A Vega-Lite Dataset and Natural Language Generation Pipeline with Large Language Models

Hyung-Kwon Ko* KAIST
Hyeon Jeon† Seoul National University
Gwanmo Park‡ Seoul National University
Dae Hyun Kim§ KAIST
Nam Wook Kim¶ Boston College
Juho Kim‖ KAIST
Jinwook Seo** Seoul National University

ABSTRACT

There is a growing trend of utilizing Visualization-oriented Natural Language Interfaces (V-NLIs) to author charts. However, researchers consistently highlight the lack of high-quality chart and natural language datasets, which impedes the development of more sophisticated and data-driven systems using V-NLIs. In this study, we present a meticulously curated collection of human-generated 1,981 Vega-Lite specifications, derived from real-world data, and use Large Language Models (LLMs) for generating natural language queries for chart generation tasks. Unlike previous datasets that relied on relatively simple and homogeneous templates, our Vega-Lite dataset contains more complex and diverse (i.e., varying interactions, composite views, and different chart types). Using this dataset, we demonstrate generating natural language queries for chart generation, and how the results can be different when different input types are used (e.g., Vega-Lite, Image, both Vega-Lite and Image).

Index Terms: Human-centered computing—Visualization; Human-centered computing—Interaction paradigms—Natural language interfaces

1 INTRODUCTION

Visualization-oriented Natural Language Interfaces (V-NLIs) have garnered significant attention due to their user-friendly nature [22, 24, 25] which lets the users focus on performing tasks rather than learning how to interact with systems [5]. To develop pragmatic V-NLIs systems, leveraging high-quality chart datasets is vital to effectively emulate natural user interactions [8]. Likewise, collecting natural language (NL) queries that accurately simulate real user utterances for a given task is essential to improve the usability and performance of the system [23]. Based on such high-quality chart and NL datasets, researchers are empowered to invent systems that enable chart authoring, while also introducing novel data-driven interaction techniques.

However, recent survey papers [3, 6, 22] have consistently pointed out that there is a scarcity of publicly accessible high-quality chart and NL datasets appropriate for building advanced V-NLIs systems. Considering that many NL queries are generated grounded in the available chart datasets [13, 16, 23], the quality of NL datasets are heavily dependent on the quality of chart datasets. We observe that current chart datasets [9, 16, 28] focus on quantity (e.g., number of charts) over quality, which are insufficient to cover real-world cases with enough complexity and diversity. Moreover, past research typically use crowdsourcing to collect different types of NL inputs (e.g., statements, phrases, questions, etc.) by asking crowd workers to come up with generation queries. However, this approach is often time-consuming, expensive [7, 27], which impacts the scalability of the datasets, as well as it is prone to issues such as participant laziness and the collection of subpar queries [1].

In this work, we introduce a comprehensive collection of human-generated 1,981 Vega-Lite specifications obtained from GitHub (Fig. 2). Our dataset contains complex and diverse specs compared to previous datasets in multiple aspects. In detail, our dataset covers varying levels of complexity from a simple scatter plot without any interaction (e.g., simple) to a chart with three views where data points are linked with a click interactions (e.g., extra complex) (see the charts highlighted with red stroke in Fig. 2), where we more focus on complex levels (more than 86% are complex and extra complex). Moreover, our dataset shows a higher average pairwise edit distance between specs, which proves that our specs are highly diverse from one another. In detail, it contains the largest number of charts with composite views, interaction (e.g., tooltips, panning & zooming, and linking), and diverse chart types (e.g., map, grid & matrix, diagram, etc.).

We also propose leveraging LLMs to generate synthetic yet realistic NL queries instead of manual collection. In this preliminary research, we focus on the task of generating NL queries for chart generation. We compare three different input types (i.e., Vega-Lite spec, Image, and both) to generate queries for four sample charts with different level of complexities. We find that there is no significant difference in the output between Vega-Lite spec and both inputs, but using only image of the chart can contain an erroneous information in the query. We plan to generalize this NL generation pipeline so that it can be utilized to make diverse types of queries needed for various tasks in V-NLIs research.

The main contributions of our work are summarized as follows:

- We collect 1,981 Vega-Lite specifications that are highly diverse and complex compared to previous datasets;
- We experiment with an LLM to generate NL queries for chart generation task to understand its capability with lower cost (e.g., time and money).

2 BACKGROUND AND RELATED WORK

According to Chen et al.'s recent survey [3], chart corpora are typically collected in three formats: bitmap graphics (e.g., `.png`), vector graphics (e.g., `.svg`), and programs (e.g., Vega-Lite specifications [18]). Among the surveyed corpora, the majority (48 out of 56) consisted of bitmap graphics, followed by vector graphics (10 out of 56), while programs were less prevalent, comprising only five instances (some works included multiple formats).

Among many program formats, we are especially interested in Vega-Lite, which is an abstract specification that enables the creation of interactive visualizations using a high-level grammar. It is represented as a nested JSON object, consisting of numerous key-value

---

*e-mail: hyungkwonko@gmail.com
†e-mail: hj@hcil.snu.ac.kr
‡e-mail: gmpark@hcil.snu.ac.kr
§e-mail: dhkim16@cs.stanford.edu
¶e-mail: nam.wook.kim@bc.edu
‖e-mail: juhokim@kaist.ac.kr
**e-mail: jseo@snu.ac.kr

pairs, which can be also seen as a tree structure (Fig. 1) [16, 28]. Each key defined in the specification is referred to as a property [26], serving a distinct role in generating charts. For example, `mark` property is used to map data to graphical marks.

Vega-Lite provides additional advantages beyond those offered by SVG formats, by facilitating easy modification and reusability and enabling the creation of diverse chart variations [11]. It provides interactive features like zooming, panning, and brushing, as well as concatenating or faceting multiple plots/views. Furthermore, it supports data-driven manipulation, allowing users to dynamically update the data and reflect changes in real time. It can be seamlessly converted to other formats like bitmaps and SVG [19], while converting from these formats to program specifications typically requires manual effort or complex external algorithms [17].

There exist three synthetic Vega-Lite datasets. A critical limitation of these artificial datasets is their reliance on pre-defined templates and rules, leading to a high degree of repetition and a limited range of chart types and functionalities (see Table 1). In detail, Poco et al. generated 4,318 Vega specifications [21] using the Compass recommendation engine [26]. They randomly selected values for a few variables (e.g., fonts, font size, legend positions, etc.) from a curated set of options. These specifications were later converted to Vega-Lite specifications in Data2Vis [9]. Zhao et al. [28] followed a similar approach to generate the Chartseer dataset, consisting of 9,925 specifications based on Data2Vis, although it is specifically designed for training a deep learning model and may not readily render into charts, making it less suitable for broader research adoption. The nvBench dataset [16] presented 7,274 specifications, representing SQL queries as tree structures and mapping them into Vega-Lite specifications.

We find two datasets that consist of human-generated real-world specifications. These datasets reveal significant variation from one specification to another, ensuring a high level of diversity in realistic scenarios. For instance, Kim et al. [13] curated 52 charts from various web sources, encompassing two chart types (bar chart and line chart). Additionally, the Vega-Lite gallery example dataset [20], the largest publicly available human-generated collection of Vega-Lite data, provides 716 high-quality examples with diverse chart types and interactions. However, due to the challenges associated with data collection, these datasets have a limited number of specifications compared to synthesized datasets. As a result, researchers often face difficulties in finding a comprehensive set of specifications for their own research purposes.

Some NL datasets are also available for V-NLIs research, including those introduced by Kim et al. and Lui et al. for chart question answering and NL to visualization generation tasks, respectively. Moreover, Quda [10] offers a dataset of 14K NL queries for various visual data analytics tasks. Srinivasan et al. [23] recently presented a dataset of 893 user utterances for natural language to visualization generation. However, these NL datasets are often task-specific and fragmented, posing challenges for researchers seeking to apply them to their own tasks. The characteristics of NL queries designed for each task can vary significantly, making a single NL dataset unsuitable for other tasks. This further reduces the availability of datasets suitable for chart generation tasks. In our preliminary research, we aim to address this limitation by utilizing LLMs to generate task-specific NL queries for chart generation.

3 VEGA-LITE DATASET

3.1 Dataset Construction

Search Queries. We utilize the GitHub API[1] to create our Vega-Lite dataset. Due to the API's limitation of providing up to 1,000 results per search query, we employ various techniques, as we elaborate below, to crawl Vega-Lite specifications in a mutually exclusive and exhaustive manner to the best of our abilities.

Table 1: **A summary of the Vega-Lite dataset construction process.** (a) First we collect all possible cases of URLs including Vega-Lite specifications. (b, c) Next, we have filtered unique URLs that are allowed to redistribute for academic purpose. (d) Finally we iteratively inspect each specification manually to check whether it is valid and unique. Since we want to collect charts with a high level of diversity.

	# of URLs / Vega-Lite specs
(a) URLs crawled	67,789
(b) URLs w/o duplicate	18,420
(c) URLs w/ license	7,408
(d) Specs after manual inspection	1,981

When building search queries, we use the keyword `https://vega.github.io/schema/vega-lite/[version]` to indicate the version of the spec that Vega-Lite uses for rendering purposes. We collect versions from v2 to v5: there are no v1 data to be found. To partition the query into a more fine-grained manner, we use keywords such as `.csv` and `.json` to gather specs with external links. Similarly, we employ keywords like `values` and `datasets` to identify ones with internally embedded data. We also leverage additional keywords using the main properties defined in the version 5 Vega-Lite specification[2]. These properties encompass essential elements for creating a single plot, including `data`, `transform`, `mark`, and `encoding`, while there are properties like `layer`, `facet`, `concat`, and `repeat`, which are specifically relevant to visualizing *composite* views [18] (e.g., layered plots, trellis plots, or multiple views). A comprehensive list of the properties we use can be found on the official documentation page[3].

Inclusion and Exclusion Criteria. We target files with extension `.json`, `vg.json`, `.vl.json`, `.vl`, and `.vg` which denotes Vega-Lite specifications. We also examine HTML and JavaScript files containing Vega-Lite specifications manually to get additional specs. Throughout the process, we exclude forked repositories to prevent redundancy. We also filter out any data from the benchmark datasets, such as Vega-Lite gallery [20].

Post-processing. To obtain a large number of unique sets of Vega-Lite specifications, we follow a step-by-step approach. During the initial stage, a total of 67,789 URLs are collected. Despite efforts to ensure a mutually exclusive and comprehensive set of specs, duplicate URLs are identified and removed, resulting in 18,420 unique URLs. Each URL is scrutinized to verify the license of the corresponding repository, ensuring compliance with copyright regulations for academic redistribution. This process yields 7,408 URLs. Lastly, we verify their validity using the Vega-Lite editor [19]. This involves identifying the URLs of the datasets used by each specification and making necessary modifications, ranging from minor adjustments such as closing unclosed brackets to more significant ones like debugging the entire code, in order to achieve successful rendering. An overview of the post-processing and the number of URLs and specifications obtained at each stage can be found in Table 1. Our dataset is publicly accessible via the following link: `https://hyungkwonko.info/chart-llm-data`.

3.2 Quantitative Analysis

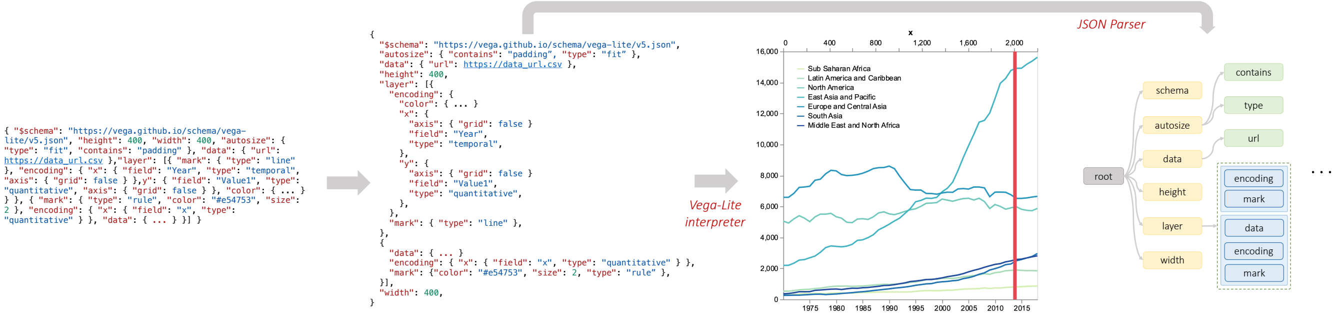
Benchmark Datasets. We compare three synthetic and two real-world Vega-Lite datasets [9, 13, 16, 20, 28] described in Section 2. While checking these datasets, we discover instances of exact code

---

[1] https://docs.github.com/en/rest

[2] https://github.com/vega/schema

[3] https://vega.github.io/vega-lite/docs



```
{ "$schema": "https://vega.github.io/schema/vega-
lite/v5.json", "height": 400, "width": 400, "autosize": {
"type": "fit", "contains": "padding" }, "data": { "url":
https://data_url.csv },"layer": [{ "mark": { "type": "line"
}, "encoding": { "x": { "field": "Year", "type": "temporal",
"axis": { "grid": false } },y": { "field": "Value1", "type":
"quantitative", "axis": { "grid": false } }, "color": { ... }
} }, { "mark": { "type": "rule", "color": "#e54753", "size":
2 }, "encoding": { "x": { "field": "x", "type":
"quantitative" } }, "data": { ... } }] }
```

(a) Raw Vega-Lite Code Data

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
  "autosize": { "contains": "padding”, "type": "fit” },
  "data": { "url": https://data_url.csv },
  "height": 400,
  "layer": [{
    "encoding": {
      "color": { ... }
      "x": {
        "axis": { "grid": false }
        "field": "Year",
        "type": "temporal",
      },
      "y": {
        "axis": { "grid": false }
        "field": "Value1",
        "type": "quantitative",
      },
    },
    "mark": { "type": "line" },
  },
  {
    "data": { ... }
    "encoding": { "x": { "field": "x", "type": "quantitative" } },
    "mark": {"color": "#e54753", "size": 2, "type": "rule” },
  }],
  "width": 400,
}
```

(b) After sorting keys and setting indentation

(c) Generated Vega-Lite chart

(d) Tree Structure

Figure 1: **Example of Vega-Lite Specification.** As previously noted in several works, Vega-Lite spec can be regarded to follow a tree structure. We evaluate the diversity of the specs, e.g., using the number of keys and the depth of JSON structure.

Table 2: **Summary statistics of our dataset and benchmark datasets that are publicly available.** Two types of datasets are presented: synthetic and real-world datasets. The best statistics within each type are highlighted in bold, while the best statistics across all datasets are also underscored.

Type	Evaluation Metric	Synthetic data (machine-generated)			Real-world data (human-generated)		
		Data2Vis [9]	Chartseer [28]	nvBench [16]	Kim et al. [13]	Gallery [20]	Ours
Quantity	# of specs	4,318	**9,897**	6,680	52	709	**1,981**
Complexity	Total # of keys across specs	101,881	**147,676**	98,074	769	26,469	**107,802**
	Average # of keys in a spec	**24**	15	15	15	37	**54**
	Simple (key ≤ 16)	0	6,164	**6,354**	41	**186**	73
	Medium (key ≤ 24)	**4,318**	3,733	326	10	170	**199**
	Complex (key ≤ 41)	0	0	0	1	179	**733**
	Extra complex (key > 41)	0	0	0	0	174	**976**
	Average depth of JSON	**4.00**	3.00	3.48	3.13	5.01	**5.19**
	Average branching factor	1.22	**1.44**	1.18	1.17	**1.41**	1.38
Diversity	Total # of unique keys	**24**	12	18	31	275	**362**
	Average pairwise edit distance	**122.62**	75.90	48.18	129.51	1,096.11	**1,549.51**
	Composite views	0	0	0	0	136	**746**
	Interaction (e.g., zoom, pan)	0	0	0	0	188	**1,010**
	# of chart types	**6**	**6**	4	2	**10**	**10**

duplication within each benchmark. To ensure a fair comparison, we implement a process to remove such redundancy. In detail, each spec is sorted in alphabetical order by the keys and edited to maintain consistent indentation. Next, we convert each file into a hash where files with identical hashes are subsequently removed from the dataset. Following this procedure, the number of specs in Chartseer dataset decrease from 9,917 to 9,897, nvBench decrease from 7,241 to 6,680, and the Vega-Lite gallery example dataset decrease from 716 to 709.

Quality Metrics. To comprehensively assess the Vega-Lite datasets, we consider three different aspects: quantity, complexity, and diversity. Initially, we count the number of collected specifications to determine the overall quantity of Vega-Lite specs, as previously done by Luo et al. [16]. However, we argue that additional metrics are necessary to gauge the quality of the Vega-Lite dataset. This is because some specs include only mandatory properties to construct a single plot without any interaction (e.g., `data`, `encoding`, `mark` for a simple bar chart), while others contain multiple plots or views linked by varying interactions. Therefore, the number of keys in a spec can highly differ depending on whether it includes properties for data pre-processing (e.g., `aggregate`, `calculate`, etc.), interactivity (e.g., `bind`, `select`, etc.), or composite views (e.g., `concat`, `repeat`, etc.). We can expect the Vega-Lite spec becomes more complex as the number of defined properties increases. Therefore, we propose a new standard to understand the complexity of a Vega-Lite dataset by counting the total number of keys present across all specifications and the average number of keys in a singe spec. To ensure a fair comparison, we only consider keys defined in the version 5 specification. We also ignore keys associated with internally embedded datasets, such as `values` and `datasets`, along with their corresponding keys. In addition to this, we also measure the average depth and branching factor of the JSON structure as they are commonly adopted to evaluate the complexity of a JSON file.

We also propose metrics for gaining insights into the diversity of dataset in terms of both the range of properties within the entire dataset and the variance between individual specs. Specifically, we count the number of unique keys employed across the entire dataset and calculate the average pairwise edit distance among all possible pairs of specs. The number of unique keys indicates how many distinct properties that can be defined in a Vega-Lite spec are used across the specs. For example, if a handful of unique keys are used within the dataset, this indicates a restricted recurrence of only a few properties. In turn, it likely signifies a low level of diversity. The average pairwise edit distance provides an overview of the dissimilarity between each pair at the code level. To perform this analysis, we sort the keys alphabetically, replace their corresponding values with empty values, and exclude keys associated with embedded datasets,

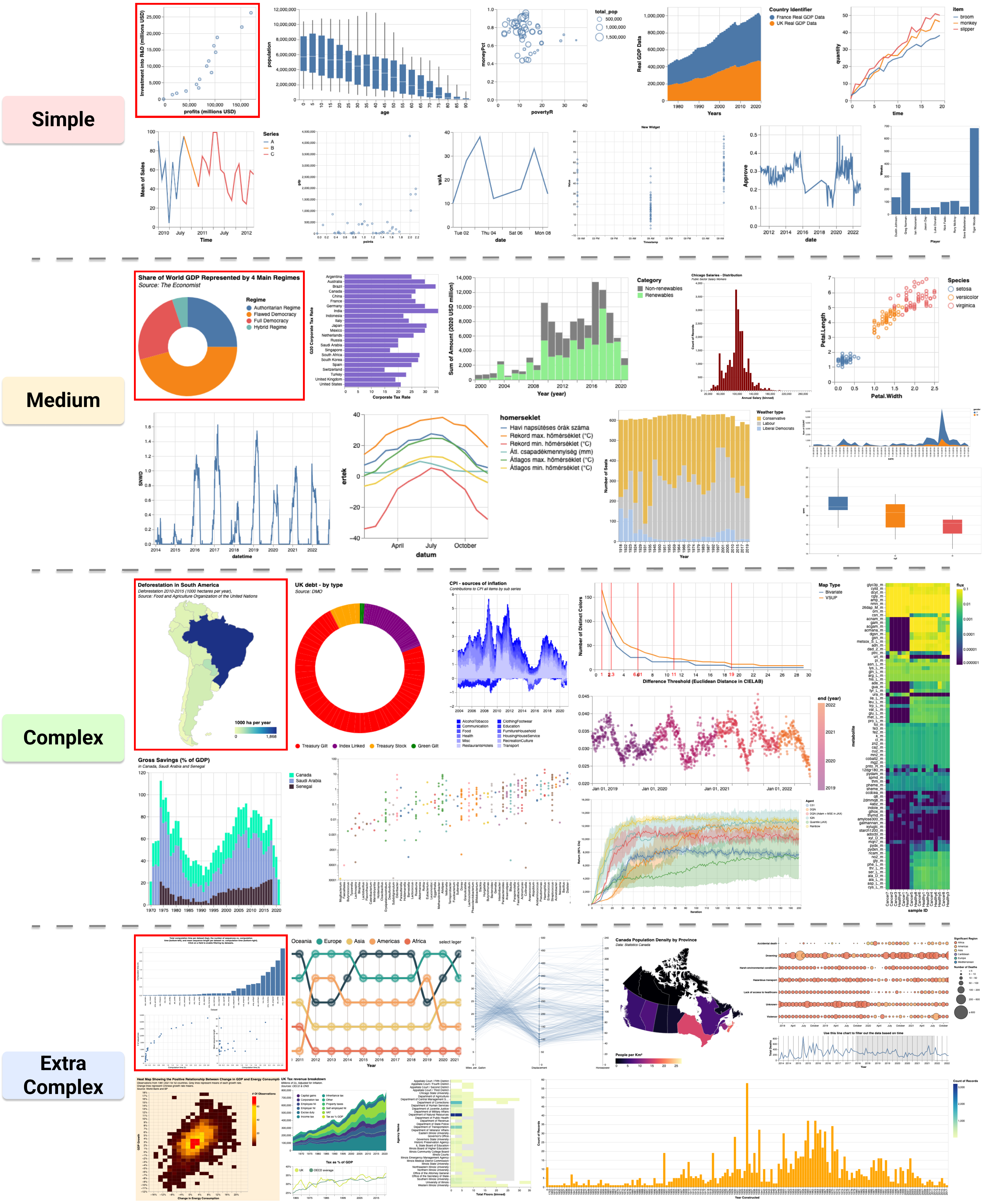


Figure 2: **Vega-Lite dataset divided by their complexity levels: simple, medium, complex, extra complex.** The level is divided based on the number of keys each spec contains. The number of keys, which are the criteria for dividing the levels, are set based on the quartiles (Q1, Q2, Q3) of Vega-Lite example gallery dataset [20]. The charts with red stroke are used for the NL generation pipeline in Section 4.

as mentioned earlier.

Complexity Levels. We observe that the existing criteria used to establish the complexity levels of chart datasets are somewhat subjective and may not possess broad applicability [12, 15, 16]. Instead, we suggest using the number of keys as a criterion for categorizing the complexity levels of charts, particularly in the context of Vega-Lite specifications. This is because, as explained above, the number of properties increases proportionately to the number of keys in a specification. To establish the standard number of keys, we refer to the Vega-Lite example gallery dataset [20] and calculate the quartiles (Q1, Q2, Q3) based on the distribution of the number of keys. These quartiles, specifically 16, 24, and 41, are utilized as reference points to divide the specs' level of complexity. For instance, a specification with a total number of keys less than or equal to 16 is classified as 'simple' complexity. Likewise, a specification with a total number of keys greater than 16 and less than or equal to 24 is classified as 'medium' complexity (Fig. 2).

Composite View, Interactivity, Chart Type Distribution. We choose three additional factors by referring to previous works [2, 14] to further assess the quality of the datasets. First, we examine the presence of composite views, which offer diverse perspectives on the same data simultaneously [4]. Secondly, considering the benefits of collecting Vega-Lite specs over static bitmap images, we count the number of charts that incorporate interactive techniques such as tooltips, zooming, and brushing. Lastly, we evaluate the number of charts types based on the taxonomy proposed by Borkin et al. [2].

Results. We present the results in terms of quantity, complexity, and diversity, highlighting the superiority of our dataset compared to the baselines. Regarding quantity, all three synthetic datasets demonstrate a higher number of specs compared to the other three real-world datasets. Among all datasets, Chartseer shows the highest number of specs (i.e., 9,897), while our dataset has 1,981 specs which outnumbers the other real-world datasets in terms of quantity.

In terms of complexity, our dataset exhibits the largest average number of keys in a single specification (i.e., 54) and ranks second in terms of the total number of keys across specs (i.e., 107,802), which is 1.4 and 4.0 times larger than the largest previous real-world Vega-Lite dataset, respectively. Chartseer presents the highest total number of keys across specs (i.e., 147,676) with the smallest average number of keys per specification (i.e., 15) among all datasets. Our dataset includes the highest number of specs classified as complex (i.e., 733) and extra complex (i.e., 976). The synthetic datasets do not contain any specs in the complex and extra complex level. Data2Vis and nvBench demonstrate the largest number of specs classified as medium (i.e., 4,318) and easy (i.e., 6,354), respectively. Our dataset also exhibits the highest average depth of JSON structure (i.e., 5.19), while Chartseer showcases the highest average branching factor (i.e., 1.44).

Lastly, with respect to diversity, our dataset demonstrates the largest total number of unique keys and the highest average pairwise edit distance among all datasets. Furthermore, our dataset includes the largest number of specs featuring composite views (i.e., 1,010) and interactions (i.e., 746), exceeding the Vega-Lite gallery dataset by 1.8 and 5.3 times, respectively. None of the synthetic datasets or Kim et al.'s dataset include specs with composite views and interactions. Both our dataset and the Vega-Lite gallery dataset cover the widest variety of chart types, encompassing ten types: Area, Bar, Circle, Diagram, Distribution, Grid & Matrix, Line, Map, Point, and Trees & Networks. Please refer to Table 2 for detailed results.

4 NATURAL LANGUAGE GENERATION PIPELINE

In V-NLI research, natural language inputs such as queries and statements are necessary in addition to the visualization dataset. Rather than relying on costly human-generated NL data, we attempt to leverage LLMs to generate this NL dataset. Our primary focus is a task to generate charts based on users' utterances. While we illustrate this task as an example in our preliminary research, our future goal is to broaden the scope of our pipeline to encompass a range of tasks such as chart understanding and chart question answering, within the field of V-NLIs research. In this section, we introduce how to write an input prompt for chart generation, analyze the generated NL queries, and discuss the implications of the results.

Prompt Design. To design the input prompt for generating NL queries in visualization generation, we begin by informing the LLMs about the task. Furthermore, to better simulate user utterances, we draw upon a recent paper by Wang et al. [25] that addresses this task by providing design practices for NL-based visualization authoring tools. They suggest that 1) users generate charts in a top-down manner, starting with basic elements such as encoding and chart type before choosing design elements, 2) users expect real-time reflection of their inquiries from the authoring tool, and 3) users use different data labels (e.g., pronouns, adjectives) to denote visual properties, which may vary among individuals. Based on these guidelines, we give the following prompt to the LLMs: *"We have an intelligent system that translates a novice user's NL query into a corresponding visualization. Your task is to generate the input from the novice user that would prompt the machine to produce the chart based on the provided information. Follow a top-down approach when generating charts, considering each step as a distinct inquiry of a sentence, starting with fundamental elements such as encoding and chart type before incorporating design elements; Ensure the NL queries reflect real-time updates and provide immediate feedback to users' inquiries through the authoring tool; Utilize diverse data labels, such as pronouns and adjectives, to convey specific visual properties, keeping in mind that these labels may vary among different individuals. Use a colloquial language."*.

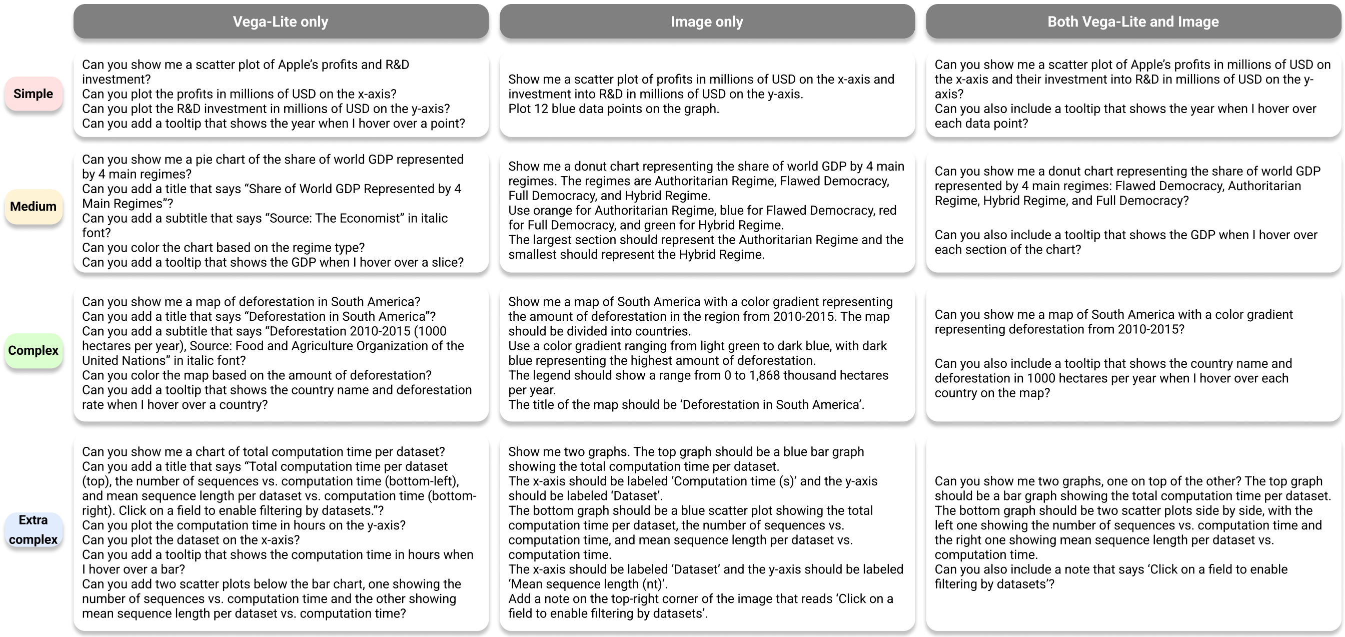
NL Generation Process. For our experiment, we select four sample charts representing different complexity levels, ranging from simple to extra complex (see the top left charts in each level with red stroke in Fig. 2). The purpose of this experiment is to compare the results across different modalities and information. To facilitate this comparison, we utilize Bing Chat[4], which allows for both text and image inputs in its chatting interface. We begin by providing the task description and three detailed guidelines to the Bing Chat for generating NL queries. Subsequently, we generate NL queries under three different settings: 1) using only chart images, 2) using only Vega-Lite spec, and 3) using both chart images and their corresponding Vega-Lite spec.

Results and Discussion. The total generated NL queries using three different types of input are listed in Fig. 3. The generated NL queries follow the instruction well, which is to draw a chart in a top-down manner. For example, in the case of complex level chart, generated NL queries using only Vega-Lite code follow 1) setting the chart type (i.e., Can you show me a map of deforestation in South America?), 2) adding styles (i.e., Can you add a subtitle that says "Deforestation 2010-2015 (1,000 hectares per year), Source: Food and Agriculture Organization of the United Nations" in italic font?), 3) adding colors (i.e., Can you color the map based on the amount of deforestation?), and 4) adding a tooltip (Can you add a tooltip that shows the country name and deforestation rate when I hover over a country?). The generated queries using other options like using only image and both Vega-Lite and image present a similar approach (e.g., chart type → color map → legend → title, chart type → tooltip).

Although some inquiries are merged into one, generated NL queries using only Vega-Lite spec and using both the image and its corresponding Vega-Lite code are similar throughout all four different levels. Specifically, while the generated NL using only Vega-Lite code results in three sequential inquiries–"Can you show me a scatter plot of Apple's profits and R&D investment?," "Can you plot the profits in millions of USD on the x-axis?," and "Can you

---

[4] https://www.bing.com



	Vega-Lite only	Image only	Both Vega-Lite and Image
Simple	Can you show me a scatter plot of Apple's profits and R&D investment? Can you plot the profits in millions of USD on the x-axis? Can you plot the R&D investment in millions of USD on the y-axis? Can you add a tooltip that shows the year when I hover over a point?	Show me a scatter plot of profits in millions of USD on the x-axis and investment into R&D in millions of USD on the y-axis. Plot 12 blue data points on the graph.	Can you show me a scatter plot of Apple's profits in millions of USD on the x-axis and their investment into R&D in millions of USD on the y-axis? Can you also include a tooltip that shows the year when I hover over each data point?
Medium	Can you show me a pie chart of the share of world GDP represented by 4 main regimes? Can you add a title that says "Share of World GDP Represented by 4 Main Regimes"? Can you add a subtitle that says "Source: The Economist" in italic font? Can you color the chart based on the regime type? Can you add a tooltip that shows the GDP when I hover over a slice?	Show me a donut chart representing the share of world GDP by 4 main regimes. The regimes are Authoritarian Regime, Flawed Democracy, Full Democracy, and Hybrid Regime. Use orange for Authoritarian Regime, blue for Flawed Democracy, red for Full Democracy, and green for Hybrid Regime. The largest section should represent the Authoritarian Regime and the smallest should represent the Hybrid Regime.	Can you show me a donut chart representing the share of world GDP represented by 4 main regimes: Flawed Democracy, Authoritarian Regime, Hybrid Regime, and Full Democracy? Can you also include a tooltip that shows the GDP when I hover over each section of the chart?
Complex	Can you show me a map of deforestation in South America? Can you add a title that says "Deforestation in South America"? Can you add a subtitle that says "Deforestation 2010-2015 (1000 hectares per year), Source: Food and Agriculture Organization of the United Nations" in italic font? Can you color the map based on the amount of deforestation? Can you add a tooltip that shows the country name and deforestation rate when I hover over a country?	Show me a map of South America with a color gradient representing the amount of deforestation in the region from 2010-2015. The map should be divided into countries. Use a color gradient ranging from light green to dark blue, with dark blue representing the highest amount of deforestation. The legend should show a range from 0 to 1,868 thousand hectares per year. The title of the map should be 'Deforestation in South America'.	Can you show me a map of South America with a color gradient representing deforestation from 2010-2015? Can you also include a tooltip that shows the country name and deforestation in 1000 hectares per year when I hover over each country on the map?
Extra complex	Can you show me a chart of total computation time per dataset? Can you add a title that says "Total computation time per dataset (top), the number of sequences vs. computation time (bottom-left), and mean sequence length per dataset vs. computation time (bottom-right). Click on a field to enable filtering by datasets."? Can you plot the computation time in hours on the y-axis? Can you plot the dataset on the x-axis? Can you add a tooltip that shows the computation time in hours when I hover over a bar? Can you add two scatter plots below the bar chart, one showing the number of sequences vs. computation time and the other showing mean sequence length per dataset vs. computation time?	Show me two graphs. The top graph should be a blue bar graph showing the total computation time per dataset. The x-axis should be labeled 'Computation time (s)' and the y-axis should be labeled 'Dataset'. The bottom graph should be a blue scatter plot showing the total computation time per dataset, the number of sequences vs. computation time, and mean sequence length per dataset vs. computation time. The x-axis should be labeled 'Dataset' and the y-axis should be labeled 'Mean sequence length (nt)'. Add a note on the top-right corner of the image that reads 'Click on a field to enable filtering by datasets'.	Can you show me two graphs, one on top of the other? The top graph should be a bar graph showing the total computation time per dataset. The bottom graph should be two scatter plots side by side, with the left one showing the number of sequences vs. computation time and the right one showing mean sequence length per dataset vs. computation time. Can you also include a note that says 'Click on a field to enable filtering by datasets'?

Figure 3: **The generated NL queries using three types of input.** Four samples charts in each level are used for the task.

plot the R&D investment in millions of USD on the y-axis?"–this is represented as a single inquiry in the NL generated using both image and Vega-Lite code–"Can you show me a scatter plot of Apple's profits in millions of USD on the x-axis and their investment into R&D in millions of USD on the y-axis?".

In case of the generated NL using only image input, the results has a tendency to present information based on the portion of area filled with a specific color, or a text that can be found in the image, which includes inaccurate information. For instance, the generated NL using only image for the medium level chart is "Use orange for Authoritarian Regime and blue for Flawed Democracy", where they are the opposite in reality. Moreover, the largest section represent the Authoritarian Regime, but it is the Flawed Democracy.

5 ONGOING WORK

To develop a generalized natural language generation pipeline for V-NLIs research, we are conducting a thorough analysis of the task space within this field. This involves examining existing research through a recent survey paper [22] and reviewing papers from key conferences in Human-Computer Interaction, Visualization, and Data Mining and Management. Using search terms like "chart" and "natural language," we will filter papers that specifically address these topics. The identified tasks will be categorized and clustered based on their characteristics, allowing us to select representative tasks of the given space. We will report whether LLMs can generate high-quality natural language queries for these selected tasks, comparing the generated queries quantitatively and qualitatively against a gold standard NL dataset created by visualization researchers. We will also discuss additional use cases for our dataset.

6 CONCLUSION

In this work, we present a dataset which comprises 1,981 Vega-Lite specifications, which is better in terms of complexity and diversity with more interactions, multiple-view visualization, and different chart types. We also propose a natural language generation pipeline so that future researchers can generate natural language queries that can suit any target tasks they want to delve into for doing NLI for data visualization research. As a future work, we plan to set diverse target tasks that can cover larger spaces of natural language queries. We hope our contributions can be helpful for building future systems using V-NLIs.

REFERENCES

[1] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pp. 313–322, 2010.

[2] M. A. Borkin, A. A. Vo, Z. Bylinskii, P. Isola, S. Sunkavalli, A. Oliva, and H. Pfister. What makes a visualization memorable? *IEEE transactions on visualization and computer graphics*, 19(12):2306–2315, 2013.

[3] C. Chen and Z. Liu. The State of the Art in Creating Visualization Corpora for Automated Chart Analysis. *Computer Graphics Forum*, 2023. doi: 10.1111/cgf.14855

[4] X. Chen, W. Zeng, Y. Lin, H. M. Ai-Maneea, J. Roberts, and R. Chang. Composition and configuration patterns in multiple-view visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1514–1524, 2020.

[5] K. Cox, R. E. Grinter, S. L. Hibino, L. J. Jagadeesan, and D. Mantilla. A multi-modal natural language interface to an information visualization environment. *International Journal of Speech Technology*, 4:297–314, 2001.

[6] K. Davila, S. Setlur, D. Doermann, B. U. Kota, and V. Govindaraju. Chart mining: A survey of methods for automated chart analysis. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3799–3819, 2020.

[7] B. Deka, Z. Huang, C. Franzen, J. Hibschman, D. Afergan, Y. Li, J. Nichols, and R. Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pp. 845–854, 2017.

[8] D. Deng, Y. Wu, X. Shu, J. Wu, M. Xu, S. Fu, W. Cui, and Y. Wu. Visimages: a corpus of visualizations in the images of visualization publications. *arXiv preprint arXiv:2007.04584*, 2020.

[9] V. Dibia and Ç. Demiralp. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE computer graphics and applications*, 39(5):33–46, 2019.

[10] S. Fu, K. Xiong, X. Ge, S. Tang, W. Chen, and Y. Wu. Quda: natural language queries for visual data analytics. *arXiv preprint arXiv:2005.03257*, 2020.

[11] J. Harper and M. Agrawala. Converting basic d3 charts into reusable style templates. *IEEE transactions on visualization and computer graphics*, 24(3):1274–1286, 2017.

[12] S. Kantharaj, R. T. K. Leong, X. Lin, A. Masry, M. Thakkar, E. Hoque, and S. Joty. Chart-to-text: A large-scale benchmark for chart summarization. *arXiv preprint arXiv:2203.06486*, 2022.

[13] D. H. Kim, E. Hoque, and M. Agrawala. Answering questions about charts and generating visual explanations. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pp. 1–13, 2020.

[14] Y. Li, Y. Qi, Y. Shi, Q. Chen, N. Cao, and S. Chen. Diverse interaction recommendation for public users exploring multi-view visualization using deep learning. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):95–105, 2022.

[15] A. Lundgard and A. Satyanarayan. Accessible visualization via natural language descriptions: A four-level model of semantic content. *IEEE transactions on visualization and computer graphics*, 28(1):1073–1083, 2021.

[16] Y. Luo, N. Tang, G. Li, C. Chai, W. Li, and X. Qin. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 1235–1247, 2021.

[17] J. Poco and J. Heer. Reverse-engineering visualizations: Recovering visual encodings from chart images. In *Computer graphics forum*, vol. 36, pp. 353–363. Wiley Online Library, 2017.

[18] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization & Computer Graphics (Proc. InfoVis)*, 2017. doi: 10.1109/tvcg.2016.2599030

[19] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega editor. Retrieved July 6 from https://vega.github.io/editor, June 2023.

[20] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite gallery. Retrieved July 6 from https://github.com/vega/vega-lite, June 2023.

[21] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE transactions on visualization and computer graphics*, 22(1):659–668, 2015.

[22] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang. Towards natural language interfaces for data visualization: A survey. *IEEE transactions on visualization and computer graphics*, 2022.

[23] A. Srinivasan, N. Nyapathy, B. Lee, S. M. Drucker, and J. Stasko. Collecting and characterizing natural language utterances for specifying data visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–10, 2021.

[24] A. Srinivasan and J. Stasko. Natural language interfaces for data analysis with visualization: Considering what has and could be asked. In *Proceedings of the Eurographics/IEEE VGTC conference on visualization: Short papers*, pp. 55–59, 2017.

[25] Y. Wang, Z. Hou, L. Shen, T. Wu, J. Wang, H. Huang, H. Zhang, and D. Zhang. Towards natural language-based visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1222–1232, 2022.

[26] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Towards a general-purpose query language for visualization recommendation. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pp. 1–6, 2016.

[27] J. Wu, S. Wang, S. Shen, Y.-H. Peng, J. Nichols, and J. P. Bigham. Webui: A dataset for enhancing visual ui understanding with web semantics. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2023.

[28] J. Zhao, M. Fan, and M. Feng. Chartseer: Interactive steering exploratory visual analysis with machine intelligence. *IEEE Transactions on Visualization and Computer Graphics*, 28(3):1500–1513, 2020.