DECONSTRUCTING AND RESTYLING SVG CHARTS USING LARGE LANGUAGE MODELS

SYED MUHAMMAD ALI RAZA ZAIDI

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF ARTS

GRADUATE PROGRAM IN INFORMATION SYSTEMS AND TECHNOLOGY
YORK UNIVERSITY
TORONTO, ONTARIO

SEPTEMBER 2024

Abstract

SVG charts are very common on the Web, however, reusing, editing and restyling these charts is very difficult. To facilitate this process, this thesis explores the challenges of extracting data and visual encodings from SVG chart images and restyling them based on user queries. We leverage large language models (LLMs) to facilitate this process using few-shot prompt approaches, enabling users to deconstructs and restyle existing Vega-Lite visualizations through natural language input. Our evaluation on 800 SVG charts and 250 natural language queries reveals that our system accurately deconstruct 93.4% charts and successfully restyled 38.6% queries. Further qualitative analysis suggests that while LLMs can effectively extract underlying data and visual encodings while accurately modifying visual elements in response to user requests, they also suffer from inaccuracies when charts are complex and unconventional. Finally, based on the above techniques, we develop a Chrome plugin tool that detects and deconstructs SVG charts from the web page and then restyles the charts based on user input. This approach highlights the potential of LLMs in improving the flexibility and usability of data visualization tools for a broader audience.

Dedication

To Professor Enamul and Ashad,

Mom, Dad,

& Mariyam

Acknowledgements

First and foremost, I would like to express my deepest gratitude to God for providing me with the strength, guidance, and perseverance to complete this thesis. Without His constant blessings, this achievement would not have been possible.

I am immensely grateful to my supervisor, Professor Enamul Hoque, for his unwavering support, insightful guidance, and invaluable feedback throughout the course of my research. His patience, encouragement, and expertise have been instrumental in shaping this work. I would also like to extend my sincere thanks to Professor Ashad Kabir for his thoughtful advice and continuous encouragement during this journey. Your input has been vital in navigating the complexities of this research. I am also grateful to Professor Arik for his invaluable insights and support.

I would like to take this opportunity to thank my parents, whose love, encouragement, and sacrifices have always motivated me to pursue my dreams. Your belief in me has been a constant source of strength.

To my fiancée Mariyam, thank you for your unwavering support, patience, and understanding throughout this journey. Your belief in me, even during the most challenging times, means the world to me.

Finally, I want to express my heartfelt thanks to my siblings for always being there for me and providing me with the encouragement I needed to stay focused and motivated.

Thank you all, Ali

Table of Contents

List of Tables

List of Figures

Introduction

1.1 Motivation

In today's data-driven world, creating effective and engaging data visualizations is essential. However, tools like D3.js [5], Vega-Lite [47], and Tableau [52], while powerful, often pose significant challenges for users without specialized design skills. These tools often require users to make critical decisions about visual encoding and design elements such as position, size, shape, color, and font to accurately represent data. This process can be both time-consuming and prone to errors, as users struggle to balance aesthetic appeal with effective data communication.

For example, while D3.js [15] offers extensive capabilities for creating complex visualizations, it demands substantial coding expertise, limiting its accessibility to non-programmers. Vega-Lite [47] simplifies this process with a concise, JSON-based syntax [13] for creating

interactive visualizations, but it still requires significant user involvement. Although graphical interfaces like Tableau [52] and Microsoft Power BI [40] aim to reduce complexity, they often fall short when it comes to re-purposing existing visualizations with users' own datasets [10].

Despite well-established principles designed to guide visualization choices [29, 17, 28], many users are either unaware of these guidelines or struggle to apply them in practice. As a result, there remains a significant gap between the theoretical understanding of good design and the practical implementation of accessible, intuitive, and coherent visualizations.

The motivation for this work arises from the need to deconstruct and restyle existing charts efficiently. Visualization practitioners often look for web-based visualizations that they wish to reuse for their own datasets [10]. Similarly, they may wish to edit and restyle existing charts that match their design criteria. Since users possess varying levels of skills in creating and editing visualizations, the ability to extract data from pre-existing charts enables them to modify and recreate visualizations tailored to their specific needs. Despite its importance, this area is underexplored, even though users frequently encounter poorly designed or visually incompatible charts.

1.2 Approach

Addressing these challenges, such as extracting data and encodings from charts and restyling them, is essential to improve the usability and accessibility of data visualizations. For

users with specific accessibility needs, the ability to customize visual elements, such as color schemes, enhances clarity and readability [20]. Furthermore, deconstructing charts can enable visually impaired users to utilize existing tools like screen readers to interpret chart data more effectively. This approach makes visualizations more informative and accessible to a broader audience, including those with blindness and vision impairment.

Existing approaches for chart deconstruction often require that charts are created using specific tools like D3.js [22]. In reality, many existing charts on the web are in SVG (Scalable Vector Graphics) format and are produced with a variety of libraries such as D3, Vega-lite, Highcharts, and Data Illustrator. One advantage of such SVG charts is that visual objects, such as bars and circles, can be extracted including their pixel coordinates, directly from the Document Object Model (DOM), providing precise and structured access to the underlying information. In contrast, bitmap images require the use of computer vision models to estimate data, which often introduces inaccuracies and ambiguities. Despite this advantage, attempts to automatically deconstruct SVG charts are rare. One notable exception is Mystique, which attempts to deconstruct SVG charts; however, it is not fully automatic and does not support restyling to meet specific user needs expressed through text queries [10]. Furthermore, dynamically restyling SVGs based on user queries remains an unexplored area. To our knowledge, no existing tool combines the ability to extract data and visual encodings from SVG charts (deconstruction) with the capability to restyle them.

This research aims to bridge that gap by enabling users to automatically extract data

from charts and then restyle them using natural language queries. Additionally, our approach provides advanced functionalities, such as querying and summarizing charts, thereby enhancing the flexibility and utility of chart manipulation.

More specifically, we leverage the capabilities of Large Language Models (LLMs), such as GPT-4 [46] and Gemini [45] to extract data from SVG charts and restyle them using natural language queries. Furthermore, we have developed a Chrome extension as a tool that serves as a pipeline, enabling a streamlined process for both data extraction and chart restyling. The plugin supports a variety of chart types, including bar, pie, scatter, line, and stacked charts, while utilizing few-shot prompt approach to improve the accuracy and efficiency [35]. It also provides interactive features, such as modifying original charts, changing chart types, adding animations, and incorporating elements like tooltips and summaries [4]. Examples of the results of our chrome extension pipeline can be seen in Figure 1.1

To understand the utility of our approach, we conducted a study and performed both quantitative and qualitative analyses. Our study is guided by two key research questions:

- How effective are LLMs at extracting data tables from SVG visualizations?
- How effective are LLMs in restyling SVG charts through natural language queries?

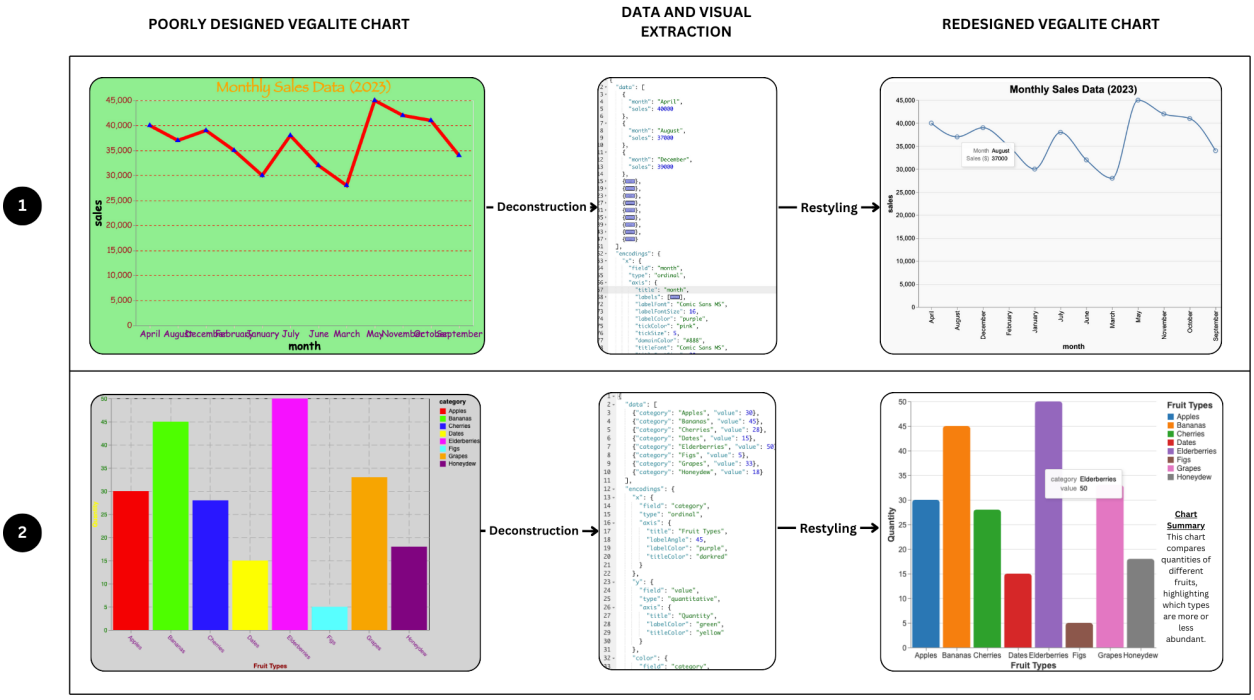

Figure 1.1: Examples 1 and 2 showing poorly designed Vegalite charts converted to well-designed charts

The original charts on the left suffer from poor design choices, including clashing colors, inconsistent fonts, cramped labels, and inefficient use of space. Our extraction and restyling process automatically extracts the data tables and visual encodings, allowing for a cleaner, more coherent redesign (shown on the right). The improved charts feature balanced color schemes, legible fonts, well-spaced labels, and enhanced readability. This transformation is achieved through natural language queries, enabling users to easily restyle charts without manually adjusting their code or layout. The first row (1) illustrates a poorly designed line chart with unreadable axis labels and inconsistent styling, which is refined into a clearer, more professional design. The second row (2) depicts a bar chart where uncoordinated color choices and visual clutter are replaced by a well-organized layout with a clear color legend and additional contextual information like chart summaries.

1.3 Research Contributions

Our contributions include:

- **Novel Data Extraction from SVG Charts:** To our knowledge, we are the first to develop a method for extracting data tables and visual encodings directly from SVG charts. By employing prompt engineering techniques, specifically few-shot prompts, we achieve a high level of accuracy with extraction of data tables and visual encodings from complex SVG structures.

- **Dynamic Restyling of SVG Images:** Our approach allows for the dynamic restyling of static SVG images based on user natural language queries. Users are able to modify the underlying data as well as change various visual encodings such as marks, ticks, scales, and color schemes, providing a flexible and powerful tool for data visualization customization.

- **Evaluation of LLMs for Data and Visual Encoding Extraction:** We conducted a comprehensive evaluation of data and visual encoding extraction using multiple large language models (LLMs). Additionally, we have manually assessed the effectiveness of various LLMs in restyling charts based on user queries, ensuring the reliability and accuracy of our approach.

- **Development of a Chrome Extension Interface:** We have also introduced a

user-friendly Chrome extension that serves as an interface tool for chart data extraction and visual restyling. This extension allows users to interact with SVG charts on any webpage, extract the underlying data and visual encodings, and customize the visual representation through natural language commands, making advanced data visualization techniques accessible to a broader audience.

1.4 Thesis Outline

The structure of this thesis is outlined as follows. Chapter 2 delves into the literature review, providing a detailed analysis of the challenges addressed in this research. Chapter 3 explains the methodology, outlining the approach and techniques employed. Chapter 4 focuses on the evaluation, presenting the results and findings. Lastly, Chapter 5 concludes the thesis, summarizing key insights and discussing future directions.

Literature Review

In this section, we review related research based on two primary themes: (i) chart deconstruction and restyling, and (ii) applications of large language models (LLMs) for data visualizations.

2.1 Chart Deconstruction

Chart deconstruction involves recovering the underlying data and visual encodings (i.e., how data is mapped to visual attributes such as position, length, and color) from an existing chart. This process has been applied to various chart formats, ranging from bitmap images to SVG charts, including those created with the D3 library.

Early work on deconstructing bitmap charts includes the ReVision system [49], which takes chart images and identifies the chart type using an SVM-based classifier, then extracts

the underlying graphical marks (e.g., bars) and data. ChartSense [26] improves on this by using a convolutional neural network (CNN) for chart type classification, followed by an interactive approach for data and mark extraction. Other efforts have focused on recovering color encodings by analyzing legends in bitmap charts [42], as well as extracting data from more complex charts like map charts [38] and scatterplots [12].

Fully automatic chart extraction pipelines were proposed by Choi et al. [11] and Siegel [51], although their methods rely on heuristics that struggle with many real-world charts, resulting in limited performance. ChartOCR [33] is another automated system that achieves high accuracy in extracting data from real-world charts. However, it primarily predicts raw data values for marks (e.g., bars) without linking them to their corresponding axes or legends. In general, data extraction from bitmap charts remains challenging, particularly for complex charts with many data points, poor image resolution, or complex functional mappings, where computer vision models face significant limitations.

In contrast, deconstructing SVG charts is relatively easier since the data is often embedded in the webpage's Document Object Model (DOM). Harper and Agrawala [20] explored deconstructing and restyling visualizations created with the D3 JavaScript library. Their tool, a Google Chrome extension, analyzes the programmatic description of the chart to extract data, marks, and encodings more accurately than bitmap-based techniques. However, these tools still struggle with unconventional and complex charts, such as map charts, or charts involving intricate functional mappings.

Mystique [10] offers an interactive approach for deconstructing SVG charts, guiding users through a series of steps to map deconstructed components to their data. Our work departs significantly from previous approaches by focusing on a fully automated pipeline that extracts data and visual encodings from charts and dynamically restyles them based on user queries. Unlike prior systems, which often address only specific aspects of the visualization process, our approach provides users with the ability to modify not only the chart's visual design but also its underlying data. This comprehensive approach enables seamless interaction, offering unparalleled flexibility and accessibility.

2.2 LLMs for Data Visualization

The intersection of Large Language Models (LLMs) and data visualization opens new avenues for simplifying complex datasets and enhancing decision-making processes. Several studies [27, 2, 35] demonstrate the potential of LLMs to automate the interpretation of visual data into more accessible formats. By leveraging their advanced language understanding capabilities, LLMs expand the scope of data visualization, enabling more interactive and user-friendly experiences.

The integration of tools such as ChatGPT, Codex, and GPT-3 within frameworks like Chat2VIS further exemplifies LLMs' ability to translate natural language into executable code for data visualizations [35]. This approach, often referred to as "prompt engineering," allows LLMs to generate Python scripts for creating visualizations. The smooth transition

from unstructured natural language to structured programming code highlights a key strength of LLMs, particularly when generating complex visual representations.

However, there are still notable challenges. Liu's study [31] underscores the need for responsible deployment of LLMs, while Kavaz [27] discusses the limitations of natural language interfaces (such as chatbot-driven systems like ChatGPT) in supporting interactive visual data manipulation. These interfaces often struggle with handling complex, multi-step queries and do not fully support advanced data visualization tasks.

Recent innovations, such as Live Charts [54], explore novel methods of presenting data by combining static charts with animations and audio narration. These approaches enhance the clarity of information through multi-sensory engagement. However, their reliance on predefined animations and the token limitations of LLMs reduce flexibility and interactivity. Additionally, Live Charts do not allow users to modify the underlying data or visual encodings, limiting customization and user participation.

Moreover, recent efforts have focused on evaluating LLMs in various benchmarks for chart comprehension and reasoning tasks, including question answering, summarization, and fact-checking [23, 16]. Despite this progress, there has been little to no exploration of LLMs' capabilities in deconstructing and restyling SVG charts—an area that motivated our study.

2.3 Applications of Chart Deconstruction

Chart deconstruction offers a range of valuable applications. Many charts suffer from poor design choices or low resolution (as seen in Figure 1.1), and researchers deconstruct these visualizations to extract the underlying data and visual encodings, allowing for the revitalization, editing, and restyling of these charts [20]. Additionally, deconstructing charts enables the indexing of large visualization collections, supporting the development of search tools that allow users to find visualizations based on specific design characteristics, facilitating exploration of the design space [22, 3]. Chart deconstruction also enhances question-answering systems that require both visual and logical reasoning over charts [36]. Furthermore, it plays a critical role in improving accessibility. Deconstructed chart data can be used to generate captions as alt-text and provide more accessible navigation of data points (e.g., [1, 50]). In this thesis, we develop a Chrome extension for the automatic deconstruction of SVG charts, focusing primarily on the application of chart restyling. However, our SVG deconstruction and restyling tool can be extended to incorporate other applications.

In summary, the research reviewed highlights the ongoing advancements in chart deconstruction and the integration of LLMs in data visualization workflows. While early efforts have made significant progress in recovering data from various chart formats, challenges remain, particularly with complex charts and bitmap images. Recent developments, such as interactive tools and LLM-driven solutions, offer new possibilities for enhancing accessibility,

customization, and interaction with visual data. Our work aims to build on these innovations by focusing on fully automated deconstruction and restyling of SVG charts, with the potential for broader applications in the future.

Methodology

This section presents the methodology of our research, building on the insights gained from the literature review. In the previous section, we examined existing works on data extraction and chart restyling, identifying the limitations of current tools, such as the complexity of SVG chart structures and the challenges in customizing visualizations. Based on these findings, we devised a research approach that integrates Vega-Lite, prompt engineering, and large language models (LLMs) to explore how these technologies can address these challenges. The purpose of this methodology is to provide a clear and detailed explanation of the tools, processes, and techniques we used to investigate data extraction from SVG charts and the potential for simplifying restyling tasks. This section outlines how we conducted our research, enabling a systematic evaluation of these technologies and their impact on making data visualization more accessible and flexible.

3.1 Problem Definition

Many charting libraries, such as D3, Vega-Lite, and Highcharts, generate SVG charts [3]. SVG (Scalable Vector Graphics) charts are popular for web-based visualizations due to their scalability and cross-platform compatibility. However, while effective for rendering visuals, SVGs are not designed to easily expose the underlying data [8]. This creates a significant hurdle when users need to extract data from these charts for analysis or repurposing. Parsing SVGs involves navigating intricate structures, a task that is often tedious, error-prone, and requires advanced technical skills [48].

Restyling existing visualizations presents another challenge. Many visualizations are created with static settings, making modifications—such as changing colors, fonts, or visual encodings—difficult without a deep understanding of the underlying code, particularly with complex libraries like D3.js. This complexity can be a barrier for users without programming expertise, limiting customization and adaptation [39].

To address these challenges, we propose using Vega-Lite in combination with prompt engineering and large language models (LLMs) to deconstruct and restyle charts. During the extraction phase, both the data and visual encodings are converted into a Vega-Lite specification. This specification provides a framework for visualizing or restyling charts. Vega-Lite's declarative syntax simplifies visualization creation and modification, allowing users to focus on what they want to visualize rather than the implementation details.

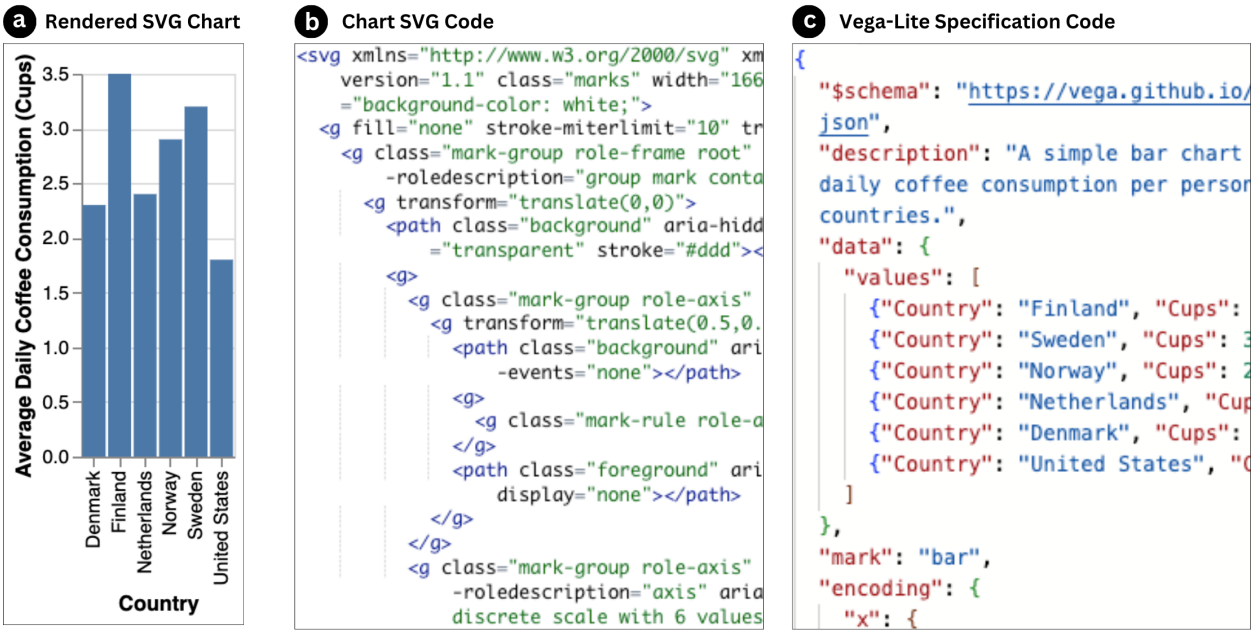

Figure 3.1: Comparison of three stages of the same chart
: a) the chart displayed as an image, b) the extracted SVG code, and c) the chart in its Vega-Lite format. This highlights the transition from the final visual output back to the underlying data representation.

Figure 3.1 illustrates a simple bar chart, along with the corresponding SVG code and the Vega-Lite specification. As shown in the figure, the SVG code is stored in an XML format, whereas the corresponding Vega-Lite specification is written in a concise JSON structure. This JSON structure defines how data fields are mapped to visual properties such as the x-position, y-height, and color of the chart elements.

Beyond simplicity, Vega-Lite also supports advanced features like data transformations and aggregations directly within its JSON structure. For example, calculating the average value per category can be accomplished with the following code:

```
"transform": [{
```

```
        "aggregate": [{"op": "average",
        "field": "value", "as": "avg_value"}],
        "groupby": ["category"]
    }]
```

Our tool takes the SVG code of a chart, extracts the data and visual encodings, and outputs the result in the Vega-Lite format. Subsequent features of the tool, including query-based restyling, take the Vega-Lite code of the deconstructed chart as input, along with any user-provided query, and return a modified Vega-Lite specification to reflect the user-suggested restyling changes.

3.2 Dataset collection

3.2.1 Chart Dataset

Our dataset was compiled from two primary sources:

(1) ChartQA [37]: ChartQA is a leading dataset created to evaluate chart question-answering systems. From this dataset, we selected 500 charts, equally distributed across five types—100 each of pie, bar, scatter, line, and stacked bar charts. To assess the accuracy of chart deconstruction, we converted the tables from ChartQA into Vega-Lite charts, generating corresponding SVG code based on the ground truth data tables.

(2) NVBench [34]: NVBench serves as a benchmark for the Natural Language to Visualization (NL2VIS) task, containing charts in Vega-Lite format. Using NVBench, we evaluated the plugin's ability to process a wide variety of chart types and its accuracy in extracting data from Vega-Lite SVG charts.

Table 3.1: Data Extraction Evaluation using ChartQA and NVBench Datasets

Dataset	**Chart Types**	**Total**
ChartQA [36]	Pie, Bar, Scatter, Line, Stacked (100 each)	500
NVBench [34]	Bar, Pie, Line (100 each)	300

3.2.2 Queries for Restyling

In order to evaluate the restyling approach, we first collected 100 natural language queries manually. These queries were designed to reflect common user requests for restyling Vega-Lite charts, covering a range of tasks related to modifying the visual representation of data. The collection process was motivated by the need to evaluate how effectively LLMs can handle different types of visual encoding commands, as well as to explore how these queries contribute to understanding the restyling capabilities of LLMs.

By creating these visual encoding-based queries, we ensured they encompassed a wide spectrum of common adjustments. In particular, we focus on gathering queries that involve visual encoding changes (e.g., changing chart type, marks and visual attributes) as well as stylistic changes that are not visual encoding specific (e.g, changing the width of bars, changing background color). By analyzing these queries, we can identify patterns and challenges in

how LLMs interpret and execute restyling commands. For instance, user queries often include requests for changes in visual encoding (e.g., colors, chart types) or adjustments to non-visual elements (e.g., tooltips, legends) [21].

In addition to these human-written queries, we also collected a total of 150 visual encoding queries that were created using prompt engineering techniques. Achieving a balance between LLM-generated and human-generated queries is crucial. Human-generated queries bring real-world context and nuance that may be missed by purely LLM-generated queries. On the other hand, LLM-generated queries can introduce variety and scale that would be difficult to achieve manually. By combining these approaches, we ensure that the dataset captures both the precision and contextual richness of human input and the broad coverage and diversity provided by LLM augmentation.

To generate the LLM-driven queries, we used few-shot prompting. Each prompt included multiple examples relevant to the category of queries being generated. In developing the prompts, we accounted for multiple dimensions of variability, including parameter ranges (e.g., color codes, pixel dimensions), chart types (e.g., bar, pie, line), and structural changes (e.g., axis reconfiguration). Additionally, each prompt underwent iterative refinement through manual tuning, ensuring that the LLM's generative capabilities were fully exploited without overfitting to specific query forms. This approach maximized both the diversity and relevance of the output queries. Below are the prompts used:

- Visual Encoding Prompt Example 1:

"Here are some examples of visual

encoding queries:

- Change the bar color to blue (#0000FF).
- Adjust the line thickness to 2 pixels for clarity.
- Switch the chart type from a bar chart to a pie chart for
  categorical representation.
- Set the y-axis scale to logarithmic for data compression.
- Increase the size of scatterplot points to 10 pixels.
- Rotate the x-axis labels by 45 degrees for better readability.
- Add a title 'Sales Report 2023' to the chart.
- Set the bar width to 15 pixels for improved visibility.
- Change the background color of the chart to light gray (#D3D3D3).
- Convert the stacked bar chart into a grouped bar chart.
- Apply a gradient color scheme to the line chart based on
  the y-axis values.
- Set the opacity of the area chart to 0.6 for enhanced transparency.
- Swap the x and y axes of the chart.
- Add grid lines to the x-axis at intervals of 10 units."

"Generate 150 more queries for visual-encoding based

restyling of charts."

To ensure robustness, the queries were generated in multiple batches and underwent a post-generation analysis phase where they were evaluated for syntactic correctness, semantic coherence, and applicability to the Vega-Lite specification. Below are some sample queries generated by LLMs:

- Set the stroke width of the scatterplot points to 3 pixels.
- Transform the line chart to use a dashed line style with a 5-pixel gap.
- Apply a gradient color scheme to the bar chart based on the y-axis values.

Query Type	**Number of Queries**
LLM Generated Visual Encoding	150
Human Generated Visual Encoding	100

Table 3.2: Analysis of Natural Language Queries for Restyling Vega-Lite Charts
This table presents a breakdown of the different types of queries used for restyling Vega-Lite charts. The queries are divided into two categories: LLM-generated queries and human-generated queries focusing on visual elements. The analysis highlights the number of queries processed for each type, with 150 queries generated by large language models (LLMs) and 100 human-generated queries.

We analyze these 250 natural language queries based on visual encoding, the breakdown of which can be seen in Table 3.2, to test the system with Gemini [53], GPT-4 [41], Llama 8B and Llama 70B. Each query was used to instruct the models to make specific changes to the Vega-Lite charts according to the provided query. The results were then manually categorized as accurate, partially accurate, or not accurate. The queries were also sub-categorized based

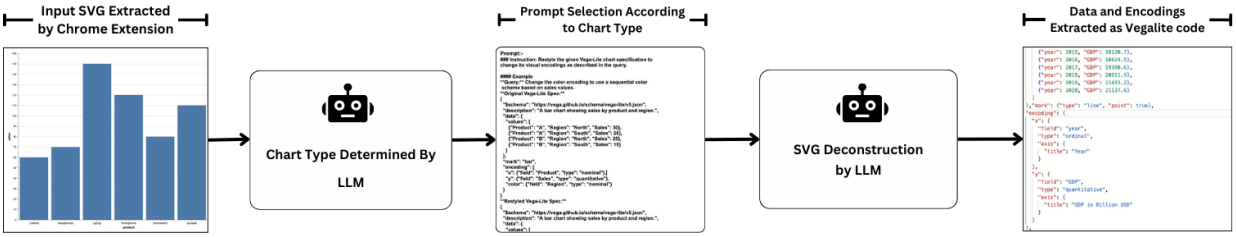

Figure 3.2: Figure showing the overview of the Data and Visual encoding extraction This figure illustrates the steps involved in extracting the data table and the visual encodings from a simple chart. The Chrome Extension automatically extracts the SVG code of a chart and then first extracts the chart type followed by extracting data and encodings as Vega-Lite code using few-shot promts.

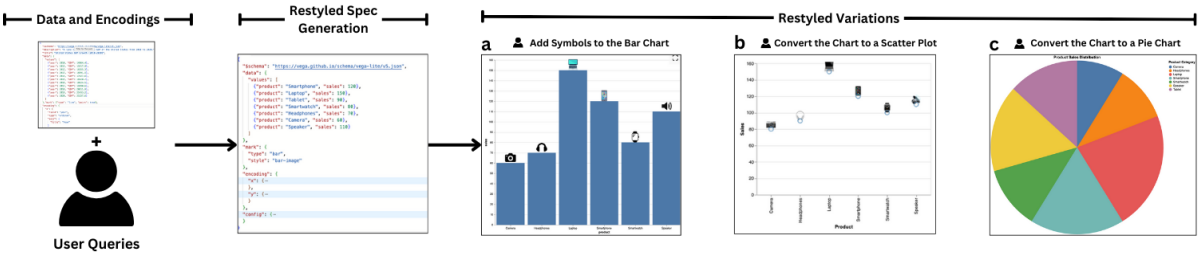

Figure 3.3: Figure showing the overview of the restlying process of charts. This figure illustrates the process of restyling charts. The tool takes the Vega-Lite specification of the chart and the user's query as input. It then generates restyled Vega-Lite specifications by utilizing LLM according to the user's query. Here, we show three examples of restyled results for the original input bar chart: a bar chart with icons representing sales items (a), a scatter plot with icons, (b), and a pie chart (c), demonstrating the tool's capability to customize chart appearances through intelligent code modifications.

on their applicability: some queries were specific to certain chart types, while others were general and applicable to all chart types. The general queries were tested across five chart types: pie, bar, grouped, line, and scatter.

3.3 Deconstruction

An overview of the chart deconstruction process is shown in Figure 3.2. The Chrome Extension scans the Document Object Model (DOM) of a webpage to identify SVG elements corresponding to Vega-Lite charts. It particularly looks for attributes or classes indicative of different chart types. Upon detection, users are prompted to confirm if they wish to proceed with deconstructing the chart, ensuring that processing is applied only to relevant charts. Once confirmed, the SVG data of the chart is transmitted to a backend server via an API request.

At the backend, the SVG is first sent to the LLM model to deduce the chart type using few-shot learning techniques [31, 6]. Based on the identified chart type, the correct few-shot prompts are selected from a collection of pre-designed prompts tailored for specific charts and sent to the LLM again for deconstruction. This process helps ensure that the model interprets the chart correctly from the beginning.

Using the few-shot prompts, curated examples help the model understand the structure of the chart, converting the SVG into a structured JSON object representing the data table. The few-shot examples ensure that the LLM can effectively interpret the relationship between SVG visual elements and their corresponding data representations, especially in more complex visualizations like line charts [43]. For instance, in deconstructing line charts, unique challenges arise due to path commands like "MoveTo" and "CurveTo" [24]. These commands dictate

how points on the line are connected and curved, requiring precise interpretation to translate pixel values into accurate data points. By utilizing these commands alongside extracted path coordinates, the model can accurately reconstruct the data and visual encodings.

To recover the **y-data values** from a line chart based on axis labels, tick marks, and their positions, we map the pixel coordinates of points on the chart to data coordinates using geometric transformations and scaling relationships.

More formally, let the chart image be represented in a 2D-pixel space in which:

- Pixel coordinates: (x_p, y_p) represent points on the chart in the image.
- Chart axis ranges: $[x_{\min}, x_{\max}]$ for the x-axis and $[y_{\min}, y_{\max}]$ for the y-axis.

The goal is to map (x_p, y_p) to **data coordinates** (x_d, y_d) using the scale provided by axis labels and tick marks. First, we identify the following pixel positions:

- $(x_{p,\min}, y_{p,\min})$: Pixel coordinates of the chart origin.
- $(x_{p,\max}, y_{p,\max})$: Pixel coordinates of the maximum extent of the axes.

We then calculate the scaling factors s_x and s_y as:

$$s_x = \frac{x_{\max} - x_{\min}}{x_{p,\max} - x_{p,\min}}$$

$$s_y = \frac{y_{\max} - y_{\min}}{y_{p,\min} - y_{p,\max}}$$

Using the scaling factors, we then map any pixel point (x_p, y_p) to data coordinates (x_d, y_d):

$$x_d = x_{\min} + s_x \cdot (x_p - x_{p,\min})$$

$$y_d = y_{\min} + s_y \cdot (y_p - y_{p,\max})$$

Here, $y_{p,\max}$ is used as the reference because pixel coordinates typically increase downward in images.

Finally, For each x_p corresponding to an x-value of interest, we simply extract y_p from the chart's line trace and apply the above transformation equations to compute (x_d, y_d).

3.4 Restyling

An overview of the chart restyling process is shown in Figure 3.3. The Chrome extension tool renders the Vega-Lite specification from the deconstruction phase and then allows the user to interactively modify the chart. To ensure the accuracy and relevance of these modifications, advanced prompts, including directives and complex prompt engineering, are used [31, 6]. These methods restrict LLMs' responses, focusing it to perform only the necessary adjustments and not to deviate into unintended actions [44]. After the modifications are applied, the chart is re-rendered to visually represent the user's customization [14]. Additionally, users can request summaries or other analytical outputs based on the restyled data visualization,

which LLMs provide by further analyzing the altered chart's data and attributes [25]. This comprehensive process allows for a dynamic and precise restyling of data visualizations, tailored specifically to the user's needs through natural language processing and prompt engineering [9]. An example of the few shot-prompt that we used can be seen below.

Few-Shot Prompt For Restyling Charts

```
Given an SVG image of a chart, the task is to
extract the underlying data table and
represent the chart's encoding using Vega-Lite
specifications. Below are two
examples demonstrating this process.

Example 1 Input:
Query: Apply a log scale to the y-axis  and use
a diverging color scheme for the bars based on
their values.
  {INPUT VEGA-LITE SPECIFICATION INSERTED HERE}

Output:
  {OUTPUT VEGA-LITE SPECIFICATION INSERTED HERE}
```

```
Example 2 Input:

Query: Adjust the chart to show stacked bars with

a gradient color scale  for values and display

labels on top of each bar.

  {INPUT VEGA-LITE SPECIFICATION INSERTED HERE}


Output:

  {OUTPUT VEGA-LITE SPECIFICATION INSERTED HERE}
```

3.5 Chrome Extension User Interface

Users interact with LLMs through a chat interface to specify desired restyling actions. As shown in Figure 3.4, the interface includes four main components: (1) the deconstructed chart rendered with the Vega-Lite specification, (2) the extracted data table, (3) the chat input field for user queries, and (4) a brief summary of the chart generated by the LLM. Users type restyling commands into the chat box, and the LLM uses few-shot prompts to perform the restyling. The system then re-renders the visualization based on the user's input. Users can submit follow-up queries to iteratively refine the chart. They can also edit the data table if they need to correct the extracted data or modify the chart's dataset.

In conclusion, this methodology section has laid the groundwork for exploring how large language models (LLMs), prompt engineering, and Vega-Lite can be used to address

key challenges in data visualization, specifically extracting data from SVG charts and restyling them. By integrating these technologies, we have detailed the tools and techniques used to streamline these complex processes, allowing for more flexible and accessible chart manipulation. This section not only explains the reasoning behind our chosen approach but also provides a structured framework for evaluating LLMs' performance. As we transition into the evaluation section, we will systematically assess the effectiveness of this methodology, focusing on the accuracy and reliability of LLMs in extracting data and performing restyling tasks based on natural language queries.

Figure 3.4: Our Chrome Extension User Interface
The user interface allows users to modify chart elements, view the underlying data table of the chart and a summary of the chart. Users enter restyling queries through a chatbox and the system dynamically re-renders the chart based on user input.

Evaluation

In this section, we conduct a detailed evaluation to address the two primary research questions that drive our study: the effectiveness of large language models (LLMs) in extracting data tables from Vega-Lite visualizations and their ability to restyle these visualizations through natural language queries. The purpose of this evaluation is to rigorously assess LLM performance across both tasks, offering a quantitative and qualitative analysis of their capabilities. By focusing on these two dimensions, we aim to explore how well LLMs can manage data extraction from a variety of chart types, such as bar, line, scatter, and pie charts, while also investigating their capacity to dynamically modify chart styles based on user-driven inputs. The evaluation process includes accuracy testing, iterative refinement of prompts, and error analysis to pinpoint challenges in both data extraction and restyling tasks. Through this assessment, we provide a comprehensive understanding of the strengths and limitations of LLMs in automating these processes, highlighting areas where they excel and identifying potential improvements for future work. This section ultimately serves as

a critical examination of LLMs' potential to streamline data visualization workflows and offers insights into their broader application in this field. For our experiments, we chose two top-performing LLMs: GPT-4 [46] and Gemini Pro [45].

4.1 Research Questions

Our study is driven by two primary research questions:

- How effective are LLMs at extracting data tables from Vega-Lite visualizations?
- How effective are LLMs in restyling Vega-Lite charts through natural language queries?

These questions laid at the heart of our investigation, guiding the evaluation of LLM performance in two critical areas: the ability to reliably extract the underlying data from diverse Vega-Lite visualizations and the capability to dynamically modify chart styles based on user-driven natural language inputs. By focusing on these dimensions, we provide comprehensive insights into how well LLMs can handle both data extraction and chart customization tasks within the context of automated visualization systems.

4.2 Effectiveness of LLMs at Extracting Data Tables

For the ChartQA dataset, after iteratively refining the two-shot prompt specific to each chart type, we achieved high accuracy in extracting data from the 500 charts. This process involved significant effort and multiple iterations to perfect the prompts, ensuring they were tailored specifically for each chart type to yield optimal results.

Similarly, for the NVBench dataset, the results were consistent, with LLMs achieving a high level of accuracy in data extraction as observed in Table 4.1 and Table 4.2. Given that NVBench charts are also Vega-Lite generated, the successful application of our refined prompts validated the robustness of our approach.

The refinement process for the few-shot prompts involved several stages. Initial attempts revealed challenges in accurately extracting data, particularly with more complex chart types. Through detailed analysis of these initial errors, we identified common issues and areas for improvement. We then iteratively modified the prompts to address these issues, focusing on the unique characteristics of each chart type. Continuous testing and refinement of the prompts based on feedback from each iteration further enhanced their effectiveness.

One notable challenge was extracting data from line charts, which required interpreting the Move-to and Curve-to elements within the SVG path data. Our algorithm accurately extracted pixel values from these path elements, converting them into actual data points to ensure precise data extraction from the line charts.

To validate the accuracy of the extracted data tables, we employed a script from the ChartQA paper [29]. This script compared the ground truth tables with the extracted tables by computing a similarity score. It calculated a cost matrix between the elements of the ground truth and extracted tables using the following distance formula:

$$\text{distance}(x1, x2) = \min\left(1, \left|\frac{x1 - x2}{x1 + 1 \times 10^{-15}}\right|\right) \tag{4.1}$$

This formula calculates the normalized absolute difference between corresponding elements in the ground truth and extracted tables. The cost matrix is then used in the Hungarian algorithm to find the optimal assignment between the elements of the two tables, minimizing the overall cost. The final similarity score is calculated using the formula:

$$\text{score} = 1 - \frac{\sum \text{cost\_matrix}[\text{row\_ind}, \text{col\_ind}]}{\max(\text{len}(\text{lst1}), \text{len}(\text{lst2}))} \tag{4.2}$$

This score represents how closely the extracted table matches the ground truth, with a score of 1 indicating a perfect match. This rigorous approach ensured that the prompts were highly effective, leading to high accuracy in data extraction across both datasets. The iterative refinement process highlights the importance of customizing prompts to handle the specific nuances of different Vega-Lite visualizations.

Evaluation	Pie	Bar	Line	Scatter	Grouped
Few-Shot (ChartQA) [29]	97%	100%	89%	93%	84%
Zero-Shot (ChartQA)	86%	91%	79%	84%	77%
Few-Shot (NVBench) [34]	94%	98%	92%	-	-
Zero-Shot (NVBench)	79%	85%	81%	-	-

Table 4.1: Results of the Data Extraction Process with GPT-4
The table presents the evaluation results of data extraction from different chart types using several approaches: ChartQA, NVBench, Few-Shot Prompt

Evaluation	Pie	Bar	Line	Scatter	Grouped
Few-Shot (ChartQA) [29]	96%	99%	81%	88%	76%
Zero-Shot (ChartQA)	82%	86%	72%	76%	63%
Few-Shot (NVBench) [34]	95%	99%	86%	-	-
Zero-Shot (NVBench)	74%	87%	70%	–	–

Table 4.2: Results of the Data Extraction Process with LLaMA 70B
This table shows values for the evaluation results of data extraction from different chart types using LLaMA 70B with several approaches: ChartQA, NVBench, Zero-Shot Prompt (both ChartQA and NVBench)

4.3 Qualitative Analysis of Data Extraction Process

The qualitative analysis of the data extraction process highlights the strengths and comparative performance of the two large language models (LLMs), GPT-4 and LLaMA 70B. Both LLMs demonstrated high accuracy in extracting data tables, as shown in Tables 4.1 and 4.2. However, GPT-4 consistently outperformed LLaMA 70B across most chart types, particularly in Few-Shot settings.

For example, GPT-4 achieved an accuracy of 97% for pie charts and 100% for bar charts in the ChartQA dataset, compared to LLaMA 70B's 96% and 99%, respectively. Similarly, for line charts, GPT-4 outperformed LLaMA 70B with an 89% accuracy compared to 81%. This trend is also evident in zero-shot prompts, where GPT-4 consistently demonstrated

higher accuracy rates.

One potential reason for GPT-4's superior performance is its robust understanding of visual encodings and its ability to handle complex chart elements. This advantage is particularly noticeable in scatter and grouped bar charts, where LLaMA 70B showed a sharper decline in accuracy. For example, LLaMA 70B scored only 63% for grouped bar charts in the ChartQA dataset, while GPT-4 achieved 84%. This disparity suggests that GPT-4 may better leverage contextual understanding and relationships between chart elements.

Despite these differences, LLaMA 70B's performance remains commendable, especially in simpler chart types like pie and bar charts. Its slightly lower accuracy might be attributed a narrower understanding of the nuances in Vega-Lite visualizations.

Overall, the high accuracy rates of both LLMs indicate that they are well-suited for data extraction tasks. However, GPT-4's consistent edge across various chart types underscores its broader applicability and efficiency in handling complex and nuanced visual data.

4.4 Effectiveness of LLMs in Restyling Vega-Lite Charts

We evaluated four language models—GPT-4, Gemini, LLaMA 8B, and LLaMA 70B—on their ability to generate Vega-Lite queries for restyling charts based on natural language prompts. The models were tested on five chart types: bar chart, pie chart, line chart, grouped bar chart, and scatter plot, using identical prompts. Results were categorized as Accurate,

Partially Accurate, or Inaccurate, as summarized in Tables 4.3 and 4.4.

In the zero-shot setting (Table 4.3), GPT-4 achieved the highest accuracy rate at 30.1%, followed closely by Gemini at 28.1%. LLaMA 70B and LLaMA 8B had lower accuracy rates of 19.1% and 9.4%, respectively. This indicates that GPT-4 and Gemini are more capable of interpreting natural language prompts without prior examples.

In the few-shot setting (Table 4.4), accuracy improved for all models. GPT-4 reached an accuracy rate of 39.1%, slightly outperforming Gemini at 38.0%. LLaMA 70B showed a significant improvement to 27.2%, while LLaMA 8B increased to 12.8%. These improvements highlight the benefit of providing examples, especially for models with less capacity or smaller architectures.

Upon an overall analysis, GPT-4 outperformed all other models in terms of accuracy in the few-shot setting. GPT-4 achieved an accuracy rate of 39.1%, with a partial accuracy rate of 8.5%, and an inaccuracy rate of 52.4%. Gemini closely followed with an accuracy rate of 38.0%, a partial accuracy rate of 6.3%, and a higher inaccuracy rate of 55.7%. LLaMA 70B and LLaMA 8B had lower accuracy rates of 27.2% and 12.8%, respectively. LLaMA 70B exhibited a partial accuracy rate of 6.6% and an inaccuracy rate of 66.1%, while LLaMA 8B had a higher partial accuracy rate of 17.0% but a high inaccuracy rate of 70.2%. The higher partial accuracy in LLaMA 8B suggests that while it often failed to produce fully correct outputs, it frequently captured some aspects of the desired restyling.

Based on an analysis, the two most common reasons for inaccurate results were:

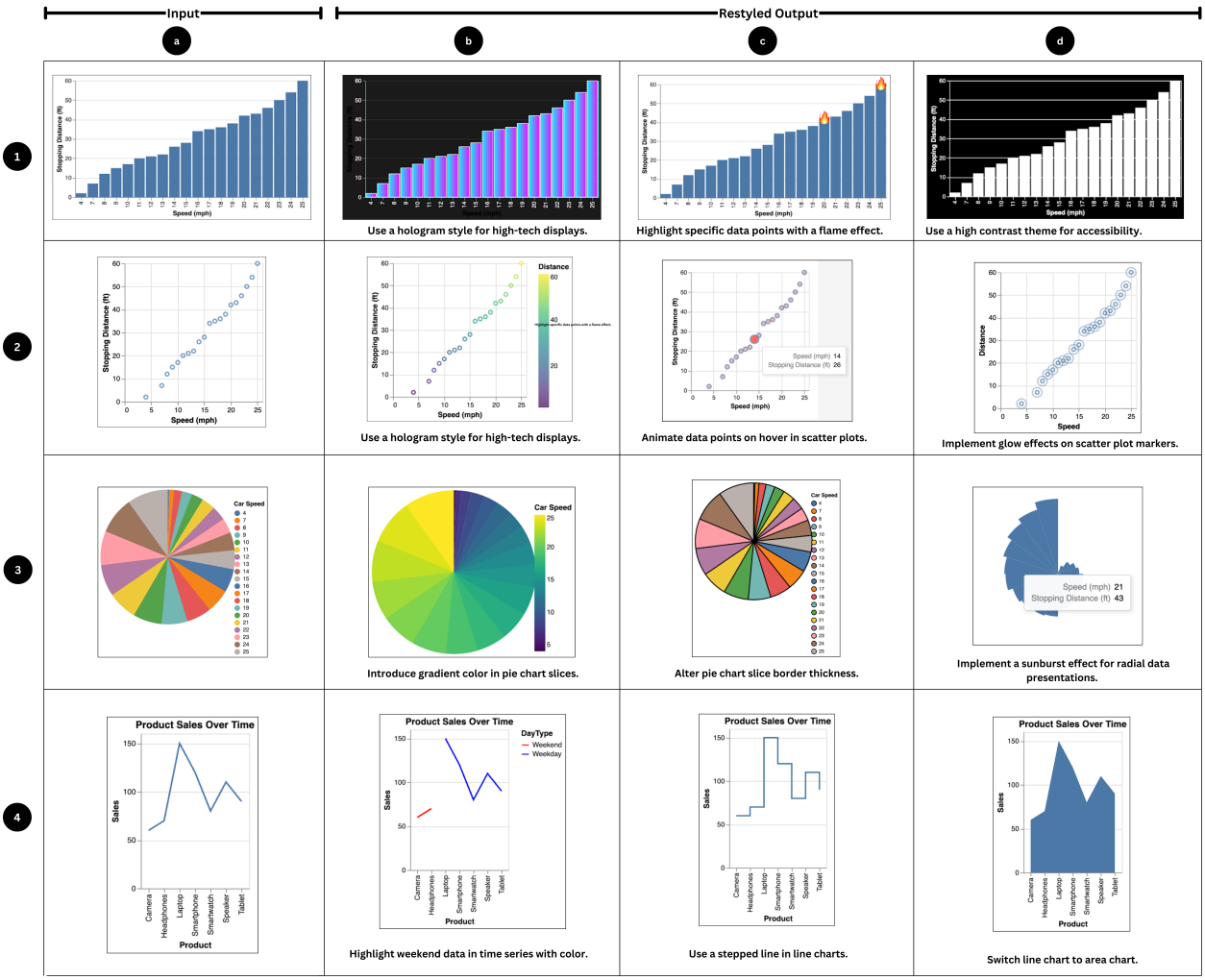

Figure 4.1: Examples of Restyled Vega-Lite Charts.
This figure shows various charts before and after restyling. The input charts are displayed in the first column (a), while columns (b), (c), and (d) illustrate restyled versions based on user-defined queries. The restyling includes effects such as holographic styles, specific point highlights, and contrast adjustments for accessibility. The rows represent different chart types: bar chart (row 1), scatter plot (row 2), pie chart (row 3), and line chart (row 4). Each chart demonstrates distinct visualization enhancements, ranging from styling to improved user interactivity.

- Prompts that required animation of any sort on or around the chart as can be seen in Figure 4.2 example C2, including for dynamic data updates, which is not inherently supported by Vega-Lite without additional JavaScript.

- Prompts that required design alterations, patterns, or images to be incorporated into the chart background rather than to chart elements as seen in Figure 4.2 example D2.

In summary, GPT-4 demonstrated superior performance in chart restyling from natural language prompts, with Gemini performing comparably. The LLaMA models showed improvements with few-shot learning, particularly LLaMA 70B, but still lagged behind GPT-4 and Gemini. These findings underscore the importance of model capacity and the diversity of training data. The effectiveness of few-shot learning is evident, especially in enhancing the performance of smaller models like LLaMA 8B and LLaMA 70B.

By providing examples, we observed that all models improved their ability to handle data visualization tasks. This suggests that incorporating few-shot learning strategies can be beneficial in practical applications where model performance needs enhancement.

Chart Type	Accurate				Partially Accurate				Inaccurate			
	GPT-4	Gemini	LLaMA 8B	LLaMA 70B	GPT-4	Gemini	LLaMA 8B	LLaMA 70B	GPT-4	Gemini	LLaMA 8B	LLaMA 70B
Bar Chart	65	62	20	35	30	28	20	15	119	128	170	155
Pie Chart	60	58	18	32	25	22	15	12	123	131	180	165
Line Chart	63	59	22	33	24	21	18	10	120	135	173	164
Scatter Plot	66	63	24	36	22	20	15	8	121	129	174	162
Grouped Chart	57	55	16	29	26	24	12	5	114	122	183	170
Total	**311**	**297**	**100**	**165**	**127**	**115**	**80**	**50**	**597**	**645**	**880**	**816**
Percentage	**30.1%**	**28.1%**	**9.4%**	**19.1%**	**12.3%**	**10.9%**	**7.4%**	**5.8%**	**57.7%**	**61.0%**	**83.0%**	**75.1%**

Table 4.3: Zero-Shot Performance of GPT-4, Gemini, LLaMA 8B, and LLaMA 70B on Chart Restyling Queries

Chart Type	Accurate				Partially Accurate				Inaccurate			
	GPT-4	Gemini	LLaMA 8B	LLaMA 70B	GPT-4	Gemini	LLaMA 8B	LLaMA 70B	GPT-4	Gemini	LLaMA 8B	LLaMA 70B
Bar Chart	81	81	25	48	20	14	34	14	109	115	150	118
Grouped Chart	74	73	23	43	16	12	33	11	104	109	133	113
Line Chart	80	77	26	49	16	12	35	8	108	115	138	112
Pie Chart	79	75	24	44	18	14	34	16	107	115	141	110
Scatter Plot	84	81	29	51	16	12	33	8	105	112	137	113
Total	**398**	**387**	**127**	**235**	**86**	**64**	**169**	**57**	**533**	**566**	**699**	**571**
Percentage	**39.1%**	**38.0%**	**12.8%**	**27.2%**	**8.5%**	**6.3%**	**17.0%**	**6.6%**	**52.4%**	**55.7%**	**70.2%**	**66.1%**

Table 4.4: Few-shot comparative Performance of GPT-4, Gemini, LLaMA 8B, and LLaMA 70B on Chart Restyling Queries with Totals and Percentages

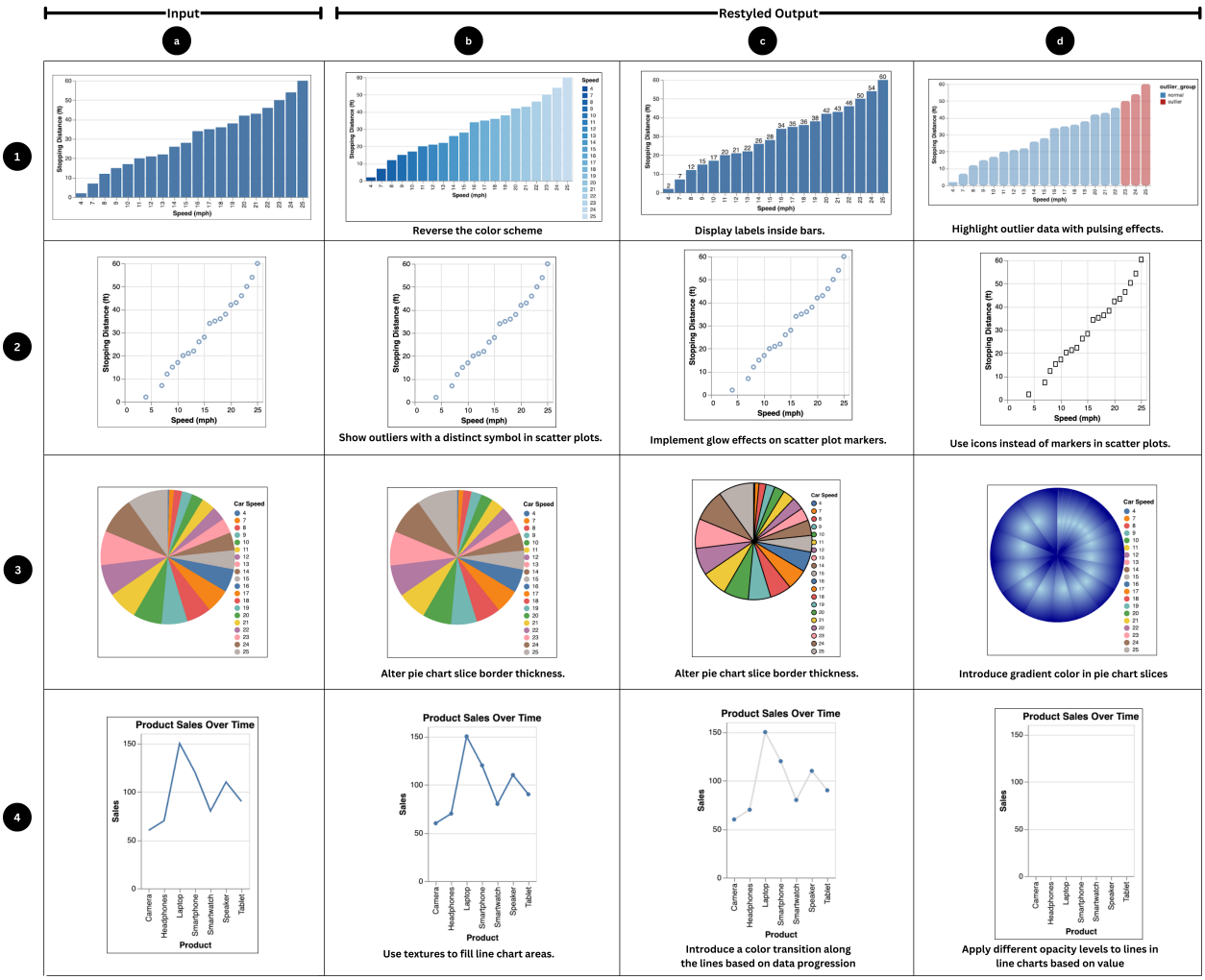

Figure 4.2: Examples of Failed Restyled Vega-Lite Charts.
This figure contrasts the original charts (column a) with various attempted restyling methods (columns b-d). These attempts include changes such as reversing color schemes, repositioning labels, highlighting outliers, and applying gradient effects. Despite the intended improvements, these restyling efforts resulted in suboptimal outcomes, particularly for bar charts, scatter plots, pie charts, and line charts, where visual clarity or intended emphasis was compromised.

4.5 Qualitative Analysis of Restyling Failures

Throughout our study, we identified several restyling failures when utilizing LLMs to modify visualizations, particularly for tasks requiring complex visual enhancements that fall outside

Vega-Lite's capabilities. The LLMs showed notable difficulties when tasked with executing advanced visual modifications or animations, often producing incomplete or inaccurate outputs.

In Figure 4.2, several examples illustrate these failures:

1-b (Reverse Color Scheme): The LLM struggled to reverse the color scheme while maintaining consistent color scales and labels. Although Vega-Lite supports simple color scale manipulations, the model could not realign the scales or reorient them appropriately, leading to a misrepresentation of the data.

1-c (Display Labels Inside Bars): The request to display labels inside the bars resulted in misaligned and overlapping text. Vega-Lite's default text positioning is limited, and the model failed to make the necessary adjustments to fit the labels within the varying bar sizes, highlighting a weakness in modifying text placement.

2-b (Outlier Symbol in Scatter Plots): In this case, the LLM failed to distinctly mark outliers with special symbols. Vega-Lite supports basic conditional formatting, but it is limited when handling unique symbols for data subsets such as outliers, which requires more detailed customization than Vega-Lite typically offers.

2-c (Glow Effects on Scatter Plot Markers): The application of glow effects to scatter plot markers did not succeed. Vega-Lite does not natively support such advanced graphical effects, resulting in a chart that lacked any glow enhancement, illustrating the model's inability to

apply visual effects that require more sophisticated rendering capabilities.

In addition to these cases, examples involving animations—such as 1-d (Pulsing Effects on Outliers) and 3-d (Transitions in Pie Charts)—further emphasize the limitations of both the LLMs and Vega-Lite. Requests for pulsing effects or dynamic transitions resulted in no functional output, as Vega-Lite does not inherently support animated transitions or complex visual dynamics. While Vega-Lite can handle interactive elements like hover effects, it lacks native support for temporal or animated effects without external tools or libraries.

While some of these failures can be attributed to Vega-Lite's inherent limitations, others clearly stem from the LLM's inability to modify the chart as specified by the user. Overall, these failures highlight a significant gap between user expectations for advanced restyling tasks and the LLM's current capacity to handle more complex queries.

In summary, the evaluation provided in this section highlights both the successes and limitations of using LLMs for data extraction and restyling within Vega-Lite visualizations. While LLMs demonstrate high accuracy in extracting data tables across a variety of chart types after prompt refinement, challenges remain, particularly with more complex visual structures like grouped bar charts or charts missing key reference points. Similarly, LLM performance in restyling visualizations shows promise, especially for straightforward tasks, but struggles with more advanced visual modifications, such as animations and intricate design changes. The findings from this evaluation emphasize the potential of LLMs to enhance automation in data visualization workflows, while also identifying areas that need

further refinement. Ultimately, this section contributes critical insights into how LLMs can be effectively leveraged in visualization tasks and suggests directions for future improvements.

Conclusion

5.1 Summary of Contributions

The key goal of our research was centered on evaluating the capabilities of large language models in two critical areas: extracting data tables and visual encodings from SVG chart images, and using user queries to restyle Vega-Lite SVG images. Our findings provide valuable insights into these aspects, demonstrating the potential and limitations of LLMs in the domain of data visualization.

First, the evaluation of LLMs in extracting data tables and visual encodings from SVG chart images revealed that these models are proficient in identifying and interpreting complex visual elements. The accuracy and consistency of data extraction were high, showcasing the effectiveness of LLMs when combined with advanced prompt engineering techniques and algorithms designed for interpreting SVG paths. These results underscore the ability of LLMs

to facilitate the conversion of visual data into a structured format, which is crucial for further analysis and usage [6, 31].

Second, our research also examined the effectiveness of LLMs in restyling Vega-Lite SVG images based on user queries. The results indicated that LLMs are capable of understanding and implementing a wide range of visual modifications as requested by users. This includes changes to color schemes, chart types, and other visual properties, making the process of customization more accessible and intuitive for users. The ability to transform user queries into dynamic visual adjustments emphasizes the potential of LLMs in enhancing the interactivity of data visualizations [9, 32]. However, we found that restyling accuracy was significantly lower than deconstruction accuracy, indicating substantial room for improvement in this task.

5.2 Conclusion and Future Works

While the development of our Chrome plugin provided a practical application for these findings, the focus remained on the broader implications of LLMs in data visualization. The success in both data extraction and restyling tasks highlights the promising role of LLMs in this field, suggesting that further exploration and refinement could lead to even more robust tools and methodologies [30, 7].

Future research will continue to refine these capabilities, aiming to improve the handling

of more complex queries and enhance the interactive aspects of visualizations. Extending Vega-Lite specification to include animation could help support a variety of queries that require animated visualizations [55]. In the future, LLM-driven chart deconstruction methods can be leveraged for other applications including chart accessibility, visualization recommendation systems and visualization education. Additionally, expanding the applicability of these techniques to other visualization platforms and exploring deeper integrations with AI advancements will be essential steps forward [19, 18].

In conclusion, our study contributes to the ongoing discourse on the integration of AI in data visualization, offering evidence of the effectiveness of LLMs in both data extraction and restyling tasks. We hope that these findings not only advance our understanding of LLM capabilities for visualization-related tasks but also pave the way for the development of more sophisticated, user-friendly visualization tools in the future. Access to all of our prompts, source code, and datasets can be found on the following github link:- https://github.com/SyedZaidi01/Syed-Ali-s-Thesis.

Bibliography

[1] Md Zubair Ibne Alam, Shehnaz Islam, and Enamul Hoque. "SeeChart: enabling accessible visualizations through interactive natural language interface for people with visual impairments". In: *Proceedings of the 28th International Conference on Intelligent User Interfaces*. 2023, pp. 46–64.

[2] Zubair Ibne Alam. *ENABLING ACCESSIBLE CHARTS THROUGH INTERACTIVE NATURAL LANGUAGE INTERFACE FOR PEOPLE WITH VISUAL IMPAIRMENTS*.

[3] Leilani Battle et al. "Beagle: Automated extraction and interpretation of visualizations from the web". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–8.

[4] Jenny Joyce Beumer, Ab De Haan, and Josine Van Der Ven. *IResearch Notes Implications of Computer-Mediated Communication for People Who Are Visually Impaired in Dealing with Complex Visualization Tasks*. 2000.

[5] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. "D3 Data-Driven Documents". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2301–2309.

[6] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *arXiv preprint arXiv:2005.14165* (2020).

[7] Meng Cai and Xin Wang. "Visualizing Data with SVG". In: *IEEE Computer Graphics and Applications* 39.2 (2019), pp. 76–85.

[8] Alberto Cairo. *The Functional Art: An introduction to information graphics and visualization*. New Riders, 2013.

[9] Banghao Chen et al. "Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review". In: (Oct. 2023). URL: http://arxiv.org/abs/2310.14735.

[10] Chen Chen et al. "Mystique: Deconstructing SVG Charts for Layout Reuse". In: *IEEE Transactions on Visualization and Computer Graphics* (2023).

[11] Jinho Choi et al. "Visualizing for the non-visual: Enabling the visually impaired to use visualization". In: *Computer Graphics Forum*. Vol. 38. 3. Wiley Online Library. 2019, pp. 249–260.

[12] Mathieu Cliche et al. "Scatteract: Automated extraction of data from scatter plots". In: *Machine Learning and Knowledge Discovery in Databases: European Conference,*

ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part I 10. Springer. 2017, pp. 135–150.

[13] Douglas Crockford. *The application/json Media Type for JavaScript Object Notation (JSON)*. RFC 4627. 2006. URL: https://www.ietf.org/rfc/rfc4627.txt.

[14] Xiaofei Cui et al. "Text-to-Chart: A Framework for Chart Generation from Natural Language Queries". In: *IEEE Access* 7 (2019), pp. 86774–86788.

[15] *D3.js - Data-Driven Documents*. 2023. URL: https://d3js.org/.

[16] Xuan Long Do et al. "Do LLMs Work on Charts? Designing Few-Shot Prompts for Chart Question Answering and Summarization". In: *arXiv preprint arXiv:2312.10610* (2023).

[17] Frank Elavsky, Cynthia Bennett, and Dominik Moritz. "How accessible is my visualization? Evaluating visualization accessibility with Chartability". In: *Computer Graphics Forum* 41 (3 June 2022), pp. 57–70. ISSN: 14678659. DOI: 10.1111/cgf.14522.

[18] Roy Thomas Fielding. "Architectural Styles and the Design of Network-based Software Architectures". In: *Doctoral dissertation, University of California, Irvine* (2000).

[19] David Flanagan. *JavaScript: The Definitive Guide*. O'Reilly Media, 2020.

[20] Jonathan Harper and Maneesh Agrawala. "Deconstructing and restyling D3 visualizations". In: Association for Computing Machinery, Oct. 2014, pp. 253–262. ISBN: 9781450330695. DOI: 10.1145/2642918.2647411.

[21] Jeffrey Heer and Mike Bostock. "Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2010), pp. 203–212.

[22] Enamul Hoque and Maneesh Agrawala. "Searching the Visual Style and Structure of D3 Visualizations". In: *IEEE Transactions on Visualization and Computer Graphics* 26 (1 Jan. 2020), pp. 1236–1245. ISSN: 19410506. DOI: 10.1109/TVCG.2019.2934431.

[23] Mohammed Saidul Islam et al. "Are Large Vision Language Models up to the Challenge of Chart Comprehension and Reasoning? An Extensive Investigation into the Capabilities and Limitations of LVLMs". In: *arXiv preprint arXiv:2406.00257* (2024).

[24] J. David Jackson. "SVG Essentials". In: *O'Reilly Media* (2017).

[25] David Johnson, Wei Li, and Ming Wang. "Interactive Data Visualization with AI: Building Effective Tools for Data Analysis". In: *Journal of Visual Languages and Computing* 65 (2022), p. 100947.

[26] Daekyoung Jung et al. "ChartSense: Interactive data extraction from chart images". In: vol. 2017-May. Association for Computing Machinery, May 2017, pp. 6706–6717. ISBN: 9781450346559. DOI: 10.1145/3025453.3025957.

[27] Ecem Kavaz, Anna Puig, and Inmaculada Rodríguez. "Chatbot-Based Natural Language Interfaces for Data Visualisation: A Scoping Review". In: *Applied Sciences* 13 (12 June 2023), p. 7025. ISSN: 20763417. DOI: 10.3390/app13127025.

[28] Dae Hyun Kim, Enamul Hoque, and Maneesh Agrawala. "Answering Questions about Charts and Generating Visual Explanations". In: Association for Computing Machinery, Apr. 2020. ISBN: 9781450367080. DOI: 10.1145/3313831.3376467.

[29] N. W. Kim et al. "Accessible Visualization: Design Space, Opportunities, and Challenges". In: *Computer Graphics Forum* 40 (3 June 2021), pp. 173–188. ISSN: 14678659. DOI: 10.1111/cgf.14298.

[30] Richard E. Ladner. "Designing Accessible Technology". In: *Communications of the ACM* 58.1 (2015), pp. 34–36.

[31] Yi Liu et al. "Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study". In: (May 2023). URL: http://arxiv.org/abs/2305.13860.

[32] Edward Loper and Steven Bird. "NLTK: The Natural Language Toolkit". In: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*. 2002, pp. 63–70.

[33] Junyu Luo et al. "ChartOCR: Data Extraction from Charts Images via a Deep Hybrid Framework". In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2021), pp. 1916–1924.

[34] Yuyu Luo, Jiawei Tang, and Guoliang Li. "nvBench: A Large-Scale Synthesized Dataset for Cross-Domain Natural Language to Visualization Task". In: (Dec. 2021). URL: http://arxiv.org/abs/2112.12926.

[35] Paula Maddigan and Teo Susnjak. "Chat2VIS: Generating Data Visualisations via Natural Language using ChatGPT, Codex and GPT-3 Large Language Models". In: (Feb. 2023). URL: http://arxiv.org/abs/2302.02094.

[36] Ahmed Masry et al. "ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning". In: *Findings of the Association for Computational Linguistics: ACL 2022*. 2022, pp. 2263–2279.

[37] Ahmed Masry et al. *ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning*. URL: https://github.com/vis-nlp/ChartQA.

[38] Angela Mayhua et al. "Extracting visual encodings from map chart images with color-encoded scalar values". In: *2018 31st SIBGRAPI Conference on graphics, patterns and images (SIBGRAPI)*. IEEE. 2018, pp. 142–149.

[39] Mico McNabb. *Data Visualization for Dummies*. John Wiley & Sons, 2020.

[40] Microsoft. *Microsoft Power BI*. Business analytics service by Microsoft. 2024. URL: https://powerbi.microsoft.com/.

[41] OpenAI. *GPT-4*. https://openai.com/research/gpt-4. 2024.

[42] Jorge Poco, Angela Mayhua, and Jeffrey Heer. "Extracting and retargeting color mappings from bitmap images of visualizations". In: *IEEE transactions on visualization and computer graphics* 24.1 (2017), pp. 637–646.

[43] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: *OpenAI Blog* 1.8 (2019), p. 9.

[44] Lena Reed et al. “Few-shot Natural Language Generation for Task-oriented Dialog Systems”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.16 (2021), pp. 142–150.

[45] Gemini Technical report. *Gemini: A Family of Highly Capable Multimodal Models*. Technical Report. Stanford InfoLab, 2024. URL: https://storage.googleapis.com/deepmind-media/gemini/gemini_1_report.pdf.

[46] GPT 4 Technical report. *GPT-4 Technical Report*. Technical Report. Stanford InfoLab, 2023. URL: https://cdn.openai.com/papers/gpt-4.pdf.

[47] Arvind Satyanarayan et al. “Vega-Lite: A Grammar of Interactive Graphics”. In: *IEEE Transactions on Visualization & Computer Graphics (Proc. IEEE InfoVis)* (2017). DOI: 10.1109/TVCG.2016.2599030. URL: http://vis.csail.mit.edu/pubs/vega-lite.

[48] Arvind Satyanarayan et al. “Vega-Lite: A Grammar of Interactive Graphics”. In: *IEEE Transactions on Visualization and Computer Graphics*. Vol. 23. 1. 2017, pp. 341–350.

[49] Manolis Savva and Nicholas Kong. *ReVision: Automated Classification, Analysis and Redesign of Chart Images*. Association for Computing Machinery, 2011, p. 658. ISBN: 9781450307161.

[50] Ather Sharif et al. “VoxLens: Making online data visualizations accessible with an interactive JavaScript plug-in”. In: *Proceedings of the 2022 CHI conference on human factors in computing systems*. 2022, pp. 1–19.

[51] Noah Siegel et al. “FigureSeer: Parsing Result-Figures in Research Papers”. In: *ECCV*. 2016.

[52] Tableau Software. *Tableau Software*. Data visualization and business intelligence tool. 2024. URL: https://www.tableau.com/.

[53] Google AI Team. *Gemini: A Family of Highly Capable Multimodal Models*. arXiv preprint arXiv:2312.11805. 2023.

[54] Lu Ying et al. *Reviving Static Charts into Live Charts*.

[55] Jonathan Zong et al. “Animated Vega-Lite: Unifying animation with a grammar of interactive graphics”. In: *IEEE Transactions on Visualization and Computer Graphics* 29.1 (2022), pp. 149–159.

Appendices

Appendix A: Data and Visual Encoding Extraction Prompts

Appendix B: Visual Encoding Queries

- Bar Chart: Change the bar colors to red.
- Line Chart: Increase line chart line thickness.
- All Charts: Set background color to yellow.
- Scatter Plot: Enhance point opacity in scatter plots.
- All Charts: Apply dashed lines to grid.
- Bar Chart: Highlight maximum values on bar charts.
- Pie Chart: Introduce gradient color in pie chart slices.

- All Charts: Expand chart padding.
- All Charts: Add text shadow effects (try with other chart type).
- All Charts: Bold the axis labels.
- All Charts: Increase title font size.
- All Charts: Change font to Times New Roman.
- Bar Chart: Round the bar chart corners.
- Pie Chart: Alter pie chart slice border thickness.
- All Charts: Reverse the color scheme.
- Line Chart: Apply a monochrome palette to line charts.
- All Charts: Change tooltip font to Verdana.
- All Charts: Lengthen x-axis tick marks.
- Grouped Bar Chart: Tighten bar spacing in grouped charts.
- All Charts: Tilt axis labels to 45 degrees.
- All Charts: Emphasize background grid lines.
- All Charts: Frame the chart with a border.
- Pie Chart: Use pattern fills for pie slices.

- Scatter Plot: Emphasize specific data points in orange.
- All Charts: Reduce opacity for non-selected chart elements.
- Bar Chart: Implement zebra striping in bar charts.
- Line Chart: Switch line chart to area chart.
- All Charts: Add a watermark to the chart background.
- All Charts: Use a vintage color scheme.
- Scatter Plot: Apply neon colors to scatter plot points.
- Bar Chart: Introduce image-based fill in bars.
- All Charts: Alternate tick color on the y-axis.
- All Charts: Display all text in italics.
- All Charts: Underline the chart title.
- All Charts: Increase grid line transparency.
- Line Chart: Add a glow effect to line chart lines.
- All Charts: Show axes in different colors.
- Scatter Plot: Make scatter plot markers square.
- Pie Chart: Decrease the size of pie chart labels.

- All Charts: Use a double line border around the chart.
- Pie Chart: Blend colors between adjacent pie slices.
- All Charts: Customize tooltip background color.
- All Charts: Brighten the colors for better visibility.
- All Charts: Display axis labels vertically.
- All Charts: Introduce a subtle gradient in the background.
- All Charts: Soften all font colors.
- All Charts: Overlay a semi-transparent panel for annotations.
- Bar Chart: Display negative values in a contrasting color.
- Scatter Plot: Use icons instead of markers in scatter plots.
- All Charts: Decrease the chart's aspect ratio.
- Bar Chart: Introduce a 3D effect on bars.
- Bar Chart: Display labels inside bars.
- Line Chart: Color lines based on value thresholds.
- Pie Chart: Use dashed borders for pie slices.
- Scatter Plot: Color-code data points by category in scatter plots.

- All Charts: Stylize the legend with a background color.
- All Charts: Apply a shadow to grid lines.
- All Charts: Increase label readability with a stroke.
- All Charts: Display chart title in uppercase.
- All Charts: Subdue all non-highlighted elements.
- All Charts: Introduce interactivity: hover to change color.
- Bar Chart: Visualize negative bars in a different style.
- Line Chart: Use textures to fill line chart areas.
- Line Chart: Highlight weekend data in time series with color.
- All Charts: Apply a vintage paper texture as chart background.
- Scatter Plot: Show outliers with a distinct symbol in scatter plots.
- Bar Chart: Alternate the opacity for rows in bar charts.
- Pie Chart: Use a radial gradient on pie charts.
- All Charts: Fade out less important data.
- All Charts: Color bands on the background based on thresholds.
- Bar Chart: Use shadows to emphasize bars in bar charts.

- Line Chart: Apply different opacity levels to lines in line charts based on value.
- Line Chart: Introduce a color transition along the lines based on data progression.
- All Charts: Highlight specific data ranges with bold colors.
- All Charts: Make axis ticks bold.
- All Charts: Use a unique font for headers and labels.
- All Charts: Apply contrasting color for tooltips.
- All Charts: Enable shadow effect on all text elements.
- All Charts: Use a script typeface for all labels.
- All Charts: Apply a light box shadow around the main chart area.
- All Charts: Highlight certain axes with a bright color.
- All Charts: Introduce blur effect for background elements.
- Scatter Plot: Use a polar coordinate system for scatter plots.
- All Charts: Highlight specific data series in bold.
- All Charts: Customize line patterns per dataset.
- Pie Chart: Use a metallic color scheme for pie charts.
- Bar Chart: Soften edges of bar chart bars.

- All Charts: Adjust label spacing to improve readability.
- All Charts: Implement a radial background pattern.
- All Charts: Highlight seasonal trends with specific colors.
- Pie Chart: Adjust the opacity of pie chart segments.
- All Charts: Introduce flashing effects for dynamic data updates.
- Scatter Plot: Use a heat map color gradient in scatter plots.
- Bar Chart: Implement variable bar widths based on data value.
- All Charts: Apply a textured background to highlight data segments.
- Line Chart: Use a stepped line in line charts.
- All Charts: Introduce a watermark with company logo.
- All Charts: Customize cursor on hover.
- All Charts: Apply a glow effect to axis lines.
- Line Chart: Use patterned fills for line chart areas.
- All Charts: Highlight new data additions in a different color.
- Scatter Plot: Animate data points on hover in scatter plots.
- All Charts: Implement data-dependent axis label color changes.

- Bar Chart: Use dual-color themes for alternating bars.
- All Charts: Present negative and positive data in contrasting styles.
- Pie Chart: Introduce shadowing under pie chart slices for depth.
- All Charts: Highlight outlier data with pulsing effects.
- All Charts: Use photo-realistic textures for chart backgrounds.
- All Charts: Introduce a chalkboard effect for background.
- All Charts: Alternate the font weight across labels.
- All Charts: Implement label hiding for smaller screen sizes.
- Bar Chart: Add a brushed metal effect to bar charts.
- All Charts: Use an outline effect for text.
- All Charts: Highlight user-selected data with border enhancements.
- All Charts: Implement a gradient effect on axis labels.
- All Charts: Use a discrete color scale for heatmaps.
- All Charts: Introduce reflective effects for chart elements.
- All Charts: Overlay a linear gradient across the entire chart.
- All Charts: Use color flashing to signify real-time data updates.

- Bar Chart: Implement striped patterns within bar charts.
- All Charts: Enhance grid lines with a 3D effect.
- All Charts: Use bright, saturated colors for urgent data highlights.
- Scatter Plot: Implement glow effects on scatter plot markers.
- All Charts: Animate chart elements on data refresh.
- All Charts: Stylize tooltips with CSS for modern appearance.
- All Charts: Highlight changes in data over time with color transitions.
- All Charts: Use color inversion for emphasis on specific chart sections.
- All Charts: Implement conditional formatting based on external data triggers.
- Scatter Plot: Highlight data points in scatter plots with neon borders.
- All Charts: Use a historical color theme for retrospective data analysis.
- All Charts: Introduce a spotlight effect for focal data.
- All Charts: Use a radial burst background pattern.
- All Charts: Highlight cyclic trends with specific pattern overlays.
- All Charts: Add detailed borders around selected chart elements.
- All Charts: Introduce fading effects for older data in time series.

- All Charts: Use a split complementary color scheme for charts.
- All Charts: Highlight user interactions with real-time color changes.
- All Charts: Implement a glass effect for background elements.
- All Charts: Use a layered approach for multiple data series.
- All Charts: Introduce a sketch effect for all chart elements.
- All Charts: Apply dynamic scaling to emphasize data variations.
- All Charts: Use color gradients to represent time evolution in data.
- All Charts: Implement shadow effects for depth perception.
- All Charts: Use a minimalist color scheme for clarity.
- All Charts: Highlight peak values with a special marker.
- All Charts: Introduce non-standard geometric shapes for data representation.
- All Charts: Use crosshatching patterns for background shading.
- All Charts: Apply artistic text effects for titles and labels.
- All Charts: Use a high contrast theme for accessibility.
- All Charts: Introduce an outline effect on hover for data points.
- All Charts: Use data-driven dynamic backgrounds.

- All Charts: Apply a neon glow to critical data points.
- All Charts: Introduce a frost effect for background areas.
- All Charts: Highlight specific time periods with color blocking.
- All Charts: Use a newsprint texture for retro appeal.
- All Charts: Apply a cut-out style for textual elements.
- All Charts: Highlight significant data trends with pulsating effects.
- All Charts: Use a depth effect for bar charts to indicate layers.
- All Charts: Implement a hand-drawn effect for all chart outlines.
- All Charts: Highlight specific data points with animation.
- All Charts: Use a muted color palette for background elements.
- All Charts: Introduce a spotlight effect on hovered data.
- Scatter Plot: Highlight correlation in scatter plots with connecting lines.
- All Charts: Use a photo overlay as chart background.
- All Charts: Implement a duotone color scheme for visual clarity.
- All Charts: Highlight specific ranges with gradient color fills.
- All Charts: Use an emboss effect for text elements.

- All Charts: Implement a scrolling effect for time series data.
- All Charts: Highlight fluctuations with color intensity changes.
- All Charts: Use a watermark effect for subtle branding.
- All Charts: Apply a pixelated effect for a digital look.
- All Charts: Use a wireframe background for a technical appearance.
- Bar Chart: Highlight specific columns in bar charts with texture.
- All Charts: Use a mirror effect for symmetry in data presentation.
- All Charts: Highlight data series intersections with special markers.
- All Charts: Use a transparent overlay for detailed comparison.
- All Charts: Implement a glow effect around chart boundaries.
- All Charts: Highlight specific data series with blinking effects.
- All Charts: Use an LED display style for digital dashboards.
- All Charts: Implement a sunburst effect for radial data presentations.
- All Charts: Highlight data trends with a shadow trail.
- All Charts: Use a cross-stitch pattern for background textures.
- All Charts: Implement a neon sign effect for chart titles.

- All Charts: Highlight specific periods with a timeline marker.
- All Charts: Use a frosted glass effect for overlay panels.
- All Charts: Highlight specific data segments with vibrancy effects.
- All Charts: Use a folded paper effect for background layers.
- All Charts: Implement a shadow effect on all non-data elements.
- All Charts: Highlight specific data points with a spotlight.
- All Charts: Use a color wash effect for background continuity.
- All Charts: Highlight comparative data with side-by-side coloring.
- All Charts: Use a ripple effect for interactive data points.
- All Charts: Implement a holographic effect for futuristic displays.
- All Charts: Highlight specific data ranges with pattern fills.
- All Charts: Use a mosaic tile background for complex data sets.
- All Charts: Highlight data density with color saturation.
- Pie Chart: Use a stained glass effect for pie chart segments.
- All Charts: Implement a depth illusion for three-dimensional data representation.
- All Charts: Highlight specific data series with neon outlines.

- All Charts: Use a watercolor wash for a soft background effect.
- All Charts: Highlight user interactions with animated borders.
- All Charts: Use a ripple effect for dynamic data interaction.
- All Charts: Implement a laser light show effect for presentation highlights.
- All Charts: Highlight specific periods with a brushed highlight.
- All Charts: Use a satin finish for background elegance.
- All Charts: Highlight specific data points with sparkle effects.
- All Charts: Use a hologram style for high-tech displays.
- All Charts: Highlight significant data changes with a flash effect.
- All Charts: Use a sand texture for a natural background feel.
- All Charts: Highlight specific data comparisons with contrasting shadows.
- All Charts: Use a crystal clear effect for transparency.
- All Charts: Highlight data transitions with color fades.
- All Charts: Use a neon outline for key data points.
- All Charts: Highlight specific areas with a magnifying effect.
- All Charts: Use a marble texture for chart backgrounds.

- All Charts: Highlight data series with glow effects.
- All Charts: Use a light burst effect for dramatic presentations.
- All Charts: Highlight specific data trends with a wave pattern.
- All Charts: Use a diamond texture for luxury data displays.
- All Charts: Highlight specific periods with a color shift.
- All Charts: Use a silk texture for a smooth background.
- All Charts: Highlight interactive elements with a pulsating glow.
- All Charts: Use a gemstone effect for valuable data points.
- All Charts: Highlight data intersections with a spotlight.
- All Charts: Use a parchment texture for a historical look.
- All Charts: Highlight specific data points with a laser pointer effect.
- All Charts: Use a velvet texture for a rich background.
- All Charts: Highlight data outliers with a glowing halo.
- All Charts: Use a brick pattern for a sturdy background.
- All Charts: Highlight seasonal data with thematic colors.
- All Charts: Use a water ripple effect for fluid data presentations.

- All Charts: Highlight data series with contrasting outlines.
- All Charts: Use a lava lamp effect for dynamic data flows.
- All Charts: Highlight specific data ranges with a spotlight.
- All Charts: Use a frosted pattern for a cold data theme.
- All Charts: Highlight specific data points with a sparkle effect.
- All Charts: Use a rain effect for fluid data visualization.
- All Charts: Highlight key data points with a neon glow.
- All Charts: Use a cloud pattern for a soft background.
- All Charts: Highlight comparative data with a split-screen effect.
- All Charts: Use a snow effect for winter data themes.
- All Charts: Highlight specific data trends with a shadow effect.
- All Charts: Use a leather texture for a premium background.
- All Charts: Highlight specific data points with a flame effect.
- All Charts: Use a mirror finish for a reflective background.
- All Charts: Highlight user selections with a radial glow.
- All Charts: Use a geometric pattern for structured data presentation.

- All Charts: Highlight data changes with a color morphing effect.
- All Charts: Use a sandblast effect for a textured background.
- All Charts: Highlight specific data points with a twinkling effect.