Text2Vis: A Challenging and Diverse Benchmark for Automated Text-to-Visualization Generation

Anonymous ACL submission

Abstract

Automated data visualization plays a crucial role in simplifying data interpretation, enhancing decision-making, and improving efficiency. While large language models (LLMs) have shown promise in generating (code to produce) visualizations from natural language, the absence of comprehensive benchmarks limits the rigorous evaluation of their capabilities. We introduce Text2Vis, a benchmark designed to assess text-to-visualization models, covering 20+ chart types and diverse data science queries, including trend analysis, correlation, outlier detection, and predictive analytics. It comprises 1,985 samples, each with a data table, natural language query, short answer, visualization code, and annotated charts. The queries involve complex reasoning, conversational turns, and dynamic data retrieval. We benchmark 10+ open-source and closed-source models, revealing significant performance gaps, highlighting key challenges, and offering insights for future advancements. We then propose an actor-critic agentic inference framework, where feedback from a critic model refines the generator's output, increasing GPT-4o's pass rate from 26% to 42% over the direct approach and improving chart quality. Finally, we introduce an automated LLM-based assessment framework for scalable evaluation that measures answer correctness, code execution success, visualization readability, and chart accuracy. We release Text2Vis at $< redacted >$.

1 Introduction

Data visualization transforms raw data into meaningful visual representations, allowing users to gain insights and make data-driven decisions across various fields such as finance, healthcare, marketing, and scientific research (Aparicio and Costa, 2015; Hoque et al., 2022). It is an integral part of the data science workflow, frequently used for exploratory data analysis, outlier detection, pattern recognition, and feature identification. However, creating accurate and intuitive visualizations is challenging due to the need to correctly interpret natural language queries, select relevant data and transform it (if needed), understand suitable visualization types, and write appropriate code for visualization (Shen et al., 2022). This problem is unique as it integrates multiple modalities—visual representation, natural language understanding, logical reasoning, and code generation—making it significantly more complex than traditional NLP tasks. This process typically requires expertise in data science, programming languages, and visualization libraries (e.g., Matplotlib, Vega-Lite (Satyanarayan et al., 2016)), creating a significant barrier for non-technical users (Ali et al., 2016; Waskom, 2021; Bisong and Bisong, 2019). While natural language interfaces (NLIs) like Tableau's Ask Data (Tableau, 2025) can generate basic charts from queries, they lack flexibility, automation, customization, and advanced analytical capabilities.

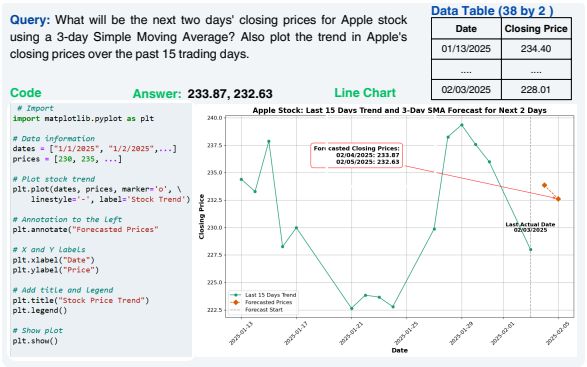

Figure 1: **Example from the Text2Vis benchmark.** **Input:** A data table containing historical stock prices and a query. **Output:** Python code for visualization, the predicted answer, and an annotated textual explanation. The chart is generated from the code.

LLMs have demonstrated strong performance in code generation and data analysis (Nejjar et al., 2025), making them promising for automated visualization tasks (Liu et al., 2021; Hoque and Islam, 2024; Maddigan and Susnjak, 2023). By under-

Dataset	Data Type	Query Type	Web Data Retrieval	Conversa-tional	Unans-werable	Multistep reasoning	Text Annotations	Text Explainability	Chart Variety	NLP to Python	Chart Specification
WikiSQL (Zhong et al., 2017)	Real	NL2SQL	✗	✗	✗	✗	✗	✗	SQL Only	No	N/A
nvBench (Luo et al., 2021)	Synthetic	NL2Vis	✗	✗	✗	✗	✗	✗	7 Types	No	Direct
NLV-Utterance (Srinivasan et al., 2021)	Real	Simple Agg.	✗	✗	✗	✗	✗	✗	10 Types	No	Direct
ADVISor (Liu et al., 2021)	Real	Aggregation	✗	✗	✗	✗	✓	✗	3 Types	No	Direct
VisEval (Chen et al., 2024)	Real + Synth.	Mid-Complex	✗	✗	✗	✓	✗	✗	7 Types	Yes	Direct
Text2Vis (Ours)	Real + Synth.	Complex Hard	✓	✓	✓	✓	✓	✓	>20 Types	Yes	Open

Table 1: Comparison of **Text2Vis** with existing text-to-visualization benchmarks.

standing natural language queries, identifying relevant data attributes, recommending chart types, and generating visualization code, LLMs can lower the barrier for non-experts to explore data without extensive technical expertise. However, as LLMs advance in coding and reasoning—often rivaling human performance on benchmarks—there is a growing need for more rigorous evaluations with complex, real-world tasks. Existing text-to-visualization benchmarks often fail to capture this complexity, limiting themselves primarily to natural language to SQL translation or simple visualization mappings based on direct mentions of data table columns (Zhong et al., 2017; Luo et al., 2021; Srinivasan et al., 2021; Liu et al., 2021). Most of them rely on rule-based methods or Vega-Lite specifications rather than code generation, limiting their practical applicability. A recent benchmark (Chen et al., 2024) evaluates LLM-driven visualization generation but lacks diverse chart types, real-world data science tasks, and multi-step reasoning. Moreover, most queries in it have explicit chart-type mentions (e.g., "draw a line chart...") rather than accommodating open-ended queries, underscoring the need for a more comprehensive benchmark.

Another major limitation of existing benchmarks is their omission of concise textual answers alongside generated visualizations, despite the fact that users often create charts to address specific data-driven questions. For instance, as shown in Fig. 1, an analyst aiming to predict Apple's stock closing price in two days using a three-day moving average would benefit not only from the visualization but also from an annotated answer. Providing such textual answers enhances task effectiveness and enables users to validate the accuracy of the generated visualization (e.g., ensuring the model correctly computes a '3-day moving average' rather than mistakenly applying a '5-day moving average').

To address these limitations, we introduce Text2Vis, a comprehensive benchmark designed to rigorously evaluate LLMs on real-world text-to-visualization tasks. Text2Vis features 1,985 queries reflecting real-life data science challenges, covering complex analytical reasoning, statistical analysis, trend analysis, outlier detection, and correlation analysis (Figure 2). Unlike previous benchmarks, it supports a wide range of visualization types, including bar charts, line charts, scatter plots, heatmaps, and specialized visualizations (Table 1). We also incorporate "retrieval-augmented" queries, which require models to fetch additional data before visualization, and conversational queries involving follow-up interactions. This makes Text2Vis a more rigorous and realistic challenge, leading to noticeable performance drops for most LLMs compared to simpler benchmarks.

To enhance visualization generation and reasoning, we also propose an actor-critic agentic framework for iterative code refinement. In this approach, the actor model generates responses and visualization code, while the critic model provides feedback to refine the output. Our experiments show that this approach significantly outperforms direct inference, producing more accurate, interpretable, and reliable visualizations. In summary, our contributions include: (*i*) **Text2Vis**, a comprehensive benchmark featuring 1,985 queries that reflect diverse, real-world data science challenges, including complex analytical reasoning and multi-step tasks; (*ii*) an **actor-critic agnetic inference approach** for iterative code refinement, which significantly enhances the accuracy, readability, and reliability of generated visualizations; (*iii*) a comprehensive **LLM-based evaluation framework** that assesses answer correctness, code execution, visualization readability, and chart accuracy, enabling scalable and consistent benchmarking; and (*iv*) **extensive evaluations** with 10 open- and closed-source models, revealing significant performance gaps and common failure patterns, providing valuable insights to guide future improvements in LLM-driven visualization generation.

2 Related Work

Text-to-Visualization Benchmarks Existing benchmarks for text-to-visualization systems often oversimplify the complexity of real-world analytical tasks by either treating the problem

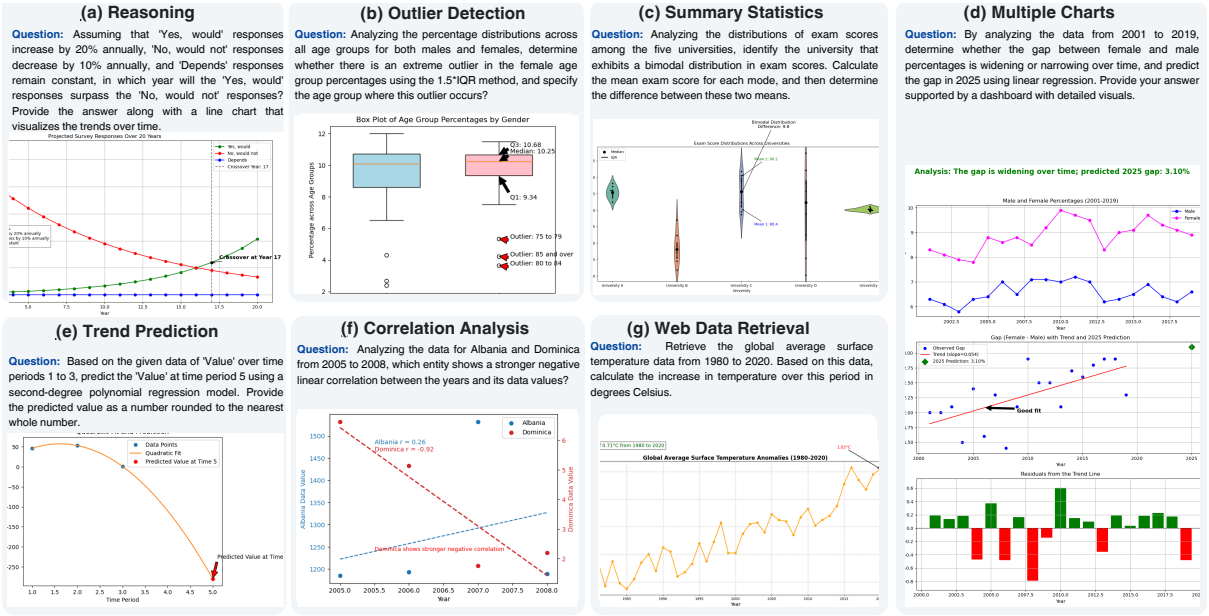

Figure 2: Examples of different question types used in data analysis, including trend prediction, reasoning, outlier detection, correlation analysis, summary statistics, and retrieval-augmented tasks.

as language-to-SQL translation or reducing it to basic visualization mapping (e.g., assigning data columns to chart axes) (Zhong et al., 2017; Luo et al., 2021; Srinivasan et al., 2021; Liu et al., 2021; Chen et al., 2024). For example, WikiSQL (Zhong et al., 2017) and nvBench (Luo et al., 2021) focus primarily on NL2SQL tasks, while NLV-Utterance (Srinivasan et al., 2021) and ADVISor (Liu et al., 2021) map textual queries to visualization specifications but do not support complex analytical queries or multi-step reasoning. Additionally, these benchmarks rely on Vega-Lite specifications rather than generating Python code, limiting their applicability to practical workflows. Quda (Fu et al., 2020) annotates user queries with analytical tasks (e.g., retrieving values, finding extrema) but lacks ground truth visualizations, making it insufficient for evaluating end-to-end visualization systems. More recently, VisEval (Chen et al., 2024) was adapted from nvBench; however, its small dataset (146 samples), limited chart variety, and reliance on queries with explicit chart-type mentions make it inadequate for assessing real-world scenarios.

As summarized in Table 1, existing benchmarks suffer from following key limitations: (1) a narrow focus on simple question and chart types, (2) a lack of multi-step analytical reasoning tasks, and (3) a disconnect from practical workflows, such as web data retrieval, conversational interactions, and handling unanswerable questions. These shortcomings hinder the evaluation of models for real-world text-to-visualization applications, underscoring the need for more comprehensive benchmarks—a gap this work aims to address.

LLMs for Automated Visualization NLP-driven visualization generation has evolved from rule-based grammar models to deep learning and LLMs. Early methods relied on predefined templates but struggled with ambiguity and scalability (Narechania et al., 2020), leading to hybrid approaches like RGVisNet (Song et al., 2022) and ADVISor (Liu et al., 2021), which improved data extraction and visualization generation. Recent advancements in LLMs have significantly enhanced their ability to generate visualization code from natural language queries (Hoque and Islam, 2024; Maddigan and Susnjak, 2023). Chat2VIS (Maddigan and Susnjak, 2023) leveraged prompt engineering to enable LLMs to generate visualizations, while ChartLlama (Han et al., 2023) further improved chart interpretation through multi-modal instruction tuning. Despite these advancements, challenges in grounding, correctness, and execution reliability persist, as highlighted by VisEval (Chen et al., 2024). To overcome these limitations, we introduce an agentic approach that enhances LLM-driven visualization generation through a structured feedback loop and contextual adaptability.

Visualization Evaluation Early works like ADVISor (Liu et al., 2021) and NLV-Utterance (Srinivasan et al., 2021) focused on verifying syntactic correctness and manually inspecting visualizations

but lacked scalability for complex queries and large datasets. Chen et al. (2023) explored LLMs for data interpretation and visualization design, though their manual grading was inefficient for large-scale benchmarking. Podo et al. (2024) introduced a structured evaluation framework assessing code correctness, visualization legality, and semantic alignment and VisEval (Chen et al., 2024) partially utilized LLMs for readability and visual accuracy. However, LLM-based chart correctness—ensuring that the generated chart accurately represents the query's intent and underlying data—alongside answer correctness remains a key bottleneck. To address this, we propose a comprehensive evaluation framework leveraging LLMs and Matplotlib to assess answer and chart correctness, as well as visual quality, readability, and execution success.

3 TEXT2VIS

We curated and synthesized a diverse dataset of data tables, queries, charts, and metadata. The dataset creation involved three key steps: (1) data table collection, (2) query generation and annotation, and (3) dataset analysis.

3.1 Data Table Construction

We started with the existing ChartQA corpus (Masry et al., 2022), which originally scraped 22K data tables from four diverse sources: (*i*) Statista (Statista, 2024), (*ii*) Pew Research (Pew, 2024), (*iii*) Our World In Data or OWID (Pew, 2024), and (*iv*) Organisation for Economic Co-operation and Development or OECD (OWID, 2024). This corpus covers a variety of topics including economics, politics, finance, climate, and health. From this collection, we manually curated 2K high-quality tables based on complexity, diversity, and analytical richness.

To broaden dataset variety and increase complexity further, we generated 173 synthetic tables using OpenAI o1-preview and Gemini Flash 1.5 Pro, incorporating missing values, multi-variable dependencies, and non-linear patterns. After rigorous validation, we removed 239 tables due to issues with table quality or overly simple/problematic queries. The final dataset comprises 1,935 carefully curated tables, providing a robust benchmark for evaluating text-to-visualization models across diverse real-world scenarios and complex analytical tasks.

3.2 Query Generation and Annotations

Query Generation and Expansion Three co-authors of this paper, who are also experts in data

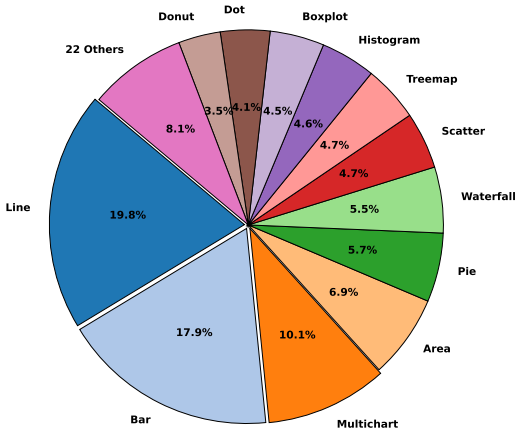

Figure 3: Common chart types in our Text2Vis.

science, manually crafted 600 high-quality queries reflecting real-world challenges such as trend analysis, statistical computations, correlation analysis, outlier detection, comparisons, deviation analysis, predictive modeling, time-series analysis, forcasting, and geospatial analysis. These queries emphasize complex reasoning, making them more challenging than those in existing benchmarks. To expand this initial set, we leveraged multiple LLMs, including OpenAI o1-preview, Gemini Flash 1.5 Pro, and Claude. Using few-shot prompting, we generated 1,624 additional queries, broadening the coverage of analytical tasks and reasoning-based challenges. After manual verification, 239 tables and queries were removed due to quality concerns.

Visualization Code and Answer Generation For each query, we generated visualization code using OpenAI o1-preview based on two libraries, Matplotlib (Bisong and Bisong, 2019) and Seaborn (Waskom, 2021), as these are among the most versatile and widely used data visualization libraries in Python. In addition, we generated short answers, visualization summaries, and metadata, including chart type and axis labels. All outputs were manually reviewed, corrected, and refined to ensure accuracy, clarity, and relevance.

3.3 Dataset Analysis

Data Table Statistics Our dataset consists of 1,985 data tables covering over 60 countries and diverse demographic and sectoral domains, including finance, healthcare, politics, energy, technology, demographics, and environment. It exhibits structural diversity, with tables containing an average of 10 rows (max: 1,000) and 3.2 columns (max: 15), ensuring a mix of compact and extensive datasets. It also includes clean data (1,789 tables), noisy data (191 tables), and hybrid cases, enabling robust

Question Category (%)					Question Complexity				Task Type			
Closed/ Open-Ended	Single query/ Conversational	Data Given/ Web-data Retrieval	Single/ Multi-Chart	Answerable/ Unanswerable	Easy	Medium	Hard	Extra Hard	Analytical	Exploratory	Predictive	Prescriptive
90/10	80/20	97/3	90/10	89/11	343	245	1098	686	700	593	191	10

Table 2: Distribution of question categories, chart types, question types based on complexity, and tasks type in Text2Vis.

evaluation of models handling real-world inconsistencies. Additionally, it focuses on query complexity, with a strong emphasis on hard queries (1,173) while maintaining a balanced range of challenges across different difficulty levels.

Query Diversity To analyze this, we used GPT-4o to automatically categorize each natural language query across three dimensions: (*i*) Question type, (*iii*) Question complexity and (*ii*) Task type.

Text2Vis encompasses a diverse set of question types that evaluate various aspects of analytical reasoning and visualization generation (Table 2). While most questions take a given data table and query as input, expecting a specific answer as output (closed-ended), others are open-ended, allowing for multiple possible visualizations. 20% questions involve multi-turn conversations, simulating natural dialogue in analytical workflows. Similarly, while many questions provide data tables, a small number of queries (3%) require models to retrieve external data before generating visualizations. Additionally, certain questions expect models to produce multiple visualizations to explore complex datasets (10%), reflecting real-world scenarios in dashboards and infographics where a single visualization is insufficient. Finally, unanswerable queries (11%) appear across all categories, adding complexity by requiring models to recognize when a valid response cannot be generated. The overall query set is highly challenging, with most questions categorized as hard (1,098) or extra hard (686). Examples of different query types are illustrated in Figure 2 and Appendix A.2.

The dataset spans a broad range of data science tasks, including analytical (700 queries), exploratory (593), predictive (191), and prescriptive (10), capturing real-world multi-step and interactive data exploration scenarios(Table 2). It also demonstrates significant linguistic richness, with an average question length of 217.87 characters and 34.15 tokens, covering a vocabulary of 6,776 unique tokens. This ensures syntactic complexity and variability. The combination of diverse data sources, multi-faceted queries, and linguistic depth makes Text2Vis a challenging and realistic benchmark for text-to-visualization models.

Code Diversity and Complexity Matplotlib and Seaborn are two of the most widely used Python libraries for visualization, and we provide code in both to ensure broad compatibility and adaptability. To measure the diversity of axis labels, we used cosine distance between the TF-IDF vectors of the axis labels (for both X and Y). The average distance was 0.97, indicating that our dataset includes a wide range of unique labels, covering different contexts and visualization types. In terms of code complexity, our scripts average 33.74 lines of code, 1,146 characters, and 123.72 tokens. Additionally, with an average of 5.34 comments per script, we prioritize clarity and maintainability, aligning with real-world visualization coding practices.

Visual Diversity Our dataset includes over 20 types of visualizations, encompassing not only common charts like bar and line charts but also more complex and less frequent types such as treemaps, boxplots, waterfall charts, and dashboard-style multi-chart visualizations (Figure 3). This diverse collection enhances model robustness by exposing it to a wide range of chart types and visual styles (e.g., color, layout). To quantify color diversity, we converted images to LAB color space and computed pairwise Euclidean distances between dominant colors, yielding a Mean CIEDE2000 (Sharma et al., 2005) Color Distance of 13.9, indicating strong variation in color schemes. To measure visual and textual diversity, we extracted text using OCR, derived visual features using CLIP (Contrastive Language-Image Pretraining) (Radford et al., 2021), and computed text embeddings via a Sentence Transformer (Reimers, 2019). This analysis produced a Mean Cosine Distance of 0.6924, highlighting strong diversity across both chart structures and annotations.

4 Methodology

We define the Text2Vis task as a text-to-visualization generation problem that evaluates how well a model can translate natural language queries into a visualization annotated with concise textual answers. The dataset consists of N examples, denoted as $D = \{t_i, q_i, a_i, v_i\}_{i=1}^{N}$, where each example includes a data table t_i, a natural language query q_i, the corresponding short answer a_i,

and the visualization code v_i. The model is tasked with generating both a_i and v_i based on t_i and q_i, with v_i producing an executable visualization code.

4.1 Models

We evaluated both state-of-the-art closed-source models and open-source models to benchmark text-to-visualization generation capabilities. For closed-source models, we tested GPT-4o (OpenAI, 2024) and Gemini 1.5-Flash (Team, 2024), which are widely used for natural language understanding and code generation. For open-source models, we prioritized deployment feasibility in the real-world and mostly selected models with less than 10B parameters. More specifically, we evaluated Qwen2.5-7B-Instruct, Qwen2.5-7B-Coder (Yang et al., 2024), Mistral-7B (Jiang et al., 2023), LLaMA 3.1-8B, DeepSeek-Coder-V2-Lite (DeepSeek-AI et al., 2024) and DeepSeek-R1-Distill-LLaMA-8B (Guo et al., 2025), as well as CodeLlama-7B-Instruct (Roziere et al., 2023). For CodeLlama, we also use its 13B and 34B versions.

4.2 Evaluation Criteria

To comprehensively assess text-to-visualization models, we define four key evaluation criteria.

Answer Match: Evaluates how accurately the generated textual response aligns with the ground truth. We use GPT-4o as an evaluator due to its impressive judging capabilities (Hackl et al., 2023).

Code Execution: Measures whether the generated visualization code executes without errors in Matplotlib. This ensures that the generated code is syntactically correct and produces an actual output.

Readability and Visualization Quality: Similar to VisEval (Chen et al., 2024), we assessed the clarity and quality of the generated charts using GPT-4o. This included evaluating aspects like layout, axis scaling, titles, labels, and color schemes.

Chart Correctness: Measures whether the generated chart accurately represents the intent of the query and the underlying data. We again use GPT-4o as an evaluator for this metric.

For scoring, Answer Match and Code Execution Success are binary (1 for success, 0 for failure), while Readability & Visualization Quality and Chart Correctness are rated on a scale from 1 to 5. A sample is considered a pass if the code executes successfully, the answer matches the ground truth, and the combined readability and chart correctness scores exceed 3.5 (implying while some readability or chart correctness issues may exist, the output remains interpretable). The evaluation prompt for chart correctness and readability, along with the scoring guide, is provided in Table 12.

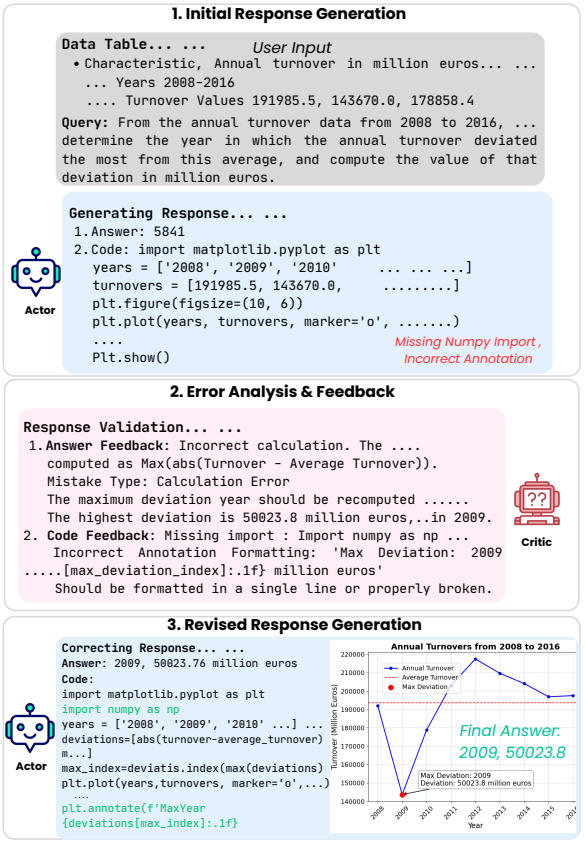

Figure 4: Our Agentic Inference Framework where the Actor (e.g., Gemini) generates an initial response, while the Critic (e.g., GPT-4o for validation, Matplotlib for visualization execution) assesses and provides feedback.

4.3 Text2Vis Inference Approaches

We use two approaches to assess the performance of text-to-visualization models: a direct inference and an agentic inference framework.

(i) Direct Inference: In this method, the model is given a prompt containing a natural language query, a corresponding data table, with instructions to generate the response in JSON format.

(ii) Agentic Inference: To enhance the quality of generated responses, we employ an actor-critic-based agentic inference framework (see Figure 4), where a critic model iteratively refines the output of the generator (actor) (Islam et al., 2024; Shinn et al., 2023). The key steps are as follows:

(1) Initial Response Generation (Actor Step): The actor model generates an initial response containing the answer, visualization code, and summary based on the given query and data table.

(2) Critic Evaluation & Feedback Generation: A separate critic model evaluates the generated

Model	Code Exec. Success	Answer Match	Visual Clarity Readability	Chart Correctness	Final Pass Rate
GPT-4o	**0.87**	**0.42**	**3.45**	**3.15**	**0.26**
Gemini-1.5-Flash	0.83	0.34	3.3	2.9	0.17
CodeLlama-7b	0.60	0.10	2.15	1.69	0.01
CodeLlama-13b	0.52	0.15	1.75	1.38	0.04
CodeLlama-34b	0.39	0.22	0.91	0.80	0.04
Llama-3.1-8B	0.72	0.24	1.68	1.59	0.07
Mistral-7B	0.39	0.24	1.4	1.31	0.06
Qwen2.5-7B	0.80	0.29	2.82	2.73	0.13
Qwen2.5-Coder-7B	0.31	0.24	1.25	1.26	0.04
DeepSeek-Coder-V2-Lite	0.75	0.22	2.93	2.63	0.10
DeepSeek-R1-Distill-Llama-8B	0.35	0.33	1.24	1.12	0.07

Table 3: Automatic evaluation results on Text2Vis using direct inference for different models. Higher values indicate better performance. Visual Clarity Readability and Chart Correctness are rated out of 5.

response based on the defined evaluation criteria. It identifies potential errors in the answer, code execution issues, and readability problems in the visualization. The critic then provides feedback to improve the initial response.

(3) Refinement & Final Response Generation: The actor model takes both the initial response and the critic's feedback into account to produce a refined final response. This iterative refinement process ensures that the final output is more aligned with the intent of the query. To ensure inference efficiency, only one round of iteration is performed.

We explore three different feedback mechanisms in the agentic framework: **(1) Self-Critique Using the Same Model**: The same model that generates the initial response acts as the critic, reviewing its own output (Saunders et al., 2022). **(2) Cross-Model Feedback**: Another external model acts as the critic. **(3) Execution-Based Feedback**: Feedback is derived from the code execution in Matplotlib, ensuring syntax correctness.

5 Experiment Results

5.1 Automatic Evaluation

Results for Direct Inference: Table 3 presents the automated evaluation results for all models assessed using the direct inference approach. The models were evaluated based on predefined criteria. GPT-4o achieved the highest performance, with 87% code execution success, 42% correct answer match, average visual clarity rating of 3.45, and a final pass rate of 26%. Among open-source models, Qwen2.5-7B performed the best, followed closely by DeepSeek-Coder-V2-Lite, both achieving a final pass rate of 13% and 10% respectively.

Despite its larger size, CodeLlama-34B performed poorly, reinforcing that increased model size does not necessarily improve structured data comprehension. Our analysis found that in over 50% of failure cases, the model struggled to extract relevant data elements from the query, leading to execution failures. However, we observe that while retrieval performance worsens with increasing model size in the CodeLlama family, the answer correctness metric shows slight improvements. This suggests that larger models may better interpret and reason about queries but still face difficulties in structured data handling.

Model	Code Exec. Success	Answer Match	Visual Clarity Readability	Chart Correctness	Final Pass Rate
GPT-4o (*without agentic*)	0.87	0.42	3.45	3.15	0.26
GPT-4o & Gemini-1.5-Flash	0.91	0.49	3.85	3.87	0.36
GPT-4o & GPT-4o	**0.94**	**0.53**	**3.99**	**4.02**	**0.42**
GPT-4o & Matplotlib	**0.94**	0.37	3.96	4.02	0.34

Table 4: Comparison of GPT-4o's direct performance (*without agentic*) and agentic inference frameworks. Higher values indicate better performance.

Results for Agentic Inference: The results in Table 4 demonstrates the effectiveness of the agentic framework in improving GPT-4o's performance across all evaluation criteria. The use of feedback mechanisms enhanced both answer correctness and code execution rates. When using self-feedback, the answer match score increased from 0.42 to 0.53, a 26% improvement, with noticeable gains in visualization clarity and correctness. Most notably, the final pass rate increased from 0.25 to 0.42, a 68% improvement, demonstrating the substantial impact of iterative refinement with only one round. The best-performing feedback method was GPT-4o with self-critique, achieving the highest code execution success rate (94%) and chart correctness (4.02). While Matplotlib execution feedback also maintained high code execution success, its final pass rate (34%) suggests that external validation alone may not be as effective as iterative language-based refinement. These results highlight the effectiveness of agentic learning.

5.2 Human Evaluation

We also selected 236 samples with a distribution similar to the original dataset for human evaluation. These samples were first verified by a new annotator (with discussions being held with the original annotator to fix any disagreements). Then the evaluation was conducted using the best-performing closed-source model: GPT-4o, and 2 open-source models (LLaMA-3.1-8B & Qwen2.5-7B).

For answer match, all responses were manually reviewed. For readability and visualization quality, we followed the same structured criteria as the automated evaluation. For chart correctness, we compared the generated visualization with the

Model	Code Exec. Success	Answer Match	Visual Clarity Readability	Chart Correctness	Final Pass Rate
GPT-4o	**0.87**	**0.39**	**3.32**	**3.30**	**0.30**
Llama-3.1-8B	0.72	0.28	1.79	1.67	0.09
Qwen2.5-7B	0.80	0.31	3.03	2.94	0.17

Table 5: Human Evaluation results on Text2Vis using direct inference for different models.

ground truth code and chart, to assign a score out of 5. An author expert in data science and visualization manually reviewed each sample and provided ratings for both aspects. The goal was to assess the reliability of LLM-based evaluation against human judgment. We found that across all 3 models, the difference between automated and manual evaluation was within 15%. The detailed result for manual evaluation is presented in Tables 5.

5.3 Ablation Studies

To evaluate the impact of dataset complexity on model performance, we conducted ablation studies across various question types. As expected, performance declined for more complex questions, such as those requiring web-based data retrieval or multiple chart generations (Table 7). Notably, GPT-4o and Gemini-Flash-1.5 outperformed open-source models on these tasks. Models also struggled with unanswerable queries, highlighting difficulties in recognizing when no valid visualization or response can be generated. Interestingly, they performed better on conversational queries, possibly due to the contextual grounding acquired during pre-training and instruction-tuning. Lastly, open-ended queries were handled slightly more effectively than closed-ended ones, indicating a stronger ability to generate diverse and flexible responses.

6 Error Analysis

We conducted a qualitative analysis to identify key error patterns. Our findings are as follows:

Code Execution Errors Syntax errors such as unterminated string literals, missing commas, and shape mismatches (e.g., *"shape mismatch: objects cannot be broadcast to a single shape"*) were common. Some models failed with plotting libraries, causing attribute errors (e.g., *"'PathPatch' object no attribute 'get_ydata'"*). Also some models exhibited naming issues (e.g., `y` instead of `years`) and indentation errors. See Figure 6(c, d, f, g).

Data Import and Retrieval Issues Several models struggled with defining datasets (*name 'df' is not defined*) and parsing date formats (*time data 'Sept 2000' does not match format %b %Y*). Most failed in web data retrieval tasks, see Figure 6h.

Logical and Analytical Reasoning Errors Mistakes in multi-step calculations, incorrect metric selection, and flawed logic led to misleading outputs. See Figure 6b.

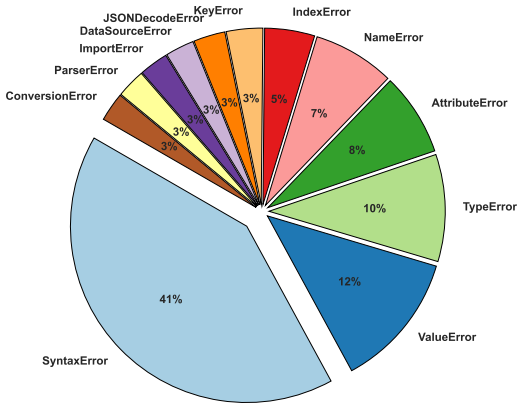

Figure 5: Error type distribution with square root transformation applied to prevent the SyntaxError category from dominating the chart.

Visualization Clarity Issues Issues like missing labels, inconsistent axis scaling, and poor color schemes impacted interpretability, even when technically correct. See Figure 6(e).

Instruction-Following Failures Many coder models failed to follow natural language instructions. CodeLlama-34B often attempted to load external CSV files (`pd.read_csv('data.csv')`) instead of processing in-context data. See Figure 6a.

Incomplete Code Generation Mistral and Llama-3.1 frequently produced incomplete implementations, lacking dataset definitions or key methods. See Figure 6g.

7 Conclusion

We introduced **Text2Vis**, a benchmark for evaluating LLMs in text-to-visualization tasks, integrating diverse datasets and over 20 chart types to assess complex questions involving multi-step reasoning, multi-chart generation, retrieval tasks, and conversational queries. Our evaluation of open- and closed-source models revealed critical limitations of LLMs, with error analysis highlighting key areas for improvement. We proposed an agentic inference framework with feedback loops that enhanced visualization accuracy, interpretability, and adaptability. This framework can further refine LLM performance by improving reasoning over structured data and enabling more accurate and explainable results. Additionally, the Text2Vis dataset will serve as a valuable resource for enhancing large language models' capabilities in code generation tasks, ensuring better alignment with real-world analytical challenges and improving their ability to generate reliable, high-quality visualizations.

Ethical Considerations

Our work focuses on sharing benchmark data and evaluation results to promote transparency and reproducibility in text-to-visualization research. All datasets used in Text2Vis are publicly available. The authors manually verified all LLM-generated queries and visualizations to ensure data integrity and accuracy.

We maintained fairness in model comparisons by applying consistent evaluation criteria across both open-source and closed-source models.

Limitations

While Text2Vis provides a comprehensive benchmark for evaluating text-to-visualization generation models, it has limitations as well. First, although our dataset incorporates diverse real-world and synthetic data, it may not fully capture the range of complexities present in specialized domains. Second, our evaluation heavily relies on LLM-based automated assessment frameworks, which, while efficient, may introduce biases in interpreting visualization quality or correctness. Although we observed strong alignment between human and automated evaluations, finer details in visualization aesthetics or interpretability may still be better captured through manual analysis.

Additionally, the benchmark provides code using Matplotlib and Seaborn libraries, but we have not included code for other popular visualization frameworks like D3.js or Vega-Lite. However, LLMs can be used for code conversion to these libraries. Furthermore, we tested multiple prompting strategies and found that our selected prompt yielded robust results. Engineering prompts may further change overall model performance. Furthermore, although this benchmark provides code in two languages, we performed all the experiments using Matplotlib, which is the most widely used visualization library. Future work can explore evaluating Seaborn-based code visualization.

Lastly, while our agentic learning framework demonstrated significant improvements, it introduces computational overhead, which may limit scalability for larger datasets or more resource-constrained environments. As an alternative, we showed how Matplotlib feedback can also provide similar improvements in performance.

References

Syed Mohd Ali, Noopur Gupta, Gopal Krishna Nayak, and Rakesh Kumar Lenka. 2016. Big data visualization: Tools and challenges. In *2016 2nd International conference on contemporary computing and informatics (IC3I)*, pages 656–660. IEEE.

Manuela Aparicio and Carlos J Costa. 2015. Data visualization. *Communication design quarterly review*, 3(1):7–11.

Ekaba Bisong and Ekaba Bisong. 2019. Matplotlib and seaborn. *Building machine learning and deep learning models on google cloud platform: A comprehensive guide for beginners*, pages 151–165.

Nan Chen, Yuge Zhang, Jiahang Xu, Kan Ren, and Yuqing Yang. 2024. Viseval: A benchmark for data visualization in the era of large language models. *IEEE Transactions on Visualization and Computer Graphics*.

Zhutian Chen, Chenyang Zhang, Qianwen Wang, Jakob Troidl, Simon Warchol, Johanna Beyer, Nils Gehlenborg, and Hanspeter Pfister. 2023. Beyond generating code: Evaluating gpt on a data visualization course. In *2023 IEEE VIS Workshop on Visualization Education, Literacy, and Activities (EduVis)*, pages 16–21. IEEE.

DeepSeek-AI, Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y. Wu, Yukun Li, Huazuo Gao, Shirong Ma, Wangding Zeng, Xiao Bi, Zihui Gu, Hanwei Xu, Damai Dai, Kai Dong, Liyue Zhang, Yishi Piao, Zhibin Gou, Zhenda Xie, Zhewen Hao, Bingxuan Wang, Junxiao Song, Deli Chen, Xin Xie, Kang Guan, Yuxiang You, Aixin Liu, Qiushi Du, Wenjun Gao, Xuan Lu, Qinyu Chen, Yaohui Wang, Chengqi Deng, Jiashi Li, Chenggang Zhao, Chong Ruan, Fuli Luo, and Wenfeng Liang. 2024. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *CoRR*, abs/2406.11931.

Siwei Fu, Kai Xiong, Xiaodong Ge, Siliang Tang, Wei Chen, and Yingcai Wu. 2020. Quda: natural language queries for visual data analytics. *arXiv preprint arXiv:2005.03257*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Veronika Hackl, Alexandra Elena Müller, Michael Granitzer, and Maximilian Sailer. 2023. Is gpt-4 a reliable rater? evaluating consistency in gpt-4's text ratings. In *Frontiers in Education*, volume 8, page 1272229. Frontiers Media SA.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*.

Enamul Hoque and M Saidul Islam. 2024. Natural language generation for visualizations: State of the art, challenges and future directions. In *Computer Graphics Forum*, page e15266. Wiley Online Library.

Enamul Hoque, Parsa Kavehzadeh, and Ahmed Masry. 2022. Chart question answering: State of the art and future directions.

Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2024. Datanarrative: Automated data-driven storytelling with visualizations and texts. *arXiv preprint arXiv:2408.05346*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Can Liu, Yun Han, Ruike Jiang, and Xiaoru Yuan. 2021. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 11–20. IEEE.

Yuyu Luo, Jiawei Tang, and Guoliang Li. 2021. nvbench: A large-scale synthesized dataset for cross-domain natural language to visualization task. *arXiv preprint arXiv:2112.12926*.

Paula Maddigan and Teo Susnjak. 2023. Chat2vis: Generating data visualisations via natural language using chatgpt, codex and gpt-3 large language models. arxiv 2023. *arXiv preprint arXiv:2302.02094*.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*.

Arpit Narechania, Arjun Srinivasan, and John Stasko. 2020. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379.

Mohamed Nejjar, Luca Zacharias, Fabian Stiehle, and Ingo Weber. 2025. Llms for science: Usage for code generation and data analysis. *Journal of Software: Evolution and Process*, 37(1):e2723.

OpenAI. 2024. Gpt-4 technical report.

OWID. 2024. Our world in data.

Pew. 2024. Pew research center.

Luca Podo, Muhammad Ishmal, and Marco Angelini. 2024. Vi (e) va llm! a conceptual stack for evaluating and interpreting generative ai-based visualizations. *arXiv preprint arXiv:2402.02167*.

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

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *CoRR*, abs/2206.05802.

Gaurav Sharma, Wencheng Wu, and Edul N Dalal. 2005. The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 30(1):21–30.

Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2022. Towards natural language interfaces for data visualization: A survey. *IEEE transactions on visualization and computer graphics*, 29(6):3121–3144.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yuanfeng Song, Xuefang Zhao, Raymond Chi-Wing Wong, and Di Jiang. 2022. Rgvisnet: A hybrid retrieval-generation neural framework towards automatic data visualization generation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1646–1655.

Arjun Srinivasan, Nikhila Nyapathy, Bongshin Lee, Steven M Drucker, and John Stasko. 2021. Collecting and characterizing natural language utterances for specifying data visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–10.

Statista. 2024. Statista.

Tableau. 2025. Tableau ask data.

Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.

Michael L Waskom. 2021. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Appendices

A.1 Data Science Question Categorization Framework

Data science plays a crucial role in uncovering insights, identifying trends, making predictions, and driving informed decision-making. However, data-related questions vary in complexity and purpose. To better organize and analyze such questions, they can be categorized into four broad groups. These categories help structure the analytical approach and determine the appropriate methods for answering each type of question.

A.1.1 Exploratory: Understanding Patterns and Structures

Some questions are aimed at understanding the overall structure of the data, identifying trends, or summarizing key characteristics. These questions do not necessarily seek to establish relationships between variables but rather focus on obtaining a broad overview of the dataset.

Exploratory questions can be categorized into the following subcategories: Insights, Trend Analysis, Statistical Summaries, Distribution Analysis, Categorical Data Analysis, Geospatial Analysis, Hierarchical Data Analysis, and Multi-Variable Analysis, each focusing on different aspects of understanding data patterns and structures.

Insights Insights-based questions focus on extracting key findings and meaningful observations from raw data. These questions often highlight notable patterns, distributions, or summary statistics.

Example: What are the top five best-selling products in the last six months?

Trend Analysis Trend analysis aims to identify changes in data over time, such as growth, decline, or seasonal fluctuations. These questions often involve historical patterns to detect trends.

Example: How have website visitor numbers changed over the past year?

Statistical Summaries Statistical summaries provide numerical insights into datasets, such as averages, variances, and standard deviations. These questions help quantify overall data characteristics.

Example: What is the median income of employees in each department?

Distribution Analysis Distribution analysis focuses on understanding how values in a dataset are

spread across a range. It helps detect skewness, uniformity, or concentration in the data.

Example: What percentage of customers fall within different age groups?

Categorical Data Analysis These questions focus on analyzing groups of categorical variables to understand their distributions, relationships, or proportions.

Example: What percentage of total sales come from each product category?

Geospatial Analysis Geospatial analysis is concerned with visualizing and understanding spatial distributions across geographic regions.

Example: What is the distribution of customer locations by city?

Hierarchical Data Analysis Hierarchical analysis examines data structured in a nested or multi-level format, often represented through tree structures.

Example: How is the company's organizational hierarchy distributed across different departments?

Multi-Variable Analysis This analysis focuses on examining interactions between multiple variables simultaneously to identify complex relationships.

Example: How do age, income, and location influence customer purchasing behavior?

A.1.2 Analytical: Explaining Relationships and Diagnosing Data

Certain questions go beyond simple observation and focus on explaining why specific patterns or anomalies exist in the data. These questions investigate relationships between variables, detect irregularities, and provide insights into underlying factors.

Analytical questions can be categorized into the following subcategories: **Reasoning, Correlation Analysis, Outlier Detection, Deviation Analysis, and Comparison Analysis**, each focusing on uncovering relationships, detecting anomalies, and understanding variations in data.

Reasoning Reasoning-based questions focus on understanding causality, hypothesis testing, and logical deductions to explain why certain patterns or anomalies exist in the data.

Example: Why do customers in certain regions spend more on our products?

Correlation Analysis Correlation analysis examines the strength and direction of relationships between two or more variables, helping to understand dependencies in data.

Example: Is there a relationship between advertising budget and sales revenue?

Outlier Detection Outlier detection identifies unusual or extreme values in the dataset that may indicate errors, fraud, or unique trends.

Example: Are there any anomalies in the monthly transaction amounts that need investigation?

Deviation Analysis Deviation analysis measures how much data deviates from expected baselines, identifying significant variations or shifts in patterns.

Example: How much does employee performance vary from the expected target levels?

Comparison Analysis Comparison analysis focuses on evaluating differences and similarities between datasets, categories, or time periods.

Example: How do customer engagement metrics compare between last year and this year?

A.1.3 Predictive: Forecasting Future Events

Some questions are forward-looking, focusing on making informed predictions about future outcomes based on historical data. These questions rely on identifying past trends to estimate what is likely to happen next.

Predictive questions can be categorized into the following subcategories: **Predictive Analysis, Time-Series Analysis, Forecasting, and Anomaly Prediction**, each focusing on using past data to estimate future outcomes and detect potential irregularities.

Predictive Analysis Predictive analysis focuses on estimating future outcomes based on historical data patterns, often using statistical models or machine learning techniques.

Example: What is the likelihood that a customer will renew their subscription next year?

Time-Series Analysis Time-series analysis involves examining data that changes over time to identify trends, cycles, and seasonal effects.

Example: How do stock prices fluctuate over different time periods?

Forecasting Forecasting predicts future values based on past trends and patterns, commonly used in sales, finance, and demand planning.

Example: What will be the expected revenue for the next quarter?

Anomaly Prediction Anomaly prediction focuses on detecting rare but significant future events that deviate from expected patterns, such as fraud detection or equipment failures.

Example: Can we predict which transactions are likely to be fraudulent?

A.1.4 Prescriptive: Recommending Data-Driven Actions

Certain questions are designed to guide decision-making by providing actionable insights. Instead of just analyzing past data or predicting future trends, these questions focus on identifying the best possible course of action.

Prescriptive questions can be categorized into the following subcategories: **Decision Support, Classification & Labeling, Clustering Analysis, and Causal Inference**, each focusing on recommending actions based on data insights and optimization techniques.

Decision Support Decision support focuses on recommending optimal strategies or actions based on data analysis. It helps businesses or individuals make informed choices by considering past trends and current conditions.

Example: What is the best pricing strategy to maximize profit while maintaining customer satisfaction?

Classification & Labeling Classification and labeling involve assigning predefined categories or labels to new data points based on learned patterns from historical data.

Example: Should this email be categorized as spam or not?

Clustering Analysis Clustering analysis identifies groups of similar data points within a dataset, allowing segmentation and targeted decision-making.

Example: Can customers be segmented into different groups based on their purchasing behavior?

Causal Inference Causal inference seeks to determine cause-and-effect relationships between variables, helping understand the impact of changes or interventions.

Example: How does increasing the marketing budget affect customer retention rates?

A.2 Query Types and Examples

To ensure a comprehensive evaluation of text-to-visualization models, the Text2Vis dataset includes diverse query types that reflect real-world data analysis scenarios. These queries are designed to test various aspects of model reasoning, retrieval capabilities, response complexity, and visualization diversity. Specifically, we categorize our dataset along the following dimensions:

- **Closed vs. Open-Ended Queries** – Distinguishes between questions expecting a specific answer as output (closed-ended) and the ones that are open-ended, allowing for multiple possible visualizations.
- **Single query vs. Conversational** – Differentiates between single query with multi-turn interactions where each query builds on prior responses and independent, standalone queries.
- **Data given vs. Web-data Retrieval** – Classifies queries based on whether they require retrieving external web data before generating visualizations.
- **Single vs. Multi-Chart** – Compares queries requiring a single visualization versus those needing multiple coordinated charts commonly found in dashboards and infographics.
- **Answerable vs. Unanswerable Queries** – Identifies whether a query has a definitive answer based on available data or if it requires additional assumptions, external knowledge, or subjective interpretation.

The following sections provide examples of few of them.

Conversational Queries: These queries simulate multi-turn interactions where each question builds on the previous answer, testing the model's ability to maintain context and continuity across queries.

- **Q1:** Can you visualize the overall trend in unemployment in the USA from 2000 to 2020? **A1 (Open-Ended):** The unemployment rate shows a significant spike during the 2008 financial crisis, peaking in 2009, followed by a steady decline until 2020. **Code:** Line chart showing the unemployment trend from 2000 to 2020.

- **Q2:** What year had the highest unemployment rate? **A2 (Short Answer):** 2009. **Code:** Bar chart highlighting the year 2009 with the highest unemployment rate.

- **Q3:** Based on the provided unemployment trend graph, what key patterns and anomalies can you identify? Discuss any significant changes, potential causes, and long-term implications. **A3 (Open-Ended):** The unemployment trend shows a sharp spike in 2009, likely reflecting the impact of the 2008 financial crisis. Post-2010, there is a gradual decline, suggesting economic recovery. However, smaller fluctuations in later years may indicate cyclical job market instabilities. A steep increase in recent years could be linked to external shocks such as a global pandemic or policy shifts. **Code:** Line chart with an outlier marker on the year 2009.

Retrieval-Augmented Queries: These queries require models to fetch additional data before visualization, testing their ability to integrate external data sources dynamically.

- **Q1:** Retrieve the unemployment data for the USA from 2000 to 2020 and visualize the trend. **A1 (Open-Ended):** The data shows a consistent trend with notable spikes during economic downturns, such as in 2009. **Code:** Line chart showing the unemployment rate in the USA from 2000 to 2020 after retrieving relevant data.

- **Q2:** Based on the retrieved data, which year had the lowest unemployment rate? **A2 (Short Answer):** 2019. **Code:** Bar chart showing the year 2019 with the lowest unemployment rate.

Short Answer vs. Open-Ended Queries: These queries distinguish between concise factual responses and detailed analytical insights.

- **Short Answer Query:** What is the highest unemployment rate recorded in the USA between 2000 and 2020? **A (Short Answer):** 9.6% in 2009. **Code:** Single bar chart highlighting 2009.

- **Open-Ended Query:** Analyze the unemployment trend in the USA from 2000 to 2020 and discuss any significant fluctuations. **A (Open-Ended):** The data indicates a sharp rise in unemployment during the 2008 financial crisis, followed by a gradual recovery. The COVID-19 pandemic in 2020 caused another spike. **Code:** Line chart with annotations on significant years (2009 and 2020).

Unanswerable Queries Unanswerable queries arise when the required data is not available in the dataset or the question cannot be logically answered based on the provided information. These queries generally fall into the following types:

- **Missing Data Queries** – When the dataset does not contain the required information. **Example:** Asking for unemployment data from 1995 when the dataset only covers 2000 onward.

- **Ambiguous Queries** – When the question lacks specificity and can have multiple interpretations. **Example:** Asking for "employment trends" without specifying sector or region.

- **Contradictory Queries** – When the query asks for information that is logically impossible. **Example:** Asking for the highest unemployment rate in 2025 when the dataset does not contain future data.

- **Hypothetical Queries** – When the question asks about alternative scenarios not represented in the data. **Example:** Asking what the unemployment rate would have been if the 2008 financial crisis had not occurred.

A.3 Common Data Visualization Error

Figure 5 highlights examples of common visualization errors, including incorrect labeling, syntax errors, and data issues. Figure 6 also provides detailed examples of model failures. Finally, Figure 7 shows a word cloud of the most common words that appeared in our evaluation error messages.

A.4 Prompt Construction

To promote transparency in query generation and reproducibility for evaluation, we provide the exact prompts used to generate our queries, evaluate our models, and develop our agentic framework. Finally, we have provided our prompts exactly, including the evaluation criteria and scoring schema used for comprehensive LLM-based evaluation.

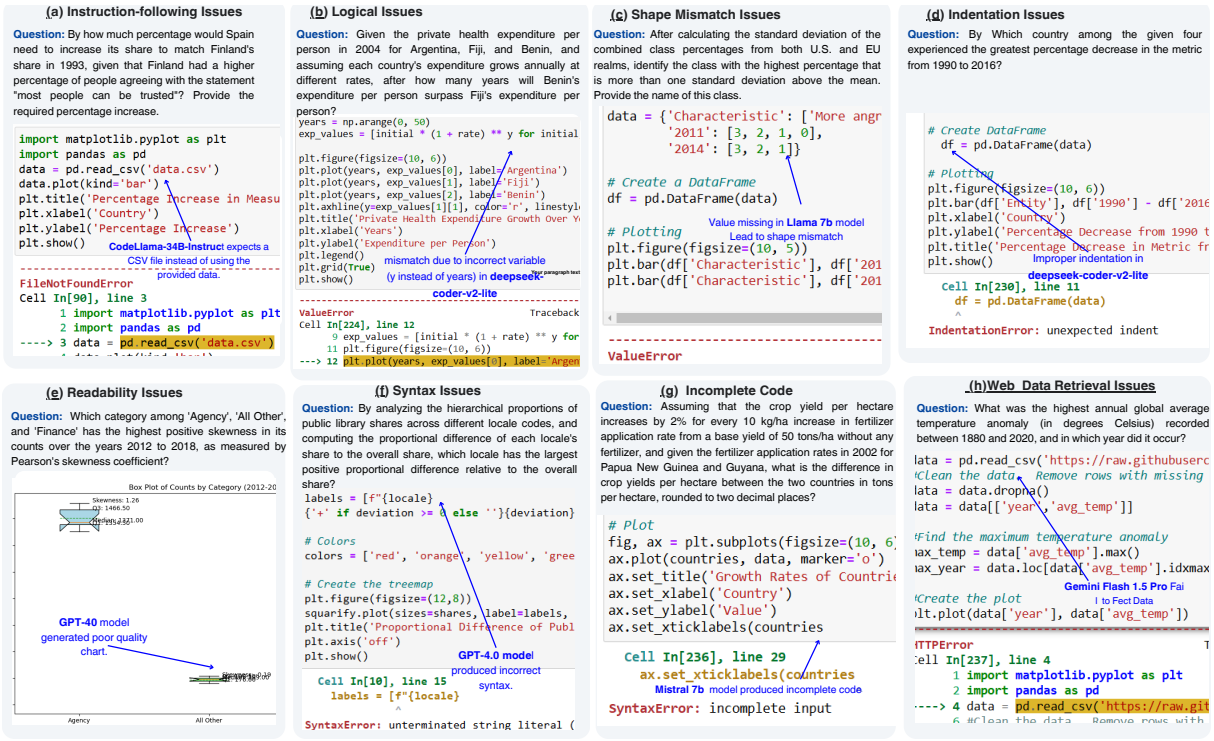

Figure 6: Common errors in Data Visualization generation.

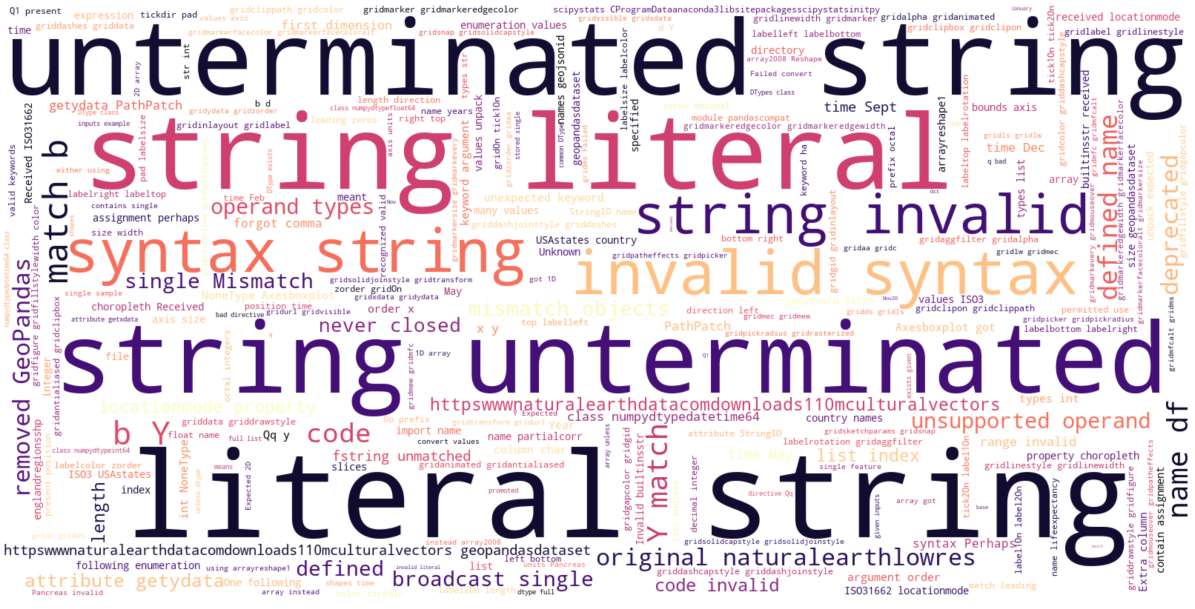

Figure 7: Most Frequent Words in Error Messages Across All Evaluated Models.

Question Type	Count
Comparison Analysis	467
Deviation Analysis	362
Trend Analysis	346
Distribution Analysis	146
Forecasting	142
Outlier Detection	140
Statistical Summaries	138
Correlation Analysis	75
Reasoning	52
Predictive Analysis	31
Hierarchical Data Analysis	22
Time-Series Analysis	18
Insights	16
Categorical Data Analysis	12
Decision Support	7
Others	11

Table 6: Distribution of various visualization tasks in the Text2Vis Dataset. Insights, Trend Analysis, Statistical Summaries, Distribution Analysis, Categorical Data Analysis, Hierarchical Data Analysis, and Multi-Variable Analysis fall under the Exploratory category. Reasoning, Correlation Analysis, Outlier Detection, Deviation Analysis, and Comparison Analysis fall under the Analytical category. Predictive Analysis, Time-Series Analysis, and Forecasting fall under the Predictive category. Decision Support falls under the Prescriptive category.

Model	Closed/ Open-Ended	Single Query/ Conversational	Data Given/ Web-data Retrieval	Single/ Multi-Chart	Answerable/ Unanswerable
GPT-4o	**0.24** / **0.26**	**0.20** / **0.50**	**0.26** / 0.08	**0.26** / **0.26**	**0.29** / 0.03
Gemini 1.5 Flash	0.17 / 0.19	0.13 / 0.33	0.18 / **0.17**	0.17 / 0.17	0.19 / 0.06
CodeLlama-7b-hf	0.02 / 0.01	0.02/ 0.01	0.00 / 0.02	0.02 / 0.03	0.02 / 0.00
CodeLlama-13b-hf	0.05 / 0.00	0.03 / 0.08	0.04 / 0.00	0.04/ 0.04	0.05 / 0.00
CodeLlama-34b-hf	0.05 / 0.02	0.02 / 0.13	0.04 / 0.00	0.05 / 0.01	0.05 / 0.00
Llama-3.1-8B	0.07 / 0.04	0.05 / 0.14	0.07 / 0.00	0.07 / 0.05	0.07 / 0.00
Mistral-7B	0.05 / 0.09	0.04 / 0.12	0.04 / 0.06	0.06 / 0.04	0.06 / 0.01
Qwen2.5-7B	0.03 / 0.15	0.11 / 0.22	0.14 / 0.00	0.14 / 0.06	0.14 / **0.07**
Qwen2.5-Coder-7B	0.04 / 0.04	0.02 / 0.11	0.14 / 0.00	0.04 / 0.03	0.04 / 0.00
DeepSeek-Coder V2-Lite	0.10/ 0.09	0.08 / 0.21	0.10 / 0.02	0.10 / 0.09	0.11 / 0.04
DeepSeek-R1-Distill-Llama-8B	0.06 / 0.10	0.06 / 0.10	0.07 / 0.02	0.07 / 0.05	0.07 / 0.02

Table 7: Performance breakdown for text-to-visualization models across different evaluation categories.

Category	Prompt Template

Conversational Query Generation	You are given a dataset in JSON format from my Data Table. Using this dataset, generate a **complex, conversational data analysis task** consisting of **4 to 5 interrelated steps**. Each step should logically build on the previous one to ensure a natural flow of analysis.
	
	To ensure clarity, **two examples** are included to demonstrate the expected structure. Please review these before generating new tasks. Then, create similar tasks that are **diverse, contextually relevant, and dependent on the new Data Table provided**.
	
	Each conversation step should include:
	
	• **Question**: A data-driven question requiring multi-step reasoning (e.g., trend analysis, variability comparison, peak detection, forecasting) that directly relates to the dataset.
	• **Answer**: Precisely answers the question.
	• **Python Code Using Matplotlib**: A self-contained code snippet that generates a relevant visualization, including clear annotations highlighting key insights.
	• **Text Summary**: A concise explanation of the insights derived from the visualization.
	• **Metadata**: Include fields such as `"ChartType"`, `"xlabel"`, and `"ylabel"` to specify the visualization type and axis labels.
	
	Example Input:
	
	Data Table
	`...`
	
	Expected JSON Output Format:
	
	`{ "Question": "...", "Answer": "...", "Code": "...", "TextSummary": "...", "ChartType": "", "xlabel": "...", "ylabel": "..." }`
	
	Ensure that each step builds on the previous one, creating a logically structured multi-step data analysis task. Maintain clarity, conciseness, and accuracy in all responses. Additionally, ensure that the generated tasks are diverse and well-aligned with the specific structure and patterns observed in the examples, while adapting to the new dataset provided.

Table 8: Prompt Templates for Conversational Query Generation.

Category	Prompt Template

Scatter Plot	You are given the following data table:
	`data_text`
	Before proceeding, evaluate whether the dataset is suitable for generating a question that is best answered by a scatter plot visualization. A dataset is considered suitable for scatter plot analysis if it contains at least two numerical variables that can be meaningfully compared.
	If the dataset is **NOT** suitable for scatter plot analysis, please output an empty JSON object with the key `"skip"` set to `true` and do not generate any further content.
	If the dataset is suitable, then perform the following tasks:
	
	1. **Generate a Single, Very Complex Data Science Question**:
	• The question must require multi-step reasoning and deep analysis.
	• Design the question specifically for a scatter plot visualization. For example, it may ask to analyze the relationship, correlation, or pattern between two numeric variables, identify outliers, or compare distributions.
	
	2. **Provide a Short Answer**:
	• The answer must be precise.
	
	3. **Output Python Code for a Scatter Plot Visualization**:
	• Use matplotlib to generate a scatter plot.
	• Ensure the code annotates key insights on the plot.
	
	4. **Include a Text Summary**:
	• Provide a concise explanation of the reasoning behind the answer, highlighting the main insight derived from the scatter plot.
	
	5. **Provide Metadata**:
	• **ChartType**: Set this to `"Scatter"`.
	• **xlabel**: The variable used for the X-axis.
	• **ylabel**: The variable used for the Y-axis (if not applicable, use `"N/A"`).
	
	To ensure clarity, two examples with scatterplot are included to demonstrate the expected structure. Please review these before generating new query and responses. Then, create similar query that are diverse, contextually relevant, and dependent on the provided data table.
	Output Requirements:
	
	• Return all the above information in a **valid JSON format** without any additional text or commentary.
	
	• Follow this exact JSON structure:
	
	Example Input:
	
	Data Table
	`...`
	
	Expected JSON Output Format:
	
	`{ "Question": "...", "Answer": "...", "Code": "...", "TextSummary": "...", "ChartType": "Scatter", "xlabel": "...", "ylabel": "..." }`

Table 9: Prompt Template for Generating a Scatter Plot Query

Category	Prompt Template
Response	You are a data visualization expert. Given a structured **data table**, respond to the following user question **based on the data**. **Input Data:** • **Data Table:** {row['Data Table']} • **Question:** {row['Question']} **Task:** 1. **Answer**: Provide a precise and concise response based on the data. If no clear answer is available, return "unanswerable". 2. **Visualization Code**: Generate Python Matplotlib code to create a meaningful visualization that accurately represents the data. Ensure annotations and highlights are included. 3. **Summary**: Briefly explain why this visualization is appropriate and how it supports the answer. **Important Requirement:** • The output must be in a **valid JSON format** without any extra text, markdown formatting, or explanations. • Ensure the JSON structure strictly follows the format below. **Expected JSON Output Format:** { "Answer": "...", "Visualization Code": "...", "Summary": "..." }

Table 10: Prompt Template for Model Response Generation

Category	Prompt Template
Agentic Framework	You are an expert in model response validation and refinement. Given a structured **data table**, Ground truth answer, a user-generated question, and an initial model response, your task is to validate and refine the model output for accuracy, correctness, and completeness. **Input Data:** • **Data Table**: `{row['Table Data']}` • **Question**: `{row['Question']}` • **Initial GPT-4o Response**: `{gpt response}` **Task:** 1. **Answer Validation**: Verify correctness and identify errors if any. 2. **Visualization Code Validation**: Check for syntax errors, readability issues, or execution problems. 3. **Summary Validation**: • Ensure the summary logically aligns with the answer and visualization. • Check for inconsistencies or misleading explanations. 4. **Refinement Task**: • Based on the feedback, refine the model response to correct errors. • Ensure the response is precise, formatted correctly, and adheres to the required JSON format. **Output Requirements:** • Ensure the final output is in a **valid JSON format** without extra text or markdown formatting. • The JSON structure must strictly follow the format below. **Expected JSON Output Format:** { "Answer": "...", "Visualization Code": "...", "Summary": "..." }

Table 11: Prompt Template for Agentic Framework

Category	Prompt Template
Evaluation	You are an evaluation expert responsible for assessing the accuracy of generated answers and the quality of visualizations. Given a structured **data table**, a user-generated question, a model-generated response, and an image-based visualization, your task is to validate the correctness of the response and evaluate the visualization quality. **Input Data:** • **Data Table**: {`row['Table Data']`} • **Question**: {`row['Generated Question']`} • **Generated Answer**: {`row['Generated Answer']`} • **Ground Truth Answer**: {`row['Answer']`} • **Generated Image**: {`row['Generated image']`} **Task:** 1. **Answer Matching**: Compare the generated answer with the ground truth using following evaluation criteria. 2. **Visualization Evaluation**: Score the visualization based on following evaluation criteria. **Evaluation Criteria:** 1. **Answer Matching (Binary: 1 or 0)** • Match if numbers are close (e.g., "48.77" vs "48.73") or equivalent percentage formats (e.g., "100" vs "100 • Match if the ground truth appears within the generated response (e.g., "100" in "The result is 100"). • For long ground truth answer, match is considered as long as the core summary remains the same, even if the wording differs. • Allow minor spelling variations or abbreviations (e.g., "Albenia" vs "Albania", "USA" vs "United States"). • No match if the meaning changes significantly (e.g., "Fragile" vs "Extreme fragility"). 2. **Readability and Quality Score (0-5)** • **Labels and Titles**: Are they clear, concise, and correctly positioned? • **Layout Spacing**: Is the layout well-organized with no clutter? • **Color Accessibility**: Are colors distinct and accessible (colorblind-friendly)? • **Axis Scaling**: Are axes correctly labeled and proportional? • **Chart Type Suitability**: Is the visualization appropriate for the data type (e.g., line chart for trends)? • **Font and Legends**: Are fonts readable, and legends properly aligned? • **Annotation Readability**: Are annotations (e.g., data labels, callouts) clear, well-placed, and non-overlapping? 3. **Chart Correctness Score (0-5)** • **Query Alignment**: Does the visualization correctly address the question? • **Data Integrity**: Are all data points accurately plotted? • **Insight Representation**: Does the chart effectively communicate its key insights based on its type? • **Handling Missing Data**: Is missing data presented appropriately without misleading distortion? • **Complexity Handling**: For multi-step queries, is the visualization logically structured? • **5.0** – Excellent: Clear, accurate, and no issues. • **4.5** – Very Good: Minor issues but does not impact understanding. • **4.0** – Good: Small flaws like minor misalignments. • **3.5** – Decent: Some readability/accuracy issues but still interpretable. • **3.0** – Average: Noticeable problems that affect clarity or correctness. • **2.5** – Below Average: Several issues that may lead to misinterpretation. • **2.0** – Poor: Significant issues making the chart unclear. • **1.5** – Very Poor: Major readability or correctness flaws. • **1.0** – Unusable: Completely unclear or misleading. • **0.0** – Failed: The visualization is unreadable or irrelevant. **Output Requirements:** • Ensure the final output is in a valid JSON format without additional text. **Expected JSON Output Format:** { "Answer Match": "...", "Readability and Quality Score": "...", "Chart Correctness Score": "..." }

Table 12: Prompt Template for Evaluating Results Using the GPT-4.0 Model.