}, \texttt{CreateNew}, \texttt{NotNeeded}\}. \end{aligned} \tag{6}$$

The `Reuse` action selects an existing corrective instruction from memory when the current failure matches a previously observed error pattern. The `CreateNew` action derives a new corrective instruction when no suitable guidance exists, while `NotNeeded` is used when the failure signal is too ambiguous to support reliable refinement. Importantly, in the `CreateNew` case, the generated instruction is not stored as an instance-specific correction. Instead, it is abstracted into a reusable category-level corrective instruction that captures the general extraction strategy needed to address similar failures in future problems of the same chart category.

Let r_t denote the refinement guidance derived after the failed verification at iteration t. The next-step guidance is then defined as

$$r_t = \begin{cases} \text{retrieve}(M^{g(c)}), & \text{if } a_t = \texttt{Reuse}, \\ \text{abstract}(\mathcal{V}_t, E_t), & \text{if } a_t = \texttt{CreateNew}, \\ \emptyset, & \text{if } a_t = \texttt{NotNeeded}. \end{cases} \tag{7}$$

$$z_t = \begin{cases} z_{t-1} \oplus r_t, & \text{if } r_t \neq \emptyset, \\ z_{t-1}, & \text{otherwise}. \end{cases} \tag{8}$$

where z_0 denotes the initial category-level guidance and $\oplus$ denotes textual concatenation. The guidance is appended to the perception prompt of the next iteration, encouraging reliable chart understanding and evidence extraction.

Across problems, BeaconChart maintains a chart-category-specific corrective instruction memory that stores reusable corrective instructions. When an instruction is reused or newly created, its outcome is recorded according to whether the final prediction is correct, allowing the memory to accumulate effective corrective guidance over time.

4. Experiments

4.1. Experimental Setup

Dataset. We evaluate BeaconChart on two chart question answering benchmarks, **ChartX** (Xia et al., 2025) and **ChartQA** (Masry et al., 2022). ChartX covers 18 chart types, 22 topics, and 7 chart-related tasks spanning both perception and cognition; in this work, we focus on its QA task. For dataset-level reporting and analysis, we additionally follow the ChartX taxonomy that groups chart types into three high-level categories: *General*, *Fine-Grained*, and *Domain-Specific*. We also evaluate on ChartQA, which contains human-authored and automatically augmented question-answer pairs over bar, line, and pie charts. Together, these datasets provide complementary testbeds with diverse chart structures, question forms, and reasoning demands.

Evaluation Metrics. We use accuracy as the primary evaluation metric under a relaxed matching criterion. A prediction is considered correct if it matches the target after basic text normalization or falls within a 5% relative error margin for numeric answers. We additionally allow scale and ratio normalization to avoid penalizing formatting differences.

Baselines. We compare BeaconChart against both general-purpose MLLMs and recent chart-specialized MLLMs. Additionally, we include Qwen3-VL (Bai et al., 2025a), LLaVA-OneVision-1.5 (Li et al., 2024), InternVL3 (Zhu et al., 2025), GPT-4.1 and GPT-4.1 mini (OpenAI, 2025) as general multimodal baselines. We further compare with chart-specialized models, including ChartInstruct (Masry et al., 2024), TinyChart (Zhang et al., 2024), VProChart (Huang et al., 2025), ChartGemma (Masry et al., 2025), and ChartVLM (Xia et al., 2025).

Implementation Details. Our current implementation uses Qwen3-VL-32B-Instruct as the shared backbone for all modules, including planning, chart-specialized percep-

Table 1. The table reports overall accuracy and performance on chart-type-specific subsets. In ChartX, Type-1/2/3 denote General, Fine-Grained, and Domain-Specific, respectively. In ChartQA, Type-1/2/3 denote Bar, Line, and Pie, respectively. Human and Augmented are reported only for ChartQA. Bold indicates the best result within each dataset block.

Model	Params	Human	Augmented	Numeric	Non-numeric	Type-1	Type-2	Type-3	Overall
ChartX (Type-1/2/3 = General / Fine-Grained / Domain-Specific)									
MLLMs									
Qwen3-VL	32B	–	–	64.03	61.65	76.20	49.80	66.45	63.45
Qwen3-VL	8B	–	–	59.11	63.44	70.00	46.80	71.71	60.16
LLaVA-OneVision-1.5	8B	–	–	57.96	62.01	71.20	42.20	73.68	58.94
InternVL3	14B	–	–	35.62	43.37	44.00	30.00	40.79	37.50
GPT-4.1	–	–	–	52.00	54.84	60.20	37.60	77.67	52.69
GPT-4.1 mini	–	–	–	41.81	49.82	46.80	33.00	69.08	43.75
Chart-Specialized MLLMs									
ChartInstruct	7B	–	–	25.09	27.60	32.40	17.60	30.26	25.69
TinyChart	3B	–	–	36.43	39.07	48.00	22.00	50.66	37.07
ChartGemma	3B	–	–	24.05	42.29	35.00	23.00	25.00	28.47
ChartVLM	7.3B	–	–	37.00	**79.57**	51.60	43.80	44.74	47.31
VProChart	7.3B	–	–	21.88	20.07	26.60	14.40	27.63	21.44
Our Framework									
BeaconChart	8B	–	–	67.01	63.80	79.40	50.40	75.00	66.23
BeaconChart	32B	–	–	**71.82**	72.76	**84.40**	**56.80**	**81.58**	**72.05**
ChartQA (Type-1/2/3 = Bar / Line / Pie)									
MLLMs									
Qwen3-VL	32B	82.80	94.24	88.32	89.27	89.05	87.28	85.47	88.52
Qwen3-VL	8B	77.60	94.48	86.54	84.18	86.51	83.29	87.21	86.04
LLaVA-OneVision-1.5	8B	80.64	**94.48**	87.91	86.25	88.43	83.79	86.63	87.56
InternVL3	14B	60.24	64.32	64.20	55.18	63.05	55.11	70.35	62.28
GPT-4.1	–	84.16	90.56	87.05	88.51	88.74	79.80	89.53	87.36
GPT-4.1 mini	–	81.12	90.32	85.73	85.69	87.23	77.56	87.79	85.72
Chart-Specialized MLLMs									
ChartInstruct	7B	48.40	81.28	67.24	55.93	66.79	63.59	45.93	64.84
TinyChart	3B	54.00	68.64	63.38	53.67	61.03	63.84	58.72	61.32
ChartGemma	3B	54.72	84.32	71.36	62.71	70.94	67.33	58.72	69.52
ChartVLM	7.3B	55.44	85.76	70.49	71.00	72.29	67.83	58.14	70.60
VProChart	7.3B	50.80	87.44	72.63	56.12	71.15	71.57	40.70	69.12
Our Framework									
BeaconChart	8B	86.40	90.48	88.47	88.32	88.43	87.53	90.70	88.84
BeaconChart	32B	**89.84**	93.36	**91.82**	**90.77**	**91.90**	**88.78**	**94.77**	**91.60**

tion, verification, and routing, with role-specific prompts for each stage. The model is loaded with 4-bit NF4 quantization and bfloat16 computation for efficient inference. We use deterministic decoding throughout and allow up to three refinement iterations per example. All experiments are conducted on NVIDIA A6000 GPUs.

4.2. Results

ChartX. Table 1 reports the results on ChartX, where BeaconChart achieves the best overall accuracy of **72.05**. This corresponds to an improvement of **8.60** points over Qwen3-VL 32B and **24.74** points over the strongest chart-specialized baseline, ChartVLM. BeaconChart performs best on Numeric questions and across all three chart-type groups. The gains on ChartX are especially notable because this benchmark exhibits broader variation in chart styles and reasoning demands, making it more sensitive to early extraction and grounding errors. One exception is the Non-numeric subset, where ChartVLM achieves the highest score of 79.57, while BeaconChart reaches 72.76. Overall, these results suggest that BeaconChart is particularly effective for numerically grounded chart reasoning, while some non-numeric cases remain more challenging.

ChartQA. Table 1 presents detailed comparisons on ChartQA. BeaconChart achieves the best overall accuracy of **91.60**, outperforming the strongest general-purpose baseline, Qwen3-VL 32B, by **3.08** points and the strongest chart-specialized baseline, ChartVLM, by **21.00** points. BeaconChart performs best on the Human split, on both Numeric and Non-numeric questions, and across all three chart types. These results indicate that the proposed framework improves

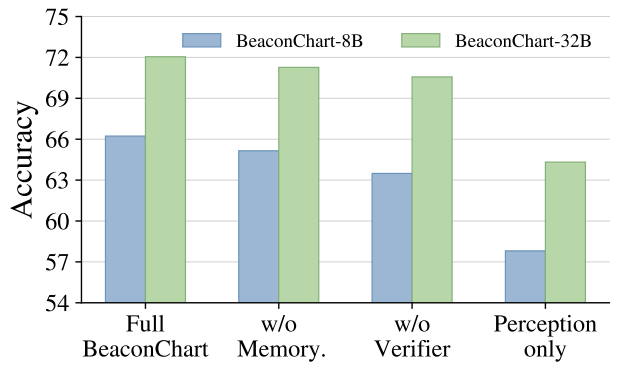

Figure 3. Ablation results of BeaconChart on ChartX with 8B and 32B backbones. *w/o Memory* removes instruction memory, *w/o Verifier* further removes the verification module, and *Perception only* uses the perception agent alone with prompt guidance.

chart understanding across diverse chart structures and question forms. The consistent gains across Bar, Line, and Pie charts show that the benefit is not limited to a single chart family. The only exception is the Augmented split, where LLaVA-OneVision slightly outperforms BeaconChart (94.48 vs. 93.36). Overall, these results suggest that BeaconChart is particularly effective when accurate evidence extraction and grounded reasoning are required.

4.3. Ablation Study

Figure 3 shows the ablation results of BeaconChart on ChartX question answering for both 8B and 32B backbones. In both settings, the full model performs best. Removing memory consistently degrades performance, and removing the verifier leads to a further drop, indicating that both components contribute to the overall effectiveness of the framework. The perception-only setting performs worst, suggesting that BeaconChart benefits not only from chart-specialized perception, but also from verification and refinement. This pattern is consistent across both model scales, indicating that the contributions of memory and verification are not specific to a particular backbone size. The gap between the full model and the perception-only variant further supports the importance of the full multi-stage pipeline beyond perception alone.

4.4. Verifier Failure Detection

Table 2 evaluates the verifier under two perspectives: final-answer level and iteration-perception level. The final-answer level measures whether the verifier correctly identifies errors in the final prediction, whereas the iteration-perception level evaluates its ability to detect errors across all intermediate reasoning steps. At the final-answer level, the verifier increases $P(\text{Error} \mid \text{Failure})$ from 27.95 to 54.15, corresponding to a 26.20-point gain and a 1.94× uplift over the base error rate. At the iteration-perception level, it achieves recall 66.45 and F1 62.02, indicating stronger sensitivity to intermediate failures. Overall, these results show that the verifier provides a reliable signal for triggering refinement when predictions are likely to be incorrect.

Table 2. Verifier failure-detection performance under two evaluation perspectives, showing that failure signals are strongly enriched with true errors and are more sensitive at intermediate steps.

Metric	Final-answer	Iteration-perception
N	1152	1636
Failure detected	277	688
Actual errors	322	602
Overall error rate	27.95%	36.80%
$P(\text{Error} \mid \text{Failure})$	54.15%	58.14%
Uplift	1.94×	1.58×
Precision	54.15%	58.14%
Recall	46.58%	66.45%
F1	50.08%	62.02%

Table 3. Consistency between relaxed and LLM-based evaluation on ChartX.

Relaxed Acc (%)	LLM Acc (%)	Cohen's κ	Agreement (%)
72.05	70.83	0.8469	93.75

4.5. Evaluation Consistency

Table 3 compares relaxed accuracy with LLM-based evaluation on ChartX. The two evaluations are highly consistent, indicating that the relaxed metric closely reflects semantic correctness. This suggests that relaxed accuracy provides a simple and reliable evaluation criterion for our experiments.

5. Limitations

BeaconChart has several limitations. First, its iterative design introduces additional inference cost compared with one-pass prediction, which may limit its practicality in large-scale deployment settings. Second, the effectiveness of refinement depends on the quality of verifier-generated judgments and corrective guidelines; if the verifier fails to diagnose an error accurately, the resulting refinement signal may be less effective. Third, the current framework assumes that the planner output is reliable and uses it as a fixed reference throughout perception and verification. If the initial plan omits necessary evidence or specifies an incorrect reasoning path, downstream modules may propagate this error rather than correct it. Finally, we evaluate BeaconChart only on chart question answering, and its applicability to other chart understanding tasks remains to be studied.

6. Conclusion

We introduced BeaconChart, a verifier-guided, memory-guided framework for chart question answering. BeaconChart improves chart reasoning by making predictions more reliable and corrective guidance reusable across similar failures. Experiments on ChartQA and ChartX show that it achieves the best overall performance against both general-purpose MLLMs and chart-specialized baselines. These results highlight the value of explicit verification and reusable corrective instruction for robust chart understanding.

LLM/Agent Usage Disclosure

The authors used LLM-based tools to assist with language polishing, structural revision, and discussion of presentation clarity. The core research idea, experimental design, implementation, analysis, and final claims were developed and verified by the authors. LLMs were not used to autonomously conduct experiments or generate results.

Acknowledgements

This work was partly supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government(MSIT) (RS-2022-00143911, AI Excellence Global Innovative Leader Education Program)

References

Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Bai, Y., Wang, M., and Vu, T.-T. MAPLE: Multi-agent adaptive planning with long-term memory for table reasoning. In *Proceedings of the 23rd Annual Workshop of the Australasian Language Technology Association*, pp. 132–158, Sydney, Australia, 2025b. Association for Computational Linguistics. URL https://aclanthology.org/2025.alta-main.10/.

Bo, X., Zhang, Z., Dai, Q., Feng, X., Wang, L., Li, R., Chen, X., and Wen, J.-R. Reflective multi-agent collaboration based on large language models. *Advances in Neural Information Processing Systems*, 37:138595–138631, 2024.

Borkin, M. A., Vo, A. A., Bylinskii, Z., Isola, P., Sunkavalli, S., Oliva, A., and Pfister, H. What makes a visualization memorable? *IEEE transactions on visualization and computer graphics*, 19(12):2306–2315, 2013.

Deng, Y., Zhang, X., Zhang, W., Yuan, Y., Ng, S.-K., and Chua, T.-S. On the multi-turn instruction following for conversational web agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8795–8812, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.477. URL https://aclanthology.org/2024.acl-long.477/.

Franconeri, S. L., Padilla, L. M., Shah, P., Zacks, J. M., and Hullman, J. The science of visual data communication: What works. *Psychological Science in the public interest*, 22(3):110–161, 2021.

Goswami, K., Mathur, P., Rossi, R., and Dernoncourt, F. Chartcitor: Multi-agent framework for fine-grained chart visual attribution, 2025. URL https://arxiv.org/abs/2502.00989.

Huang, K.-H., Chan, H. P., Fung, M., Qiu, H., Zhou, M., Joty, S., Chang, S.-F., and Ji, H. From pixels to insights: A survey on automatic chart understanding in the era of large foundation models. *IEEE Transactions on Knowledge and Data Engineering*, 37(5):2550–2568, 2024.

Huang, M., Zhang, L., Lai, H., Wu, W., Zhang, X., and Liu, J. Vprochart: Answering chart question through visual perception alignment agent and programmatic solution reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 3689–3696, 2025.

Jiang, H., Zhang, X., Garg, S., Arora, R., Kuo, S.-Z., Xu, J., Colak, A., and Dong, X. L. Memory-qa: Answering recall questions based on multimodal memories. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 24255–24277, 2025.

Kafle, K., Price, B., Cohen, S., and Kanan, C. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2018.

Kahou, S. E., Michalski, V., Atkinson, A., Kádár, Á., Trischler, A., and Bengio, Y. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*, 2017.

Keim, D. A. Information visualization and visual data mining. *IEEE transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

Li, B., Zhang, Y., Guo, D., Zhang, R., Li, F., Zhang, H., Zhang, K., Zhang, P., Li, Y., Liu, Z., et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

Liang, X., Tao, M., Xia, Y., Wang, J., Li, K., Wang, Y., He, Y., Yang, J., Shi, T., Wang, Y., et al. Sage: Self-evolving agents with reflective and memory-augmented abilities. *Neurocomputing*, 647:130470, 2025.

Liu, F., Piccinno, F., Krichene, S., Pang, C., Lee, K., Joshi, M., Altun, Y., Collier, N., and Eisenschlos, J. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12756–12770, 2023.

Masry, A., Do, X. L., Tan, J. Q., Joty, S., and Hoque, E. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022.

Masry, A., Kavehzadeh, P., Do, X. L., Hoque, E., and Joty, S. Unichart: A universal vision-language pretrained model for chart comprehension and reasoning. *arXiv preprint arXiv:2305.14761*, 2023.

Masry, A., Shahmohammadi, M., Parvez, M. R., Hoque, E., and Joty, S. Chartinstruct: Instruction tuning for chart comprehension and reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10387–10409, 2024.

Masry, A., Thakkar, M., Bajaj, A., Kartha, A., Hoque, E., and Joty, S. Chartgemma: Visual instruction-tuning for chart reasoning in the wild. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pp. 625–643, 2025.

Methani, N., Ganguly, P., Khapra, M. M., and Kumar, P. Plotqa: Reasoning over scientific plots. In *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pp. 1527–1536, 2020.

Moured, O., Alzalabny, S., Osman, A., Schwarz, T., Müller, K., and Stiefelhagen, R. Chartformer: A large vision language model for converting chart images into tactile accessible svgs. In *International Conference on Computers Helping People with Special Needs*, pp. 299–305. Springer, 2024.

OpenAI. Introducing gpt-4.1 in the api. https://openai.com/index/gpt-4-1/, 2025. Accessed: 2026-04-20.

Quan, X., Valentino, M., Dennis, L. A., and Freitas, A. Verification and refinement of natural language explanations through LLM-symbolic theorem proving. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2933–2958, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.172. URL https://aclanthology.org/2024.emnlp-main.172/.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Tang, B., Boggust, A., and Satyanarayan, A. Vistext: A benchmark for semantically rich chart captioning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7268–7298, 2023.

Tang, L., Kim, G., Zhao, X., Lake, T., Ding, W., Yin, F., Singhal, P., Wadhwa, M., Liu, Z. L., Sprague, Z., et al. Chartmuseum: Testing visual reasoning capabilities of large vision-language models. *arXiv preprint arXiv:2505.13444*, 2025.

Vogel, A., Moured, O., Chen, Y., Zhang, J., and Stiefelhagen, R. Refchartqa: Grounding visual answer on chart images through instruction tuning. In *International Conference on Document Analysis and Recognition*, pp. 523–537. Springer, 2025.

Wadhwa, M., Zhao, X., Li, J. J., and Durrett, G. Learning to refine with fine-grained natural language feedback. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12237–12257, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.716. URL https://aclanthology.org/2024.findings-emnlp.716/.

Ware, C. *Information visualization: perception for design*. Morgan Kaufmann, 2019.

Wu, A., Wang, Y., Shu, X., Moritz, D., Cui, W., Zhang, H., Zhang, D., and Qu, H. Ai4vis: Survey on artificial intelligence approaches for data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):5049–5070, 2021.

Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., and Yu, D. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Xia, R., Ye, H., Yan, X., Liu, Q., Zhou, H., Chen, Z., Shi, B., Yan, J., and Zhang, B. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *IEEE Transactions on Image Processing*, 2025.

Xu, Z., Du, S., Qi, Y., Xu, C., Yuan, C., and Guo, J. Chartbench: A benchmark for complex visual reasoning in charts. *arXiv preprint arXiv:2312.15915*, 2023.

Yu, C., Zhang, Y., Liu, Z., Ding, Z., Sun, Y., and Jin, Z. FRAME: Feedback-refined agent methodology for enhancing medical research insights. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 7690–7704, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.400. URL https://aclanthology.org/2025.findings-acl.400/.

Zhang, L., Hu, A., Xu, H., Yan, M., Xu, Y., Jin, Q., Zhang, J., and Huang, F. Tinychart: Efficient chart understanding with visual token merging and program-of-thoughts learning. *arXiv preprint arXiv:2404.16635*, 2024.

Zhu, J., Wang, W., Chen, Z., Liu, Z., Ye, S., Gu, L., Tian, H., Duan, Y., Su, W., Shao, J., et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

A. Stability Analysis

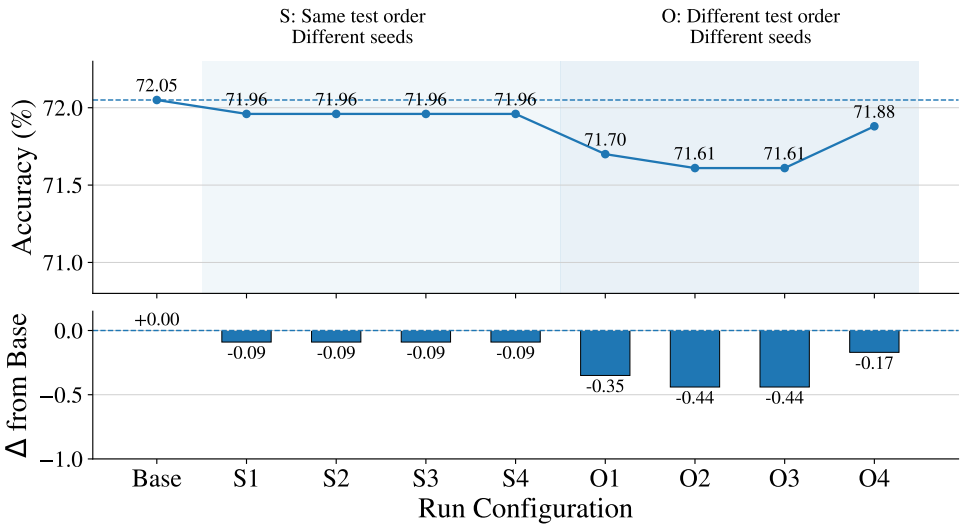

Figure 4. Stability of **BeaconChart** on ChartX QA under different random seeds and test-set orders. *S* denotes runs with the same test order but different seeds, while *O* denotes runs with both different test orders and different seeds.

Figure 4 shows that BeaconChart is stable under different random seeds and test-set orders on ChartX QA. When only the random seed is changed while the test order is fixed, the accuracy remains nearly identical to the baseline, decreasing only from 72.05 to 71.96 across all runs. When both the seed and the test order are changed, the variation becomes slightly larger, but the drop remains small, ranging from 0.17 to 0.44 points relative to the baseline. Overall, these results suggest that BeaconChart is robust to minor stochastic and ordering variations, with consistently similar performance across runs.

B. Chart Type Definitions in ChartX

In this section, we provide an overview of the chart types used in ChartX, together with their structural characteristics and reasoning requirements. ChartX contains 18 chart types spanning diverse visualization styles. To support chart-aware inference, we group these fine-grained chart types into broader chart categories $g(c)$ according to their structural characteristics.

Area and Line-based Charts. This category includes *Area* and *Line* charts, which represent continuous trends over an ordered domain. These charts often require reasoning about temporal changes, overall trends, and cumulative patterns.

Bar-based Charts. This category includes *Bar* and *3D Bar* charts. These charts encode values through discrete bars and require accurate alignment with axes and category labels. Typical reasoning patterns include value lookup, comparison, and arithmetic operations.

Proportion-based Charts. This category includes *Pie*, *Ring*, and *Rose* charts, which represent part-to-whole relationships. Reasoning over these charts often involves comparing relative proportions, identifying dominant or minor segments, and interpreting visually small regions.

Distribution-based Charts. This category includes *Histogram* and *Box* charts, which summarize data distributions. These charts require interpreting statistical properties such as ranges, medians, quartiles, and frequency patterns.

Complex Structured Charts. This category includes *Treemap*, *Multi-Axes*, *Radar*, *Bubble*, *Heatmap*, *Candlestick*, and *Funnel* charts. These chart types involve more complex layouts or multi-dimensional encodings, often requiring multi-step reasoning, cross-referencing of multiple visual cues, and careful interpretation of chart-specific semantics.

C. Memory Effectiveness Across Chart Types

Table 4. Analysis of corrective instruction memory usage and effectiveness across chart types on ChartX question answering.

Chart Type	Samples	Activated	Iter1 Acc	Final Acc	Δ	Recovery Rate
treemap	50	20	50.00	58.00	+8.00	26.67
multiaxes	50	23	42.00	46.00	+4.00	6.67
ring	50	15	66.00	70.00	+4.00	16.67
bar3d	50	25	28.00	30.00	+2.00	5.56
radar	50	20	34.00	36.00	+2.00	14.29
rose	50	12	48.00	50.00	+2.00	12.50
heatmap	52	9	82.69	84.62	+1.92	16.67
bar	200	26	81.50	81.50	0.00	0.00
pie	100	7	90.00	90.00	0.00	0.00
box	50	8	82.00	82.00	0.00	0.00
bubble	50	18	72.00	72.00	0.00	0.00
candlestick	50	18	62.00	62.00	0.00	0.00
funnel	50	1	98.00	98.00	0.00	0.00
histogram	50	9	84.00	84.00	0.00	0.00
line	200	29	85.00	84.50	-0.50	7.14
area	50	37	44.00	40.00	-4.00	5.00

Table 4 presents an analysis of corrective instruction memory usage and its impact on performance across chart types. The results show that the effectiveness of memory varies substantially across chart types. We observe that memory yields the largest gains on structurally complex chart types. For example, *treemap*, *multiaxes*, and *ring* charts improve by 8.00, 4.00, and 4.00 points, respectively. These chart types often involve hierarchical layouts, multiple axes, or non-trivial spatial relationships, which increase the likelihood of extraction and reasoning errors in early iterations. In such cases, corrective instruction memory can provide guidance that helps later iterations recover from early mistakes.

For simpler chart types such as *bar*, *pie*, and *histogram*, memory has little effect on final accuracy. These charts are generally easier to interpret, and the perception agent is often already reliable in the first iteration, leaving limited room for refinement. We also observe performance degradation in a few categories, such as *line* and *area*, where refinement may introduce over-correction or fail to provide useful guidance, especially when the task depends more on continuous trend interpretation than on discrete value recovery. Overall, these results suggest that corrective instruction memory is most beneficial for chart types with higher structural ambiguity, while having limited or mixed impact on simpler cases. This highlights the importance of selective memory usage and suggests that further gains may come from adapting refinement strategies to chart-specific characteristics.

D. Memory Statistics

Table 5. Memory activation and recovery statistics of BeaconChart on ChartX question answering.

Metric	Value
Total number of samples	1152
Memory-activated samples	277
Memory activation rate	24.05%
Accuracy on memory-activated samples	45.85%
Memory reuse samples	263
Recoverable reused samples (initially incorrect)	150
Recovery rate after memory reuse	9.33%
Overall iteration-1 accuracy	71.27%
Overall final accuracy	72.05%
Overall improvement (final − iter1)	+0.78%p

Table 5 summarizes how corrective instruction memory is used during inference and how it contributes to performance gains. Out of 1152 total samples, memory is activated in 277 cases, indicating that the framework invokes corrective guidance selectively rather than uniformly. The accuracy on these activated cases is 45.85, suggesting that memory is primarily

triggered on more challenging instances where the initial prediction is less reliable. Among the activated cases, 263 involve memory reuse, and 150 of them correspond to samples whose first-iteration prediction is initially incorrect and therefore potentially recoverable through later refinement.

This corresponds to a recovery rate of 9.33 among the reused cases that were initially incorrect. At the overall level, accuracy improves from 71.27 at the first iteration to 72.05 after refinement, yielding a gain of 0.78 points. Although the absolute gain is modest, it comes from targeted recovery in difficult cases rather than uniform improvements across all samples. These results suggest that corrective instruction memory serves as a selective mechanism that provides reusable guidance and supports targeted recovery in difficult chart question answering cases.

E. Verifier Behavior Across Chart Types

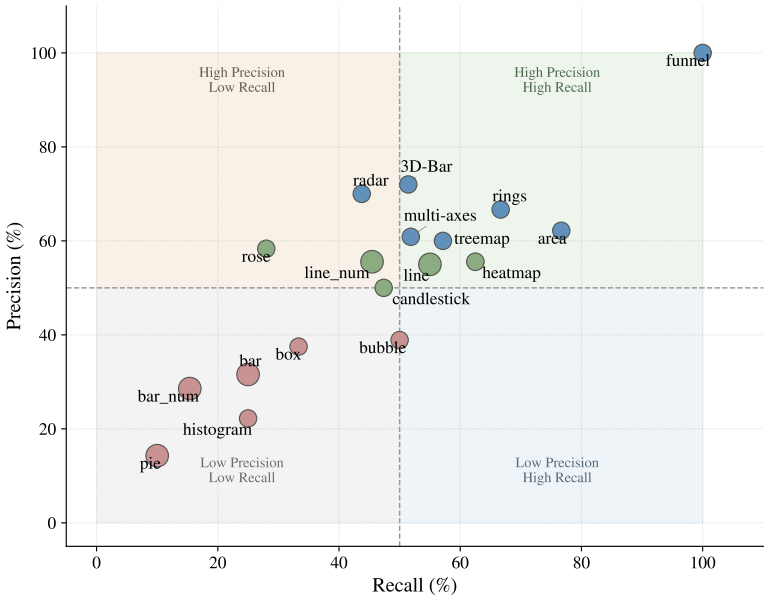

Figure 5. Precision–recall characteristics of the verifier across chart types on ChartX. Each point corresponds to a chart type, with bubble size proportional to the number of samples. The background shading denotes four regimes defined by 50% precision and recall thresholds.

Figure 5 visualizes the precision–recall characteristics of the verifier across chart types. Each point corresponds to a chart type, where precision measures the correctness of failure detection and recall reflects the proportion of actual errors identified by the verifier. Overall, the verifier exhibits non-uniform behavior across chart types. Several chart types, including *funnel*, *area*, *ring*, and *3D bar*, lie in the upper-right region of the plot, indicating that the verifier achieves both high precision and high recall. These chart types often involve more complex structures or multi-step reasoning, where initial predictions are more prone to error. In such cases, the verifier more reliably identifies unsupported predictions and triggers corrective refinement.

In contrast, chart types such as *pie*, *histogram*, and *bar_num* tend to lie in the lower-left region. This suggests that the verifier has fewer opportunities to improve performance on these chart types, as the perception model already produces relatively reliable predictions in the first iteration. Overall, these observations indicate that verifier-guided refinement is most useful in structurally complex or error-prone cases, rather than contributing uniformly across all chart types.

F. Illustrative Example: Planning and Initial Perception

BeaconChart organizes each question into planner-defined evidence fields and uses those fields to guide the initial perception step. The framework first specifies what evidence should be extracted and how the answer should be formed before performing structured evidence extraction, as illustrated in Figure 6.

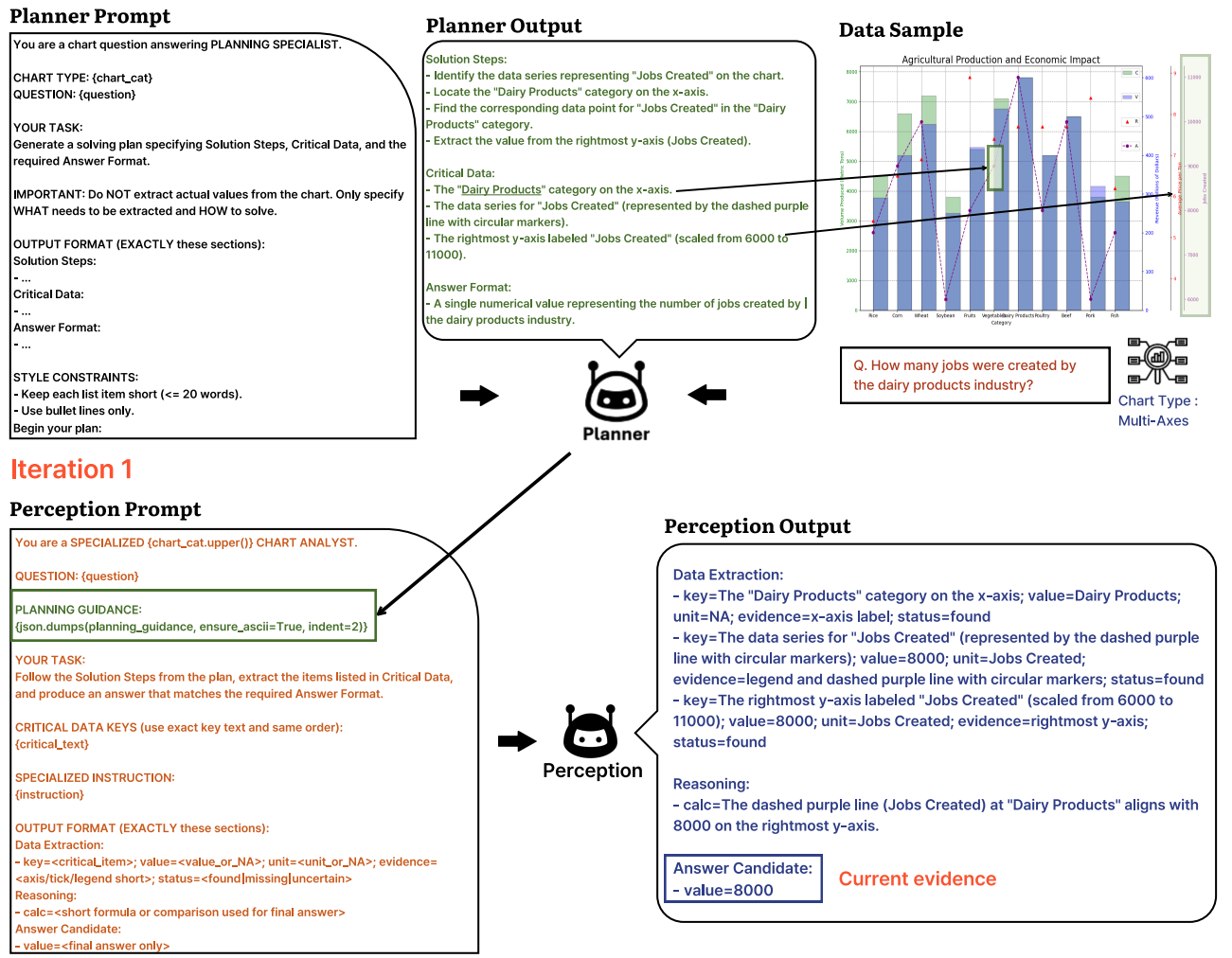

Figure 6. Example of planning and initial perception in BeaconChart. The planner first generates structured guidance without extracting actual values, and the perception module then produces the initial evidence state conditioned on that guidance.

Planner stage. The planner receives the chart, question, and chart category, and outputs three structured fields: *Solution Steps*, *Critical Data*, and *Answer Format*. It is instructed not to extract actual values, but instead to specify what evidence must be read from the chart and how the problem should be solved. In this example, the planner identifies the *Dairy Products* category, the dashed purple *Jobs Created* series, and the rightmost *Jobs Created* axis as key evidence. These outputs define the evidence structure for the next stage by specifying which visual elements the perception module should attend to.

Initial perception stage. The perception module then takes the planner guidance and produces structured *Data Extraction*, *Reasoning*, and *Answer Candidate* outputs. Because the prompt is conditioned on the planner output, the module is encouraged to fill the required evidence slots rather than directly generate a free-form answer. In this example, it extracts the target value as 8000, yielding an incorrect initial answer. This output becomes the initial evidence state for the subsequent verification stage.

G. Illustrative Example: Two-Stage Verification

After the initial perception step, BeaconChart applies a two-stage verifier to determine whether the current evidence is sufficient to support the answer. In the illustrated multi-axes case, the first perception pass predicts 8000, while the verifier independently recovers 11000 from the chart. This mismatch causes the verifier to reject the current prediction and provide grounded feedback for the next iteration.

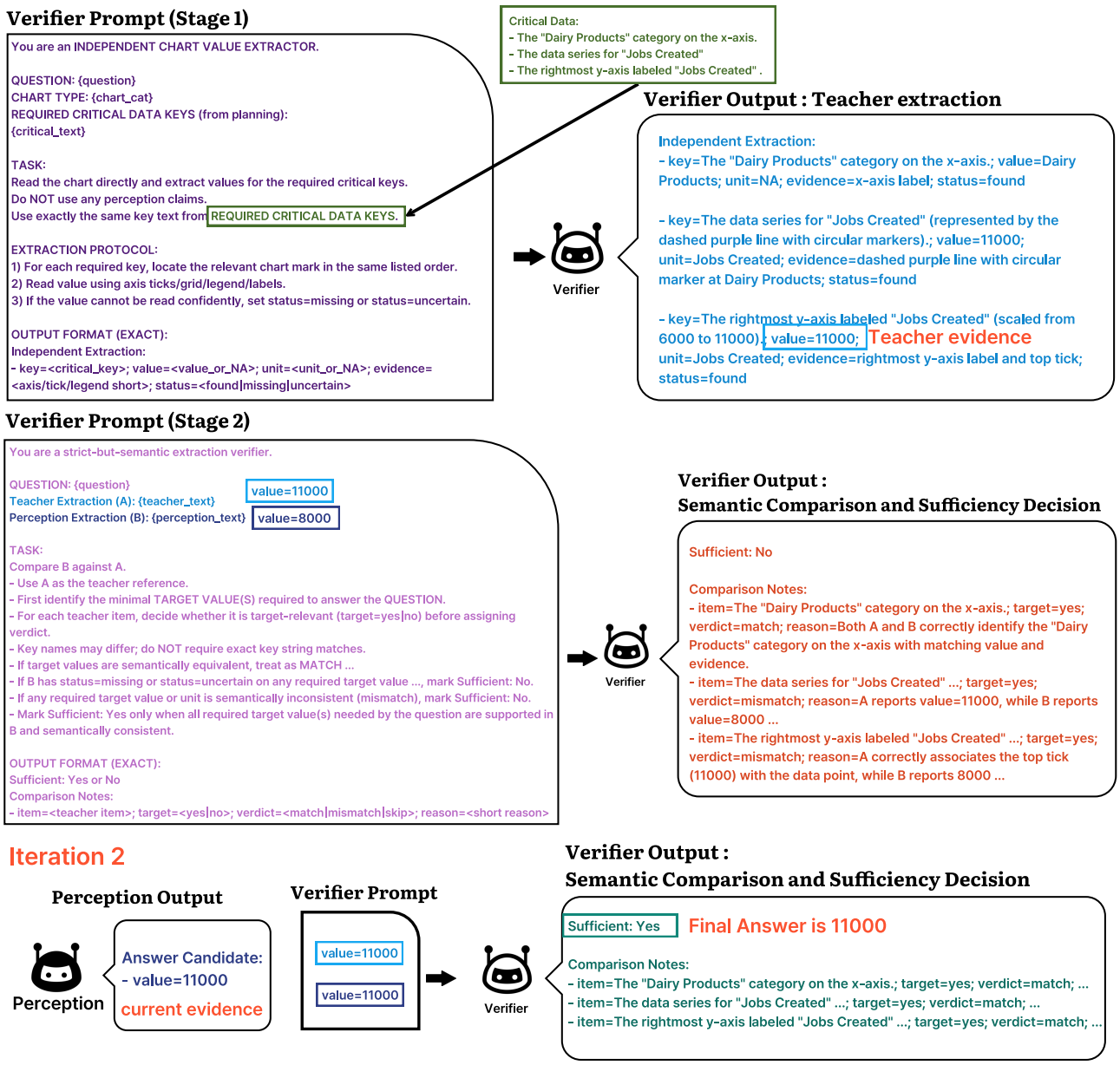

Figure 7. Example of two-stage verification in BeaconChart. The verifier first constructs teacher evidence by independently re-reading the chart and then compares the current perception output against it to make a sufficiency decision.

Verifier stage 1. The first verifier prompt takes the chart, question, chart category, and planner-defined critical-data keys as input, and is explicitly instructed not to use any perception claims. It reconstructs the required target evidence directly from the chart and outputs a structured *Independent Extraction*. In this example, Stage 1 reads the dashed purple *Jobs Created* series and the rightmost *Jobs Created* axis, and identifies the correct value as 11000.

Verifier stage 2. The second verifier prompt compares the teacher extraction with the current perception extraction and returns both item-level judgments and a final sufficiency decision. In the first pass, the perception output still supports 8000, so the verifier returns *Sufficient: No*. In the next iteration, however, the perception output is updated to 11000, and the verifier now finds it consistent with the teacher evidence, returning *Sufficient: Yes*. This example shows that the verifier both triggers corrective refinement and terminates the loop once the perception state is aligned with chart-grounded evidence.