Text2Chart31: Instruction Tuning for Chart Generation with Automatic Feedback

Fatemeh Pesaran zadeh[1] **Juyeon Kim**[2*] **Jin-Hwa Kim**[1,3] **Gunhee Kim**[1†]
[1]Seoul National University, [2]KAIST AI, [3]NAVER AI Lab
fatemehpesaran@vision.snu.ac.kr, juyeonkim@kaist.ac.kr,
j1nhwa.kim@navercorp.com, gunhee@snu.ac.kr

Abstract

Large language models (LLMs) have demonstrated strong capabilities across various language tasks, notably through instruction-tuning methods. However, LLMs face challenges in visualizing complex, real-world data through charts and plots. Firstly, existing datasets rarely cover a full range of chart types, such as 3D, volumetric, and gridded charts. Secondly, supervised fine-tuning methods do not fully leverage the intricate relationships within rich datasets, including text, code, and figures. To address these challenges, we propose a hierarchical pipeline and a new dataset for chart generation. Our dataset, Text2Chart31, includes 31 unique plot types referring to the Matplotlib library, with 11.1K tuples of descriptions, code, data tables, and plots. Moreover, we introduce a reinforcement learning-based instruction tuning technique for chart generation tasks without requiring human feedback. Our experiments show that this approach significantly enhances the model performance, enabling smaller models to outperform larger open-source models and be comparable to state-of-the-art proprietary models in data visualization tasks. We make the code and dataset available at https://github.com/fatemehpesaran310/Text2Chart31.

1 Introduction

Recently, a range of NLP tasks has been addressed by leveraging the remarkable ability of Large Language Models (LLMs). This advancement has been possible largely through the process of instruction-tuning (Ouyang et al., 2022; Yoo et al., 2024), which fine-tunes LLMs to rely on intuitive natural language instructions and skillfully solve intricate tasks, encompassing fields like question answering (Sanh et al., 2022; Liu and Low, 2023), summarizing (Goyal et al., 2023; Fetahu et al., 2023), and

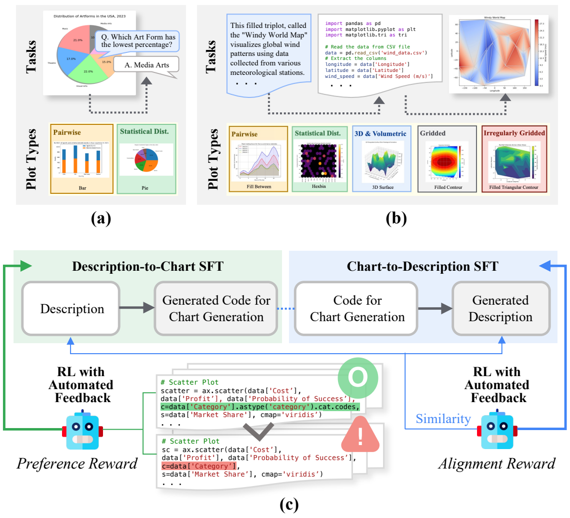


Figure 1: Illustration of the contributions of our method. (a): Existing datasets rarely cover a full range of chart types and primarily focus on QA tasks rather than chart generation. (b): Our dataset focuses on chart generation tasks and covers 31 unique plot types with tuples that combine descriptions, code, data tables, intermediate reasoning steps, and plots. (c): We further adopt RL-based instruction tuning method that leverage automated feedback and cycle consistency.

sentiment analysis (Varia et al., 2023). However, available LLMs continue to suffer from the difficult tasks of visualizing complex, fact-based, real-world data through charts and plots, mainly because of two challenges.

Firstly, the current datasets (Methani et al., 2020; Masry et al., 2022; Kahou et al., 2018; Zhu et al., 2021; Kantharaj et al., 2022; Han et al., 2023) primarily focus on QA in the chart domain rather than chart generation, and they rarely cover a full range of chart types and their varied applications. Several chart forms like 3D, volumetric, gridded, and irregularly gridded remain largely unexplored or insufficiently studied. These forms are important for evaluating the capabilities of LLMs in under-

∗ Work done as an intern at Seoul National University
† Corresponding author.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 11459–11480
November 12-16, 2024 ©2024 Association for Computational Linguistics

standing multidimensional data, spatial data, and vector field data. Developing such instructional datasets typically entails significant expenses due to the complex nature of text-to-chart processes, incorporating various data components such as text, code, and data tables. This complexity, along with the lack of specific online sources containing these plot types, makes their collection difficult and time-consuming. It necessitates human expert intervention to ensure quality, which drives up costs.

Secondly, existing instruction-tuning methods based on supervised fine-tuning do not fully utilize the potential of rich datasets; for example, chart data include multiple components like text descriptions, code, and figures. Supervised fine-tuning struggles to effectively extract and leverage all the intricate information and relationships within these components, leading to suboptimal performance.

To address the first challenge, we propose a novel hierarchical pipeline for chart generation by leveraging the advanced linguistic skills of GPT-3.5-turbo (Ouyang et al., 2022) and code generation and data analysis capabilities of GPT-4-0613 (OpenAI, 2023). We contribute a dataset encompassing 31 unique plot types from the Matplotlib library (Hunter, 2007), featuring 11.1K tuples that combine descriptions, code, data tables, and plots, covering a wide range of use cases. Our pipeline is structured into the following steps: topic generation, description creation, code production, data table and reasoning step formulation, and cycle consistency verification. This approach reduces biases towards common topics or plot types, and ensures consistent and accurate generation of multiple data elements. By minimizing the human supervision in our proposed pipeline, we can generate a high-quality large-scale dataset that includes comprehensive descriptions, codes, data tables, reasoning steps, and illustrated graphs.

We further propose a novel reinforcement learning-based instruction tuning technique to address the second challenge. This method is tailored for chart generation tasks without costly human feedback. We propose two different reward functions: the preference reward and alignment reward. For the preference reward, we construct a preference dataset from the supervised fine-tuned model's output and the ground truth code. For the alignment reward, we optimize the model to increase the similarity between ground truth description and regenerated description from the code, exploiting the cycle consistency between code and description. We jointly optimize two sequential policy models using the PPO (Schulman et al., 2017).

Finally, we make the following contributions:

- We develop a novel dataset generation pipeline that populates data samples and filters out the low-quality ones, exploiting the cycle consistency in the task. This approach is scalable to increase the volume of data as needed.
- We introduce the Text2Chart31 dataset, comprising 31 plot types with 11.1K tuples that combine descriptions, code, data tables, intermediate reasoning steps, and plots, covering a wide range of use cases.
- We introduce an RL-based instruction tuning method that utilizes novel reward functions that leverage automated feedback and cycle consistency. The experiments demonstrate that our fine-tuned models outperform state-of-the-art open and closed-source models on data visualization tasks. To the best of our knowledge, this is the first work to adopt an RL-based instruction tuning approach for the chart generation task.

2 Text2Chart31 Dataset

Our newly contributed Text2Chart31 dataset supports 31 plot types based on Matplotlib with 11.1K data points. We outline its key characteristics in Table 1 comparing with existing datasets in the data visualization domain. The Text2Chart31 dataset $\mathcal{D}$ consists of 11,128 data points, each of which contains a tuple of (x, c, d, r, y): a textual plot description (x), its corresponding code (c), the resulting plots (y). For 8,166 data points, we additionally include a raw data table (d) and intermediate reasoning steps (r) to generate descriptions.

For the dataset, we develop a hierarchical plot generation pipeline leveraging GPT-3.5-turbo and GPT-4. Despite their impressive capabilities for text and code generation, collecting high-quality data points is challenging for two primary reasons: (1) GPT-3.5-turbo exhibits bias towards particular topics or narrow plot types that are commonly represented in its training data, and (2) the text-to-chart data involves multiple data elements including descriptions, code, and data tables, making it difficult to generate accurate and consistent data points in a single step. Consequently, we claim that a hierarchical approach is essential for producing higher-quality chart-generation data points. This pipeline is illustrated in Figure 2.

2.1 Task Definition

Our benchmark is designed to evaluate three tasks. (1) *Description-to-Chart:* Given a plot description x, an algorithm generates its corresponding code c that creates a chart by the Matplotlib library[1](Hunter, 2007). (2) *Raw Data-to-Chart:* When provided with only a raw data table d, the algorithm generates intermediate reasoning steps r that analyze the raw data and then generates a description d for the most suitable plot type based on the characteristics of the data. (3) *Code-to-Description:* Given the code c for a plot, the model generates a detailed description x of the plot.

2.2 Dataset Construction Pipeline

Our pipeline initiates by generating a topic from which a description x is derived. To ensure both diversity of the topic and alignment with the intended plot type, each topic is filtered before proceeding to the next step. We additionally generate code c, raw data table d and intermediate reasoning step r corresponding to the description. Lastly, we use the cycle-consistency verification to ensure the high quality of the data points. Please refer to Appendix C for the detailed process with examples.

Topic generation. We generate distinct topic pools for five different plot categories: pairwise, statistical, gridded, irregularly gridded, and 3D/volumetric data. To maintain diversity within each topic pool, we include only topics with low similarity scores compared to those already being presented. To assess similarity, the ROUGE-L metric (Lin, 2004) is employed as a common practice from previous studies (Wang et al., 2023b).

Description generation. For each plot type, we start by manually writing 5 to 10 descriptions as seed points that contain all the necessary information for a plot to be illustrated. To generate a description (x), we randomly sample two descriptions and pair them with a topic from the topic pool. This assembled data is prompted into GPT-3.5-turbo, which generates a similar format plot description for the sampled topic. We remove the topic from the pool after a new description is generated to uphold the diversity. Inspired by the studies on the reasoning capabilities of LLMs (Wei et al., 2023; Kojima et al., 2023; Wang et al., 2023a), we instruct GPT-4 to *self-evaluate* the generated descriptions for quality control. This step is crucial to exclude any incompatible instructions that can lead to the creation of unsuitable plots, thereby avoiding computational waste.

Code generation. We input descriptions into GPT-4, which is instructed to generate Python code for the Matplotlib library. This code aims to visualize the described plot. We add the generated code (c) to the dataset only if it successfully generates the corresponding plot (y) without a runtime error.

Data table and reasoning step generation. For plots derived from data files in $\mathcal{D}$, GPT-4 is prompted to generate either a raw data table d or Python code that can generate the data table. 3D volumetric, gridded, and irregularly gridded plots often require specific patterns or mathematical relations between variables; therefore, code is created and executed to generate the data table instead of directly generating it. We further generate intermediate reasoning steps r using GPT-4, which is instructed to analyze the characteristics of the data and CSV file, explore possible plot types, determine the most suitable plot type, and consider additional aspects of the description. This process results in data points (x, c, d, r, y).

Cycle-Consistency verification. We argue that given the complex and fact-based nature of text-to-chart datasets, employing human evaluation to check the quality of generated data points is inefficient. To this end, we propose an AI-assisted method using cycle consistency, to assure the quality of the data point. This process involves regenerating an instruction that describes the plot from the generated code and comparing it against the original one. We keep the data only if the regenerated description closely aligns with the original one based on pre-defined criteria, indicating the high quality of the data. We provide further details on the cycle consistency method in Appendix D.

2.3 Analysis of Text2Chart31 Dataset

As shown in Table 1, we can effectively balance the data points per plot type with equal distribution in the dataset, which is quantified by the Shanon Diversity metric (Friedman and Dieng, 2023). Shannon Diversity is computed through $H = -\sum_{i=1}^{S} p_i \log(p_i)$, where S is the total number of classes in the dataset, and p_i is the proportion of instances belonging to the i-th class. Our Text2Chart31 dataset achieve the highest score of 0.981. Figure 6 in Appendix shows a detailed com-

[1]We use Matplotlib 3.8 version.

Dataset	# Data				# Plot Type				Quality Analysis	
	Figures	Instruction Tuning	Description to Code	Raw Data to Description	Pairwise& Stat. Dist.	(Irregularly) Gridded	3D & Volumetric	Total	Dataset Balance†	Content Diversity‡
PlotQA	224.3K	28.9M	✗	✗	3	✗	✗	3	0.786	0.038
ChartQA	21.9K	32.7K	✗	✗	3	✗	✗	2	0.422	-
FigureQA	180K	2.3M	✗	✗	4	✗	✗	4	0.960	-
Unichart	611K	7M	✗	✗	3	✗	✗	3	0.821	0.157
AutoChart	10.2K	23.5K	✗	✗	3	✗	✗	3	0.978	0.027
Chart-to-Text	44K	44K	✗	✗	5	1	✗	6	0.327	0.421
ChartLlama	11K	160K	7.8K	✗	8	2	✗	10	0.738	-
ChartX	6K	48K	6K	✗	13	2	1	16	0.953	0.534
Text2Chart31	11.1K	19.3K	**11.1K**	**8.2K**	**16**	**10**	**5**	**31**	0.980	0.674
Text2Chart31-v2§	28.2K	50.2K	**28.2K**	**22K**	**16**	**10**	**5**	**31**	**0.993**	**0.696**

Table 1: Comparison with other chart datasets: PlotQA (Methani et al., 2020), ChartQA (Masry et al., 2022), FigureQA (Kahou et al., 2018), Unichart (Masry et al., 2023), Autochart (Zhu et al., 2021), Chart-to-Text (Kantharaj et al., 2022), ChartLlama (Han et al., 2023), and ChartX (Xia et al., 2024). We report the total number of figures and instruction tuning data, including the tasks like QA, summarization, code generation, and plot recommendation. Additionally, we provide the number of data points for the tasks of *Description to Chart* and *Raw Data to Chart*, specifying data for Description to Code (visualization code) and Raw Data Analysis to Description (analyzing raw data to generate a corresponding description). We also detail the number of plot types in each dataset. †We measure the dataset balance score using the Shannon Diversity Index (Friedman and Dieng, 2023). ‡We evaluate the content diversity by calculating average distinct n-grams (n from 1 to 5) (Li et al., 2016). For PlotQA, Chart-to-Text, and AutoChart, we use chart titles, captions, and descriptions to evaluate content diversity, respectively. For Unichart and ChartX, we use summarizations. ChartQA and FigureQA are excluded due to lack of descriptions/titles, and ChartLlama is private. Finally, content diversity of Text2Chart31 is computed using the topics. § Text2Chart31-v2 is constructed and published at the camera ready version of the paper, and the experiment results in this paper is conducted with Text2Chart31.

parison of the distribution per chart type between datasets using pie charts. We further evaluate the content diversity of datasets via Distinct-n score (Li et al., 2016). Our dataset achieves a score of 0.674, indicating that our pipeline effectively reassures the diversity of topics.

3 Instruction Tuning Approach

We discuss our proposed instruction tuning methods for fine-tuning LLMs to tackle the three data visualization tasks: (1) Description-to-Chart, (2) Raw-Data-to-Chart, and (3) Code-to-Description, using the Text2Chart31 dataset. We respectively denote three specialized models for the three tasks: π_{θ_1}, π_{θ_2}, and π_{θ_3}. We train these models with two phases: supervised fine-tuning (SFT), followed by reinforcement learning (RL) with two types of reward that are specifically tailored to improve chart generation performance. Initially, all three tasks undergo supervised fine-tuning. Afterward, using PPO algorithm (Schulman et al., 2017), we jointly optimize π_{θ_1} with the preference reward and π_{θ_3} with the alignment reward that ensures cycle consistency and coherence of outputs. Algorithm 1 summarizes the overall procedure.

3.1 Supervised Fine-tuning

We perform supervised fine-tuning of π_{θ_1}, π_{θ_2}, and π_{θ_3} using the cross-entropy loss with the Text2Chart31 dataset. For Task 1, the model π_{θ_1} maximizes the probability of outputting the ground truth code for a given description by minimizing cross-entropy loss in the Line 3 of Algorithm 1. For Task 2, we design the model π_{θ_2} to generate descriptions from raw data in two stages. First, the model generates a reasoning step r from the raw data d, which involves analyzing data characteristics and determining the appropriate plot type. Then, the model is fine-tuned to generate the description x using the data and the reasoning step as in the Line 4. Lastly, we fine-tune the model π_{θ_3} for Task 3 to maximize the probability of predicting the ground truth description for a given visualization code as in the Line 5 of Algorithm 1.

3.2 RL via Automatic Feedback

We design two reward functions, which are the preference reward and the alignment reward, specifically tailored for the chart generation task. It is

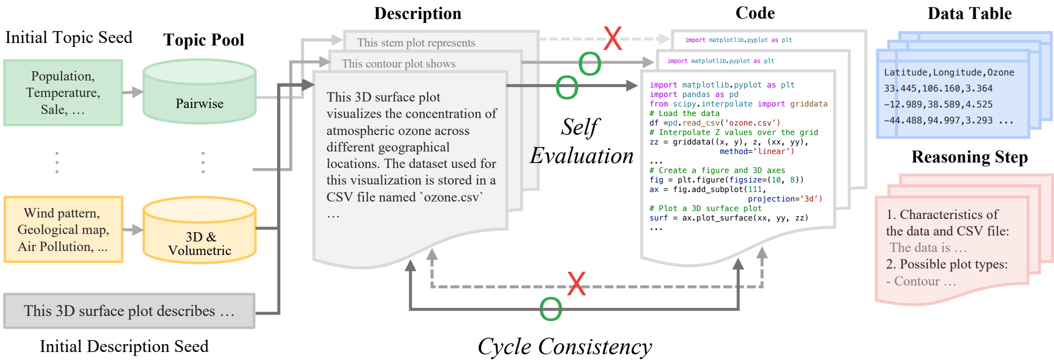


Figure 2: Illustration of our hierarchical chart generation process with an example of a single plot type. The process begins by randomly selecting a topic from a topic pool. Two instructional samples are then chosen from an instruction pool and given to GPT-3.5-turbo to generate a new instruction, which undergoes a self-evaluation process by GPT-4 for qualification. If it meets the criteria, which includes compatibility with the data points and the plot type, it is added to the instruction pool. Simultaneously, the new instruction is sent to GPT-4 for data table creation using a long data table format and code generation. Finally, the generated tuple (x, d, c, y) goes through a final filtering of cycle-consistency to validate the produced data point with high quality and correctness.

worth noting that we remove human supervision during these processes and solely rely on automatic feedback.

Preference reward. We propose an automatic way of designing a preference dataset based on the output of the supervised fine-tuned model π_{θ_1}. We define preference dataset $\mathcal{D}_{\text{pref}} = (c_i^+, c_i^-)_{i=1}^n$, where a preferred code c^+ is the ground truth code, while a less preferred one c^- is a corresponding code output of SFT. Afterward, we train a preference reward model $R_\phi(c)$ following Ouyang et al. (2022) and employ this reward model to train π_{θ_1} via proximal policy optimization (PPO) algorithm (Schulman et al., 2017) as follows:

$$\underset{\theta_1}{\text{maximize}}\ \mathbb{E}_{x\sim\mathcal{D},\, \hat{c}\sim\pi_{\theta_1}(\cdot|x)}\Big(R_\phi(\hat{c})\Big) - \beta D_{\text{KL}}(\pi_{\theta_1} \parallel \pi_{\theta_{\text{sft}1}}).$$

Alignment reward. The alignment reward leverages cycle consistency between a chart's description and code. First, π_{θ_1} generates a code from the original description, then π_{θ_3} uses this code to produce a regenerated description. The alignment reward is defined as the similarity between the original and regenerated descriptions, measured by BertScore (Zhang et al., 2020; Black et al., 2024). We optimize π_{θ_3} via maximizing the alignment reward $R(\cdot,\cdot)$ using PPO algorithm as follows:

$$\underset{\theta_3}{\text{maximize}}\ \mathbb{E}_{x\sim\mathcal{D},\, \hat{c}\sim\pi_{\theta_1}(\cdot|x),\, \hat{x}\sim\pi_{\theta_3}(\cdot|\hat{c})}\Big(R(x, \hat{x})\Big) - \beta D_{\text{KL}}(\pi_{\theta_3} \parallel \pi_{\theta_{\text{sft}3}}).$$

4 Experiments

Baselines. For the evaluation of the three target tasks, we compare with the state-of-the-art open-source baseline models as follows: (i) Description-to-Chart: Code Llama Instruct (Rozière et al., 2024), Llama 3 Instruct (Meta AI, 2024), StarCoder (Li et al., 2023), and Instruct CodeGen (Nijkamp et al., 2023), (ii) Raw Data-to-Description: Llama 2 Chat (Touvron et al., 2023) and Llama 3 Instruct model, and (iii) Code-to-Description: Code Llama, Llama 2 Chat, and Llama 3 Instruct models. We also compare with proprietary models including GPT-3.5-turbo (Ouyang et al., 2022), GPT-4-0613, GPT-4-turbo-2024-04-09 (OpenAI, 2023), GPT-4o-2024-05-13 (OpenAI, 2024), and Claude 3 Opus (Anthropic, 2024).

Evaluation metrics. For the three target tasks, we report the following evaluation measures.

(i) Description-to-Chart: We report the total error ratio and plot-type error ratio. The total error ratio indicates the percentage of code executions that result in errors. We categorize and report plot-type errors based on Matplotlib classifications. We further evaluate the similarity between the predicted code and the ground truth (GT) code by reporting the METEOR (Banerjee and Lavie, 2005) and CodeBLEU metrics (Ren et al., 2020).

(ii) Raw Data-to-Description: We report the Jaccard similarity and the Hit Rate. The former measures the intersection ratio between the recommended plot list derived from generated reasoning steps and the GT reasoning steps. The latter is the

Algorithm 1 Chart generation instruction tuning

Require: Description-to-chart policy network π_{θ_1}, Raw data-to-chart policy network π_{θ_2}, code-to-description policy network π_{θ_3}, Text2Chart31 dataset $\mathcal{D}$

for iter = $1, 2, \ldots, N_{\text{sft}}$ **do** ▷ Supervised fine-tuning
    Sample data (x, c, d, r, y) from dataset $\mathcal{D}$
    Optimize $\mathcal{L}_{\text{code}}(\theta_1) = -\sum_t \log \pi_{\theta_1}(c_{(t)}|x, c_{(<t)})$
    Optimize $\mathcal{L}_{\text{reason}}(\theta_2) + \mathcal{L}_{\text{desc}}(\theta_2) = -\sum_t \log \pi_{\theta_2}(r_{(t)}|d, r_{(<t)}) - \sum_t \log \pi_{\theta_2}(x_{(t)}|d, r, x_{(<t)})$
    Optimize $\mathcal{L}_{\text{desc}}(\theta_3) = -\sum_t \log \pi_{\theta_3}(x_{(t)}|x_{(<t)}, c, d)$
end for
$\pi_{\theta_{\text{sft1}}} \leftarrow \pi_{\theta_1}, \pi_{\theta_{\text{sft2}}} \leftarrow \pi_{\theta_2}, \pi_{\theta_{\text{sft3}}} \leftarrow \pi_{\theta_3}$
Generate automatic preference dataset $\mathcal{D}_{\text{pref}}$ from $\pi_{\theta_{\text{sft1}}}$ and $\mathcal{D}$
Train preference reward model $R_\phi(c)$ from $\mathcal{D}_{\text{pref}}$
for iter = $1, 2, \ldots, N_{\text{rl}}$ **do** ▷ Reinforcement learning (PPO)
    Sample x from dataset $\mathcal{D}$
    Generate $\hat{c}$ from $\pi_{\theta_1}(\cdot|x)$, and generate $\hat{x}$ from $\pi_{\theta_3}(\cdot|\hat{c})$
    Calculate preference reward $R_\phi(\hat{c})$
    Calculate alignment reward $R(x, \hat{x}) = \text{BertScore}(x, \hat{x})$
    Jointly optimize
$$\left(J_{\text{PPO}}(\theta_1) = R_\phi(\hat{c}) - \beta \log\left(\frac{\pi_{\theta_1}(\hat{c}\,|\,x)}{\pi_{\theta_{\text{sft1}}}(\hat{c}\,|\,x)}\right), \quad J_{\text{PPO}}(\theta_3) = R(x, \hat{x}) - \beta \log\left(\frac{\pi_{\theta_3}(\hat{x}\,|\,\hat{c})}{\pi_{\theta_{\text{sft3}}}(\hat{x}\,|\,\hat{c})}\right) \right) \text{ with PPO}$$
end for

percentage of recommended lists containing the GT plot type. To evaluate the quality of the generated descriptions, we first use these descriptions to generate code with both the SFT Llama3 Instruct-8B model and the GPT-3.5-turbo, and then calculate the error ratio for the generated codes. Additionally, we report ROUGE-L and BertScore metrics to assess the similarity between the generated descriptions and the GT descriptions.

(iii) Code-to-Description: We measure ROUGE-1/2/L and BertScore to evaluate the similarity between the generated descriptions and the GTs. Lastly, as done for Task 2, we generate the code by giving the predicted descriptions to the GPT-3.5-turbo and report the error ratio.

Training setup. We begin the supervised fine-tuning using LoRA fine-tuning (Hu et al., 2021). When we further fine-tune the model with RL, we merge the original SFT LoRA parameters into the base model and fine-tune separate LoRA parameters. For SFT, we utilize a total of 11.1K data points for Task 1, 3, and 7.84K for Task 2. On the other hand, RL fine-tuning is conducted using 0.5K randomly selected data points, representing 4.8% of our $\mathcal{D}_{\text{pref}}$ dataset. For SFT, we use 2 RTX A6000 GPUs and the training requires 6 to 12 hours, depending on the tasks. For RL, we use 6 RTX A6000 GPUs and the training takes less than 12 hours. Further details of the experiments can be found in Appendix B.

4.1 Results of Description-to-Chart

Table 2 presents the results for the Description-to-Chart task. We fine-tune Llama 3 Instruct-8B and Code Llama Instruct-13B on our Text2Chart31 dataset for five epochs. We run RL fine-tuning on the Llama 3 Instruct and Code Llama Instruct-13B using preference reward, denoted as RL_{pref}. The results show that our fine-tuned models outperform all open-source baselines that we compared. Specifically, the 13B model with SFT and RL achieves even a lower total error ratio than the state-of-the-art closed-source models like GPT-3.5-turbo, GPT-4, GPT-4-turbo, GPT-4o, and Claude 3 Opus. The RL fine-tuning reduces the total error ratio of the Llama 3 Instruct-8B model from 16.09 to 14.55, making it superior to the Claude 3 Opus. Particularly, our models excel in generating underexplored plot types such as gridded, irregularly gridded, and 3D and volumetric plots, compared to open-source models.

Human evaluation. We additionally conduct human evaluation to check the correctness of the generated plot and its alignment with the descrip-

Model	Error ratio (%) ↓					Code similarity ↑	
	Pairwise	Statistical distribution	(Irregularly) gridded	3D & Volumetric	Total	METEOR	CodeBLEU
Open-source							
CLI-7B	22.67	29.42	77.94	52.20	41.32	0.485	0.402
L3I-8B	20.76	28.98	66.76	34.59	35.91	0.519	0.437
[SFT] L3I-8B	19.07	13.27	**13.53**	**20.75**	16.09	0.562	**0.464**
[SFT+RL$_{\text{pref}}$] L3I-8B	**13.14**	**11.50**	15.00	26.42	**14.55**	**0.567**	0.461
CLI-13B	18.86	29.42	71.76	57.23	39.14	0.489	0.413
StarCoder-15.5B	23.31	32.08	51.18	25.16	32.89	0.347	0.328
Instruct CodeGen-16B	38.56	45.13	62.94	40.25	46.66	0.388	0.330
[SFT] CLI-13B	**6.36**	6.19	**12.06**	22.64	9.49	**0.581**	**0.481**
[SFT+RL$_{\text{pref}}$] CLI-13B	**6.36**	**5.53**	12.35	**21.38**	**9.21**	0.566	0.467
Closed-source							
GPT-3.5-turbo	11.02	13.50	28.82	19.59	18.62	0.524	0.453
GPT-4-0613	13.56	11.06	28.53	39.62	19.26	0.535	0.441
GPT-4-turbo	11.02	14.16	11.76	29.56	14.27	0.540	0.448
GPT-4o	13.98	6.86	13.53	26.42	13.00	0.552	0.450
Claude 3 Opus	7.84	7.74	30.59	23.27	14.90	0.515	0.435

Table 2: Results of the Description-to-Chart task. The plot type error ratio is categorized based on Matplotlib classifications (Hunter, 2007). CLI and L3I stand for Code Llama Instruct and Llama 3 Instruct, respectively. **SFT** and **RL**$_*$ indicate our fine-tuned models.

Method	Error ratio (%) ↓		Plot type ↑		Desc. sim. ↑	
	w/ GPT	w/ SFT	HitRate	Jac.	R-L	BertScore
Open-source						
L2C-7B	40.82	56.45	0.175	0.359	0.232	0.820
L2C-13B	35.64	37.99	0.205	0.384	0.237	0.825
L3I-8B	27.25	38.48	0.269	0.406	0.196	0.800
[SFT] L2C-7B	15.92	15.82	0.329	0.396	0.381	0.903
[SFT] L3I-8B	**15.53**	**15.62**	**0.413**	**0.432**	**0.389**	**0.905**
Closed-source						
GPT-3.5-turbo	21.00	29.10	0.239	0.412	0.232	0.816
GPT-4	16.41	34.67	0.286	0.428	0.202	0.829
GPT-4-turbo	27.05	37.01	0.313	0.461	0.184	0.808
GPT-4o	15.82	31.64	0.339	0.436	0.170	0.786
Claude 3 Opus	15.62	27.34	0.294	0.451	0.188	0.813

Table 3: Results of the Raw Data-to-Chart task. Description similarity, error ratio, and plot type prediction are compared for various open-source and closed-source methods. The error ratio is evaluated using **SFT** L3I-8B from Task 1 denoted as 'w/ SFT', or GPT-3.5-turbo denoted as 'w/ GPT'. **SFT** indicates our fine-tuned models. L2C and L3I stand for Llama 2 Chat and Llama 3 Instruct, respectively.

Method	Description similarity ↑				Err. ratio (%) ↓
	R-1	R-2	R-L	BertScore	w/ GPT
Open-source					
L2C-7B	0.419	0.182	0.260	0.812	38.86
CLI-7B	0.411	0.173	0.260	0.823	42.73
L3I-8B	0.453	0.206	0.276	0.834	36.82
[SFT] L3I-8B	0.592	0.343	0.436	0.881	21.36
[SFT+RL$_{\text{algn}}$] L3I-8B	**0.594**	**0.346**	**0.440**	**0.884**	**20.31**
Closed-source					
GPT-3.5-turbo	0.463	0.212	0.286	0.845	45.26
GPT-4	0.426	0.175	0.252	0.809	23.12
GPT-4-turbo	0.416	0.168	0.242	0.795	29.80
GPT-4o	0.442	0.198	0.269	0.775	11.81
Claude 3 Opus	0.453	0.207	0.276	0.827	19.33

Table 4: Results of the Code-to-Description task. **SFT** and **RL**$_*$ indicate our fine-tuned models. L2C and L3I stand for Llama 2 Chat and Llama 3 Instruct-8B, respectively.

tion. We randomly sample a subset of 155 data points, consisting of 5 samples from each of the 31 plot types. For each sample, three crowd workers are recruited to compare the generated plot images with the GT reference plot images based on chart type, data representation, and visual appearance. If both images are equally similar or neither is similar, it is voted as a tie. More details can be found in Appendix E. Figure 3 presents the results of human evaluation. The inter-annotator agreement is measured using Krippendorff's α, whose value is 0.519 for the three classes (win, lose, and tie). Our fine-tuned models consistently have higher win rate compared to Llama 3 Instruct-8B and GPT-3.5-turbo. Specifically, **SFT** CLI-13B model has the higher win rate (47.7%) against L3I-8B, while also achieving a lower lose rate (4.5%). Our **SFT+RL$_{\text{pref}}$** L3I-8B model wins over GPT-3.5-turbo with 25.2% win rate and 20.6% lose rate.

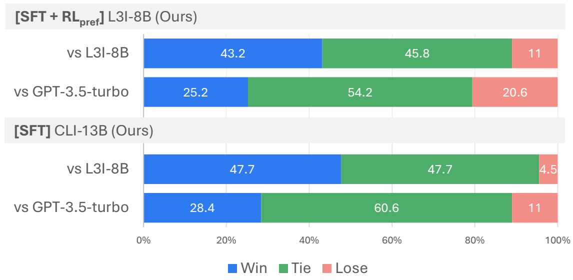


Figure 3: Human evaluation results on a randomly sampled subset of the test set. We compare **SFT+RL$_{\text{pref}}$** L3I-8B and **SFT** CLI-13B with GPT-3.5-turbo and L3I-8B.

4.2 Results of Raw Data-to-Chart

Table 3 presents the results of the Raw Data-to-Chart task. We fine-tune Llama 2 Chat-7B and Llama 3 Instruct-8B using our Text2Chart31 dataset. We report the error ratio after visualizing the generated descriptions using our supervised fine-tuned Llama 3 Instruct-8B (w/ SFT) from task 1 and GPT-3.5-turbo (w/ GPT). Notably, our fine-tuned Llama 3 Instruct-8B outperforms all open-source models across all metrics. Furthermore, this model surpasses closed-source models (GPT-3.5-turbo, GPT-4-turbo) in terms of error ratio and generated description similarity.

4.3 Results of Code-to-Description

Table 4 presents the results on the Code-to-Description task. We fine-tune the Llama 3 Instruct-8B using our dataset and evaluate the description similarity with ROUGE and BertScore. Our fine-tuned model outperforms all open-source and closed-source models across Description similarity. Furthermore, RL fine-tuning with alignment reward consistently increases the description similarity across all metrics. We also provide the generated descriptions to GPT-3.5-turbo and report the error ratio to highlight the quality of the descriptions produced by our fine-tuned models. After RL fine-tuning, the error ratio decreases from 21.36% to 20.31%, and the description similarity consistently improves.

5 Related Work

Chart datasets. There are several existing chart datasets, including PlotQA (Methani et al., 2020), ChartQA (Masry et al., 2022), FigureQA (Kahou et al., 2018), Unichart (Masry et al., 2023), Autochart (Zhu et al., 2021), Chart-to-Text (Kantharaj et al., 2022). These datasets primarily focus on question and answer (QA) tasks on a limited range of plot types. More recently, ChartLlama (Han et al., 2023) proposes a text-to-chart dataset that includes QA tasks and generates visualization code from provided descriptions. However, these datasets still lack coverage in certain plot categories such as 3D/volumetric plots and vector field plots, and they do not cover the use case of analyzing the raw data and predicting the most suitable plot types. On the other hand, our Text2Chart31 dataset encompasses 31 plot types with 11.1K tuples that combine descriptions, code, data tables, and plots, thereby covering a wide range of use cases.

Instruction tuning. Employing reinforcement learning with human feedback is a prevalent strategy for enhancing (un)supervised finetuned models, whether by integrating human feedback into the learning loop (Arakawa et al., 2018; Arumugam et al., 2019) or by leveraging preference data generated by human (Ouyang et al., 2022; Glaese et al., 2022; Bai et al., 2022; Stiennon et al., 2022). However, we argue that this methodology might not offer the most practical solution for plot visualization tasks, given the intricate and fact-intensive nature of plot types. Moreover, considering the limitations of human cognition, there is a risk of overlooking crucial small details essential for validating the accuracy of generated plots. To address this, we propose a novel automatic method that constructs a preference dataset using supervised fine-tuned output.

Cycle consistency. Exploiting cycle consistency to enhance the performance of the generative model has been mainly studied in the image domain (Zhu et al., 2020). Recently, DDPO (Black et al., 2024) adopts the LLaVA model (Liu et al., 2023) to increase the alignment between the image and the text. Following this line of research, we propose an alignment reward that exploits cycle consistency between description and code to improve LLM for chart generation tasks. This is made possible because of the rich nature of our Text2Chart31 dataset, which consists of diverse textual modalities, including visualization code and description.

6 Conclusion

We introduce a novel hierarchical pipeline and a comprehensive dataset for chart generation. The proposed Text2Chart31 dataset, encompassing 31 unique plot types, provides a robust foundation for diverse visualization tasks with its 11.1K tuples of descriptions, code, data tables, and plots. Additionally, we proposed an RL-based instruction tuning technique employing preference and alignment rewards, improving LLMs in data visualization.

Limitations

There are certain considerations to note. First, our dataset is based on Matplotlib version 3.8. As such, if earlier versions of Matplotlib are used where function names may have changed, the generated code could potentially cause errors. This is a natural consequence of advancements and updates in software libraries. Additionally, the descriptions provided are exclusively in English. This focus ensures clarity and consistency in our current scope but can be expanded to include multiple languages in future iterations. Lastly, our primary focus was on chart generation through large language models (LLMs), rather than on question answering. However, exploring question answering capabilities is a promising direction for future research.

Ethics Statement

All data points generated in Text2Chart31 were created using large language models (LLMs) and are intended solely for visualization purposes. These data points do not represent real-world facts and should not be referenced as accurate depictions of actual data distributions. Furthermore, they do not contain offensive contents. Matplotlib library is based on PSF license. We have used open source models, libraries, and closed source models for their intended uses, and not use other than research purposes.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback. This work was financially supported by SNU-NAVER Hyperscale AI Center, Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II191082, SW StarLab, No. RS-2022-II220156, Fundamental research on continual meta-learning for quality enhancement of casual videos and their 3D metaverse transformation, and No. RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2023R1A2C2005573).

References

Anthropic. 2024. Introducing the next generation of claude.

Riku Arakawa, Sosuke Kobayashi, Yuya Unno, Yuta Tsuboi, and Shin ichi Maeda. 2018. Dqn-tamer: Human-in-the-loop reinforcement learning with intractable feedback. *Preprint*, arXiv:1810.11748.

Dilip Arumugam, Jun Ki Lee, Sophie Saskin, and Michael L. Littman. 2019. Deep reinforcement learning from policy-dependent human feedback. *Preprint*, arXiv:1902.04257.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. 2024. Training diffusion models with reinforcement learning. *Preprint*, arXiv:2305.13301.

Besnik Fetahu, Zhiyu Chen, Oleg Rokhlenko, and Shervin Malmasi. 2023. Instructpts: Instruction-tuning llms for product title summarization. *Preprint*, arXiv:2310.16361.

Dan Friedman and Adji Bousso Dieng. 2023. The vendi score: A diversity evaluation metric for machine learning. *Preprint*, arXiv:2210.02410.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Soňa Mokrá, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. 2022. Improving alignment of dialogue agents via targeted human judgements. *Preprint*, arXiv:2209.14375.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2023. News summarization and evaluation in the era of gpt-3. *Preprint*, arXiv:2209.12356.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *Preprint*, arXiv:2203.05794.

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. *Preprint*, arXiv:2311.16483.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Akos Kadar, Adam Trischler, and Yoshua Bengio. 2018. Figureqa: An annotated figure dataset for visual reasoning. *Preprint*, arXiv:1710.07300.

Shankar Kantharaj, Rixie Tiffany Ko Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq Joty. 2022. Chart-to-text: A large-scale benchmark for chart summarization. *Preprint*, arXiv:2203.06486.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. *Preprint*, arXiv:1510.03055.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. Starcoder: may the source be with you! *Preprint*, arXiv:2305.06161.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Preprint*, arXiv:2304.08485.

Tiedong Liu and Bryan Kian Hsiang Low. 2023. Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks. *Preprint*, arXiv:2305.14201.

Ahmed Masry, Parsa Kavehzadeh, Xuan Long Do, Enamul Hoque, and Shafiq Joty. 2023. Unichart: A universal vision-language pretrained model for chart comprehension and reasoning. *Preprint*, arXiv:2305.14761.

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. *Preprint*, arXiv:2203.10244.

Meta AI. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. *Preprint*, arXiv:1909.00997.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2023. Codegen: An open large language model for code with multi-turn program synthesis. *Preprint*, arXiv:2203.13474.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024. Hello gpt-4o.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco, and Shuai Ma. 2020. Codebleu: a method for automatic evaluation of code synthesis. *Preprint*, arXiv:2009.10297.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. *Preprint*, arXiv:2110.08207.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2022. Learning to summarize from human feedback. *Preprint*, arXiv:2009.01325.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Siddharth Varia, Shuai Wang, Kishaloy Halder, Robert Vacareanu, Miguel Ballesteros, Yassine Benajiba, Neha Anna John, Rishita Anubhai, Smaranda Muresan, and Dan Roth. 2023. Instruction tuning for few-shot aspect-based sentiment analysis. *Preprint*, arXiv:2210.06629.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023a. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. *Preprint*, arXiv:2212.10560.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Min Dou, Botian Shi, Junchi Yan, and Yu Qiao. 2024. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *Preprint*, arXiv:2402.12185.

Kang Min Yoo, Jaegeun Han, Sookyo In, Heewon Jeon, Jisu Jeong, Jaewook Kang, Hyunwook Kim, Kyung-Min Kim, Munhyong Kim, Sungju Kim, et al. 2024. Hyperclova x technical report. *arXiv preprint arXiv:2404.01954*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.

Jiawen Zhu, Jinye Ran, Roy Ka wei Lee, Kenny Choo, and Zhi Li. 2021. Autochart: A dataset for chart-to-text generation task. *Preprint*, arXiv:2108.06897.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2020. Unpaired image-to-image translation using cycle-consistent adversarial networks. *Preprint*, arXiv:1703.10593.

A Details of Text2Chart31 Dataset

In this section, we provide a comprehensive overview of the Text2Chart31 dataset, including its categories, examples, summary statistics, topic distribution and plot type distribution.

A.1 Categories and Examples

Figure 4 illustrates the diverse range of plot types included in the Text2Chart31 dataset. The dataset covers 31 different plot types, grouped into 5 categories: Pairwise Chart, Statistical Distribution Chart, Gridded Chart, Irregularly Gridded Chart, and 3D & Volumetric Chart. The examples provided for each plot type illustrate the variety of data and plot types present in the dataset.

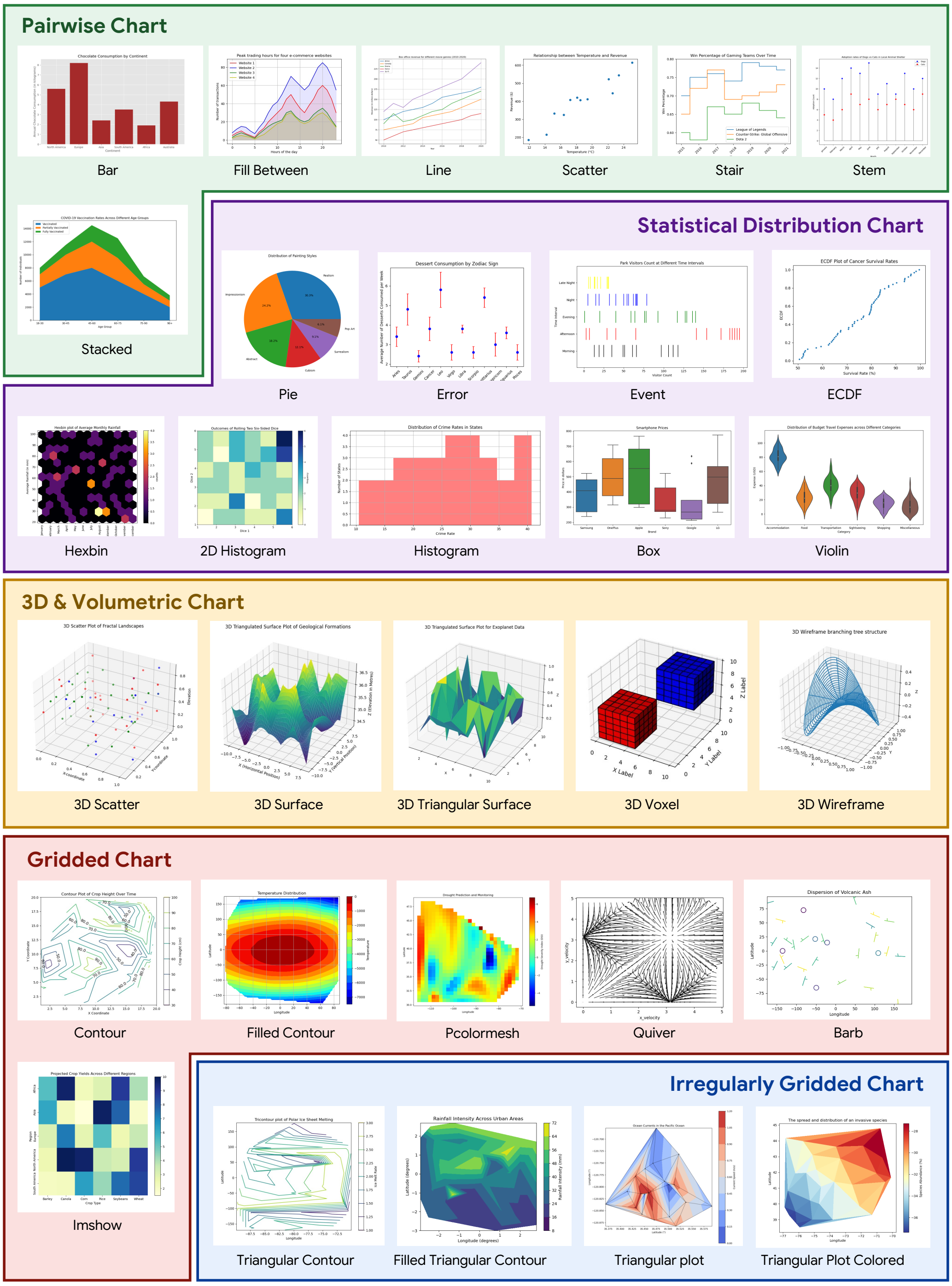


Figure 4: Examples from the 31 plot types in Text2Chart31 dataset, grouped into 5 chart categories.

A.2 Dataset Summary

The Text2Chart31 dataset consists of 11,128 data points, with 9,705 in the training set and 1,423 in the test set. The dataset is categorized into five categories of charts: Pairwise, Statistical Distribution, Gridded, Irregularly Gridded, and Statistical Distribution and 3D & Volumetric chart. Among the total data points, 8,166 include both data tables (d) and reasoning steps (r), with 7,142 in the training set and 1,024 in the test set.

	Train	**Test**	**Total**
Pairwise Chart	3026 (1557)	472 (241)	3498 (1798)
Statistical Distribution Chart	2878 (1784)	452 (284)	3330 (2068)
Gridded Chart	1305 (1305)	192 (192)	1497 (1497)
Irregularly Gridded Chart	1145 (1145)	148 (148)	1293 (1293)
3D & Volumetric Chart	1351 (1351)	159 (159)	1510 (1510)
Total	9705 (7142)	1423 (1024)	11128 (8166)

Table 5: Summary of the Text2Chart31 dataset. The numbers in parentheses indicate the data points that include both data tables (d) and reasoning steps (r).

A.3 Topic Distribution

Figure 5 shows the distribution of keywords within the topic pool extracted using BERTopic(Grootendorst, 2022). The generated topic pool encompasses a diverse range of fact-based and natural topics, ensuring comprehensive coverage across various subject areas.

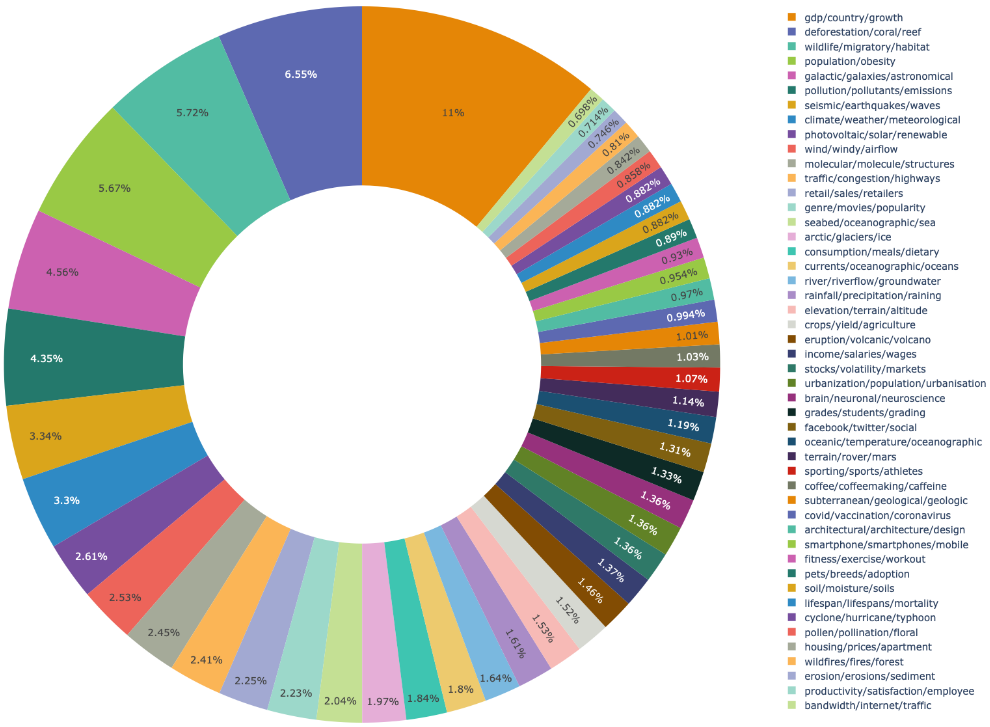
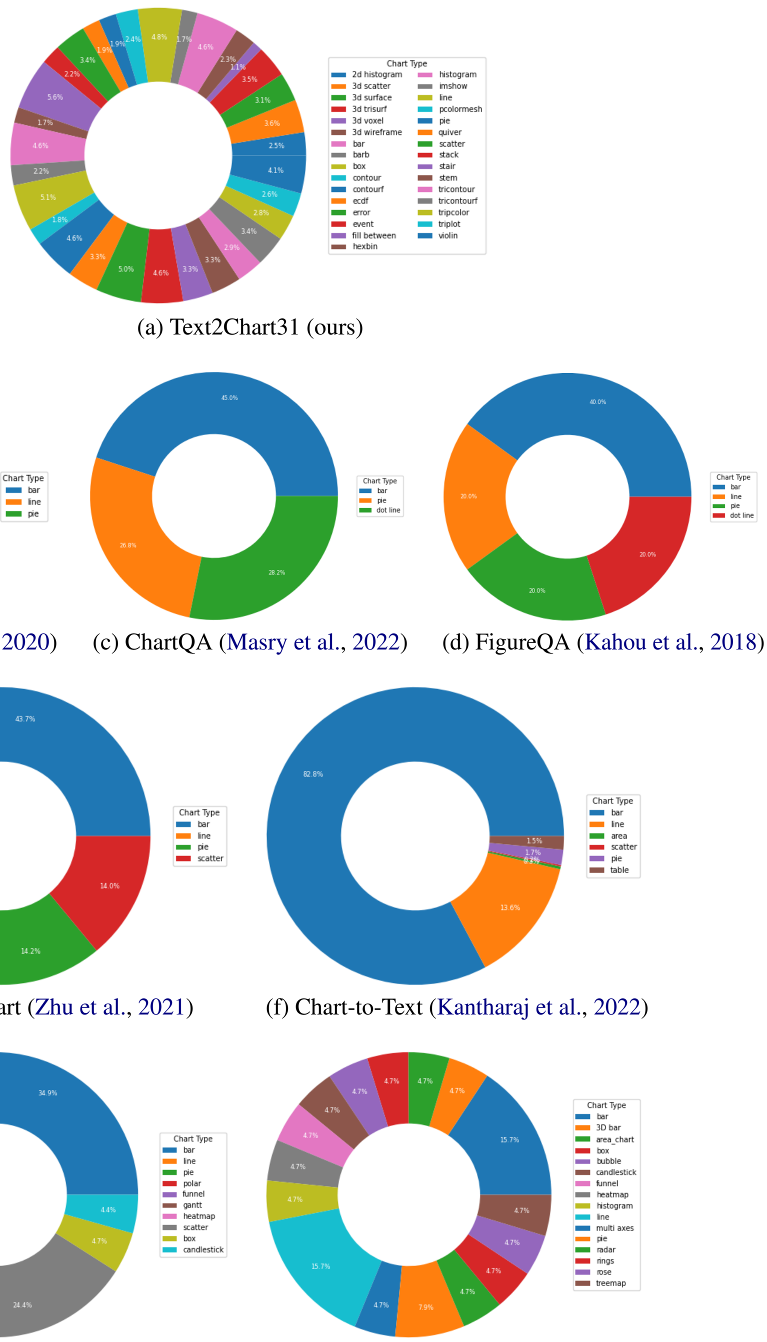


Figure 5: Distribution of keywords within the topic pool, showcasing the diverse and balanced coverage of topics in the Text2Chart31 dataset.

A.4 Plot Type Distribution

As shown in Figure 6, our dataset, Text2Chart31, exhibits the most diverse and well-balanced distribution across various plot types when compared to other existing datasets. While existing datasets have primarily focused on common plot types such as bar charts and line charts, our dataset provides comprehensive coverage across a diverse range of plot types. This includes more complicated plot types like 3D surface plots and contour plots.



(a) Text2Chart31 (ours)

(b) PlotQA (Methani et al., 2020) (c) ChartQA (Masry et al., 2022) (d) FigureQA (Kahou et al., 2018)

(e) Autochart (Zhu et al., 2021) (f) Chart-to-Text (Kantharaj et al., 2022)

(g) ChartLlama (Han et al., 2023) (h) ChartX (Xia et al., 2024)

Figure 6: Comparison of the distribution of chart types with other datasets. Each pie chart shows the distribution of the plot types for each dataset, respectively.

A.5 Example of Text2Chart31

This section shows example from the Text2Chart31 dataset, providing a overall view of the data point including description, code, reasoning step, and data table, as shown in Table 6.

Description

This contour plot showcases the temperature variations along a coastal region using the dataset 'coastal_temperature.csv'. The dataset includes information on latitude, longitude, and temperature (in degrees Celsius) for different coastal locations.
By establishing a grid of points (X, Y) representing latitude and longitude, we can model the temperature variations using the equation 'Z = a*X + b*Y + c', where a and b are regression coefficients determined through analysis.
Here are the first six rows of the dataset:
Latitude, Longitude, Temperature (°C)
40.7128, -74.0060, 15
48.8566, 2.3522, 10
37.7749, -122.4194, 18
51.5074, -0.1278, 12
35.6895, 139.6917, 20
42.3601, -71.0589, 14
Visualizing this data on a contour plot helps us analyze the spatial distribution of temperature along the coast. The contour lines indicate regions with similar temperature values, enabling us to identify temperature patterns and variations based on geographical location. This plot can be useful for understanding coastal climate, identifying areas with specific temperature ranges, and providing insights for planning activities related to beach tourism, coastal agriculture, and coastal development.

Code

```
...
# Define the grid over which we will interpolate the data
x_range = np.linspace(data['Longitude'].min(), data['Longitude'].max(), len(data['Longitude'].unique()))
y_range = np.linspace(data['Latitude'].min(), data['Latitude'].max(), len(data['Latitude'].unique()))
x, y = np.meshgrid(x_range, y_range)

# Interpolate the measured temperature data onto the defined grid
z = griddata((data['Longitude'], data['Latitude']), data['Temperature (°C)'], (x, y), method='cubic')

# Configure the contour plot
plt.figure(figsize=(10, 6))
contours = plt.contourf(x, y, z, cmap='RdYlBu_r')
plt.colorbar(contours)
...
```

Reasoning Step

1. Characteristics of the data and CSV file:
- The data includes three columns: Latitude, Longitude, and Temperature (°C).
- Latitude and Longitude are geographical coordinates specifying north-south and east-west positioning on the Earth's surface.
- Temperature is a numerical measure indicating the amount of heat energy.
- The data thus represents the temperature at different geographical locations around the world. ....

2. Possible plot types:
- **Map-based plots**: Utilizing the latitude and longitude values to plot geographical data on a global map.
- **Scatter Plot**: To visualize how temperature varies with latitude or longitude.
- **Contour or Heatmap**: Can be plotted over a map to show temperature distributions.
- **3D Surface Plot**: Making use of X (Longitude), Y (Latitude), and Z (Temperature) to form a 3D visualization of temperature variations.

3. Most suitable plot type:
- **Map-based Heatmap or Contour Map**: This is especially suitable due to the geographical nature of the data (latitude and longitude) combined with a third variable (temperature). This type of visualization will allow viewers to easily understand temperature distributions across various geographical locations. ...

4. Further considerations for the description:
- Verify the accuracy and intent of the negative temperature values. If these are anomalies, it must be clearly stated to avoid misinterpretation. - Include proper color scales to represent temperature values, ensuring that the map is accessible and readable.
- Consider adding interactive elements such as tooltips or zoom features if the visualization is digital, to enable detailed inspection of data at specific locations. ...

Data Table

Latitude,Longitude,Temperature (°C)
5.2927,-173.8335,31.8
-12.0503,72.5485,-39.1
86.2247,137.7667,27.6
-43.7762,-120.4625,44.5
16.645,-132.0993,-48.7
23.3972,-46.3823,-9.7
...

Table 6: An example data point in Text2Chart31, comprising a description, code, reasoning step, and CSV data table (top to bottom). The description elucidates the contour plot, coastal_temperature.csv dataset, and insights from the visualization. The code utilizes Matplotlib for generating the contour plot. The reasoning step delineates the rationale behind crafting the data table and visualization, factoring in data characteristics, plot types, and additional consideration. Finally, the data table shows the dataset columns: Latitude, Longitude, and Temperature (°C).

B Experimental Details

Training setup and hyperparameters We report the hyperparameters for training supervised fine-tuning and joint reinforcement learning based fine-tuning in Table 7 and Table 8. For supervised fine-tuning, we fine-tune base model with LoRA adapter with the configuration in Table 7. For reinforcement learning-based fine-tuning, we start with the supervised fine-tuned model and merge the LoRA parameters into the original model parameters. Then, we apply an additional LoRA adapter according to the configuration in Table 8. Finally, we fine-tune both Task 1 and Task 3 models jointly using the PPO algorithm.

	Task 1		Task 2		Task 3
Model	L3I-8B	CLI-13B	L2C-7B	L3I-8B	L3I-8B
Training epochs	5	5	5	5	5
Training set size	9705	9705	7142	7142	9705
Batch size	16	16	16	16	16
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate	5e-4	5e-4	5e-5	5e-5	5e-4
Learning rate scheduling	Constant	Constant	Constant	Constant	Constant
Mixed precision	BF16	BF16	BF16	BF16	BF16
LoRA rank	16	16	32	32	32
LoRA alpha	16	16	32	32	32
LoRA dropout	0.1	0.1	0.1	0.1	0.1

Table 7: Training hyperparameters for supervised fine-tuning. L3I-8B, CLI-13B, and L2C-7B denote Llama 3 Instruct-8B, Code Llama Instruct-13B, and Llama 2 Chat-7B, respectively.

	Task 1		Task 3
Model	L3I-8B	CLI-13B	L3I-8B
Batch size	8	8	8
Training steps	63	94	63
Training data size	504	752	504
Optimizer	Adam	Adam	Adam
Learning rate	1.41e-5	7.05e-6	1.41e-5
Learning rate scheduling	Constant	Constant	Constant
Mixed precision	BF16	BF16	BF16
LoRA rank	16	16	32
LoRA alpha	16	16	32
LoRA dropout	0.1	0.1	0.1

Table 8: Training hyperparameters for RL fine-tuning. L3I-8B and CLI-13B denote Llama 3 Instruct-8B and Code Llama Instruct 13B, respectively.

C Cycle Consistency Details

This method leverages the capabilities of language models to verify the consistency between the original plot description and the generated code, without the need for manual human evaluation. Figure 7 and Figure 8 illustrate examples of data points that fail and pass the cycle consistency verification, respectively. By employing this method, we ensure that the generated code and plot are well aligned with the intended visualization described in the original description, maintaining the quality of the Text2Chart31 dataset.

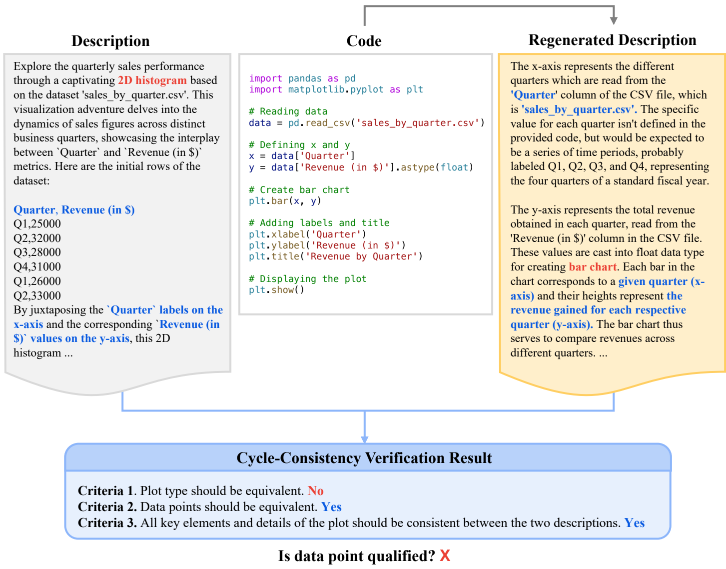


Figure 7: Example of cycle consistency verification for a description and generated code, showcasing inconsistency in the plot type (2D histogram vs. bar chart) despite consistent data source and sufficient detail in both descriptions.

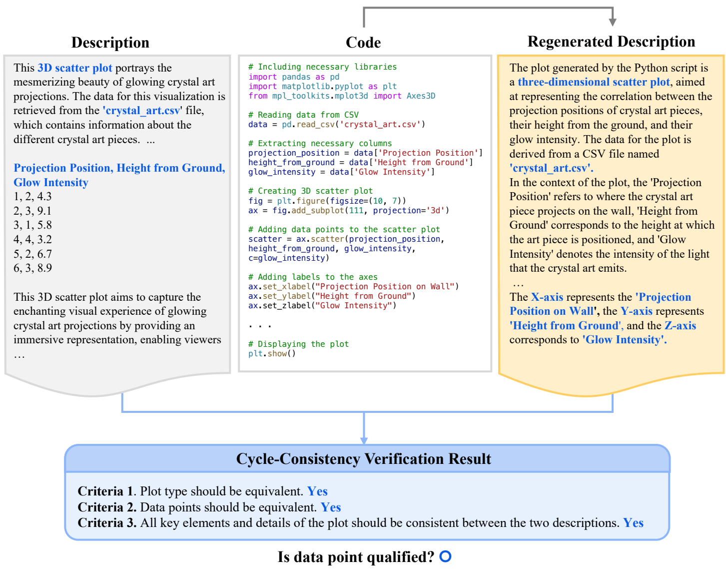


Figure 8: Example of cycle consistency verification for a description and generated code, showcasing consistency in the plot type (3D scatter plot), data source (crystal_art.csv), and sufficient detail in both descriptions to accurately redraw the plot.

D Prompt Template

D.1 Prompt Template used for Data Generation

This section presents the prompt templates used for various stages of the data generation process in the Text2Chart31 dataset. Figure 9 illustrates the prompt template used for topic generation, while Figure 10 shows the template for description generation. Figure 11 presents the template for description self-evaluation, and Figure 12 illustrates the template for code generation. Figure 13 shows the template used for cycle consistency verification, and Figure 14 presents the template for data table generation. Figure 15 illustrates the template for generating code that creates the data table, and Figure 16 shows the template for reasoning step generation.

Topic Generation Prompt Template

"system" :
You are good at thinking about different topics. I am going to write different topics suitable for types like 3D wireframe plots, 3d tri-surface plots, 3d voxel plots, 3d scatter plots, 3d surface plots and more, that can be used in generating descriptions for plots that illustrate 3D data or volumetric data types.
"user" :
Write 5 different topics that are commonly used for making 3D plots. Keep each topic to 8 words or less and be creative.
"assistant" :
Topic 1. {topic 1} Topic 2. {topic 2} Topic 3. {topic 3} Topic 4. {topic 4} Topic 5. {topic 5}
"user" :
Generate 5 more plots, 3D surface plots and more. Keep each topic under 8 words. creative topics for 3D data or volumetric data, suitable for 3D wireframe plots, 3D tri-surface plots, 3D voxel plots, 3D scatter

Figure 9: Prompt template used for topic generation

Description Generation Prompt Template

"system":
You are good at describing various/different 3D Surface Plots for data visualization. Make sure when you describe a graph, mention the data points that are going to be used; otherwise, we won't be able to sketch the plot.
"user" :
Write 5 descriptions that describe a 3D Surface Plot. You can write code to illustrate the 3D Surface Plot based on the given description
"assistant" :
Description 1. {description 1} Description 2. {description 2} Description 3. {description 3}
Description 4. {description 4} Description 5. {description 5}
"user" :
Write a description for a 3D Surface Plot based on the given topic. You can write code to illustrate the 3D Surface Plot based on the description. Include the FIRST 6 ROWS of DATA POINTS from the CSV file that will be used, along with the name of the CSV file. Without this information, we won't be able to draw the plot. Do not use ellipsis (...) to represent omitted data. Ensure column names are readable if necessary.
Please include a meaningful and interpretable pattern describing how the dependent variable Z changes with respect to the 2 independent variables X and Y across a two-dimensional space.

Topic: {topic}
Generated Description :

Figure 10: Prompt template used for description generation

Description Self-Evaluation Prompt Template

"system":
You are an expert in data visualization.
"user" :
Please check the correctness of given description for the given criteria.
Criteria 1. Check if the Data type in the description suits the Plot type (compatible).
Criteria 2. Check if the mentioned mathematical relation between variables suits the plot type.
Description : {description}
Answer if it satisfies each criteria in the following format:
"Criteria 1 : Yes/No"
"Criteria 2 : Yes/No"

Figure 11: Prompt template used for description self-evalution

Code Generation Prompt Template

"system":
You are an expert in python.
"user" :
Please write a python code to illustrate the plot in the given description. Make sure to include all the necessary libraries. Make sure to include all the necessary libraries. Please make labels in the plot readable if necessary. Please generate the python code in ``` ``` format.

Description : {description}
Generated Code :

Figure 12: Prompt template used for code generation

Cycle-Consistency Verification Prompt Template

Step 1. Regenerating the description given the code

The code generates a plot. Please generate a detailed and clear description of the plot.
Make sure to follow the following criteria:
1. The description should be detailed enough for one to redraw the plot using only the description.
2. Include every data point used in the plot in the description (or the CSV filename if mentioned).
3. Do not omit any detail from the plot.
Code : {code}
Generated Description :

Step 2. Comparing the original description with the regenerated description

Please determine if these two descriptions describe the same plot by considering the following criteria:
Criteria 1. Plot type should be equivalent.
Criteria 2. Data points should be equivalent.
Criteria 3. All key elements and details of the plot should be consistent between the two descriptions.

Description 1: {original description}
Description 2: {regenerated description}

Answer if each criterion is satisfied using the following format:
"Criteria 1 : Yes/No"
"Criteria 2 : Yes/No"
"Criteria 3 : Yes/No"

Figure 13: Prompt template used for cycle consistency verification

Data Table Generation Prompt Template

"system":
You are an expert in generating data tables
"user" :
Please write a Python script that generates a CSV file based on the given description. Ensure that the data points provided in the description are included in the generated CSV file. The column and row names should be relevant to the data being represented.
Focus on generating the code for the CSV file in this section and do not include any code related to chart generation. The output of the code should generate a set of random data points, consisting of {number} data points, that you believe would be suitable for the plot described in the description. Please generate the python code in ``` ``` format.

Description : {description}
Generated Code for CSV file:

Figure 14: Prompt template used for data table generation

Code for Data Table Generation Prompt Template

"system":
You are an expert in python.
"user" :
Please generate a CSV file based on the given description. Generate {number} random data points. Ensure that the data points provided in the description are included in the generated CSV file. The column and row names should be relevant to the data being represented. Do not use ellipsis (...) to represent omitted data. Please generate the CSV file in ``` ``` format.

Description : {description}
Generated CSV :

Figure 15: Prompt template used for code used for data table generation

Reasoning Step Generation Prompt Template

"system":
You are an expert in data visualization.
"user" :
I need an intermediate logical flow explaining why the following raw data table is best visualized using the provided description. Please write a proper intermediate reasoning step in the following format.
1. Characteristic of data and CSV file:
2. Possible plot types:
3. Most suitable plot type:
4. Further considerations for the description:

Description: {description}
Raw Data Table :{data table}
Reasoning Steps:

Figure 16: Prompt template used for reasoning step generation

D.2 Prompt Template for Tasks

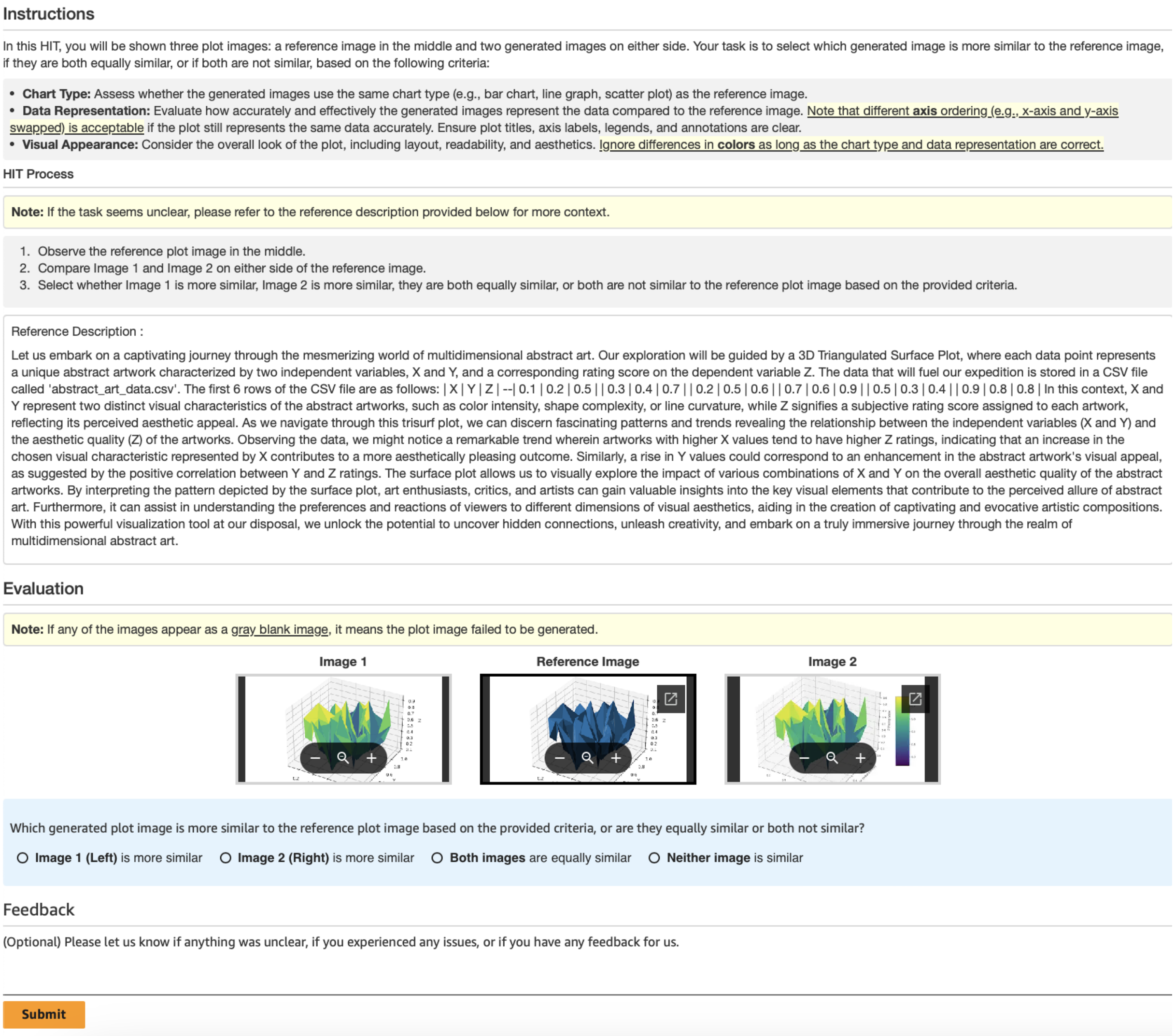
This section presents the prompt templates used for three tasks using the Text2Chart31 dataset, including description-to-chart, raw data-to-chart, and chart-to-description tasks. Figure 17 illustrates the prompt template used for the description-to-chart task, Figure 18 shows the template for the raw data-to-chart task, and Figure 19 presents the template used for the chart-to-description task.

Task 1: Description-to-Chart Task Prompt Template

"system":
You are good at generating complete python code from the given chart description.
"user" :
Your task is to generate a complete python code for the given description. Make sure to include all necessary libraries.
Description: {description}
Please generate the corresponding code that generates the plot that has the above description.
"assistant" :
Code:
```Python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

Figure 17: Prompt template used for description to chart task

**Task 2: Raw Data-to-Chart Task** Prompt Template

**"system":**
You an expert in chart generation and data visualization.
**"user":**
Given the Raw Data Table, generate the reasoning steps to determine the most suitable plot for visualizing the data, taking into account the characteristics of the data.
Raw Data Table in {name of CSV file}:
{data table}
Provide the reasoning steps in the following format:
1. Characteristics of the data and CSV file:
2. Possible plot types:
3. Most suitable plot type:
4. Further considerations for the description:
**"assistant" :**
Reasoning Steps : {reasoning step}
**"user":**
Given the reasoning step above and the raw data table in {name of CSV file}.
Please describe the plot you would generate to visualize this data, including:
Plot type, CSV file name, First 6 rows of the raw data table, Variables assigned to each axis and Any styling, formatting, or additional elements you would include.
Description :

Figure 18: Prompt template used for raw data-to-chart task

**Task 3: Chart-to-Description Task** Prompt Template

**"system":**
You are good at describing about the given data visualization code. Make sure when you describe a graph, mention the data points or csv file that are going to be used; otherwise, we won't be able to sketch the graph.
**"user":**
Your task is to generate a description of the chart based on the provided code, please make sure to include labels from the graph.
Code: {code}
Please generate the corresponding description.
**"assistant":**
Description:

Figure 19: Prompt template used for chart-to-description task
```

E Details on Human Evaluation

Figure 20 illustrates the user interface designed for the human evaluation task. The interface presents the crowd workers with a reference plot image and two generated plot images (Image 1 and Image 2) from different models, where the order of the generated images is randomly determined. The workers are asked to select one of the following options: Image 1 (Left) is more similar to the reference image, Image 2 (Right) is more similar to the reference image, both images are equally similar to the reference image, or neither image is similar to the reference image. The workers make their selection based on the similarity of the generated images to the reference image in terms of chart type, data representation, and visual appearance. We use Amazon Mechanical Turk and gather annotators from English speaking countries. We pay maximum $0.4 per HIT. We explain annotators that the provided answers are going to be used as a research purpose in our qualification HIT.



Instructions

In this HIT, you will be shown three plot images: a reference image in the middle and two generated images on either side. Your task is to select which generated image is more similar to the reference image, if they are both equally similar, or if both are not similar, based on the following criteria:

- **Chart Type:** Assess whether the generated images use the same chart type (e.g., bar chart, line graph, scatter plot) as the reference image.
- **Data Representation:** Evaluate how accurately and effectively the generated images represent the data compared to the reference image. Note that different **axis** ordering (e.g., x-axis and y-axis swapped) is acceptable if the plot still represents the same data accurately. Ensure plot titles, axis labels, legends, and annotations are clear.
- **Visual Appearance:** Consider the overall look of the plot, including layout, readability, and aesthetics. Ignore differences in **colors** as long as the chart type and data representation are correct.

HIT Process

Note: If the task seems unclear, please refer to the reference description provided below for more context.

1. Observe the reference plot image in the middle.
2. Compare Image 1 and Image 2 on either side of the reference image.
3. Select whether Image 1 is more similar, Image 2 is more similar, they are both equally similar, or both are not similar to the reference plot image based on the provided criteria.

Reference Description :

Let us embark on a captivating journey through the mesmerizing world of multidimensional abstract art. Our exploration will be guided by a 3D Triangulated Surface Plot, where each data point represents a unique abstract artwork characterized by two independent variables, X and Y, and a corresponding rating score on the dependent variable Z. The data that will fuel our expedition is stored in a CSV file called 'abstract_art_data.csv'. The first 6 rows of the CSV file are as follows: | X | Y | Z | --| 0.1 | 0.2 | 0.5 | | 0.3 | 0.4 | 0.7 | | 0.2 | 0.5 | 0.6 | | 0.7 | 0.6 | 0.9 | | 0.5 | 0.3 | 0.4 | | 0.9 | 0.8 | 0.8 | In this context, X and Y represent two distinct visual characteristics of the abstract artworks, such as color intensity, shape complexity, or line curvature, while Z signifies a subjective rating score assigned to each artwork, reflecting its perceived aesthetic appeal. As we navigate through this trisurf plot, we can discern fascinating patterns and trends revealing the relationship between the independent variables (X and Y) and the aesthetic quality (Z) of the artworks. Observing the data, we might notice a remarkable trend wherein artworks with higher X values tend to have higher Z ratings, indicating that an increase in the chosen visual characteristic represented by X contributes to a more aesthetically pleasing outcome. Similarly, a rise in Y values could correspond to an enhancement in the abstract artwork's visual appeal, as suggested by the positive correlation between Y and Z ratings. The surface plot allows us to visually explore the impact of various combinations of X and Y on the overall aesthetic quality of the abstract artworks. By interpreting the pattern depicted by the surface plot, art enthusiasts, critics, and artists can gain valuable insights into the key visual elements that contribute to the perceived allure of abstract art. Furthermore, it can assist in understanding the preferences and reactions of viewers to different dimensions of visual aesthetics, aiding in the creation of captivating and evocative artistic compositions. With this powerful visualization tool at our disposal, we unlock the potential to uncover hidden connections, unleash creativity, and embark on a truly immersive journey through the realm of multidimensional abstract art.

Evaluation

Note: If any of the images appear as a gray blank image, it means the plot image failed to be generated.

Image 1 | Reference Image | Image 2

Which generated plot image is more similar to the reference plot image based on the provided criteria, or are they equally similar or both not similar?

○ **Image 1 (Left)** is more similar ○ **Image 2 (Right)** is more similar ○ **Both images** are equally similar ○ **Neither image** is similar

Feedback

(Optional) Please let us know if anything was unclear, if you experienced any issues, or if you have any feedback for us.

Submit

Figure 20: User interface for human evaluation comparing generated plot images.