

CHARTREF: BENCHMARKING FINE-GRAINED VISUAL ELEMENT LOCALIZATION IN CHARTS

Anonymous authors

Paper under double-blind review

ABSTRACT

Humans interpret charts by first localizing visual elements—such as bars, markers, and segments—before reasoning over the data. In contrast, current multimodal models primarily rely on text reasoning, limiting their ability to leverage fine-grained visual information. To address this, we introduce CHARTREF, a dataset of 38,846 questions, answers, referential expressions, and bounding boxes across 1,141 figures and 11 chart types. Our key insight is that the chart-rendering code makes it possible to generate visual element localizations that are aligned with question-answer pairs. Given only the Python script, a large language model infers the semantics of plotted data, maps data series to visual encodings, and programmatically extracts bounding boxes, yielding visual annotations for charts at scale. Using CHARTREF, we benchmark multimodal LLMs and find 3–7% accuracy improvements on chart question answering when models are provided with ground-truth bounding boxes. We further evaluate vision and multimodal models on chart object detection and visual grounding, where models localize an expression referring to the data in the chart. While object detection exceeds 80 AP@50, visual grounding accuracy is only 2.8, revealing a significant gap: current models can recognize chart elements perceptually but struggle to integrate context cues from axes, legends, labels, and data to ground fine-grained textual references.

1 INTRODUCTION

When reading charts, humans localize relevant visual elements before extracting insights, grounding their reasoning in the visualized data. Building AI models with similar chart localization and reasoning capabilities propel applications in finance (Shu et al., 2025) and healthcare (Lee et al., 2024), support interactive systems where models communicate with humans through visual annotations (Hu et al., 2024), and advance document understanding (Ma et al., 2024) and scientific discovery (Lála et al., 2023; Yang et al., 2023b). Despite the importance of chart grounding, prior work has focused exclusively on evaluating models through question answering (Masry et al., 2022; Wang et al., 2024; Xia et al., 2025; Tang et al., 2025), which only measures the accuracy of textual answers and overlooks whether models actually localize and reason over the relevant visual evidence.

While vision-language models are proficient at localizing objects in natural scenes (Liu et al., 2024; Li et al., 2022; Zhong et al., 2022; Xiao et al., 2024; Fu et al., 2025), grounding phrases in charts requires fundamentally different visual reasoning skills, such as the ability to interpret elements such as legends and axis labels, understand coordinate systems, and navigate multi-panel subplot arrangements. For example, in Figure 1 (right), identifying the visual marker that corresponds to the phrase “average response for Control Group in session 5” requires understanding the correspondence between the legend and the markers, parsing the correct data series, and comparing the x-axis to the plotted data. This makes chart localization a process that current models have limited exposure to, as their pretraining focuses primarily on natural image-text alignment rather than supervision over chart-specific structures.

Additionally, unlike visual grounding for natural images, there are relatively few datasets for evaluating and enhancing chart visual grounding. Previous work has focused intensely on treating charts as a computer vision problem (Methani et al., 2020; Suri et al., 2025), applying off-the-shelf detection models to raw images to extract bounding box annotations. These approaches inherently limit

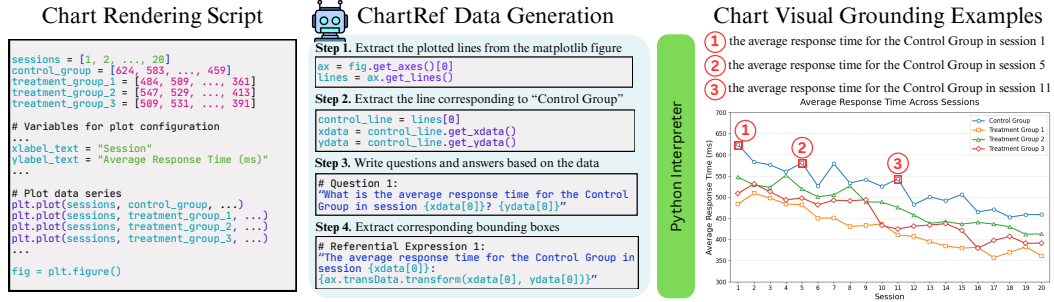


Figure 1: **CHARTREF dataset**. To generate a large-scale, diverse dataset of bounding box annotations paired with questions, answers, and referential expressions, we propose a data generation pipeline that takes as input the Python rendering script. We leverage the insight that the code representation contains information on the semantics of the plotted data, e.g. that `ax.get_lines()[0]` corresponds to the data for "Control Group", enabling generation of accurate questions and referential expressions that are grounded in contextual cues such as labels and ticks. Additionally, the code provides a means to extract bounding boxes for visual representations of the data by querying the underlying figure rendering. Our data generation results in CHARTREF, a benchmark for evaluating the chart visual grounding capabilities of multimodal models.

the diversity of the data to a select few chart types, such as bars, lines, and pie charts, to which standard object detection can be directly applied.

To address these shortcomings, we leverage the power of code as an intermediate medium to procedurally generate potentially unlimited source of chart images, visual element localizations, questions, and answers. This results in CHARTREF, a dataset of 38,846 paired examples across 11 figure types. Specifically, our data generation takes as input Python rendering scripts that encompass charts of diverse types and visual complexity. Our key insight is that without relying on the rendered image and given only the rendering script, a large language model can infer the semantic meaning of plotted data, associate data series with their corresponding visual encodings, and programmatically extract bounding boxes by querying the figure rendering as shown in Figure 1.

Using CHARTREF, we first evaluate multimodal foundation models on chart-question answering with and without the ground truth bounding boxes. In comparison to both standard prompting and chain-of-thought, models achieve a significant improvement in accuracy of 3-7% when given the ground truth annotation, motivating the development of models capable of fine-grained chart element localization. We next evaluate the capabilities of state-of-the-art vision and multimodal models on two tasks: 1) object detection, where the model detects all visual elements corresponding to the plotted data, and 2) visual grounding, where the model localizes an expression referring to a data point in the chart. Finetuning models on CHARTREF enhances performance significantly on both tasks – 10.6 AP@50 to 80.6 AP@50 for object detection and 0.3 Acc@1 to 2.8 Acc@1 for visual grounding. However, visual grounding performance is still far below that of object detection. These results highlight a critical gap in chart understanding and motivate future work on improving vision–language alignment to enable fine-grained chart visual grounding.

2 CHARTREF DATA CURATION

In Section 2.1, we first detail our pipeline for generating chart questions and answers with corresponding referential expressions and bounding boxes for the visual elements. In Section 2.2, we describe our procedure for postprocessing the generated questions and answers to facilitate evaluating numerical answers and eliminate ambiguous questions with a multimodal LLM verifier. In Section 2.3, we detail key statistics of CHARTREF.

2.1 DATA GENERATION

Given a matplotlib Python script that renders a chart, our data generation pipeline uses an LLM to synthesize code that extracts bounding boxes from the rendered visualization. This approach

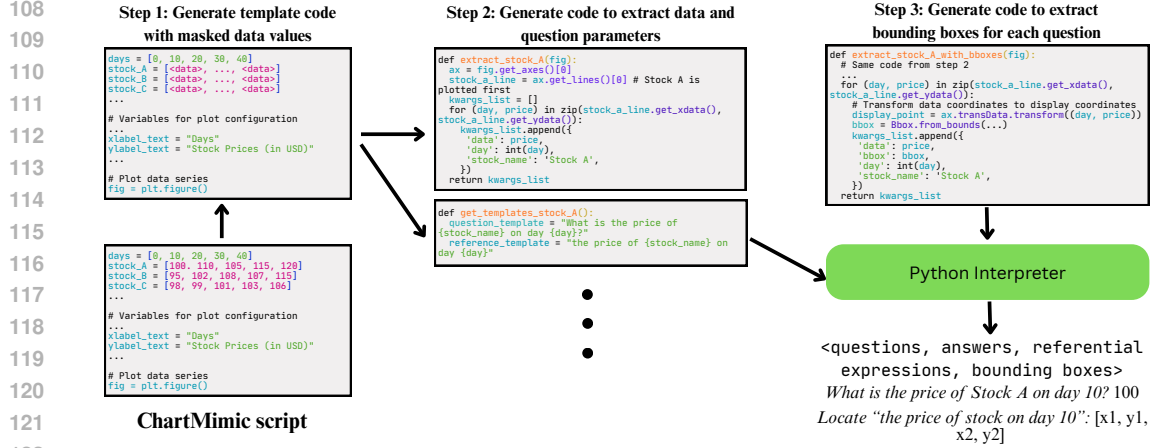


Figure 2: **Data generation pipeline.** Given the Python script from ChartMimic, we generate paired $\langle$ question, answer, referential expression, bounding box $\rangle$ using the following data generation pipeline. In Step 1, we prompt a LLM to mask out the plotted data values, while preserving information required to interpret the chart, such as labels and coordinates at which the data points appear. In Step 2, the model identifies all data series that are masked out and writes code that extracts the data values and identifying parameters, as well as the question and referential expression templates. Applying the identifying parameters to the templates results in the questions and referential expressions that align with the data values. In Step 3, the model modifies the data extraction code from Step 2 to additionally extract the bounding boxes.

is necessary because unlike SVG or HTML formats where spatial coordinates are directly embedded in the markup, matplotlib scripts do not explicitly expose this spatial information. Instead, the figure object (e.g., `figure = plt.figure(...)`), stores both the ground truth data and the coordinates of the visualized data in image space. To extract both the underlying data and their bounding boxes, an LLM generates code that programmatically queries the figure object’s properties. For example, the axes can be extracted with `figure.get_axes()`, and all line objects can be further collected with `ax.get_lines()` for each axis. This allows access to the data points via `line.get_xdata()` and `line.get_ydata()`, which can be converted to pixel coordinates using `ax.transData.transform()`.

Concretely, our data generation pipeline is as follows. As shown in Figure 2, we first prompt an LLM to generate template versions of a given Python script, where the plotted data is masked out but all other code is unchanged. This step ensures that the model writes code that extracts data directly from the figure object rather than hardcoding values, while still providing sufficient context—labels, axis titles, and visualization parameters—to infer the correspondence between data values and their textual descriptions. In the next stage, the LLM is prompted with the template to first identify all data series that are plotted by the code and for each data series, the LLM synthesizes a function that extracts the visualized data and parameters specifying how each series is visualized. These parameters are then used to instantiate question and referential expression templates. For example, if a data series represents “Stock A” values over a set of days, the generated questions correspond to individual data values, such as “What is the value of Stock A at day 10?”, with the referential expression highlighting the corresponding visual element in the chart. After the data is extracted, the model is prompted to augment the data extraction code to additionally extract the bounding box for the visual element that corresponds to the visual element. When there are no explicit markers, we instruct the model to generate a 10×10 bounding box. See Fig 3 (top) for an overview of the types of questions, answers, and bounding boxes generated for each chart type. We use Claude Sonnet 4 (Anthropic, 2025) for all stages of the pipeline. See Appendix B for the data generation prompts and Appendix E for examples of the data in CHARTREF.

We organize the generated data into two tasks: chart question answering and chart element localization as shown in Figure 3 (bottom). To assess chart question answering with and without bounding boxes, we use the generated questions, answers, and bounding boxes. For chart element localization,

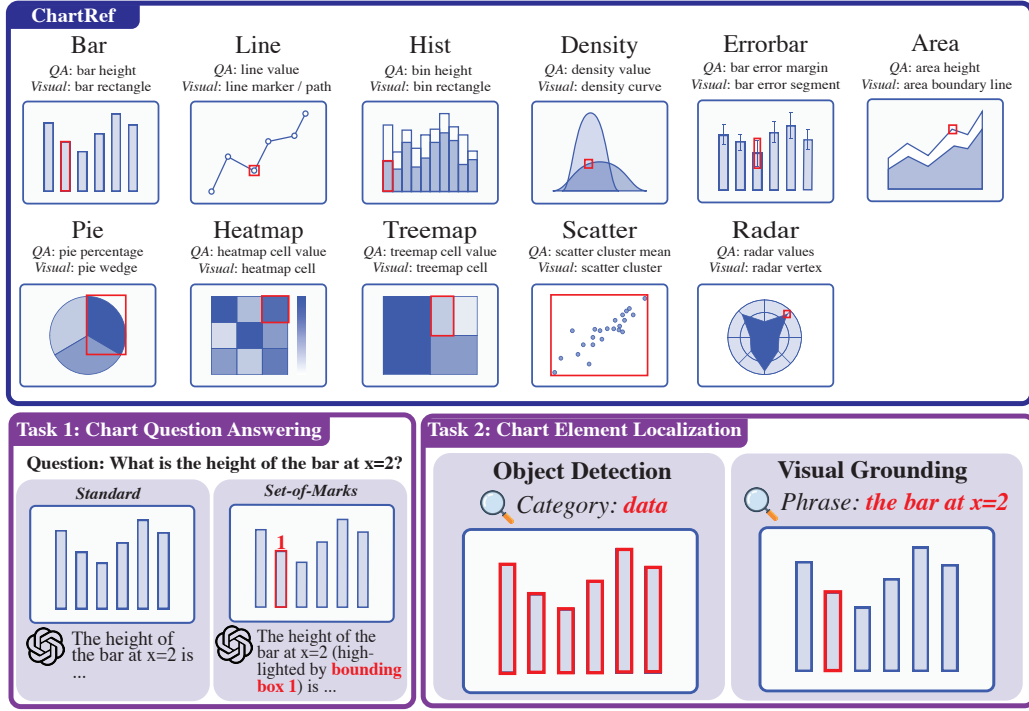


Figure 3: **CHARTREF overview**. **Top**: CHARTREF consists of figures across 11 chart types, with questions that require correctly extracting the underlying data and are paired with the corresponding answers, referential expressions, and bounding boxes around individual visual elements. For example, questions for bar plots ask about values that are visualized by the bar rectangle. **Bottom Left**: CHARTREF allows us to investigate the capabilities of multimodal foundation models on chart question answering, with both the original image and annotated with the ground truth bounding box. **Bottom Right**: With the ground truth bounding boxes, we benchmark models on two vision tasks: object detection, where all visual elements representing the data are identified, and visual grounding, where given a phrase, the model localizes a specific data element.

we consider two vision tasks: object detection and visual grounding. For object detection, we use all annotated bounding boxes for a given figure as ground truth. For visual grounding, by the nature of our data generation, there is a one-to-one mapping between referential expressions and bounding boxes.

2.2 DATA POSTPROCESSING

After generating (question, answer, referential expression, bounding box) tuples, we post-process the questions and answers as follows. We employ a multimodal LLM to determine the appropriate precision level for each ground truth answer and establish an error margin based on the spacing of relevant axis ticks. For numerical answers, we consider a prediction correct if $|\text{round}(y_g, p) - y_p| \leq \epsilon$, where y_g is the ground truth, y_p is the predicted answer, p is the determined precision level, and ϵ is the error margin. For text answers, we use exact string matching.

We then validate the quality of the questions using a multimodal LLM to classify each question into one of three categories: 1) ambiguous questions, where there is not enough context in the chart to answer the question, 2) defective questions, where the answer is trivially answered without looking

Table 1: **Data generation statistics.**

Statistic	Value
<i>Total</i>	
# of Examples	44345
# of Figures	1259
# of Chart Types	11
<i>Object Detection</i>	
# of Annotations	40336
# of Figures	1148
<i>Visual Grounding</i>	
# of Examples	38846
# of Figures	1141

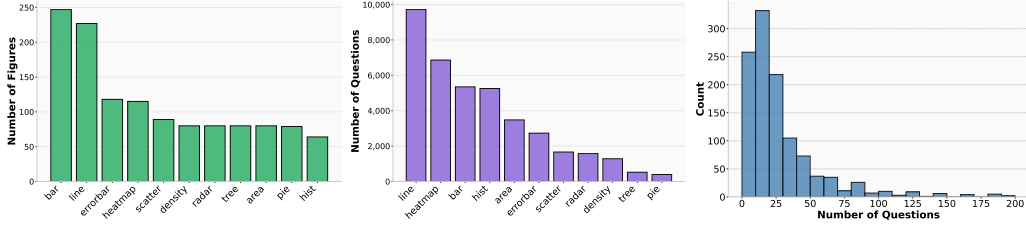


Figure 4: **CHARTREF statistics.** **Left:** The number of figures per type shows that the dataset is balanced across figure types. **Middle:** The number of questions per type reflects that certain figure types tend to reflect much denser information – for example a pie chart with several wedges will have fewer examples than a line plot reflecting tens or hundreds of individual examples. **Right:** The distribution of the number of questions show that the plots are densely annotated with bounding boxes, with many figures having over 20 paired examples.

at the chart, and 3) valid questions that do not fall into the above two categories. This ensures that the resulting questions do not reference information that is not available in the chart image. See Table 1 for the statistics. We use Claude Sonnet 4 as the multimodal LLM and detail the postprocessing prompts in Appendix C.

2.3 DATASET STATISTICS

We apply our pipeline to ChartMimic (Yang et al., 2025), a diverse dataset of human-curated Python chart rendering scripts with 2400 total figures across 22 different chart types. We select 11 of the chart types for which we could generate accurate bounding box annotations. In total, we generated 44345 paired question, answer, referential expression, and bounding box from 1259 figures. For object detection, we filtered out bounding box annotations with greater than 0.95 IoU overlap and did not include density plots, which represent continuous functions or distributions rather than discrete points, resulting in 40336 bounding boxes across 1148 figures. For visual grounding, we filtered out ambiguous and defective questions as discussed in Section 2.2, in addition to duplicate referential expressions. This resulted in a total of 38846 filtered examples across 1141 figures. In Figure 4, we analyze the distribution of figures per type, showing that although there is a higher proportion of bar and line plots, our data is balanced. The number of questions per figure type reflects the information density of plots – for example, line plots contain many individual data points. The distribution of the number of questions shows that the figures are densely annotated, with many having over 20 paired examples.

3 RELATED WORK

We outline prior work in assessing chart question answering, which have paired (chart image, question, answer) but lack visual annotations, as well as work that curates visual annotations for charts and other structured visuals, but often are limited to a subset of simple chart types, lack alignment between annotations and question-answers, or require human annotators.

Chart Question Answering Datasets. One line of work (Masry et al., 2022; Xu et al., 2024; Wang et al., 2024; Masry et al., 2024; 2025; Xia et al., 2025; Tang et al., 2025) focuses on evaluating chart question answering (CQA) that involve visual and logical reasoning over charts. Some benchmarks are human annotated – for example, ChartQA (Masry et al., 2022) consists of bar, line, and pie charts collected through web-crawling and human-annotated questions and answers. ChartQAPro (Masry et al., 2025) has been proposed as a CQA dataset with enhanced visual diversity. CharXiv (Wang et al., 2024) sources its charts from arXiv papers and evaluates multimodal LLMs on descriptive questions on extracting information from basic chart elements and reasoning questions requiring synthesizing information across multiple visual elements. ChartMuseum (Tang et al., 2025) contains expert-annotated questions focusing on visual reasoning that is difficult to perform with textual chain-of-thought. Other benchmarks are curated in an automated manner, involving multimodal LLMs in the loop. ChartX (Xia et al., 2025) uses GPT-4 to automatically generate chart title and

types that are aligned with CSV data for chart perception questions and task templates to generate cognition questions that builds on the perception questions. ChartBench (Xu et al., 2024) first generates the chart using LLMs to synthesize chart themes and JSON data and subsequently synthesizes questions and answers from chart templates. ChartGemma (Masry et al., 2024) proposes a data curation pipeline that leverages a multimodal LLM to directly generate questions from chart images.

Visual Annotation Datasets. Prior work has collected bounding box annotations for chart elements but typically only for a narrow range of chart types. FigureQA (Kahou et al., 2018) generates bounding box annotations from Bokeh (de Ven, 2018) for bar, lines, and pie charts, but these annotations are limited in visual diversity and lack alignment between annotations and questions. Other works leverage off-the-shelf vision models, such as PlotQA (Methani et al., 2020), which uses Faster-RCNN (Ren et al., 2017) to detect all bars, lines, and chart interpretation elements such as titles and labels. ChartLens (Suri et al., 2025) uses instance segmentation to annotate bar and pie charts and a specialized Transformer model to detect lines. Because these approaches depend on computer vision models for localization, they are inherently constrained to chart types where such models perform reliably. Orthogonal to our work, prior works (Battle et al., 2018; Zhu et al., 2025a) have developed pipelines for extracting annotations from SVG scripts, which have spatial coordinates directly embedded in the source code. Beagle (Battle et al., 2018) extracts circles, rectangles, line, and paths from charts. OrionBench (Zhu et al., 2025a) focuses on automatically synthesizing annotations for chart and human-readable objects embedded in infographics. Recently, RADAR (Rani et al., 2025) has curated human-annotated bounding boxes that correspond to individual CQA reasoning steps.

4 EXPERIMENTS

4.1 DO VISUAL ANNOTATIONS ENHANCE CHART PERCEPTION?

To motivate the use of bounding boxes to locate relevant visual elements for chart question answering, we evaluate the performance of multimodal LLMs on our synthesized dataset, where questions and answers are paired with ground truth bounding boxes that localizes the visual element corresponding to the answer. This enables us to evaluate the impact of annotating the ground truth bounding box on chart perception. We evaluate three closed source models, GPT-5 (OpenAI, 2025b), GPT-o3 (OpenAI, 2025a), Gemini-2.5-Pro (Comanici et al., 2025), and two open source models, Qwen2.5-VL (Bai et al., 2025), and InternVL3 (Zhu et al., 2025b). For all models, we experiment with three settings: standard prompting, chain-of-thought, and set-of-marks (SoM) prompting (Yang et al., 2023a), a visual prompting technique designed to ground model responses in visual cues. In our SoM prompting, the ground truth bounding box is overlaid on the image with a numerical label, allowing models to reference it in their answers. Results are shown in Table 2.

In comparison to standard prompting, set-of-marks prompting enhances performance by 2-3% for open-source models and 3-7% for closed-source models. In contrast, chain-of-thought prompting does not improve performance and even degrades it for Qwen2.5-VL, InternVL3, and GPT-o3. These results demonstrate that when the relevant chart visual elements are localized, downstream question answering is improved as the model is able to ground its reasoning in the bounding boxes.

4.2 OBJECT DETECTION

Using the bounding boxes, we evaluate vision models on object detection, where the task is to detect all visual elements corresponding to discrete data elements. We consider two types of models: traditional object detection models that must be trained to evaluate new categories, as well as foundation models capable of zero-shot detection. For traditional object detection models, we evaluate YOLOv3 (Redmon & Farhadi, 2018), Faster-RCNN (Ren et al., 2017), RTMDet (Lyu et al., 2022),

Table 2: **Set-of-Marks with bounding boxes enhances chart perception.**

Model	Direct	CoT	SoM
Qwen2.5-VL-72B	67.6	66.6	70.2
InternVL3-78B	67.9	66.5	69.6
GPT-5	80.0	80.0	83.8
GPT-o3	78.4	76.8	81.4
Gemini-2.5-Pro	72.9	-	79.7

Table 3: **Object detection performance. Left:** Traditional object detection and foundation models finetuned on CHARTREF. **Right:** Zero-shot inference with foundation models.

Model	AP	AP ₅₀	AP ₇₅	AR
YOLOv3	25.0	52.8	20.4	37.7
Faster-RCNN	54.1	60.4	60.4	67.4
Co-DETR	64.0	79.8	69.7	74.8
RTMDet	38.0	56.2	40.1	16.0
MM-GD	61.5	79.1	67.6	72.8
LLMDet	64.2	80.6	68.5	75.1

Model	AP	AP ₅₀	AP ₇₅	AR
GroundingDINO (T)	5.2	8.7	5.1	20.3
GroundingDINO (B)	4.7	7.8	4.7	20.2
GLIP (T)	3.8	6.6	3.8	13.2
GLIP (L)	5.8	8.7	5.8	16.3
MM-GD (T)	4.1	8.0	3.6	26.4
MM-GD (B)	5.8	10.6	5.3	27.1
MM-GD (L)	3.6	6.2	3.7	26.7
LLMDet (T)	5.8	10.6	5.3	28.2
LLMDet (B)	6.3	12.2	5.6	30.0
LLMDet (L)	3.3	6.6	2.9	23.8

and CoDeTR (Zong et al., 2023). For vision models, we evaluate Grounding DINO (Liu et al., 2024), MM-Grounding-DINO (MM-GD) (Zhao et al., 2024), GLIP (Li et al., 2022), and LLMDet (Fu et al., 2025). As these models are not adapted to charts, we evaluate both the pretrained and finetuned models on our dataset.

Results and Analysis. Results for zero-shot inference and finetuned models on CHARTREF are shown in Table 3. For zero-shot inference, we benchmark vision foundation models across different sizes and show that the best performance achieved is 6.3 AP and 30.0 AR. Notably, increasing model size does not always enhance performance, demonstrating the out-of-domain shift from natural images to chart object detection. After finetuning on CHARTREF, AP improves from 6.3 to 64.2 and AR improves from 30.0 for 80.6 for LLMDet, with similar improvements seen in MM-GD. The best traditional object detection model, Co-DETR, achieves 64.0 AP and 74.8 AR.

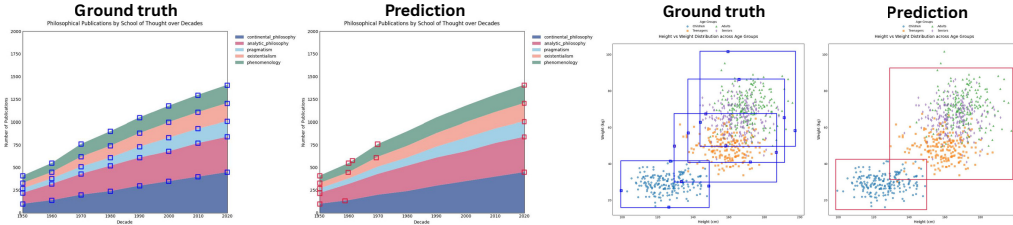


Figure 5: **Object Detection Example Errors.** We visualize errors made by Co-DETR, with ground truth bounding boxes displayed in blue and predicted bounding boxes displayed in red. **Left:** For area charts, data is plotted at each x-tick. However, the predictions are not able to capture the correct data points and are missing bounding boxes where the slope of the lines are constant and lack salient features. **Right:** For scatter plots, the ground truth visual elements include clusters of data points. Although the predictions are able to capture the blue cluster as it is well-separated from the other data points, the model does not correctly separate the overlapping orange, purple, and green clusters.

Additionally, we conduct a qualitative analysis in Figure 5 of Co-DETR’s errors. We find that the models have an overreliance on salient perceptual features. For example, on the left chart of Fig. 5, the model struggles to identify individual data points at the x-ticks. On the right, Co-DETR does not correctly separate overlapping clusters that denote separate data series.

We further analyze Co-DETR’s performance across different chart types in Table 4. We find that the lowest performance is for line and scatter plots, as the bounding boxes around individual markers in the line or scatter plot are typically small, whereas the model performs best for treemaps, pie charts, and bars, which contain fewer fine-grained bounding boxes and are represented by well-defined shapes, e.g. rectangles for treemaps and bars and pie wedges for bar charts.

Table 4: **Object detection performance across chart types.** Both AP@50 and AR scores are above 0.8 for errorbar, bar, tree, pie, and heatmap plots. The data for these plot types are represented by larger geometric shapes, such as rectangles, wedges, and grid cells, making it easier for the model to detect. In contrast, detecting individual data points for line and scatter plots are more difficult, as these visual elements are more fine-grained.

Metric	Tree	Bar	Errorbar	Line	Radar	Hist	Area	Scatter	Heatmap	Pie
AP ₅₀	92.3	91.3	80.0	60.4	74.3	84.8	85.6	69.4	95.6	82.4
AR	99.0	96.4	86.7	50.5	65.8	78.2	65.5	69.8	92.7	98.4

4.3 VISUAL GROUNDING

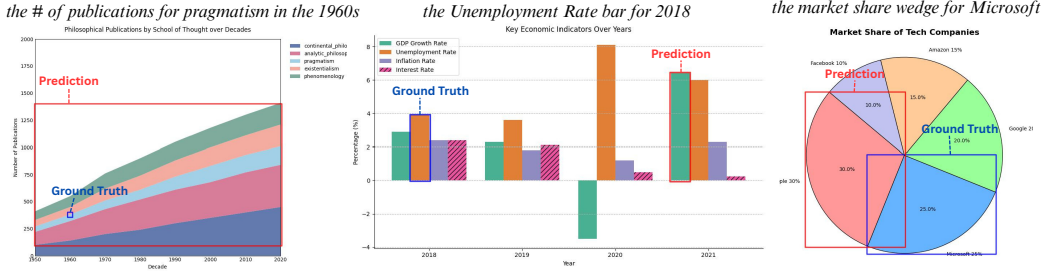
We benchmark both vision models and multimodal LLMs on chart visual grounding, where given a phrase, such as “the bar for Model A”, the model outputs the correct bounding box that localizes the phrase. For vision models, we consider the same vision foundation models discussed in Section 4.2. We evaluate both zero-shot capabilities as well as finetune models on our CHARTREF. For reference, we additionally benchmark multimodal LLMs with zero-shot referential capabilities, including GPT-4o, Qwen2.5-VL, and InternVL3. We show the results in Table 5.

Results and Analysis. Among zero-shot vision models, performance is low, with all models achieving below 1 Acc@1. Larger vision models do not offer significant performance improvements, demonstrating that current models are not well-adapted to the task. In comparison, multimodal LLMs perform similarly to the zero-shot foundation models, in spite of the increased amount of training on diverse image sources, which often include chart data, and on chart question-answering. Finetuning vision models yields a significant improvement: MM-GroundingDINO improves from 0.3 to 2.8 and LLMDet’s accuracy improves from 0.6 to 2.3. However, the performance on visual grounding is still far below the object detection capabilities, which are close to perfect. This suggests that even though the vision backbone can be adapted to detect visual elements in charts, both vision foundation models and multimodal LLMs lack the vision-language alignment to understand the semantic information of charts beyond perceptual features.

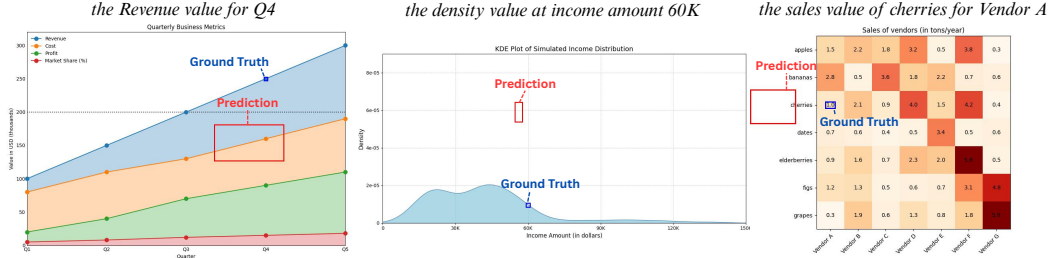
For MM-GroundingDINO and GPT-4o, we visualize the visual grounding results in Figure 6. We find that MM-GroundingDINO tends to predict bounding boxes that do correspond to a visual element in the plot; however, it is not able to correctly identify the visual element referred to by the text. In contrast, GPT-4o’s predictions tend to be imprecise, even in the presence of visual markers and text. These findings highlight that the main bottleneck lies in linking text references to the correct visual elements: vision foundation models can localize objects but fail to disambiguate which element is referenced, while multimodal LLMs struggles with precise localization altogether. Closing this gap will require improving models’ ability to jointly reason over visual structure and textual cues, rather than relying solely on perceptual or linguistic features.

Table 5: **Visual grounding performance**

Model	Acc@1	Acc@5	Acc@10
<i>Zero-shot</i>			
GroundingDINO (T)	0.6	2.8	4.9
GroundingDINO (B)	0.7	3.0	5.3
GLIP (T)	0.4	0.5	0.5
GLIP (L)	0.5	0.5	0.6
MM-GroundingDino (T)	0.3	2.7	5.3
MM-GroundingDino (B)	0.2	2.8	5.3
MM-GroundingDino (L)	0.3	3.1	5.9
LLMDet (T)	0.5	3.9	7.4
LLMDet (B)	0.6	3.6	6.9
LLMDet (L)	0.6	3.3	5.7
<i>Finetuned</i>			
MM-GroundingDino (T)	2.8	13.5	22.9
LLMDet (T)	2.3	12.0	21.3
GPT-4o	0.7	-	-
Qwen-2.5-VL-72B	0.5	-	-
InternVL3-78B	0.2	-	-



(a) Examples of MM-GroundingDINO visual grounding errors.



(b) Examples of GPT-4o visual grounding errors.

Figure 6: **Visual grounding errors.** Although MM-GroundingDINO tends to produce bounding boxes that are more closely aligned to the visual elements in the plot, it is not able to localize the correct visual element. **Top Left:** MM-GroundingDINO predicts a bounding box that contains multiple data points. **Top Middle:** The predicted bar does not match ground truth. **Top Right.** Even for a plot that represents only 6 distinct data points, MM-GroundingDINO localizes the wrong pie wedge. In contrast, GPT-4o’s predictions often do not precisely localize the prediction. **Bottom Left:** The predicted bounding box is vertically offset from the ground truth and is also much larger than the visualized markers. **Bottom Middle:** GPT-4o’s prediction is vertically offset and localizes an empty region of the plot. **Bottom Right:** The prediction is horizontally offset from the ground truth.

5 CONCLUSION

In this work, we present CHARTREF, a large scale dataset of 38846 paired (chart question, answer, referential expression, and bounding box) across 11 figure types. Our data curation pipeline leverages the chart’s Python rendering code to programatically extract bounding boxes of visual elements that are aligned with questions and answers. With CHARTREF, we demonstrate that access to ground-truth bounding boxes improves chart question answering, motivating approaches capable of chart visual grounding. We thus benchmark vision models and multimodal LLMs on chart object detection and chart visual grounding. Through finetuning on CHARTREF, vision models can be adapted to detect all visual elements corresponding to the underlying data. However, both vision models and multimodal LLMs struggle to achieve comparable visual grounding performance when given a referential phrase. This gap highlights that chart visual grounding requires novel advances in text-vision alignment that would allow models to integrate diverse contextual cues, such as the legend, axis ticks, and subplot arrangement, to localize fine-grained information in charts. CHARTREF serves as a tool for both evaluating and training models on chart visual grounding, inspiring future work in models capable of human-like visual grounding.

Limitations. Because CHARTREF is generated from Python rendering code, it may not fully capture the stylistic variability, noise, or imperfections of real-world charts, potentially limiting model generalization to figures from research papers, reports, or scanned images. Additionally, the referential expressions in CHARTREF only require extracting individual data from the chart and do not cover more complex structures, such as comparisons or relationships between multiple data points. Finally, while the dataset spans 11 chart types, it does not encompass all visualization formats, including network diagrams, 3D plots, or interactive charts.

6 ETHICS STATEMENT

Our contribution represents a step towards developing models capable of localizing relevant visual elements in charts, which enhances the interpretability of model outputs and supports more transparent AI systems. This advancement contributes to society by enabling better understanding and validation of automated chart analysis, ultimately benefiting applications in data visualization, accessibility, and decision-making processes.

We have carefully reviewed all applicable ethical guidelines and believe our work adheres to the principles of scientific excellence, transparency, and responsible research conduct. Our dataset generation uses ChartMimic as input, which is available under the Apache 2.0 license, and we will adhere to the terms of this license when releasing CHARTREF, thus respecting intellectual property, privacy, and confidentiality guidelines.

7 REPRODUCIBILITY STATEMENT

To ensure reproducibility of our work, we provide comprehensive details of our methodology and experimental setup. The main text includes a detailed overview of our data generation pipeline, explaining the key steps for creating CHARTREF. Appendix B contains all prompts used in our LLM-based data curation process, enabling the replication of our dataset generation approach. Additionally, we detail experimental settings, hyperparameters, and evaluation protocols in Appendix D. We plan to release CHARTREF and code upon publication to facilitate further research and reproduction of our results.

REFERENCES

- Anthropic. Introducing Claude 4. <https://www.anthropic.com/news/claude-4>, May 2025.
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report. *CoRR*, abs/2502.13923, 2025. doi: 10.48550/ARXIV.2502.13923.
- Leilani Battle, Peitong Duan, Zachery Miranda, Dana Mukusheva, Remco Chang, and Michael Stonebraker. Beagle: Automated Extraction and Interpretation of Visualizations from the Web. In Regan L. Mandryk, Mark Hancock, Mark Perry, and Anna L. Cox (eds.), *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, pp. 594. ACM, 2018. doi: 10.1145/3173574.3174168.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Naveen Sachdeva, Inderjit S. Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, Ahmed Omran, Nikunj Saunshi, Dara Bahri, Gaurav Mishra, Eric Chu, Toby Boyd, Brad Hekman, Aaron Parisi, Chaoyi Zhang, Kornrapphop Kawintiranon, Tania Bedrax-Weiss, Oliver Wang, Ya Xu, Ollie Purkiss, Uri Mendlovic, Ilai Deutel, Nam Nguyen, Adam Langley, Flip Korn, Lucia Rossazza, Alexandre Ramé, Sagar Waghmare, Helen Miller, Nathan Byrd, Ashrith Sheshan, Raia Hadsell Sangnie Bhardwaj, Pawel Janus, Tero Rissa, Dan Horgan, Sharon Silver, Ayzaan Wahid, Sergey Brin, Yves Raimond, Klemen Kloboves, Cindy Wang, Nitesh Bharadwaj Gundavarapu, Ilia Shumailov, Bo Wang, Mantas Pajarskas, Joe Heyward, Martin Nikoltchev, Maciej Kula, Hao Zhou, Zachary Garrett, Sushant Kafle, Sercan Arik, Ankita Goel, Mingyao Yang, Jiho Park, Koji Kojima, Parsa Mahmoudieh, Koray Kavukcuoglu, Grace Chen, Doug Fritz, Anton Bulyenov, Sudeshna Roy, Dimitris Paparas, Hadar Shemtov, Bo-Juen Chen, Robin Strudel, David Reitter, Aurko Roy, Andrey Vlasov, Changwan Ryu, Chas Leichner, Haichuan Yang, Zelda Mariet, Denis Vnukov, Tim Sohn, Amy Stuart, Wei Liang, Minmin Chen, Praynaa Rawlani, Christy Koh, JD Co-Reyes, Guangda Lai, Praseem Banzal, Dimitrios Vytiniotis, Jieru Mei, and Mu Cai. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. *CoRR*, abs/2507.06261, 2025. doi: 10.48550/ARXIV.2507.06261.

- Bryan Van de Ven. Bokeh. <https://bokeh.org/>, 2018.
- Shenghao Fu, Qize Yang, Qijie Mo, Junkai Yan, Xihan Wei, Jingke Meng, Xiaohua Xie, and Wei-Shi Zheng. LLMDet: Learning Strong Open-Vocabulary Object Detectors under the Supervision of Large Language Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 14987–14997. Computer Vision Foundation / IEEE, 2025. doi: 10.1109/CVPR52734.2025.01396.
- Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual Sketchpad: Sketching as a Visual Chain of Thought for Multimodal Language Models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.
- Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Akos Kadar, Adam Trischler, and Yoshua Bengio. FigureQA: An Annotated Figure Dataset for Visual Reasoning, February 2018.
- Jakub Lála, Odhran O’Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G. Rodrigues, and Andrew D. White. PaperQA: Retrieval-Augmented Generative Agent for Scientific Research. *CoRR*, abs/2312.07559, 2023. doi: 10.48550/ARXIV.2312.07559.
- Chanseo Lee, Kimon A. Vogt, and Sonu Kumar. Prospects for AI clinical summarization to reduce the burden of patient chart review. *Frontiers in Digital Health*, 6, November 2024. ISSN 2673-253X. doi: 10.3389/fdgth.2024.1475092.
- Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. Grounded Language-Image Pre-training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 10955–10965. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01069.
- Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with Grounded Pre-training for Open-Set Object Detection. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XLVII*, volume 15105 of *Lecture Notes in Computer Science*, pp. 38–55. Springer, 2024. doi: 10.1007/978-3-031-72970-6_3.
- Chengqi Lyu, Wenwei Zhang, Haian Huang, Yue Zhou, Yudong Wang, Yanyi Liu, Shilong Zhang, and Kai Chen. RTMDet: An Empirical Study of Designing Real-Time Object Detectors. *CoRR*, abs/2212.07784, 2022. doi: 10.48550/ARXIV.2212.07784.
- Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, Pan Zhang, Liangming Pan, Yu-Gang Jiang, Jiaqi Wang, Yixin Cao, and Aixin Sun. MMLONGBENCH-DOC: Benchmarking Long-context Document Understanding with Visualizations. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.
- Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning, March 2022.
- Ahmed Masry, Megh Thakkar, Aayush Bajaj, Aaryaman Kartha, Enamul Hoque, and Shafiq Joty. ChartGemma: Visual Instruction-tuning for Chart Reasoning in the Wild, November 2024.
- Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md. Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, Megh Thakkar, Md. Rizwan Parvez, Enamul Hoque, and Shafiq Joty. ChartQAPro: A More Diverse and Challenging Benchmark for Chart Question Answering. In Wanxiang Che,

- Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 19123–19151. Association for Computational Linguistics, 2025.
- Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, and Pratyush Kumar. PlotQA: Reasoning over Scientific Plots, February 2020.
- OpenAI. OpenAI o3 and o4-mini System Card, 2025a.
- OpenAI. GPT-5 System Card. <https://cdn.openai.com/gpt-5-system-card.pdf>, September 2025b.
- Anku Rani, Aparna Garimella, Apoorv Saxena, Balaji Vasan Srinivasan, and Paul Pu Liang. RADAR: A Reasoning-Guided Attribution Framework for Explainable Visual Data Analysis, August 2025.
- Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *CoRR*, abs/1804.02767, 2018.
- Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. doi: 10.1109/TPAMI.2016.2577031.
- Dong Shu, Haoyang Yuan, Yuchen Wang, Yanguang Liu, Huopu Zhang, Haiyan Zhao, and Mengnan Du. FinChart-Bench: Benchmarking Financial Chart Comprehension in Vision-Language Models. *CoRR*, abs/2507.14823, 2025. doi: 10.48550/ARXIV.2507.14823.
- Manan Suri, Puneet Mathur, Nedim Lipka, Franck Deroncourt, Ryan A. Rossi, and Dinesh Manocha. ChartLens: Fine-grained Visual Attribution in Charts, May 2025.
- Liyan Tang, Grace Kim, Xinyu Zhao, Thom Lake, Wenxuan Ding, Fangcong Yin, Prasann Singhal, Manya Wadhwa, Zeyu Leo Liu, Zayne Sprague, Ramya Namuduri, Bodun Hu, Juan Diego Rodriguez, Puyuan Peng, and Greg Durrett. ChartMuseum: Testing Visual Reasoning Capabilities of Large Vision-Language Models, May 2025.
- Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, Alexis Chevalier, Sanjeev Arora, and Danqi Chen. ChartX: Charting Gaps in Realistic Chart Understanding in Multimodal LLMs, June 2024.
- Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Peng Ye, Min Dou, Botian Shi, Junchi Yan, and Yu Qiao. ChartX & ChartVLM: A Versatile Benchmark and Foundation Model for Complicated Chart Reasoning, March 2025.
- Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 4818–4829. IEEE, 2024. doi: 10.1109/CVPR52733.2024.00461.
- Zhengzhuo Xu, SiNan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. ChartBench: A Benchmark for Complex Visual Reasoning in Charts. October 2024.
- Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, Gongye Liu, Xiaomei Nie, Deng Cai, and Yujiu Yang. ChartMimic: Evaluating LMM’s Cross-Modal Reasoning Capability via Chart-to-Code Generation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.
- Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-Mark Prompting Unleashes Extraordinary Visual Grounding in GPT-4V. *CoRR*, abs/2310.11441, 2023a. doi: 10.48550/ARXIV.2310.11441.

- Zhishen Yang, Raj Dabre, Hideki Tanaka, and Naoaki Okazaki. Scicap+: A knowledge augmented dataset to study the challenges of scientific figure captioning. In Amir Pouran Ben Veyseh, Franck Dernoncourt, Thien Huu Nguyen, and Viet Dac Lai (eds.), *Proceedings of the Workshop on Scientific Document Understanding co-located with 37th AAAI Conference on Artificial Intelligence (AAAI 2023), Remote, February 14, 2023*, volume 3656 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023b. URL <https://ceur-ws.org/Vol-3656/paper13.pdf>.
- Xiangyu Zhao, Yicheng Chen, Shilin Xu, Xiangtai Li, Xinjiang Wang, Yining Li, and Haian Huang. An Open and Comprehensive Pipeline for Unified Object Grounding and Detection. *CoRR*, abs/2401.02361, 2024. doi: 10.48550/ARXIV.2401.02361.
- Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, and Jianfeng Gao. RegionCLIP: Region-based Language-Image Pretraining. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 16772–16782. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01629.
- Jiangning Zhu, Yuxing Zhou, Zheng Wang, Juntao Yao, Yima Gu, Yuhui Yuan, and Shixia Liu. OrionBench: A Benchmark for Chart and Human-Recognizable Object Detection in Infographics. *CoRR*, abs/2505.17473, 2025a. doi: 10.48550/ARXIV.2505.17473.
- Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. InternVL3: Exploring Advanced Training and Test-Time Recipes for Open-Source Multimodal Models. *CoRR*, abs/2504.10479, 2025b. doi: 10.48550/ARXIV.2504.10479.
- Zhuofan Zong, Guanglu Song, and Yu Liu. DETRs with collaborative hybrid assignments training. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 6725–6735. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00621.

A USE OF LARGE LANGUAGE MODELS (LLMs)

Large Language Models (LLMs) were used to improve the phrasing and clarity of the paper, particularly in the introduction and abstract, where we asked the model to improve an existing draft. LLMs were also used to enhance the prompts used in the data curation and verification pipeline. They did not play a significant role in research ideation.

B DATA GENERATION PROMPTS

Below is the prompt for generating templates from the input Python script.

Stage 1: Template Generation Prompt

```
You are a helpful assistant. Given a python script as follows:
```python
{code}
```

Your task is to replace the data that is plotted to placeholder <data>
tokens.

## Replace with <data>:
- Data values that appear as visual elements within the plot, such as
bars, lines, points, etc.
- Text labels that appear INSIDE the plotted area (annotations, data
labels on visual elements)

## Keep unchanged:
- Axis labels, titles, legends (all legend text)
- Tick labels and categorical labels
- All plotting parameters (colors, sizes, styles, limits)

Example:
```python
Original
data = [10, 25, 30]
labels = ['A', 'B', 'C']
plt.bar(labels, data)
plt.title('Chart')

Template
data = [<data>, <data>, <data>]
labels = ['A', 'B', 'C'] # Keep unchanged
plt.bar(labels, data)
plt.title('Chart') # Keep unchanged
```
```

Below is the prompt for extracting individual data elements and identifying arguments, as well as generating question and referential expression templates that use these arguments. Note that we prompt the model with additional guidelines for specific chart types, denoted by `chart_type_prompt`.

Stage 2: Extract Data and Generate Question Templates Prompt

```
You are a helpful assistant. Given a template python script as follows:
```python
{code}
```

Notice that data values are intentionally filled in with <data>
placeholders. You have access to the matplotlib fig object that this
script with the filled in data values creates.
```

```

756 ## Your Task
757 Create extraction functions that extract the actual data values plotted/
758 displayed in the rendered matplotlib figure. These values generally
759 correspond to the <data> placeholders but may be approximated, rounded,
760 or otherwise modified during visualization.
761
762 ## Steps:
763 1. Identify visual attributes in the script. Different attributes
764 represent distinct data series or groupings, such as such as individual
765 line series, bar groups, scatter plot series, or sets of data labels
766 displayed within the chart. Do not extract hardcoded values that are not
767 masked out with <data> placeholders - these are not the underlying data
768 being plotted.
769 2. For each attribute, create two functions following the pattern below
770 3. Extract data as it appears visually - approximated/binning values are
771 acceptable
772
773 ## Data Extraction and Kwargs Generation Functions
774 - Name: extract_and_generate_kwargs_<attribute_name>(fig)
775 - Extract the actual data values for the given attribute that is plotted/
776 displayed in the chart. Infer the attribute name from the context of the
777 template script.
778 - DO NOT extract axis labels, legend labels, tick labels, titles, or
779 other chart annotations used in interpreting the chart
780 - DO NOT round the data values after extracting them from the matplotlib
781 figure
782 - Return a list of dictionaries, where each dictionary contains:
783   - A "data" key with the extracted value (number or string only),
784   which is used to format the answer template
785   - Additional kwargs that capture all dimensions of variation needed
786   to identify the data value, which are used to format the question
787   template
788 - Values should match what's visually plotted, which may be approximated/
789 rounded from original data
790 - Extracted data can be either numerical values or text strings,
791 depending on what is plotted
792
793 ## Question and Answer Template Functions
794 - Name: get_templates_<attribute_name>()
795 - Return a tuple containing (question_template, answer_template) as
796 strings with {{placeholder}} variables
797 - Questions must be answerable from the chart image alone
798 - Answer template should simply be "{{data}}"
799 - DO NOT round any formatted values in the question or the answer
800 template
801 - Use descriptive references to identify the extracted data based on
802 visual elements observable in the chart image itself (e.g., category
803 names, legend labels, axis values, colors, positions). DO NOT USE generic
804 index-based references like "first bin", "second point", or "item 1",
805 which are ambiguous when looking at the rendered chart
806 - Templates should work for all data values in the attribute and yield
807 unique questions for each data value
808
809 ## Additional Instructions
810 - Access legend elements: Use ax.get_legend() to get the legend object,
811 then legend.get_children() to access the individual legend components (
812 text labels, colored patches, lines).
813 {chart_type_prompt}
814 ## Format:
815 ```python
816 # Imports here (matplotlib.pyplot, numpy, etc.)
817
818 def extract_and_generate_kwargs_<attribute_name>(fig):
819     \"\\"

```

```

810     Extract plotted data values and generate kwargs for question
811     formatting
812     \"\"\"
813     kwargs_list = []
814     # Extract data and build kwargs dictionaries
815     # Each dict should include 'data' key, and other identifying
816     parameters
817     return kwargs_list # List of dicts, each with 'data' and other
818     kwargs
819
820 def get_templates_<attribute_name>():
821     \"\"\"
822     Return question and answer templates with {{placeholders}}
823     \"\"\"
824     question_template = "What is the {{metric}} for {{category}}?" #
825     Example
826     answer_template = "{{data}}" # Always just {{data}}
827     return question_template, answer_template
828
829 ...
830
831 Create separate code blocks for each visual attribute. Each function must
832 be self-contained.

```

Below is the prompt for extracting bounding boxes.

Stage 3: Extract Bounding Boxes Prompt

```

832 You are a helpful assistant. Given a previously created data extraction
833 function for a single attribute, modify it to add bounding box generation
834 capabilities.
835
836 ```python
837 {extract_functions}
838 ```
839
840 ## Task
841 Modify the provided extract_and_generate_kwargs_<attribute_name> function
842 and get_templates_<attribute_name> function to include bounding box
843 functionality.
844
845 ## Required Modifications:
846
847 ### Modify Data Extraction Function
848 - Update extract_and_generate_kwargs_<attribute_name>(fig) to include "
849 bbox" key in returned dictionaries
850 - The "bbox" key refers to the bounding box as a matplotlib Bbox object
851 that localizes the visual element representing this data point. The
852 bounding box must be in display coordinates. Follow the below tips:
853   - Target the most specific visual element for each data point (e.g.
854     marker, bar segment, text label)
855   - Use ax.get_window_extent() when possible.
856   - Avoid converting to display coordinates manually when the bounding
857     box can be obtained directly from the matplotlib object.
858 - CRITICAL: Each data value must have a unique bounding box
859
860 ### Modify Template Function
861 - Update get_templates_<attribute_name>() to return 4-tuple: (
862   question_template, answer_template, bbox_question_template,
863   bbox_answer_template)
864 - Bbox question template: "Provide the bounding box coordinates of the
865   region this sentence describes: [description using same placeholders as
866   data question]"
867 - Bbox answer template: always "{{bbox}}"
868 {chart_type_prompt}
869 ## Expected Output Format:
870 ```python

```

```

864 # Imports here
865
866 def extract_and_generate_kwargs_<attribute_name>(fig):
867     \"\"\"Extract plotted data values and generate kwargs with bounding
868     boxes\"\"\"
869     kwargs_list = []
870     # Extract data and build kwargs dictionaries with 'data', 'bbox', and
871     # identifying parameters
872     return kwargs_list
873
874 def get_templates_<attribute_name>():
875     \"\"\"Return question, answer, and bbox templates\"\"\"
876     question_template = "What is the {{metric}} for {{category}}?"
877     answer_template = "{{data}}"
878     bbox_question_template = "Provide the bounding box coordinates of the
879     region this sentence describes: the {{metric}} bar for {{category}}"
880     bbox_answer_template = "{{bbox}}"
881     return question_template, answer_template, bbox_question_template,
882     bbox_answer_template
883
884 Provide the complete modified functions for this specific attribute.
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

```

Table 6: Additional Instructions for Extracting Data for Each Chart Type

| Chart Type | Additional Instructions |
|------------|---|
| Area | <p>At any given x-coordinate in a stackplot, each layer’s polygon has both a top edge and a bottom edge. The individual layer height is simply: <code>individual_height = y_top - y_bottom</code> at that x-coordinate. Follow the below approach to extract individual layer heights:</p> <ul style="list-style-type: none"> • Extract x-coordinates: Get the unique x-values (data points) from the polygon vertices • For each collection (layer) and each x-coordinate: Find all vertices at that exact x-coordinate (within small tolerance). Get <code>y_top = max(y_values_at_x)</code> and <code>y_bottom = min(y_values_at_x)</code>. Calculate <code>individual_height = y_top - y_bottom</code> • Generate kwargs: Include the individual height, layer name, and x-coordinate |
| Errorbar | <ul style="list-style-type: none"> • <code>ax.containers</code> will contain both <code>BarContainer</code> and <code>ErrorbarContainer</code> objects. Check the type of the container and process accordingly. • For <code>ErrorbarContainer</code> objects, check <code>container.has_xerr</code> and <code>container.has_yerr</code> to determine error bar orientation. • <code>container.lines</code> is a tuple with structure like <code>(data_line, (cap_line1, cap_line2), (line_collection,))</code>. Use <code>container.lines[2][0]</code> to get the error bar stems. • When processing the unpacked line objects, use the <code>has_xerr/has_yerr</code> results to determine which coordinates contain the error information. |
| Scatter | <p>If individual points are not uniquely identifiable by categorical labels, focus on aggregate statistics:</p> <ul style="list-style-type: none"> • Use <code>ax.collections</code> to get scatter data points, then compute aggregate values from the x,y coordinates, such as cluster means, ranges of values, etc. The bounding box template should refer to the cluster of points used to compute the aggregate statistic. |
| Density | <p>When extracting values from a continuous distribution or function, you must first identify the tick values from <code>ax.get_xticks()</code> or <code>ax.get_yticks()</code>. Extract ONLY the data points at these tick values. Do NOT extract data at other arbitrary points.</p> |
| Hist | <ul style="list-style-type: none"> • Each question should ask about the height of a specific bin within a specific interval. • Determine intervals by obtaining the x-tick positions using <code>ax.get_xticks()</code>. For every pair of adjacent x-ticks, define an interval <code>[xtick_i, xtick_{i+1}]</code>. • For each interval, include only bars whose centers fall within that range. You MUST refer to these bins using ordinal numbers (first, second, third, etc.) from left to right within each interval. DO NOT use generic index-based references like “bin 1”, “bin 2”, or “item 1”, which are ambiguous when looking at the rendered chart. |

Table 7: Additional Bounding Box Generation Instructions for Each Chart Type

| Chart Type | Additional Instructions |
|------------|--|
| Area | When visible markers are present, compute the bounding box around each marker. Otherwise, when the extracted values are heights of layers, compute the center of the bounding box at the top of the layer at each x-coordinate and create a square bounding box with 10×10 pixel dimensions centered at that point. Do NOT use other bounding box dimensions. |
| Line | When visible markers for the data are present, compute the bounding box around each marker. Otherwise, when there are no markers, create a square bounding box with 10×10 pixel dimensions for each data point. Do NOT use other bounding box dimensions. |
| Radar | When visible markers for the data are present, compute the bounding box around each marker. Otherwise, when there are no markers, create a square bounding box with 10×10 pixel dimensions for each data point. Do NOT use other bounding box dimensions. |
| Density | If the extracted values are the density at specific coordinates, create a square bounding box with 10×10 pixel dimensions centered at the point on the density curve corresponding to each coordinate. Do NOT use other bounding box dimensions. |
| Scatter | If the extracted values are aggregate statistics, compute the bounding box over the relevant cluster of points. If the extracted values are individual points, compute the bounding box over each point. |
| Error Bar | Manually calculate the bounding box using the segment endpoints: use <code>ax.transData.transform</code> to convert the two segment points to display coordinates. Create a Bbox from the transformed coordinates using <code>Bbox.from_bounds(min_x, min_y, width, height)</code> . For the dimension perpendicular to the error bar, use a fixed width of 5 pixels. |

C DATA POSTPROCESSING PROMPTS

Below is the prompt to determine the error margin for generated questions and answers.

Postprocessing: Error Margin Prompt

```

You are given a chart image and several questions about the chart.

{questions}

## Your Task:
Analyze the chart's visual elements to determine the appropriate
precision level of the ground truth answer and establish a reasonable
error margin for evaluating predicted answers. You should reason about
how you would answer these questions based on the chart image and
identify the relevant visual elements.

### Step-by-Step Analysis:

**STEP 1: Identify Answer Type**
- **Direct readings**: Values explicitly shown as text labels, legend
items, category names, or data point labels
- **Estimated values**: Values that must be visually interpolated from
axis positions, calculated, or derived

**STEP 2: For Direct Readings**
- **Ground truth precision**: If the answer is a number, use the
precision of the number in the text label (e.g., "42.5" has precision 1)
- **Error margin**: 0 (exact match required for directly labeled values)
- **Special case**: If answer is categorical/text, set precision to null
and error margin to 0

```

```

1026 **STEP 3: For Estimated Values: Determine Ground Truth Precision and
1027 Error Margin**
1028
1029 Look at the axis tick labels and identify the **smallest meaningful unit
1030 of difference** between consecutive ticks. The ground truth should be **
1031 one decimal place more precise** than the tick interval, while the error
1032 margin accounts for visual estimation uncertainty.
1033
1034 ### Tick Interval Analysis Examples:
1035
1036 **Interval = 1 (ticks: 1, 2, 3, 4, 5)**
1037 - Ground truth precision: 1 (tenths place) - one level more precise than
1038 interval
1039 - Error margin: 0.5 (half the tick interval)
1040 - Reasoning: Ground truth can be 2.3, 4.7, etc. Visual tolerance is
1041 +/-0.5
1042
1043 **Interval = 10 (ticks: 10, 20, 30, 40, 50)**
1044 - Ground truth precision: 0 (ones place) - one level more precise than
1045 interval
1046 - Error margin: 5 (half the tick interval)
1047 - Reasoning: Ground truth can be 23, 47, etc. Visual tolerance is +/-5
1048
1049 **Interval = 20 (ticks: 20, 40, 60, 80, 100)**
1050 - Ground truth precision: 0 (ones place) - one level more precise than
1051 interval
1052 - Error margin: 10 (half the tick interval)
1053 - Reasoning: Ground truth can be 33, 67, etc. Visual tolerance is +/-10
1054
1055 **Interval = 0.1 (ticks: 0.1, 0.2, 0.3, 0.4)**
1056 - Ground truth precision: 2 (hundredths place) - one level more precise
1057 than interval
1058 - Error margin: 0.05 (half the tick interval)
1059 - Reasoning: Ground truth can be 0.23, 0.37, etc. Visual tolerance is
1060 +/-0.05
1061
1062 **Interval = 0.5 (ticks: 0.5, 1.0, 1.5, 2.0)**
1063 - Ground truth precision: 2 (hundredths place) - one level more precise
1064 than interval
1065 - Error margin: 0.25 (half the tick interval)
1066 - Reasoning: Ground truth can be 1.23, 1.67, etc. Visual tolerance is
1067 +/-0.25
1068
1069 ### Precision Scale Reference:
1070 The precision can be **any integer** (positive, negative, or zero) or **
1071 null**:
1072
1073 - **null**: Categorical/text values
1074 - **3**: Thousandths place (0.001)
1075 - **2**: Hundredths place (0.01)
1076 - **1**: Tenths place (0.1)
1077 - **0**: Ones place (1)
1078 - **-1**: Tens place (10)
1079 - **-2**: Hundreds place (100)
1080 - **-3**: Thousands place (1000)
1081
1082 ### Ground Truth Precision Rule:
1083 **For estimated values**: Ground truth precision = interval_precision + 1
1084 **For direct readings**: Ground truth precision matches the precision
1085 shown in labels
1086
1087 ### Complex Cases:
1088 - **Multiple axes**: Use the axis most relevant to the answer
1089 - **Logarithmic scales**: Focus on the linear spacing between major ticks

```

```

1080 - **Mixed scales**: Choose the most restrictive precision that's
1081 reasonable for the ground truth
1082 - **Percentage charts**: Disregard the percentage sign, focus on
1083 numerical values
1084 - **Calculated values**: Consider the precision of the least precise
1085 component
1086
1086 Output your analysis as a JSON object:
1087 ```json
1088 {{
1089     "answer_type": "direct_reading" or "estimated_value",
1089     "ground_truth_precision": <integer or null>,
1090     "error_margin": <number or 0>,
1091     "tick_interval": <the interval between consecutive ticks, if
1092     applicable>,
1093     "reasoning": "Detailed explanation: Ground truth precision [X] (one
1094     level more precise than tick interval [Y]) because [reason]. Error
1095     margin [Z] (half tick interval) because [visual estimation tolerance
1096     reasoning]"
1097 }}
1098 ```

```

Below is the prompt to evaluate question quality.

Postprocessing: Question Quality Prompt

```

1102 You are given a chart image and several questions about the chart.
1103
1104 {questions_section}
1105
1106 ## Your Task:
1107 Evaluate the overall quality and answerability of this set of questions
1108 given the chart image. You should reason about how you would answer these
1109 questions based on the chart image. Since all questions follow a similar
1110 reasoning pattern, judge them as a group and determine which category
1111 they fall into:
1112
1113 1. **VALID**: Questions that can be answered by examining the chart, even
1114 if they require:
1115     - Making reasonable assumptions about standard chart elements (axes,
1116     legends, data points)
1117     - Basic chart reading skills (identifying trends, comparing values,
1118     reading labels)
1119     - Standard domain knowledge (e.g., knowing that "Q1" means first
1120     quarter)
1121     - IMPORTANT: Questions with precise numerical parameters are valid as
1122     long as the underlying data is represented in the chart.
1123
1124 2. **AMBIGUOUS**: Questions that are genuinely unanswerable because:
1125     - Key terms are completely undefined AND cannot be inferred from chart
1126     context
1127     - The question refers to chart elements that definitively don't exist
1128
1129 3. **DEFECTIVE**: Questions where:
1130     - The answer is explicitly stated in the question text itself
1131     - No chart examination is needed because the answer is given away in
1132     the question
1133
1134 Output a JSON object with the following fields:
1135 ```json
1136 {{
1137     "questions_quality": "<AMBIGUOUS|VALID|DEFECTIVE>",
1138     "justification": "<Brief explanation for your judgment of the
1139     question set>"
1140 }}

```

D EXPERIMENT DETAILS

D.1 MULTIMODAL LLM EVALUATION

In Table 8, we detail the standard, chain-of-thought, and set-of-marks prompts that we use for evaluating chart question-answering. Table 9 shows the prompt we use for evaluating visual grounding. Additionally, for GPT-4o, we instruct the model to output normalized coordinates in $[0, 1]$. Qwen2.5-VL natively outputs coordinates relative to the resized image, and InternVL3 is prompted to normalize coordinates by 1000.

Table 8: Chart Question Answering Prompts

| Prompt Type | Prompt |
|--------------|--|
| Standard | <p>You are an expert in analyzing charts. Your task is to answer the question based on the chart provided.</p> <p>At the end of your response, provide your final answer in the format "Answer: X" where X is your final answer.</p> <p>Question: {question}</p> |
| CoT | <p>You are an expert in analyzing charts. Your task is to answer the question based on the chart provided. Think step by step and provide your reasoning before giving the final answer.</p> <p>At the end of your response, provide your final answer in the format "Answer: X" where X is your final answer.</p> <p>Question: {question}</p> |
| Set-of-Marks | <p>You are an expert in analyzing charts. Your task is to answer the question based on the chart provided. Think step by step, and in your reasoning, refer to relevant bounding boxes labeled with numbers.</p> <p>At the end of your response, provide your final answer in the format "Answer: X" where X is your final answer.</p> <p>Question: {question}</p> |

Table 9: Visual Grounding Prompt

| Prompt Type | Template Content |
|-------------|--|
| Standard | <p>You are an expert in analyzing charts. Your task is to localize the visual element corresponding to a given region.</p> <p>At the end of your response, provide your final answer in the format "Answer: [x1, y1, x2, y2]" where [x1, y1] are the coordinates of the top-left corner of the bounding box and [x2, y2] are the coordinates of the bottom-right corner of the bounding box.</p> <p>Provide the bounding box coordinates of the region this sentence describes: {referential_expression}</p> |

D.2 CHART ELEMENT LOCALIZATION

For object detection, we finetune 6 models: MM-GD (Zhao et al., 2024), LLMDeT (Fu et al., 2025), Faster R-CNN (Ren et al., 2017), YOLOv3 (Redmon & Farhadi, 2018), RTMDeT (Lyu et al., 2022), and Co-DETR Zong et al. (2023). For visual grounding, we finetune MM-GD and LLMDeT.

For each model, we specify the number of epochs, batch size, and learning rate when fine-tuning on CHARTREF’s training set in Table 10 shows the fine-tuning hyperparameters. The training was done on 4 NVIDIA H200s.

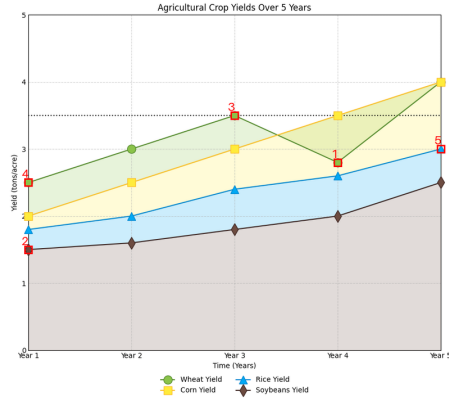
Table 10: Training hyperparameters

| Object Detection | | | | | | |
|------------------|--------|--------|--------------|--------|--------|---------|
| Hyperparameters | MM-GD | LLMDet | Faster R-CNN | YOLOv3 | RTMDet | Co-DETR |
| Optimizer | AdamW | AdamW | SGD | SGD | AdamW | AdamW |
| E | 100 | 100 | 100 | 120 | 50 | 30 |
| B | 4 | 16 | 4 | 4 | 16 | 16 |
| lr | $1e-4$ | $1e-4$ | 0.02 | $1e-3$ | $4e-3$ | $1e-4$ |
| weight_decay | $1e-4$ | $1e-4$ | $1e-4$ | $5e-4$ | 0.05 | $1e-4$ |

| Visual Grounding | | | | | | |
|------------------|--------|--------|---|---|---|---|
| Optimizer | AdamW | AdamW | - | - | - | - |
| E | 8 | 16 | - | - | - | - |
| B | - | - | - | - | - | - |
| lr | $1e-4$ | $1e-4$ | - | - | - | - |
| weight_decay | $1e-4$ | $1e-4$ | - | - | - | - |

E CHARTREF EXAMPLES

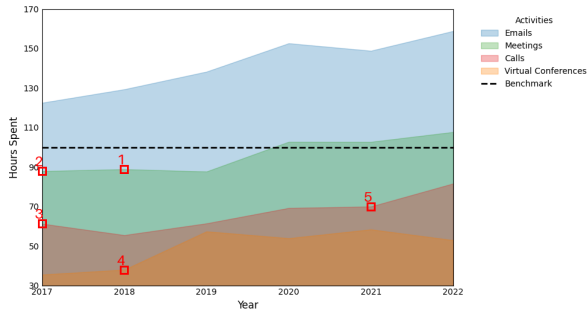
Below are visual grounding examples, where each referential expression corresponds to a red bounding box labeled with a number.



1. the yield in tons/acre for Wheat in Year 4
2. the yield in tons/acre for Soybeans in Year 1
3. the yield in tons/acre for Wheat in Year 3
4. the yield in tons/acre for Wheat in Year 1
5. the yield in tons/acre for Rice in Year 5

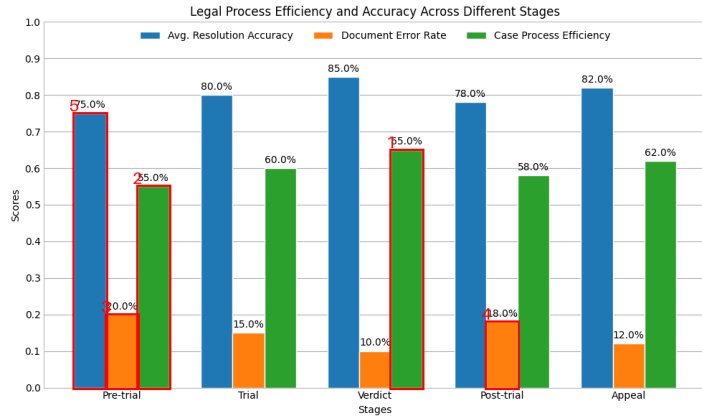
Figure 7: Area Chart, Example 1

Annual Hours Spent on Communication Activities



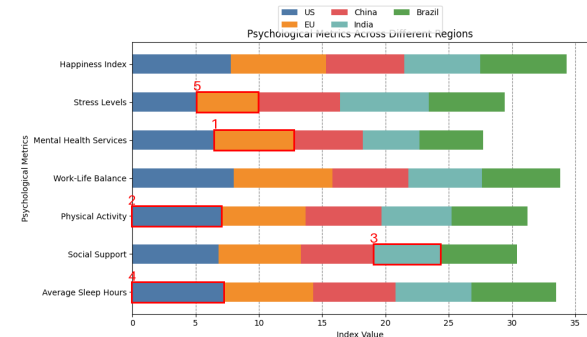
1. the annual hours spent on Meetings in 2018
2. the annual hours spent on Meetings in 2017
3. the annual hours spent on Meetings in 2017
4. the annual hours spent on Virtual Conferences in 2018
5. the annual hours spent on Calls in 2021

Figure 8: Area Chart, Example 2



1. the Case Process Efficiency bar for the Verdict stage
2. the Case Process Efficiency bar for the Pre-trial stage
3. the Document Error Rate bar for the Pre-trial stage
4. the Document Error Rate bar for the Post-trial stage
5. the Average Resolution Accuracy bar for the Pre-trial stage

Figure 9: Bar Chart, Example 1



1. the Mental Health Services bar segment for EU
2. the Physical Activity bar segment for US
3. the Social Support bar segment for India
4. the Average Sleep Hours bar segment for US
5. the Stress Levels bar segment for EU

Figure 10: Bar Chart, Example 2

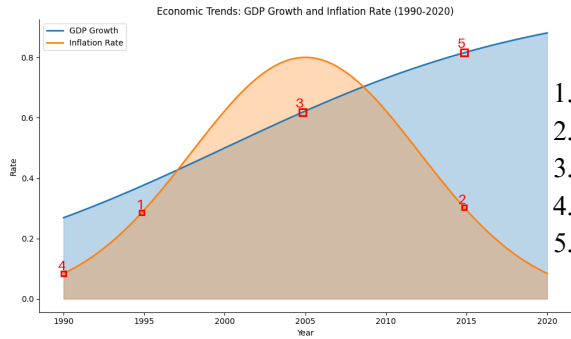
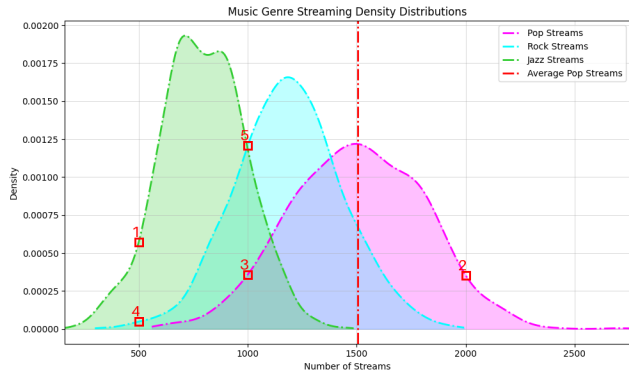
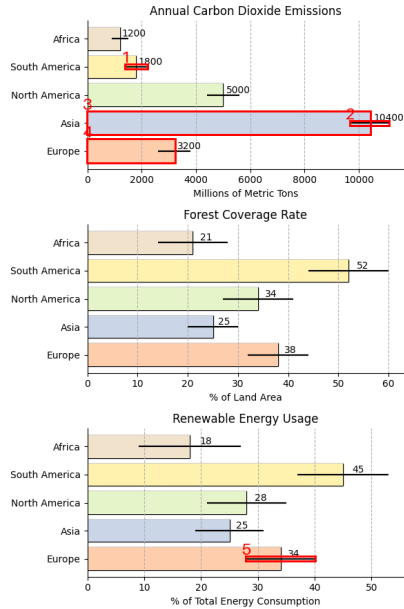


Figure 11: Density Plot, Example 1



1. the density value for Jazz Streams at 500 streams
2. the density value for Pop Streams at 2500 streams
3. the density value for Pop Streams at 1000 streams
4. the density value for Rock Streams at 500 streams
5. the density value for Rock Streams at 1000 streams

Figure 12: Density Plot, Example 2



1. the CO2 emissions error margin for South America
2. the CO2 emissions error margin for Asia
3. the CO2 emissions for Asia
4. the CO2 emissions for Europe
5. the renewable energy usage error margin for Europe

Figure 13: Error Bar Chart, Example 1

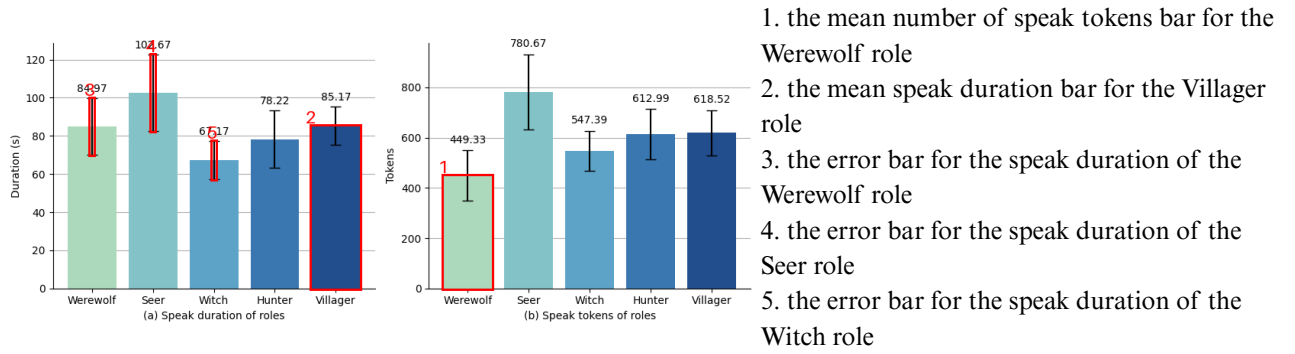


Figure 14: Error Bar Chart, Example 2

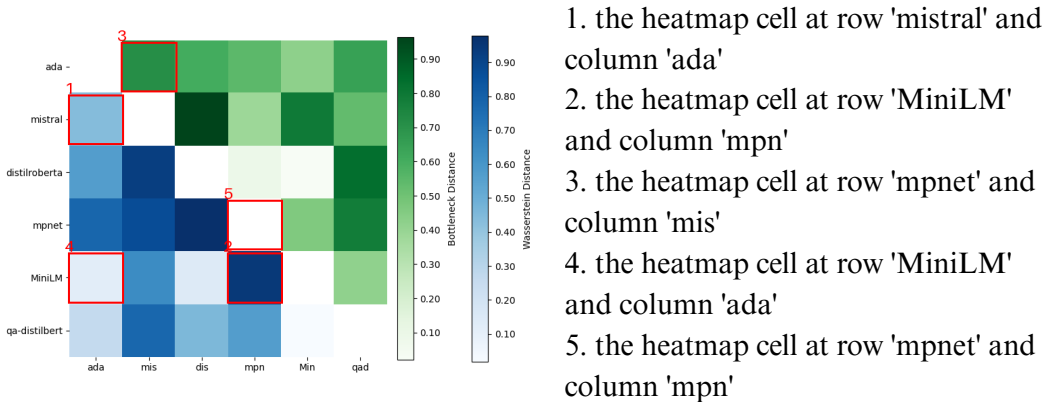


Figure 15: Heatmap, Example 1

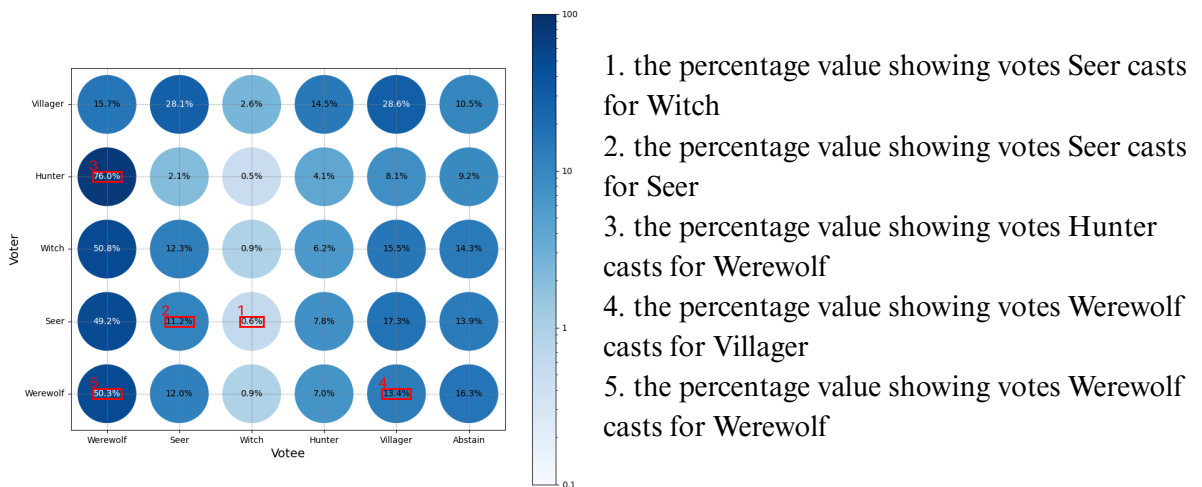


Figure 16: Heatmap, Example 2

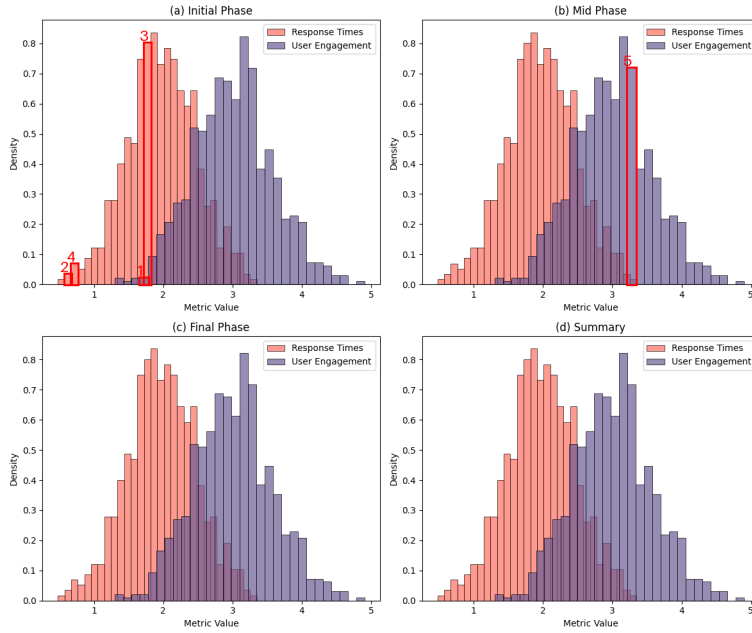


Figure 17: Histogram, Example 1

1. the fourth User Engagement bin in the interval $[1, 2]$ in subplot (a) Initial Phase
2. the second Response Times bin in the interval $[0, 1]$ in subplot (a) Initial Phase
3. the ninth Response Times bin in the interval $[1, 2]$ in subplot (a) Initial Phase
4. the third Response Times bin in the interval $[0, 1]$ in subplot (a) Initial Phase
5. the third User Engagement bin in the interval $[3, 4]$ in subplot (b) Mid Phase

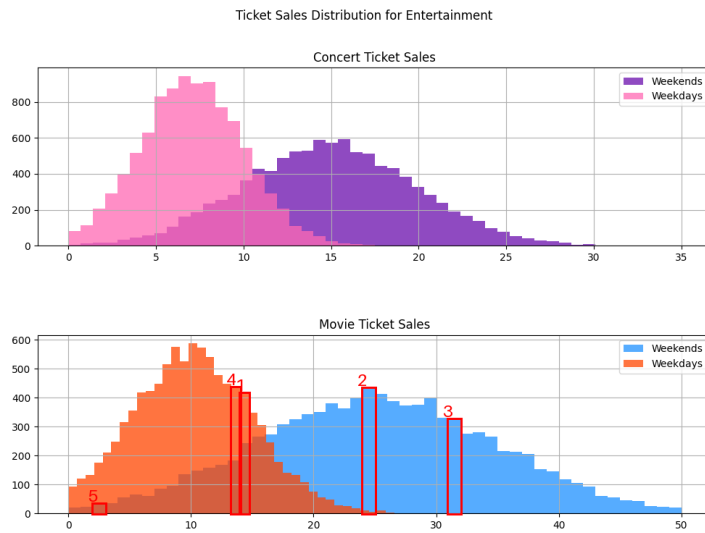


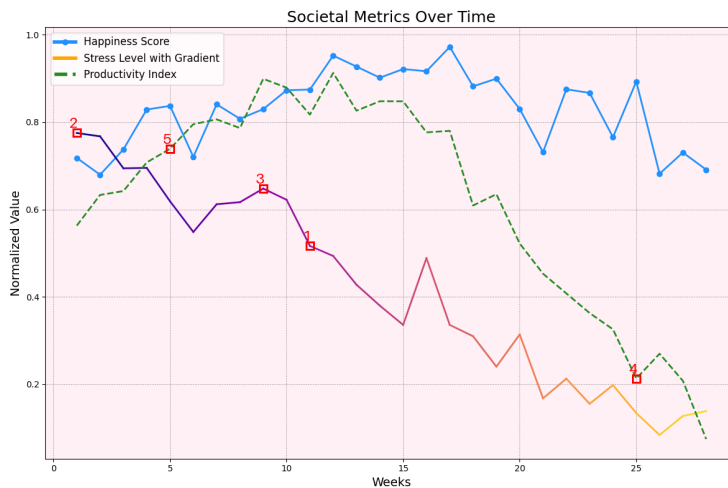
Figure 18: Histogram, Example 2

1. the seventh bar for Weekdays in the interval $[10, 20]$ in the Movie Ticket Sales histogram
2. the fifth bar for Weekends in the interval $[20, 30]$ in the Movie Ticket Sales histogram
3. the second bar for Weekends in the interval $[30, 40]$ in the Movie Ticket Sales histogram
4. the sixth bar for Weekdays in the interval $[10, 20]$ in the Movie Ticket Sales histogram
5. the first bar for Weekends in the interval $[20, 30]$ in the Movie Ticket Sales histogram



1. the content decay value at time point 3 months
2. the spikes in engagement value at time point 9 months
3. the user growth value at time point 6 months
4. the spikes in engagement value at time point 1 months
5. the user growth value at time point 2 months

Figure 19: Line Chart, Example 1



1. the Stress Level for week 11
2. the Stress Level for week 1
3. the Stress Level for week 9
4. the Productivity Index for week 25
5. the Productivity Index for week 5

Figure 20: Line Chart, Example 2

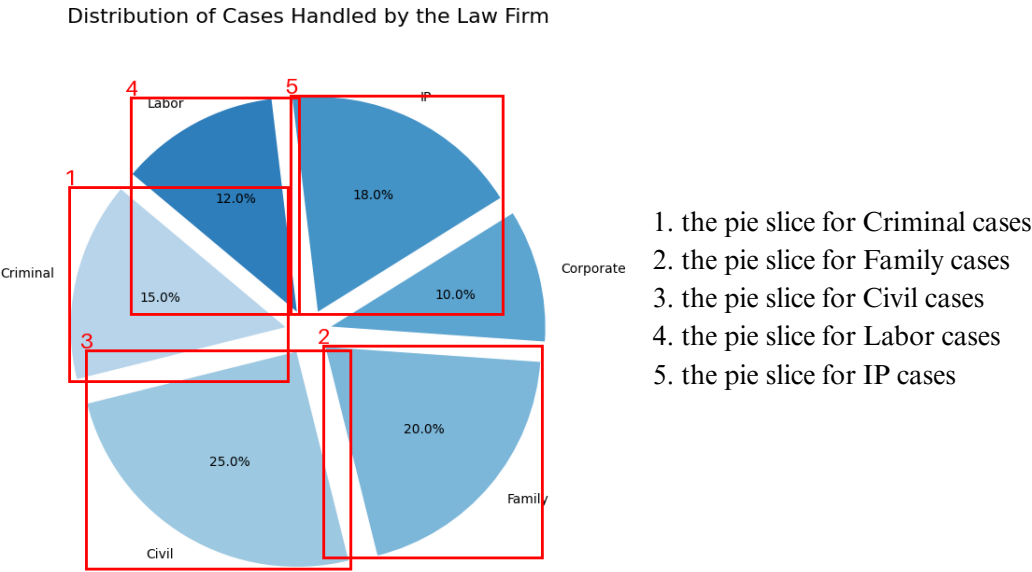


Figure 21: Pie Chart, Example 1

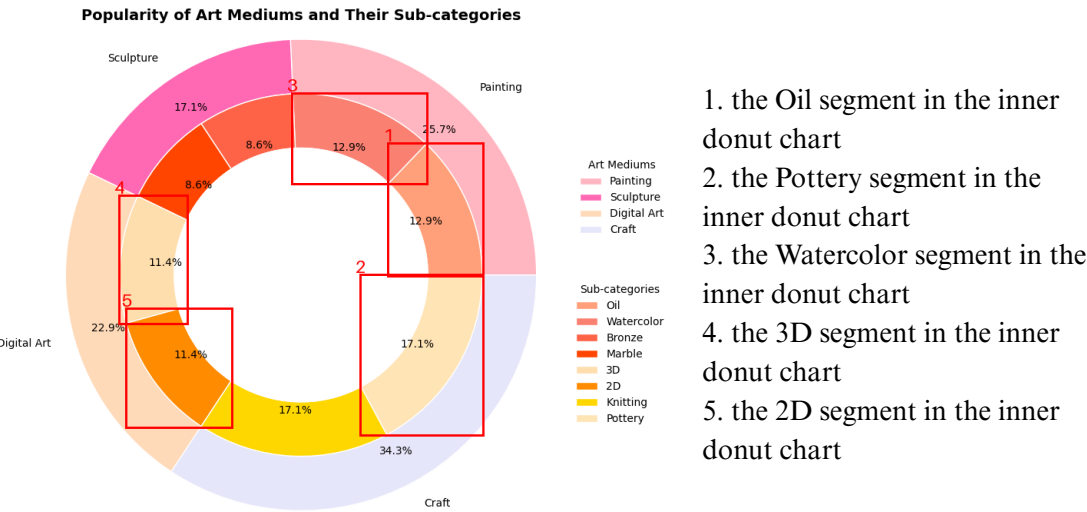
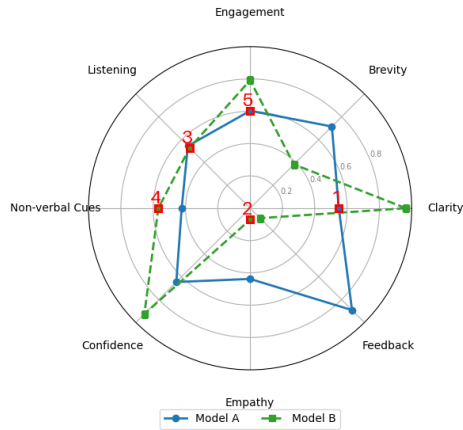
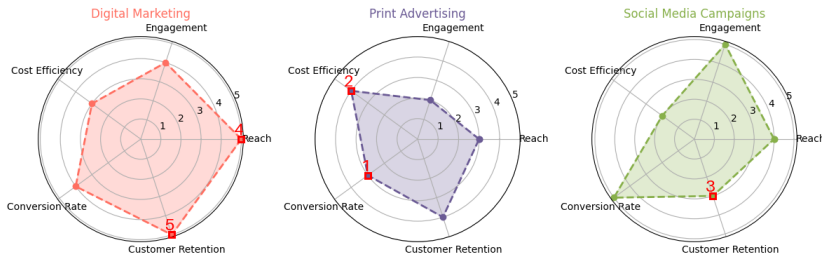


Figure 22: Pie Chart, Example 2



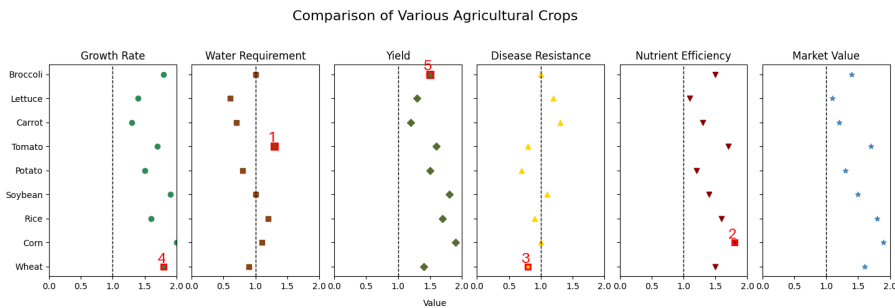
1. the Clarity score for Model A
2. the Empathy score for Model B
3. the Listening score for Model B
4. the Non-verbal Cues score for Model B
5. the Engagement score for Model A

Figure 23: Radar Chart, Example 1



1. the Conversion Rate value for Print Advertising
2. the Cost Efficiency value for Print Advertising
3. the Customer Retention value for Social Media Campaigns
4. the Reach value for Digital Marketing
5. the Customer Retention value for Digital Marketing

Figure 24: Radar Chart, Example 2



1. the Water Requirement data point for Tomato
2. the Nutrient Efficiency data point for Corn
3. the Disease Resistance marker for Wheat
4. the Growth Rate marker for Wheat
5. the Yield data point for Broccoli

Figure 25: Scatter Plot, Example 1

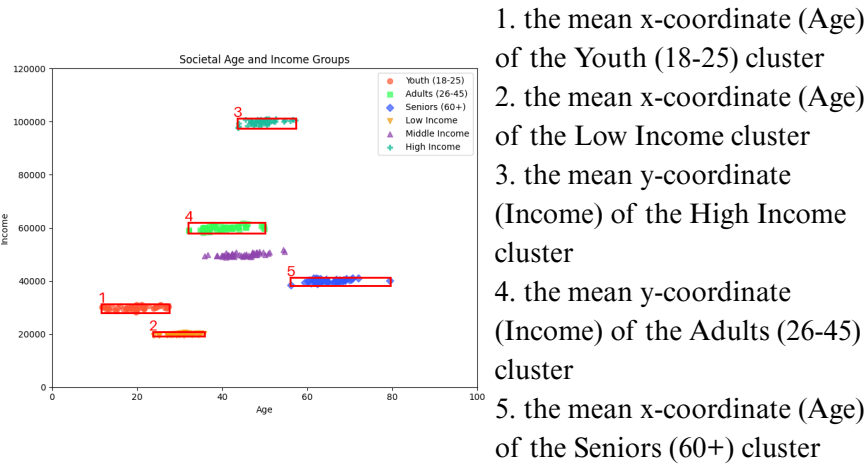


Figure 26: Scatter Plot, Example 2

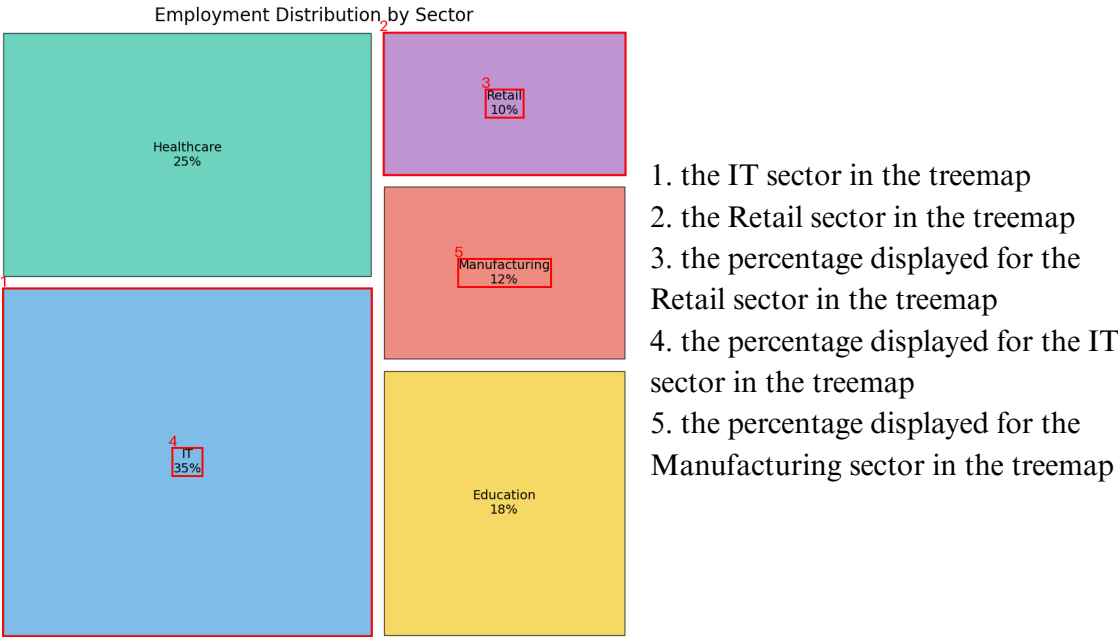
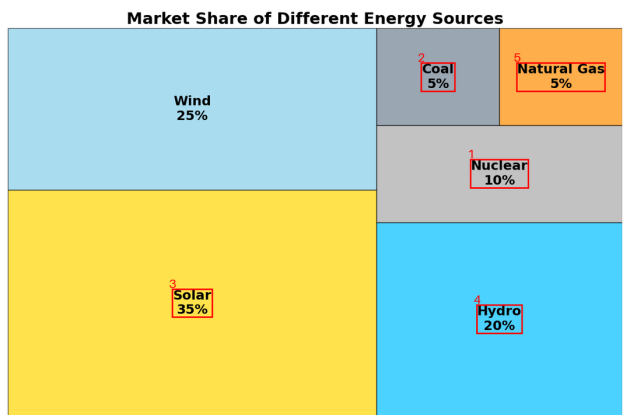


Figure 27: Treemap, Example 1



1. the market share percentage for Nuclear
2. the market share percentage for Coal
3. the market share percentage for Solar
4. the market share percentage for Hydro
5. the market share percentage for Natural Gas

Figure 28: Treemap, Example 2