Text2Vis: A Challenging and Diverse Benchmark for Generating Multimodal Visualizations from Text

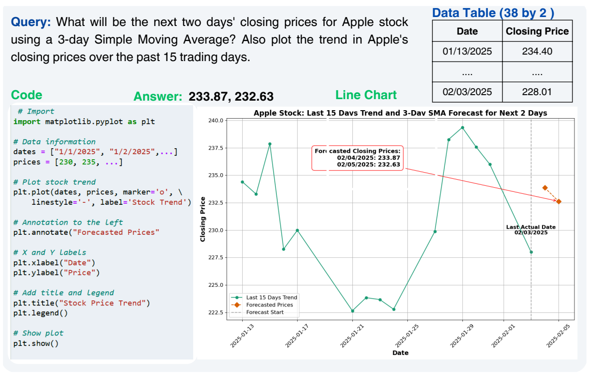
Mizanur Rahman[‡,*], **Md Tahmid Rahman Laskar**[‡,§], **Shafiq Joty**[¶,#], **Enamul Hoque**[‡,*],
[‡]York University, [§]Dialpad, [¶]Salesforce AI Research, [#]Nanyang Technological University

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 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 11 open-source and closed-source models, revealing significant performance gaps, highlighting key challenges, and offering insights for future advancements. To close this gap, we propose the first cross-modal actor-critic agentic framework that jointly refines the textual answer and visualization code, increasing GPT-4o's pass rate from 26% to 42% over the direct approach and improving chart quality. We also introduce an automated LLM-based evaluation framework that enables scalable assessment across thousands of samples without human annotation, measuring answer correctness, code execution success, visualization readability, and chart accuracy. We release Text2Vis at `https://github.com/vis-nlp/Text2Vis`.

1 Introduction

Data visualization transforms raw data into meaningful visual representations, allowing users to gain insights and make data-driven decisions (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, locate and, if necessary, transform the relevant data, identify the appropriate chart type, and generate the correct visualization code (Shen et al., 2022). This problem integrates multiple modalities—visual representation, natural language understanding, logical reasoning, and code generation—making it more challenging than traditional NLP tasks. Moreover, 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) could be helpful for non-technical users as these NLI platforms can generate basic charts from queries, they lack flexibility, automation, customization, and advanced analytical capabilities.


Query: What will be the next two days' closing prices for Apple stock using a 3-day Simple Moving Average? Also plot the trend in Apple's closing prices over the past 15 trading days.

Data Table (38 by 2)

Date	Closing Price
01/13/2025	234.40
....	....
02/03/2025	228.01

Code

```
 # Import
import matplotlib.pyplot as plt

# Data information
dates = ["1/1/2025", "1/2/2025",...]
prices = [230, 235, ...]

# Plot stock trend
plt.plot(dates, prices, marker='o', \
    linestyle='-', label='Stock Trend')

# Annotation to the left
plt.annotate("Forecasted Prices"

# X and Y labels
plt.xlabel("Date")
plt.ylabel("Price")

# Add title and Legend
plt.title("Stock Price Trend")
plt.legend()

# Show plot
plt.show()
```

Answer: 233.87, 232.63

Line Chart

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.,

[*] Contact Emails: {mizanurr,enamulh}@yorku.ca

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.

2025), making them promising for automated visualization tasks (Liu et al., 2021; Hoque and Islam, 2024; Maddigan and Susnjak, 2023). In addition to LLMs understanding natural language queries, they can also identify the relevant data attributes and the appropriate chart types for visualization. By 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 basic visualizations that map explicitly mentioned data columns to chart elements (Zhong et al., 2017; Luo et al., 2021; Srinivasan et al., 2021; Liu et al., 2021). In addition, most rely on rule-based approaches or declarative specifications like Vega-Lite, which restrict flexibility and hinder support for advanced customization or logic, unlike executable code generation. While the recent ViSEval benchmark (Chen et al., 2024) evaluates LLM-driven visualization generation, it lacks diverse chart types, real-world data science tasks, and multi-step reasoning. Moreover, most queries in this benchmark have explicit chart-type mentions (e.g., "draw a line chart...") rather than accommodating open-ended queries.

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 example, as shown in Figure 1, a query asking for a 3-day moving average to predict a stock's closing price benefits not only from the generated chart, but also from an answer that clarifies the logic used. Such multimodal outputs—visual and textual—are essential for robust, interpretable systems.

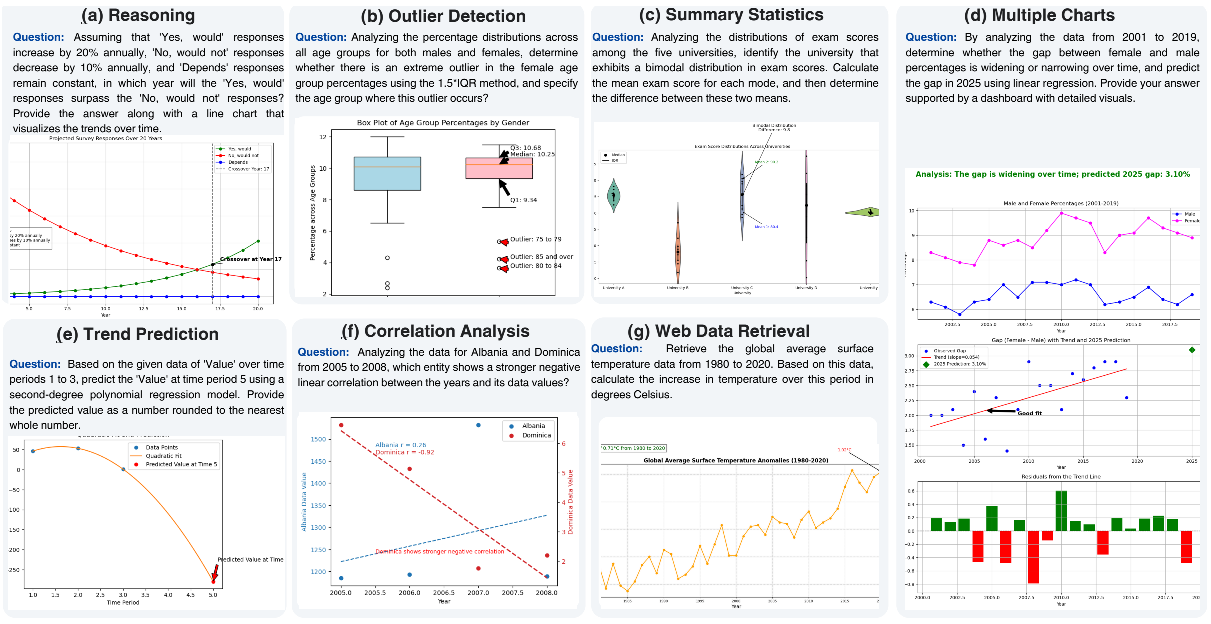
To address these gaps, we introduce Text2Vis, a comprehensive benchmark for evaluating LLMs on real-world text-to-visualization tasks. It features 1,985 diverse samples, each comprising a data table, a natural language query, a short answer, visualization code, and annotated charts. Unlike previous benchmarks, Text2Vis covers over 20 chart types and supports a wide range of realistic analytical scenarios, including retrieval-augmented queries, multi-turn conversations, multi-chart outputs, and unanswerable questions (Table 1). It also covers complex reasoning tasks such as statistical analysis, trend forecasting, outlier detection (Figure 2).

To advance LLM capabilities, we propose a cross-modal actor-critic inference framework that jointly refines textual answers and visualizations using multimodal feedback. This improves answer accuracy and chart quality, outperforming direct inference. We also introduce a fully automated LLM-based evaluation framework to assess answer and chart correctness, chart readability, and visual accuracy—validated on 1,985 samples across 11 models, closely aligning with human judgments and eliminating the need for manual annotation.

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*) a cross-modal **actor-critic agentic inference framework** that simultaneously refines both answer and visualization code, which significantly enhances the accuracy, readability, and reliability of generated outputs; (*iii*) a scalable **LLM-based evaluation framework** that systematically assesses answer correctness, visualization readability, and chart accuracy—enabling consistent, large-scale benchmarking without human annotation; and (*iv*) **extensive evaluations** with 11 open- and closed-source models, revealing significant performance gaps and common failure patterns, providing valuable directions for future research.

2 Related Work

Text-to-Visualization Benchmarks Existing text-to-visualization benchmarks often oversimplify



(a) Reasoning

Question: Assuming that 'Yes, would' responses increase by 20% annually, 'No, would not' responses decrease by 10% annually, and 'Depends' responses remain constant, in which year will the 'Yes, would' responses surpass the 'No, would not' responses? Provide the answer along with a line chart that visualizes the trends over time.

(b) Outlier Detection

Question: Analyzing the percentage distributions across all age groups for both males and females, determine whether there is an extreme outlier in the female age group percentages using the 1.5*IQR method, and specify the age group where this outlier occurs?

(c) Summary Statistics

Question: Analyzing the distributions of exam scores among the five universities, identify the university that exhibits a bimodal distribution in exam scores. Calculate the mean exam score for each mode, and then determine the difference between these two means.

(d) Multiple Charts

Question: By analyzing the data from 2001 to 2019, determine whether the gap between female and male percentages is widening or narrowing over time, and predict the gap in 2025 using linear regression. Provide your answer supported by a dashboard with detailed visuals.

(e) Trend Prediction

Question: Based on the given data of 'Value' over time periods 1 to 3, predict the 'Value' at time period 5 using a second-degree polynomial regression model. Provide the predicted value as a number rounded to the nearest whole number.

(f) Correlation Analysis

Question: Analyzing the data for Albania and Dominica from 2005 to 2008, which entity shows a stronger negative linear correlation between the years and its data values?

(g) Web Data Retrieval

Question: Retrieve the global average surface temperature data from 1980 to 2020. Based on this data, calculate the increase in temperature over this period in degrees Celsius.

Analysis: The gap is widening over time; predicted 2025 gap: 3.10%

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.

real-world analytical tasks by framing them as NL-to-SQL translation—assuming visualizations can be derived from SQL outputs—or by reducing the task to visualization specification mapping (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. Most rely on Vega-Lite rather than Python code, limiting practical use in customizable workflows. VisEval (Chen et al., 2024), adapted from nvBench, is constrained by its small set of tables (146), limited chart variety, and explicit chart-type mentions in queries, reducing its real-world relevance.

As summarized in Table 1, current benchmarks have three core limitations: (1) limited coverage of questions and chart types, (2) lack of multi-step analytical reasoning, and (3) weak alignment with real-world workflows such as web data retrieval, conversational input, and unanswerable queries. These gaps motivate the need for a more comprehensive benchmark, which we present in this work.

LLMs for Automated Visualization Visualization generation has progressed from rule-based templates 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 LLMs have significantly advanced code generation capabilities (Hoque and Islam, 2024; Maddigan and Susnjak, 2023). Chat2VIS (Maddigan and Susnjak, 2023) applied prompt engineering for visualizations, while ChartLlama (Han et al., 2023) enhanced chart understanding through instruction tuning. Yet, challenges in grounding, correctness, and execution persist (Chen et al., 2024). We address these gaps with a cross-modal agentic inference framework that jointly refines answers and code using multimodal feedback.

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) used LLMs for interpreting data and designing visualizations, but relied on manual evaluation. Podo et al. (2024) proposed a structured framework assessing code correctness, chart legality, and semantic alignment. ViSEval (Chen et al., 2024) partially leveraged LLMs, using GPT-4V only for readability scoring, while relying on rule-based checks for code validity and chart legality. Although LLMs show high agreement with human judgments (Gu et al., 2024), their broader use in evaluating visualizations remains limited. We propose a fully automated LLM-based framework for assessing answer and chart correctness, visual quality, and readability, enabling scalable benchmarking with minimal human effort.

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 Cooperation and Development or OECD (OWID, 2024). From this collection, we manually curated 2001 high-quality tables based on complexity, diversity, and analytical richness.

To broaden the dataset variety and increase complexity further, we generated 173 synthetic tables using OpenAI o1-preview and Gemini Flash 1.5 Pro (Table 14), incorporating missing values, multivariable dependencies, and non-linear patterns.

3.2 Query Generation and Annotations

Query Generation and Expansion Three coauthors of this paper, who are also experts in data 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, forecasting, 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 3.5 Haiku. Using few-shot prompting, we generated 1,624 additional queries (see Table 13), broadening the coverage of analytical tasks and reasoning-based challenges. After manual verification, 239 table-query pairs were removed due to issues with table quality or overly simple queries. The final dataset comprises 1,985 samples: 1,935 based on curated tables, and 50 designed for web-based data retrieval, providing a robust benchmark for evaluating text-to-visualization models across complex analytical tasks in real-world scenarios.

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 samples 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. In addition to clean data tables, the dataset also contains noisy tables (191 tables) featuring missing values or inconsistencies, as well as hybrid cases, enabling robust evaluation of models handling real-world inconsistencies.

Query Diversity To characterize query types, we used GPT-4o to automatically categorize each natural language query across three dimensions: (*i*) Question type, (*ii*) Question complexity and (*iii*) Task type. See Appendix (Table 19) for an example prompt used in complexity classification.

Text2Vis supports 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 Appendix A.3. 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, unanswer-

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	1173	224	1098	686	191	10

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

able 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 (1,173) or extra hard (224). Examples of different query types are illustrated in Figure 2 and Appendix A.3.

The dataset spans a broad range of data science tasks, including analytical (1098 queries), exploratory (686), 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, covering 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 4). 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. For multi-modal feature analysis, we used OCR to extract chart text, derived visual features using CLIP (Contrastive Language-Image Pretraining) (Radford et al., 2021) - whose effectiveness for representing chart images was validated via a small controlled experiment, and computed text embeddings via a Sentence Transformer (Reimers and Gurevych, 2019). This produced a Mean Cosine Distance of 0.69, highlighting strong diversity in chart structures and annotations.

4 Methodology

4.1 Problem Formulation

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.

4.2 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 (Grattafiori et al., 2024), 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.

4.3 Text2Vis Inference Approaches

We use two approaches to assess the performance of text-to-visualization models: a direct inference and an agentic inference framework (Fig 3).

(i) Direct Inference: In this method, the model is given a prompt containing a natural language query, a data table and instructions to generate a JSON response containing both a short answer and visualization code. We evaluate three prompting strategies: (a) zero-shot prompting, (b) 3-shot prompting, and (c) retrieval-augmented 3-shot prompting with dynamically selected examples (Appendix A.4).

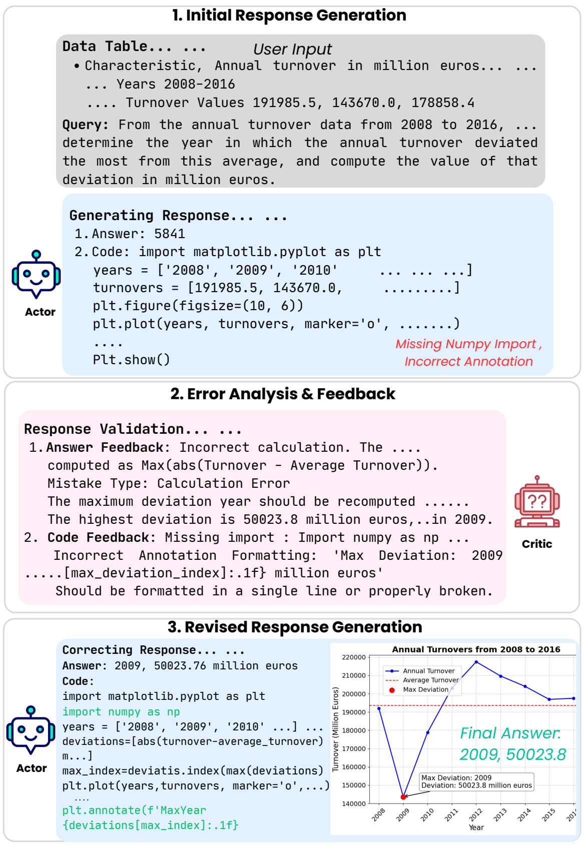
(ii) Agentic Inference: We propose a cross-modal actor-critic inference framework, inspired by reflective LLM workflows (Islam et al., 2024; Shinn et al., 2023; Madaan et al., 2023). To our knowledge, this is the first agentic setup for text-to-visualization that incorporates multimodal feedback on answer correctness, code quality, and the resulting visualization (Figure 3). The framework is also model-agnostic: it takes the initial $\langle$answer, code$\rangle$ output from any baseline inference model f_θ and routes it into the actor–critic refinement loop. The key steps are:

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

(2) Critic Evaluation & Feedback Generation: A separate critic model analyzes the initial response and provides structured feedback across three modalities: answer feedback (numerical correctness), code feedback (syntax/semantic checks), and visual feedback (clarity and correctness of the generated chart). The visual feedback is for the chart produced by executing the generated code.

(3) Refinement & Final Response Generation: The actor 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.

Feedback Strategies. We explore three critic configurations: (1) self-critique (same model) (Saunders et al., 2022), (2) cross-model critique (external model), and (3) execution-based feedback (e.g., Matplotlib error traces). Each variant affects the reliability of downstream refinement.



```
Data Table... ...
 • Characteristic, Annual turnover in million euros... ...
   ... Years 2008-2016
   .... Turnover Values 191985.5, 143670.0, 178858.4
Query: From the annual turnover data from 2008 to 2016, ...
determine the year in which the annual turnover deviated
the most from this average, and compute the value of that
deviation in million euros.
```

```
Generating Response... ...
 1. Answer: 5841
 2. Code: import matplotlib.pyplot as plt
    years = ['2008', '2009', '2010'     ... ... ...]
    turnovers = [191985.5, 143670.0,     .........]
    plt.figure(figsize=(10, 6))
    plt.plot(years, turnovers, marker='o', .......)
    ....
    Plt.show()
```

```
Response Validation... ...
 1. Answer Feedback: Incorrect calculation. The ....
   computed as Max(abs(Turnover - Average Turnover)).
   Mistake Type: Calculation Error
   The maximum deviation year should be recomputed ......
   The highest deviation is 50023.8 million euros,..in 2009.
2. Code Feedback: Missing import : Import numpy as np ...
    Incorrect  Annotation  Formatting:  'Max  Deviation:  2009
.....[max_deviation_index]:.1f} million euros'
    Should be formatted in a single line or properly broken.
```

```
Correcting Response... ...
Answer: 2009, 50023.76 million euros
Code:
import matplotlib.pyplot as plt
import numpy as np
years = ['2008', '2009', '2010' ...] ...
deviations=[abs(turnover-average_turnover)
m...]
max_index=deviatis.index(max(deviations)
plt.plot(years,turnovers, marker='o',...)
  ....
plt.annotate(f'MaxYear
{deviations[max_index]:.1f}
```

Figure 3: 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.

4.4 Evaluation Criteria

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

Answer Match: Assesses how well the generated text aligns with the ground truth.

Code Execution: Measures whether the generated visualization code executes successfully, ensuring syntactic correctness and output generation.

Readability and Visualization Quality: Assesses the visual clarity and design quality of the chart, including layout, axis scaling, labels, titles, and color usage (Chen et al., 2024).

Chart Correctness: Measures whether the generated chart accurately represents the intent of the query and the underlying data.

Scoring is binary for Answer Match and Code Execution (1 = success, 0 = failure), while Readability and Chart Correctness are rated from 1 to 5 (Table 18). A sample is considered a pass if the code executes successfully, the answer matches the ground truth, and both the readability score and the chart correctness score are at least 3.5, implying that minor readability or correctness issues may exist, but the output remains interpretable.

Automatic Judge: We use GPT-4o as the evaluation judge for all criteria, including Answer Match, Readability, Visualization Quality, and Chart Correctness, based on its strong alignment with human judgments (Zheng et al., 2023; Gu et al., 2024). The model is invoked with a fixed system prompt and deterministic parameters (see A.6). The evaluation rubric and prompt are detailed in Table 18. To demonstrate scalability, we evaluated 1,985 samples in 5 minutes for a total cost of $2.0 (see A.7.5).

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, chart correctness of 3.15, 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. Over 50% of its failures were due to incorrect extraction of relevant data elements, leading to execution errors.

Results for Agentic Inference: Table 4 shows that our agentic framework consistently improves GPT-4o's performance across all metrics, outperforming zero-shot, few-shot, and RAG baselines. With one round of Answer + Code feedback, the answer match improved from 42% to 53%, readability from 3.45 to 3.99, chart correctness from 3.15 to 4.02, and the final pass rate from 26% to 42% (+62%). Adding visual feedback further improved readability (4.23) and chart correctness (4.24), though answer match remained stable—indicating its impact is primarily visual. Among all settings, Answer + Code feedback yielded the strongest final pass rate. In contrast, code-only and execution-only feedback achieved high execution rates but low final pass rates due to poor answer accuracy. Also, all improvements in final pass rate over the zero-shot baseline are statistically significant ($p < 0.01$), based on McNemar's test on all 1,985 paired samples. These findings highlight the benefit of multimodal feedback in agentic refinement.

5.2 Human Evaluation

To validate the reliability of our automated evaluation, we manually annotated 236 samples stratified to reflect the full dataset's distribution. Three representative models were assessed: GPT-4o (closed-source, high-performing), LLaMA-3.1-8B (open-source, lower-performing), and Qwen2.5-7B (open-source, best-performing), using the same rubric as the automated setup.

Across all three models, automated and manual evaluations differed by under 15% on all criteria: *Answer Match*, *Readability*, and *Chart Correctness*. We found strong correlation between human and model judgments for each metric, with high Pearson (r) and Spearman (ρ) coefficients (see Table 9). To further assess agreement on the final pass rate, we computed Cohen's Kappa, obtaining an average score of 0.78—indicating substantial alignment.

Evaluation Consistency and Robustness To assess the generalizability of our evaluation framework, we included Gemini 1.5 Pro as an alternative judge alongside GPT-4o. Across all models, Gemini-based pass rates closely align with GPT-4o (Table 3), with strong correlation between the two judges (Table 8) and comparable agreement with human annotations (Table 10). Although both LLMs demonstrate high alignment with human judgments, GPT-4o achieved slightly better correlation and is more widely used as a standard evaluator. Accordingly, we report GPT-4o-based scores as the primary metric, while including Gemini results for robustness. We also conducted a repeatability study on the same stratified 236-sample subset using GPT-4o, evaluating each query five times with identical prompts. Results showed minimal variation, with over 97.5% of samples yielding consistent final pass rates. Further details (App. A.7.2).

5.3 Ablation Studies

We conducted ablation studies to assess feedback contributions in our agentic framework and analyze robustness across task types.

Feedback Modality Ablation. As shown in Table 4c, we compared full tri-modal feedback (answer, code, and visual) against reduced variants: answer-only, code-only, visual-only, and execution-based feedback using Matplotlib. Removing any

Model	Code Exec. Success (%)	Answer Match (%)	Visual Clarity Readability	Chart Correctness	Final Pass Rate (%)	
					GPT	Gemini
GPT-4o	**87**	**42**	**3.45**	**3.15**	**26**	**24**
Gemini-1.5-Flash	83	34	3.30	2.90	17	15
CodeLlama-7B	60	10	2.15	1.69	1	1
CodeLlama-13B	52	15	1.75	1.38	4	3
CodeLlama-34B	39	22	0.91	0.80	4	3
Llama-3.1-8B	72	24	1.68	1.59	7	6
Mistral-7B	39	24	1.40	1.31	6	4
Qwen-2.5-7B	80	29	2.82	2.73	13	12
Qwen-2.5-Coder-7B	31	24	1.25	1.26	4	3
DeepSeek-Coder-V2-Lite	75	22	2.93	2.63	10	8
DeepSeek-R1-Distill-Llama-8B	35	33	1.24	1.12	7	8

Table 3: Automatic evaluation on Text2Vis using direct inference across models. Higher scores indicate better performance. Visual Clarity, Readability, and Correctness are rated out of 5 by GPT-4o. The last column gives Final Pass Rate from Gemini 1.5 Pro for comparison.

Model Setup	Strategy	Code Exec. Success (%)	Answer Match (%)	Clarity Readability	Chart Correctness	Final Pass Rate (%)
(A) Baseline						
GPT-4o	0-shot	87	42	3.45	3.15	26
Gemini 1.5 Flash	0-shot	83	34	3.30	2.90	17
GPT-4o	3-shot	88	42	3.45	3.15	26
Gemini 1.5 Flash	3-shot	81	29	3.36	3.38	20
GPT-4o	RAG + 3-shot	88	38	3.65	3.75	31
Gemini 1.5 Flash	RAG + 3-shot	80	31	3.30	3.45	22
(B) Agentic Inference (LLM Feedback)						
GPT-4o + Gemini 1.5	Answer + Code	91	49	3.85	3.87	36
GPT-4o + GPT-4o	Answer + Code	**94**	**53**	**3.99**	**4.02**	**42**
GPT-4o + GPT-4o	Answer + Code + Visual	93	46	4.02	4.23	41
(C) LLM Feedback Ablation						
GPT-4o + GPT-4o	Answer Only	86	47	3.51	3.20	28
GPT-4o + Matplotlib	Code Exec Only	94	37	3.96	4.02	34
GPT-4o + GPT-4o	Code Only	94	36	3.99	4.19	32
GPT-4o + GPT-4o	Visual Only	94	38	4.03	4.24	33

Table 4: Model performance for **(A) Baseline**, **(B) Agentic Inference**, and **(C) LLM Feedback Ablation**. Code execution and pass rate are shown as percentages.

Model	Code Exec. Success (%)	Answer Match (%)	Visual Clarity Readability	Chart Correctness	Final Pass Rate (%)
GPT-4o	**87**	**39**	**3.32**	**3.30**	**30**
Llama-3.1-8B	72	28	1.79	1.67	9
Qwen2.5-7B	80	31	3.03	2.94	17

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

modality significantly degraded performance, confirming that multi-modal integration is essential for effective agentic refinement. We also studied text-only and visual-only generation settings and found that jointly generating both answer and code does not degrade performance, see A.7.4 and Tab 12.

Task-Type Robustness. We evaluated model performance across task categories (Table 11). Model performance declined for complex queries—especially those requiring web-based retrieval or multi-chart generation—across all metrics , reflecting LLM limitations in grounding external information and reasoning over interdependent outputs. GPT-4o and Gemini-1.5 Flash consistently 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 more effectively than closed-ended ones, indicating a stronger ability to generate diverse and flexible responses.

6 Error Analysis

We conducted a qualitative error analysis by manually evaluating 200 randomly selected samples for each of the 11 models to identify key error patterns (see fig.5). Our findings are as follows:

Code Execution Errors – Common syntax and runtime failures such as unterminated strings, missing commas, shape mismatches (e.g., *"shape mismatch: objects cannot be broadcast to a single shape"*), and attribute errors (e.g., *"'PathPatch' object has no attribute 'get_ydata'"*). Naming issues (e.g., `y` instead of `years`) and indentation errors also disrupted execution. See Figure 6(c, d, f, g)

Data Import and Retrieval Issues – Frequent failures in defining in-context datasets (*'df' is not defined*), handling non-standard date formats (*time data 'Sept 2000' does not match %b %Y*), and executing web-based data retrieval. 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.

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 Several models ignored task constraints or attempted irrelevant actions. For example, CodeLlama-34B frequently used `pd.read_csv('data.csv')` instead of processing the provided data. See Figure 6a.

Incomplete Code Generation Outputs from models like Mistral and LLaMA-3.1 were often missing dataset definitions, key methods or steps, resulting in unusable code. See Figure 6g.

7 Conclusion

We introduce **Text2Vis**, a benchmark for evaluating LLMs in text-to-visualization tasks, featuring diverse datasets and over 20 chart types to support complex queries involving multi-step reasoning, retrieval, multi-chart generation, and conversations. Our evaluation of open- and closed-source models revealed critical limitations, with error analysis highlighting key areas for improvement. To address performance gaps, we propose a cross-modal agentic inference framework that enables LLMs to refine both textual answers and visualization code using structured feedback. This framework significantly enhances answer accuracy, chart correctness, and overall pass rates. In addition, we present a scalable and fully automated LLM-based evaluation pipeline that assesses correctness, code execution, and visual quality. We believe Text2Vis is a valuable resource for advancing LLM capabilities in visualization code generation, promoting better alignment with real-world tasks and enabling more accurate, 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 some limitations. First, although our dataset incorporates diverse real-world and synthetic data, it may not fully capture the range of complexities found in specialized domains. Second, our evaluation relies on LLM-based automated assessment frameworks which, while efficient, may introduce biases in interpreting visualization quality or correctness. Although we tested with two different LLM-based judges and observed strong alignment with human evaluations as well as high correlation between the two judges, finer aspects of visualization aesthetics or interpretability may still be better assessed through manual review.

Additionally, our benchmark focuses on Python-based visualization using Matplotlib and Seaborn, which are widely adopted for their versatility, compatibility, and extensive support in data science workflows. While we do not provide native code for other frameworks like Vega-Lite or D3.js, the Python code can be readily converted to these libraries using existing LLMs or code converters. Future work may extend Text2Vis by enabling direct generation of Vega-Lite, D3.js, or Plotly code, allowing broader support for interactive and web-based visualizations.

Furthermore, while we describe our method as an agentic inference framework, the current implementation follows a fixed actor–critic loop, without dynamic planning or adaptive tool selection. Future extensions could incorporate a planning component that enables the model to autonomously decide when to revise, retrieve external information, or switch visualization libraries—moving closer to a truly autonomous agent architecture.

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. The agentic loop introduces 3× latency over zero-shot inference; while effective, this raises scalability challenges in resource-constrained settings. 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, and 21 others. 2024. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *CoRR*, abs/2406.11931.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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. In *Computer Graphics Forum*, volume 41, pages 555–572. Wiley Online Library.

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, and 1 others. 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*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Paula Maddigan and Teo Susnjak. 2023. Chat2vis: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. *Ieee Access*, 11:45181–45193.

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. *Preprint*, arXiv:2303.08774.

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, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, and 1 others. 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. *Preprint*, arXiv:2403.05530.

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, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

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

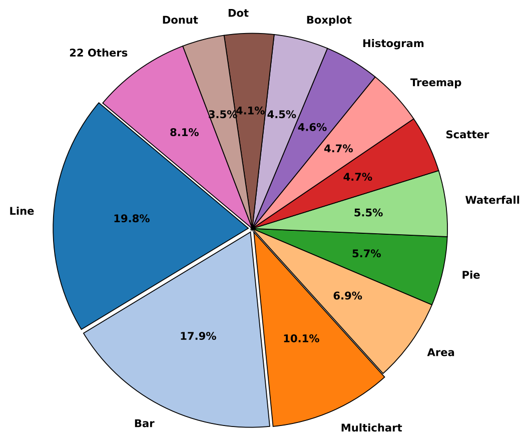


Figure 4: Common chart types in our Text2Vis.

A Appendices

A.1 Common Chart Types

Text2Vis includes a wide range of chart types that reflect the diversity of real-world data analysis tasks 4. Line charts are essential for visualizing trends over time, making them ideal for time-series forecasting and moving averages. Bar charts are widely used for comparing categorical variables, especially in demographic and economic data. Multichart visualizations simulate dashboard-style insights, often used in business analytics. Pie charts and donut charts are useful for showing part-to-whole relationships. Scatter plots and boxplots support statistical analysis such as correlation and distribution. Waterfall charts are often applied in financial reporting to track cumulative effects. By including this variety, Text2Vis ensures robust evaluation of models across diverse analytical scenarios.

A.2 Data Science Question Taxonomy

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.2.1 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.2.2 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.2.3 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.2.4 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.3 Query Types & Illustrative 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.

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.

A.4 Zero/Few-Shot and Retrieval-Augmented Prompting

In our experiments, we employed three distinct setups for the direct inference approach to comprehensively evaluate model performance. Below, we provide detailed descriptions of each approach used:

(i) Zero-Shot Prompting: In the zero-shot prompting scenario, the model receives only the target query and the corresponding data table directly, without any preceding examples or additional context. The prompt explicitly instructs the model to generate the response in JSON format. This setup assesses the model's inherent ability to interpret and respond to queries based solely on its pre-trained knowledge and instruction-following capabilities.

(ii) 3-Shot Prompting: In the 3-shot prompting approach, the model is initially provided with three randomly selected examples, each consisting of a natural language query, the relevant data table, and the correct annotated answer. These three examples precede the actual target query and its associated data table. The purpose of this setup is to evaluate the model's capability to leverage few-shot learning, enabling it to infer and adapt patterns from a minimal number of illustrative examples before generating the response for the target query.

(iii) Retrieval-Augmented Generation (RAG) + 3-Shot Prompting: The RAG + 3-shot prompting approach further enhances the 3-shot method by dynamically retrieving relevant examples from the dataset based on semantic similarity to the target query. Specifically, we employed the Sentence-BERT model (`all-MiniLM-L6-v2`) to encode queries into embedding vectors. We then computed cosine similarity scores between the target query embedding and all available query embeddings within the dataset. The top three most semantically similar queries and their corresponding annotated answers and data tables were retrieved. These retrieved examples served as the few-shot context preceding the target query, providing contextually relevant information intended to improve the model's comprehension and response accuracy.

A.5 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.

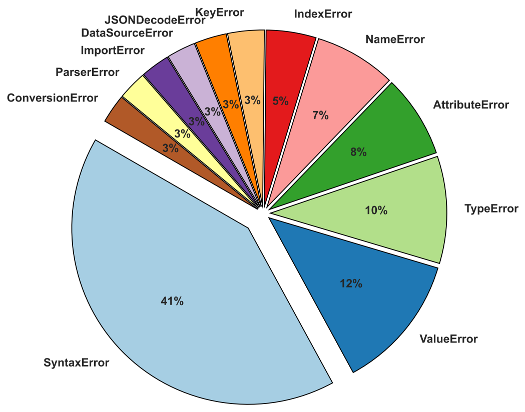


Figure 5: Error type distribution with square root transformation applied to prevent the SyntaxError category from dominating the chart.

Model	Code Exec. Success (%)	Answer Match (%)	Visual Clarity Readability	Chart Correctness	Final Pass Rate (%)
GPT-4o	**87**	**40**	**3.38**	**3.10**	**24**
Gemini-1.5-Flash	83	34	3.35	2.69	15
CodeLlama-7b	60	10	1.97	1.23	1
CodeLlama-13b	52	14	1.90	0.90	3
CodeLlama-34b	39	21	0.92	0.60	3
Llama-3.1-8B	72	24	1.74	1.27	6
Mistral-7B	39	22	1.45	1.10	4
Qwen2.5-7B	80	28	2.86	2.40	12
Qwen2.5-Coder-7B	31	22	1.30	1.08	3
DeepSeek-Coder-V2-Lite	75	21	2.97	2.10	8
DeepSeek-R1-Distill-Llama-8B	35	36	1.70	1.21	8

Table 7: 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. Gemini 1.5 Pro as judge

A.6 Experimental Setup & Inference Protocol

A.6.1 Model Parameters

We used the following fixed seed and controlled decoding parameters for output generation, while all other settings were retained as default values provided by each model's API.

- **Seed:** 42
- **Temperature:** 0.1
- **Top-p:** 1.0
- **Max New Tokens (Model Response Generation):** 2048

A.6.2 Inference Protocols

We use three setups:

- **Zero-Shot Prompting:** Query + table only, no examples.
- **Few-Shot Prompting (3-shot):** Three examples precede the query.
- **RAG + 3-Shot Prompting:** Three most semantically similar examples retrieved via SBERT, then appended before the query.

A.6.3 Prompt Construction Summary

For consistency, all prompting strategies use the same system template structure and JSON formatting requirements. Full prompt templates are listed in Appendix A.8.

A.7 Extended Evaluation & Robustness

A.7.1 Human Evaluation Setup

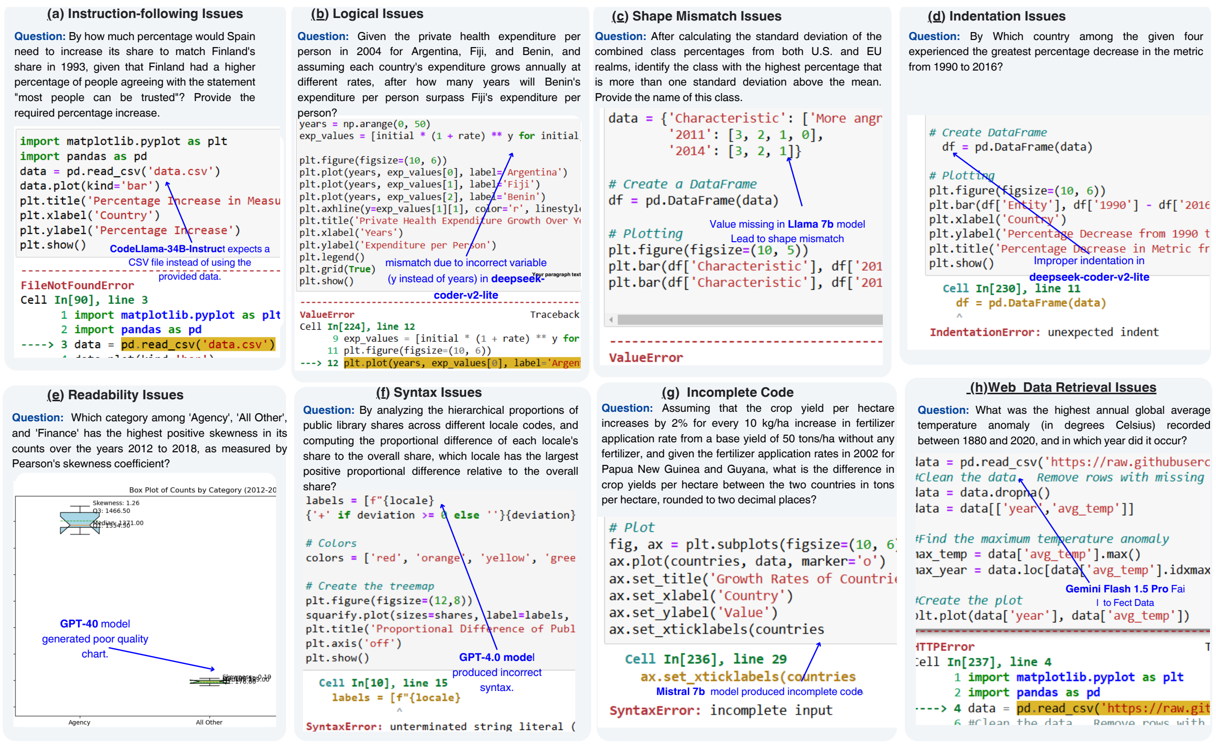
The 236 samples were first annotated by a primary annotator—an expert in data science and visualization. The annotations were then independently verified by a second annotator, with any disagreements resolved through discussion. Each model output was rated for answer correctness, chart readability, and visual accuracy using the same scoring rubric as in automated evaluation.

To quantify agreement between human and automated scores, we computed Pearson (r) and Spearman (ρ) correlations across all three core dimensions, (see Table 9). The average Pearson correlations were 0.87 for *Answer Match*, 0.88 for *Clarity & Readability*, and 0.867 for *Chart Correctness*. The corresponding Spearman correlations were 0.87, 0.83, and 0.857, respectively. Additionally, we computed Cohen's Kappa for agreement on final pass rate, obtaining an average $\kappa = 0.78$—indicating substantial alignment.

These findings support the alignment of automated and human judgments, reinforcing the robustness of our LLM-based evaluation framework.

A.7.2 LLM Judge Agreement

To evaluate consistency across LLM judges, we compared GPT-4o and Gemini 1.5 Pro on all benchmarked models. As shown in Table 8, the two judges exhibit strong agreement across all evaluation dimensions. The average Pearson correlations were 0.953 for *Answer Match*, 0.907 for *Clarity & Readability*, and 0.828 for *Chart Correctness*. Similarly, the average Spearman correlations were 0.953, 0.868, and 0.857 for the same dimensions, respectively. Gemini also showed similarly strong correlations with human scores (Table 10), with average Pearson correlations of 0.91, 0.863, and 0.813, and Spearman correlations of 0.91, 0.823, and 0.807 for *Answer Match*, *Clarity & Readability*, and *Chart Correctness*, respectively. These results confirm the consistency and robustness of our automated evaluation framework across state-of-the-art LLMs.



(a) Instruction-following Issues

Question: By how much percentage would Spain need to increase its share to match Finland's share in 1993, given that Finland had a higher percentage of people agreeing with the statement "most people can be trusted"? Provide the required percentage increase.

```
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv('data.csv')
data.plot(kind='bar')
plt.title('Percentage Increase in Measu
plt.xlabel('Country')
plt.ylabel('Percentage Increase')
plt.show()
-----------------------------------------
FileNotFoundError
Cell In[90], line 3
      1 import matplotlib.pyplot as plt
      2 import pandas as pd
----> 3 data = pd.read_csv('data.csv')
```

CodeLlama-34B-Instruct expects a CSV file instead of using the provided data.

(b) Logical Issues

Question: Given the private health expenditure per person in 2004 for Argentina, Fiji, and Benin, and assuming each country's expenditure grows annually at different rates, after how many years will Benin's expenditure per person surpass Fiji's expenditure per person?

```
years = np.arange(0, 50)
exp_values = [initial * (1 + rate) ** y for initial

plt.figure(figsize=(10, 6))
plt.plot(years, exp_values[0], label='Argentina')
plt.plot(years, exp_values[1], label='Fiji')
plt.plot(years, exp_values[2], label='Benin')
plt.axhline(y=exp_values[1][1], color='r', linestyl
plt.title('Private Health Expenditure Growth Over Ye
plt.xlabel('Years')
plt.ylabel('Expenditure per Person')
plt.legend()
plt.grid(True)
plt.show()
-----------------------------------------
ValueError                          Traceback
Cell In[224], line 12
      9 exp_values = [initial * (1 + rate) ** y for
     11 plt.figure(figsize=(10, 6))
---> 12 plt.plot(years, exp_values[0], label='Argen
```

mismatch due to incorrect variable (y instead of years) in **deepseek-coder-v2-lite**

Your paragraph text

(c) Shape Mismatch Issues

Question: After calculating the standard deviation of the combined class percentages from both U.S. and EU realms, identify the class with the highest percentage that is more than one standard deviation above the mean. Provide the name of this class.

```
data = {'Characteristic': ['More angr
        '2011': [3, 2, 1, 0],
        '2014': [3, 2, 1]}

# Create a DataFrame
df = pd.DataFrame(data)

# Plotting
plt.figure(figsize=(10, 5))
plt.bar(df['Characteristic'], df['201
plt.bar(df['Characteristic'], df['201
-----------------------------------------
ValueError
```

Value missing in **Llama 7b** model Lead to shape mismatch

(d) Indentation Issues

Question: By Which country among the given four experienced the greatest percentage decrease in the metric from 1990 to 2016?

```
# Create DataFrame
  df = pd.DataFrame(data)

# Plotting
plt.figure(figsize=(10, 6))
plt.bar(df['Entity'], df['1990'] - df['2016
plt.xlabel('Country')
plt.ylabel('Percentage Decrease from 1990 t
plt.title('Percentage Decrease in Metric fr
plt.show()

  Cell In[230], line 11
    df = pd.DataFrame(data)
    ^
IndentationError: unexpected indent
```

Improper indentation in **deepseek-coder-v2-lite**

(e) Readability Issues

Question: Which category among 'Agency', 'All Other', and 'Finance' has the highest positive skewness in its counts over the years 2012 to 2018, as measured by Pearson's skewness coefficient?

GPT-40 model generated poor quality chart.

(f) Syntax Issues

Question: By analyzing the hierarchical proportions of public library shares across different locale codes, and computing the proportional difference of each locale's share to the overall share, which locale has the largest positive proportional difference relative to the overall share?

```
labels = [f"{locale}
{'+' if deviation >= 0 else ''}{deviation}

# Colors
colors = ['red', 'orange', 'yellow', 'gree

# Create the treemap
plt.figure(figsize=(12,8))
squarify.plot(sizes=shares, label=labels,
plt.title('Proportional Difference of Publ
plt.axis('off')
plt.show()

  Cell In[10], line 15
    labels = [f"{locale}
              ^
SyntaxError: unterminated string literal (
```

GPT-4.0 model produced incorrect syntax.

(g) Incomplete Code

Question: Assuming that the crop yield per hectare increases by 2% for every 10 kg/ha increase in fertilizer application rate from a base yield of 50 tons/ha without any fertilizer, and given the fertilizer application rates in 2002 for Papua New Guinea and Guyana, what is the difference in crop yields per hectare between the two countries in tons per hectare, rounded to two decimal places?

```
# Plot
fig, ax = plt.subplots(figsize=(10, 6)
ax.plot(countries, data, marker='o')
ax.set_title('Growth Rates of Countrie
ax.set_xlabel('Country')
ax.set_ylabel('Value')
ax.set_xticklabels(countries

  Cell In[236], line 29
    ax.set_xticklabels(countries
SyntaxError: incomplete input
```

Mistral 7b model produced incomplete code

(h)Web Data Retrieval Issues

Question: What was the highest annual global average temperature anomaly (in degrees Celsius) recorded between 1880 and 2020, and in which year did it occur?

```
lata = pd.read_csv('https://raw.githubuserc
#Clean the data   Remove rows with missing
lata = data.dropna()
lata = data[['year','avg_temp']]

#Find the maximum temperature anomaly
nax_temp = data['avg_temp'].max()
nax_year = data.loc[data['avg_temp'].idxmax

#Create the plot
plt.plot(data['year'], data['avg_temp'])
-----------------------------------------
HTTPError                                 T
Cell In[237], line 4
      1 import matplotlib.pyplot as plt
      2 import pandas as pd
----> 4 data = pd.read_csv('https://raw.git
      6 #Clean the data   Remove rows with
```

Gemini Flash 1.5 Pro Fai l to Fect Data

Figure 6: Common errors in Data Visualization generation.

A.7.3 Evaluation Repeatability

To test evaluation stability, we re-evaluated the same stratified 236-sample subset five times using GPT-4o with fixed prompts and identical sampling settings. The results showed minimal variation across runs, with standard deviations of ±0.0 for *Answer Match*, ±0.015 for *Readability*, ±0.023 for *Chart Correctness*, and ±0.003 for the *Final Score*. Notably, 97.49% of the samples maintained the same final pass rate across all runs, confirming the reliability and consistency of our evaluation framework.

A.7.4 Disjoint vs Joint Generation (Ablation Study)

We also ablated the task formulation by comparing *text-only*, *visual-only*, and *joint (both)* generation settings using GPT-4o. As shown in Table 12, the joint setup achieved similar performance to the disjoint settings across answer accuracy and chart quality metrics. These results suggest that combining answer and visualization generation does not degrade performance, supporting the feasibility of our unified task formulation.

Setup	Code Exec. Success (%)	Answer Match (%)	Visual Clarity Readability	Chart Correctness
Text only	–	44	–	–
Visual only	89	–	3.61	3.30
Both	**87**	**42**	**3.45**	**3.15**

Table 12: Ablation study comparing GPT-4o performance in text-only, visual-only, and joint (both) generation settings.

A.7.5 Runtime & Cost Analysis

To demonstrate scalability, we split the full 1,985-sample set into ten parallel GPT-4o API calls, completing all evaluations in just 5 minutes at a total cost of approximately $2.0—compared to an estimated 33 human-hours (assuming 1 minute per sample)—yielding a significant reduction in annotation time and cost.

A.8 Prompt Templates

We present the key prompt templates used throughout the Text2Vis. While we developed multiple query generation prompts to support various chart types and analytical tasks, representative examples are provided here.

A.8.1 Query Generation Prompts

To ensure diversity and realism in question types, we employ multiple prompts for structured query generation. A few representative examples include:

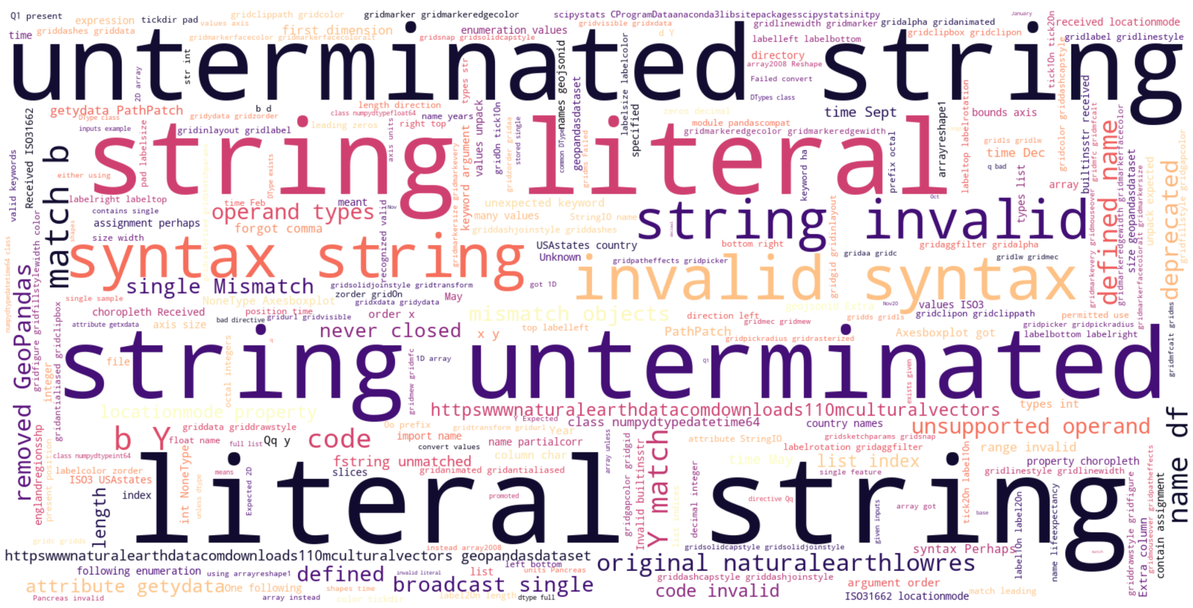


Figure 7: Most Frequent Words in Error Messages Across All Evaluated Models.

- **Conversational Prompt:** Used to generate interdependent multi-turn queries over a dataset. See Table 13.
- **Histogram Prompt:** Generates synthetic frequency-based data and a complex reasoning question best answered with a histogram. See Table 14.
- **Scatter Plot Prompt:** Used when the dataset contains multiple numerical columns, enabling queries about relationships and outliers. See Table 15.

Additional prompt types used during dataset creation include those for pie charts, bar charts, multi-chart dashboard layouts, and unanswerable question construction.

A.8.2 Inference Prompts

These prompts guide how models generate answers and visualizations, ensuring consistency across inference settings:

- **Model Response Prompt:** A unified system prompt applied across zero-shot, few-shot, and RAG settings. It defines task instructions, output format (JSON), and required libraries (Matplotlib/Seaborn). See Table 16.
- **Agentic Refinement Prompt:** Used in the self-refinement loop to assess and correct model outputs based on answer and chart quality. See Table 17.

A.8.3 Evaluation Prompts

We adopt LLM-based evaluation prompts to assess answer and chart correctness and chart quality. Theis evaluates answer match, chart readability, and chart correctness using detailed rubrics. See Table 18.

A.8.4 Complexity Prompt

This prompt is used to classify each query as `Simple`, `Medium`, `Hard`, or `Extra Hard`. See Table 19.

Model	**Pearson**			**Spearman**		
	Answer Match	**Clarity & Readability**	**Chart Correctness**	**Answer Match**	**Clarity & Readability**	**Chart Correctness**
GPT-4o	0.93	0.83	0.90	0.93	0.71	0.87
Gemini-1.5-Flash	0.98	0.84	0.88	0.98	0.71	0.87
CodeLlama-7B	0.97	0.77	0.63	0.97	0.74	0.67
CodeLlama-13B	0.97	0.95	0.77	0.97	0.95	0.83
CodeLlama-34B	0.97	0.97	0.84	0.97	0.99	0.92
Llama-3.1-8B	0.96	0.96	0.83	0.96	0.97	0.91
Mistral-7B	0.94	0.97	0.86	0.94	0.98	0.92
Qwen-2.5-7B	0.95	0.95	0.84	0.95	0.85	0.86
Qwen-2.5-Coder-7B	0.95	0.96	0.88	0.95	0.97	0.87
DeepSeek-Coder-V2-Lite	0.95	0.94	0.79	0.95	0.83	0.83
DeepSeek-R1-Distill-Llama-8B	0.91	0.84	0.89	0.91	0.85	0.88

Table 8: Pearson and Spearman correlations between the **GPT-4o judge** and the **Gemini 1.5 Pro judge** across three evaluation dimensions: Answer Match, Clarity & Readability, and Chart Correctness.

Model	**Pearson**			**Spearman**		
	Answer Match	**Clarity & Readability**	**Chart Correctness**	**Answer Match**	**Clarity & Readability**	**Chart Correctness**
GPT-4o	0.85	0.84	0.81	0.85	0.71	0.78
Llama-3.1-8B	0.89	0.91	0.87	0.89	0.91	0.89
Qwen2.5-7B	0.87	0.89	0.92	0.87	0.87	0.90

Table 9: Pearson and Spearman correlations between human evaluation and GPT-4o-based evaluation across three core dimensions: Answer Match, Clarity and Readability, and Chart Correctness.

Model	**Pearson**			**Spearman**		
	Answer Match	**Clarity & Readability**	**Chart Correctness**	**Answer Match**	**Clarity & Readability**	**Chart Correctness**
GPT-4o	0.92	0.86	0.68	0.92	0.72	0.66
Llama-3.1-8B	0.91	0.89	0.83	0.91	0.90	0.91
Qwen2.5-7B	0.90	0.84	0.93	0.90	0.85	0.85

Table 10: Pearson and Spearman correlations between human evaluation and Gemini 1.5 Pro-based evaluation across three core dimensions: Answer Match, Clarity and Readability, and Chart Correctness.

Model	**Closed / Open-Ended**	**Single Query / Conversational**	**Data Given / Web-data Retrieval**	**Single / Multi-Chart**	**Answerable / Unanswerable**
GPT-4o	**24 / 26**	**20 / 50**	**26** / 8	**26 / 26**	**29** / 3
Gemini 1.5 Flash	17 / 19	13 / 33	18 / **17**	17 / 17	19 / 6
CodeLlama-7b-hf	2 / 1	2 / 1	0 / 2	2 / 3	2 / 0
CodeLlama-13b-hf	5 / 0	3 / 8	4 / 0	4 / 4	5 / 0
CodeLlama-34b-hf	5 / 2	2 / 13	4 / 0	5 / 1	5 / 0
Llama-3.1-8B	7 / 4	5 / 14	7 / 0	7 / 5	7 / 0
Mistral-7B	5 / 9	4 / 12	4 / 6	6 / 4	6 / 1
Qwen2.5-7B	3 / 15	11 / 22	14 / 0	14 / 6	14 / **7**
Qwen2.5-Coder-7B	4 / 4	2 / 11	14 / 0	4 / 3	4 / 0
DeepSeek-Coder V2-Lite	10 / 9	8 / 21	10 / 2	10 / 9	11 / 4
DeepSeek-R1-Distill-Llama-8B	6 / 10	6 / 10	7 / 2	7 / 5	7 / 2

Table 11: Performance breakdown (Final Pass Rate in percentages) 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 13: Prompt Templates for Conversational Query Generation.

Category	Prompt Template

Synthetic Data Table and Histogram Question Generation	I want you to perform the following tasks:
	
	1. **Generate a Synthetic Data Table**:
	- Randomly choose **one** domain from:
	Healthcare, Technology, Finance, Marketing, Retail, Education, Sports, Energy, Logistics, or any other relevant field.
	- Clearly specify the selected **domain** in the output.
	- Ensure **domain-specific numerical scales** (e.g., revenue in 100k+, sales in thousands, ratings 1.0-5.0, percentages 0-100%).
	- The dataset should include **5-8 rows** and **1 numerical column** with **frequency-based data** to represent a distribution.
	- The structure must **naturally lead to a complex data science question requiring a Histogram** to analyze **distribution, frequency, or variability**.
	- Return the dataset as a **valid JSON object** with column names as keys and values as lists.
	
	2. **Generate a Single, Very Complex Data Science Question**:
	- The question must require **multi-step reasoning** and **deep analysis** related to **data distributions, frequency analysis, or variability**.
	- The question should involve **detecting patterns, finding skewness, assessing data spread, or identifying peaks and outliers**.
	- Ensure the question requires **multi-step computations** such as **mean, median, standard deviation, percentiles, or comparisons of histogram bins** before deriving the final answer.
	- **Do not modify the original dataset in any way (e.g., do not add missing values or create new data points).**
	- The question should be best answered using a **Histogram**.
	
	3. **Provide a Short Answer**:
	- The answer must be **exactly one word or one number**.
	
	4. **Output Python Code Using Matplotlib**:
	- The code should create a **Histogram** that effectively visualizes the dataset and addresses the generated question.
	- **Data should be binned appropriately to represent the distribution**.
	- **Ensure clear labeling of axes and meaningful annotations** to highlight key insights.
	- **Use colors, bin adjustments, and density plots if necessary** to enhance clarity.
	- **Must include text annotations** to enhance the chart's clarity and insight.
	- You may use **pandas** for data handling if necessary.
	
	5. **Include a Text Summary**:
	- Provide a concise summary.
	- Highlight the **main insight** derived from the visualization.
	
	6. **Provide Metadata**:
	- **Domain**: The selected domain.
	- **ChartType**: Must be `"Histogram"`.
	- **xlabel**: The numerical variable representing the bins.
	- **ylabel**: The frequency count of values.
	
	Output Requirements:
	- Return **all** the above information in a **valid JSON format** without any additional text or commentary outside the JSON object.
	- Follow this exact JSON structure:
	
	`{`
	`"Domain": "...",`
	`"GeneratedDataTable": { "Value": [...] },`
	`"Question": "...",`
	`"Answer": "...",`
	`"Code": "...",`
	`"TextSummary": "...",`
	`"ChartType": "Histogram",`
	`"xlabel": "...",`
	`"ylabel": "..."`
	`}`

Table 14: Prompt Template for Synthetic Data Table and Histogram Question 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 15: 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. **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": "..." }

Table 16: 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. **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": "..." }

Table 17: 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 18: Prompt Template for Evaluating Results Using the GPT-4.0 Model.

Category	Prompt Template
Complexity Classification	Prompt: You are given a data science question based on a table. Your task is to classify the complexity of the question into one of the following four categories: • **Simple**: The answer can be directly retrieved by locating a single value or label, with no calculation required. • **Medium**: The question requires one or two reasoning steps, such as comparing values, calculating a difference or percentage, or sorting a small set of entries. • **Hard**: The question involves multiple steps of reasoning—combining comparisons, aggregations, or filtering across rows or categories. It may require intermediate calculations to arrive at the answer. • **Extra Hard**: The question demands complex, multi-step reasoning such as identifying trends, interpreting grouped patterns, performing advanced aggregations. It may also involve retrieving external information from the web to fully answer the query. Based on the criteria above, classify the question using one of the following labels: `Simple`, `Medium`, `Hard`, or `Extra Hard`. Provide **only the label** as your final output.

Table 19: Prompt Template for Question Complexity Classification