

ChartGPT: Leveraging LLMs to Generate Charts from Abstract Natural Language

Yuan Tian, Weiwei Cui, Dazhen Deng, Xinjing Yi, Yurun Yang, Haidong Zhang, Yingcai Wu

Abstract—The use of natural language interfaces (NLIs) to create charts is becoming increasingly popular due to the intuitiveness of natural language interactions. One key challenge in this approach is to accurately capture user intents and transform them to proper chart specifications. This obstructs the wide use of NLI in chart generation, as users' natural language inputs are generally abstract (i.e., ambiguous or under-specified), without a clear specification of visual encodings. Recently, pre-trained large language models (LLMs) have exhibited superior performance in understanding and generating natural language, demonstrating great potential for downstream tasks. Inspired by this major trend, we propose ChartGPT, generating charts from abstract natural language inputs. However, LLMs are struggling to address complex logic problems. To enable the model to accurately specify the complex parameters and perform operations in chart generation, we decompose the generation process into a step-by-step reasoning pipeline, so that the model only needs to reason a single and specific sub-task during each run. Moreover, LLMs are pre-trained on general datasets, which might be biased for the task of chart generation. To provide adequate visualization knowledge, we create a dataset consisting of abstract utterances and charts and improve model performance through fine-tuning. We further design an interactive interface for ChartGPT that allows users to check and modify the intermediate outputs of each step. The effectiveness of the proposed system is evaluated through quantitative evaluations and a user study.

Index Terms—Natural language interfaces, large language models, data visualization.

I. INTRODUCTION

Natural language interfaces (NLIs) have become a popular interactive strategy for data analysis and visualization creation [1], [2]. For example, a user can easily create a histogram showing the distribution of IMDB ratings for a movie dataset by simply saying “create a histogram showing the distribution of IMDB ratings.” Compared to traditional methods, NLIs provide a shortcut for analysts not proficient in visualization programming, such as D3.js or Vega-Lite [3], to create visualizations. Even for senior visualization users, NLIs can free them from tedious programming issues or interactive editings on visualization toolkits (e.g., Tableau [4]).

The key of NLIs is to precisely capture user intents and generate appropriate visualizations under the ambiguity and underspecification of natural languages. While experts in visual analytics are capable of specifying all necessary information for visualization generation in one utterance, including

Y. Tian, D. Deng, X. Yi, Y. Yang, and Y. Wu are with Zhejiang University. E-mail: {yuantian, dengdazhen, yixinjing, yurunyang, ycwu}@zju.edu.cn. Dazhen Deng and Yingcai Wu are the corresponding authors.

W. Cui and H. Zhang are with Microsoft. E-mail: {weiwei.cui, haidong.zhang}@microsoft.com.

Manuscript received xxx, xx, 2024; revised xxx, xx, 2024.

data fields, data transformations, chart types, and visual encodings, beginners in visualization programming may struggle to provide all the information. Demonstrated by previous studies [5], [6], user queries are underspecified in many cases. For instance, the utterance “What type of movies make the most money?” implicitly refers to the field of “gross profit.” The term “type” can be understood differently (e.g., genre, rating, etc.) in various contexts. Such ambiguity makes it hard to map utterances to concrete chart specifications. Traditional methods combine lexical parsing and predefined rules to support abstract inference to some extent [6]. For example, NL4DV [1] facilitates attribute inference from computing the similarity with data fields, values, and defined aliases and enables task and chart type inference through predefined rules. However, such methods are limited by the ability of parsers to understand natural language. In addition, the predefined aliases and rules can be hard to maintain, modify, and expand [7].

Recently, large language models (LLMs), such as Bert [8], GPT-3 [9] and ChatGPT [10], have demonstrated outstanding performance in natural language understanding. These models, pre-trained on a massive corpus of text, have acquired a vast amount of knowledge and can be utilized for various downstream tasks [11], such as data transformation [12], narration generation [13], [14], and web design [15]. The remarkable success of these LLM applications inspires us to investigate their potential for visualization generation. However, using LLMs to generate visualizations from abstract utterances presents two main challenges.

Controlling chart parameters with LLMs. The process of visualization generation involves complex parameters and operations. Users have to specify parameters such as mark, field, encoding, and aggregation, which are then rendered by visualization systems (e.g., Vega-Lite and Tableau) to transform the original data table and produce the chart. While language models (LLMs) can generate fluent and informative answers to human questions, they may not always be accurate, which is well-known as the “hallucination problem” [16]. This makes it challenging to use LLMs directly in visualization generation, as a single incorrect parameter could negatively impact the subsequent operations and potentially compromise the entire process. To tackle this challenge, we adopt a systematic approach by breaking down the chart generation process into a series of interrelated sub-tasks, following the principle of least-to-most idea [17]. This decomposition allows us to leverage the strengths of LLMs to produce well-defined and manageable outputs for complex parameters and operations involved in chart creation.

Lacking approaches to inject visualization knowledge.

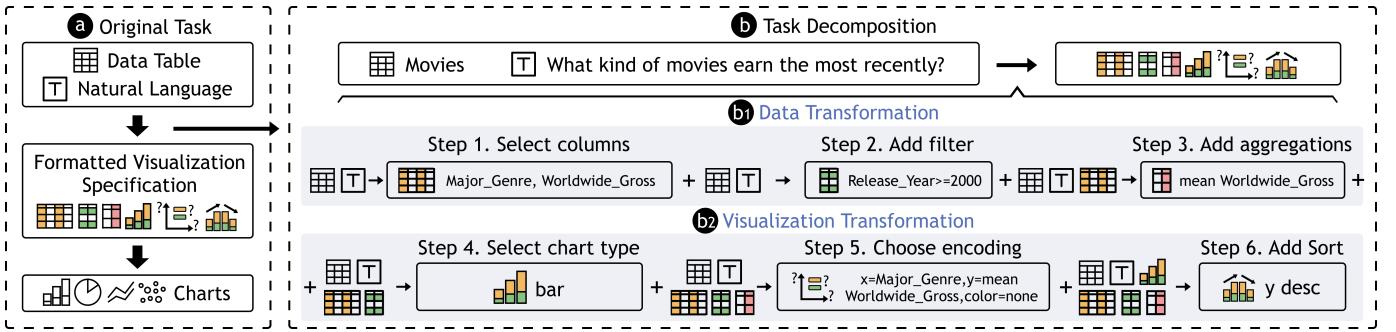


Fig. 1. An example of chart generation problem formulation. (a) The task comprises three stages: input context (data table and natural language), formatted visualization specification, and charts. (b) We decompose the first stage transformation process into two successive transformations: data transformation (b1) and visualization transformation (b2), involving six steps. At each step, the model utilizes the input context and previous answers to generate the next output.

LLMs are designed and trained to handle general language-related tasks, such as text generation, recognition, and summarization. To make LLMs more domain-specific, two methods are commonly used: prompting and fine-tuning. Prompting refers to providing the model with a text that includes the context of domain tasks and expected outputs. Although effective, this approach is not always practical, especially when the model needs to be provided with a large amount of knowledge (e.g., Draco rules in our scenario) in a single prompt. Fine-tuning the LLM with appropriate datasets can provide more examples and knowledge. While there are well-established datasets in NL2VIS [18], [19], these datasets mainly consist of explicit natural language descriptions or cover limited datasets, which are not suitable for our scenario. To address this challenge, we constructed a dataset of abstract utterances with corresponding charts. The dataset enables LLMs to learn user intents in visual data analysis and generate chart configurations with the desired formats.

In this study, we introduce ChartGPT, leveraging LLMs to generate charts from abstract utterances. We broke down the chart generation process into a series of sub-tasks for the LLM to solve sequentially and constructed an abstract utterance dataset to fine-tune the model (FLAN-T5-XL [20]). Based on the fine-tuned model, we developed an interactive interface that allows users to explore and modify the intermediate steps of chart generation. We evaluated our proposed method through quantitative experiments and a comparative user study with the state-of-the-art NL2VIS methods. We also summarized the feedback from the usability study and discussed future work for improving the system. The main contributions of this study are as follows.

- We propose a framework to generate charts from abstract utterances using fine-tuned LLMs.
- We construct a dataset of abstract utterances and charts for LLM fine-tuning. The dataset could facilitate future machine learning research in this direction.
- We conduct quantitative experiments and user studies to prove the usefulness of the proposed method. The feedback could shed light on future applications of LLMs in visualizations.

II. RELATED WORK

A. Visualization Recommendation

Recently, there has been a growing interest in exploring visualization recommendation techniques that can assist data workers in tackling the laborious task of creating visualizations [21]–[25]. These techniques are mainly classified into two categories: rule-based and machine learning (ML)-based [7], [26]. Rule-based methods map data to visual encoding according to visualization knowledge, such as the conclusions from empirical studies. A large number of recommendation systems, such as APT [27], Show Me [28], CompassQL [29], and Voyager [30], [31], are compiled from visualization rules. To improve the usability of visualization rules, Moritz et al. [32] translated the rules into answer set programming and formulated a knowledge base. Though effective, rule-based methods might suffer from flexibility, as the manually specified rules by experts are difficult to update, modify, and maintain. This limits their adaptability to diverse data types or changing conditions.

In contrast, ML-based methods have the advantage of being able to learn from data and adapt to changing conditions, making them more flexible and robust [33]–[35]. For example, DeepEye [36] and Draco-learn [32] use machine learning algorithms to rank recommended visualizations based on visualization design rules. Other studies, such as Data2Vis [37] and Table2Charts [38] utilize sequence-to-sequence models to map datasets to visual representations. KG4Vis [39], [40] uses knowledge graphs to support explainability for recommendations. To generate multiple-view visualizations [41], Multi-Vision [42] and Dashbot [43] adopt deep learning methods to model datasets. These studies primarily focus on creating visualizations from data tables. In this work, we take a more challenging approach, comprehending natural language intentions to generate charts.

B. Natural Language Interfaces for Data Visualization

Natural language interfaces are proven efficient in specifying data visualizations [44]–[46]. Many studies utilized semantic or lexical parsing techniques to infer user intent and generate visualizations. Articulate [47] extracted visual tasks and attributes and selected visualizations with a graph reasoner algorithm. DataTone [48] proposed interactive ambiguity

widgets to help users resolve ambiguity in natural language. FlowSense [49] utilized semantic parsers to assist dataflow diagram construction. Users can expand and adjust dataflow diagrams via natural languages. Eviza [50] employed a probabilistic grammar-based approach and allowed an interactive query dialog with an existing visualization. Evizeon [51] further applied language pragmatics principles to support visual analytical conversations. NL4DV [1] incorporated lexical and dependency parsing techniques to infer attributes and tasks from user utterances and generated visualizations. With recent advancements in natural language processing, attempts have been made to utilize deep learning-based language models to produce visualizations. For example, ncNet [2] employed a Transformer-based sequence-to-sequence model to convert natural language queries into visualizations.

However, these studies mainly aim at explicit requests and are difficult to deal with incomplete or implicit utterances. Some studies, such as NL4DV, enable implicit attribute inference by computing the similarity with data fields, values, and defined aliases. Ask Data [6] resolved partial utterances based on syntactic and semantic constraints and produced an intermediate language to generate visualizations. However, The performance of these methods is greatly limited by the capability of the language parsers. Additionally, the predefined aliases and rules may hinder flexibility, as they are hard to maintain, modify, and expand [7]. In this paper, we aim to utilize the language comprehension ability of pre-trained LLMs to tackle the challenge of abstract utterances.

C. Large Language Models for Data Analysis

Recently, there have been significant advancements in large language models (LLMs), such as online models GPT-3 [9] and GPT-4 [52], as well as open-source models flan-T5 [20] and LLaMa [53]. Pre-training on tens of TB of text data, LLMs have demonstrated superior performance in understanding and generating natural language. These models have been applied to various domains, including data transformation [12], narration generation [13], [14], and web design [15].

Specifically, recent studies have explored utilizing LLMs for data analysis. Some studies employ LLMs to generate visualization code (e.g., Python and Vega-Lite) directly. For example, CHAT2VIS [54] generates visualization code in Python by prompting LLMs with table schema, column types, and utterances. Similarly, LIDA [55] defines visualization generation as a four-stage generation problem and leverages GPT-3.5 to generate visualization code. Other studies explored a broader application of LLMs in data analysis. GPT4-Analyst [56] proposes a framework that utilizes prompts to direct GPT-4 in performing data collection, visualization, and analysis. Data-Copilot [57] can generate requests, select the needed interfaces, and invoke the corresponding interface tools sequentially or in parallel. All of these works are based on prompt engineering and depend on online models such as GPT-3.5 and GPT-4, which are not fully controllable and stable [16], [52]. These models might suffer from inherent hallucination problems that occasionally provide unstable output with incorrect answers, leading to failure to follow the designed pipeline.

Different from the above methods that use generic LLMs, we opt to train a visualization-specific LLM to address the problem of chart recommendation. Specifically, we adopt the chain-of-thought [17], [58] idea to decompose the task and then solve it sequentially. Instead of relying on prompt engineering, we fine-tuned an open-source LLM on our constructed abstract utterances dataset. Additionally, we developed a template for the model input and output, enhancing parsing and applicability across various visualization representations, with Vega-Lite serving as an example.

III. BACKGROUND AND PROBLEM FORMULATION

This section introduces the background of reasoning strategies and describes how we formulate the chart generation problem into step-by-step reasoning sub-tasks.

A. Reasoning Strategies in LMs

For language models (LMs), reasoning is defined as the process of breaking down a complex task into simpler sub-tasks for LMs to handle effectively [59]. Specifically, in the least-to-most reasoning strategy [17], the original task is divided into a sequence of sub-tasks, starting with the simplest and gradually increasing in complexity. Through the reasoning process, LMs can solve more complicated sub-tasks with the help of previously solved sub-tasks.

We also adopt a decomposition approach to tackle the chart generation task. We formulate the task as a fixed sequence of sub-tasks and tackle them with an LLM that generates an answer based on the problem context and the outputs from previous sub-tasks. Notably, all sub-tasks are handled by the same model, as LLMs possess the capability to generalize across various tasks. Finally, the answers from all sub-tasks are consolidated to produce a complete chart.

B. Problem Formulation

We formulate our problem based on the Information Visualization Data State Reference Model [60], [61], which outlines the visualization pipeline as a sequence of data stages and explains how data undergoes various transformations from one stage to the next.

As illustrated in Figure 1a, we formulate the problem into three data stages: table data, formatted visualization specification, and charts. Specifically, a formatted visualization specification is a text sequence that satisfies a specific visualization grammar and can be parsed and rendered into a chart. Examples include Vega-Lite [3], Vega-Zero [2], and the chart templates defined in Table2Charts [38]. We also propose a formatted template compatible with our method and pipeline.

Our key challenge is the first transformation of stages: generating visualization specifications from table data based on user utterances. We decompose the process into a series of sub-tasks and formulate each sub-task as a formatted sequence-to-sequence problem, illustrated in Figure 1b.

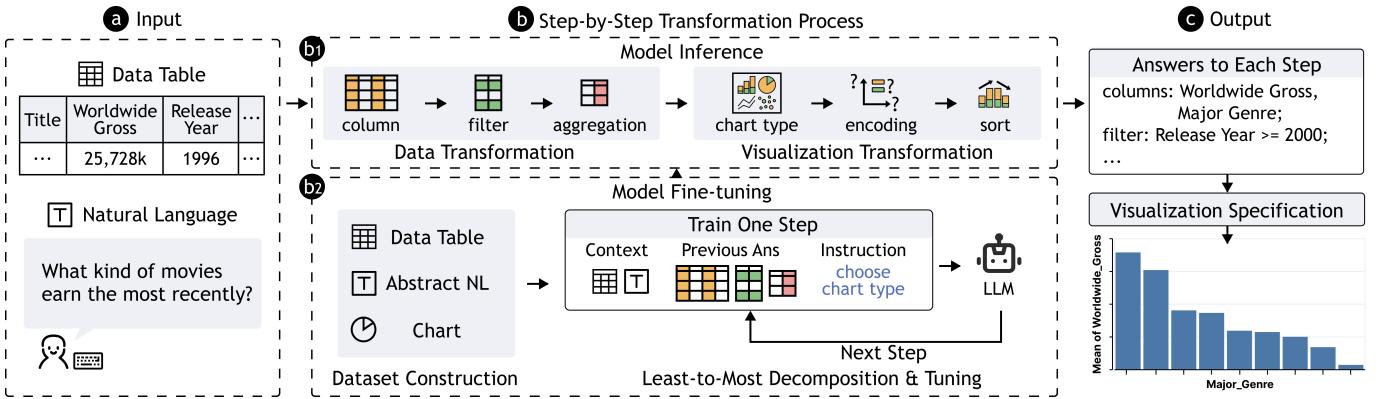


Fig. 2. ChartGPT overview. ChartGPT takes a data table and an utterance provided by the user as input (a). To generate the chart, ChartGPT employs a step-by-step transformation process (b) that decomposes the chart generation task into six sequential steps (b1). Each step is solved by the LLM fine-tuned on our constructed dataset (b2). By leveraging the output from each step, ChartGPT generates visualization specifications and presents charts to the user (c).

1) Problem Decomposition: Inspired by grammars of graphics [62]–[64], we divided the process from data to visualization specification into two successive transformations: data transformation and visualization transformation (Figure 1b1 and Figure 1b2). Both consist of three sub-tasks, resulting in a sequence of six sub-tasks performed step-by-step.

Data transformation. Data transformation contains operations that deal with table data. After this transformation, the transformed data can be encoded directly into visual channels. The data transformation process includes three sub-tasks: selecting columns, filtering rows, and adding aggregations. **First**, relevant columns are selected based on the data and user utterance, usually involving 1-3 columns. **Second**, data rows are filtered if needed. **Third**, data columns are aggregated using functions such as *count*, *average*, and *sum* if necessary.

Visualization transformation. After obtaining the transformed data, we should determine the appropriate encoding of visual channels. This process also contains three sub-tasks: choosing chart type, determining visual encoding, and adding optional operations. **First**, the model needs to infer which chart type is suitable for the selected data, aggregation, and utterance. **Second**, the model is required to map the data fields to visual channels. Notably, the fields in this sub-task have been transformed. For example, if executing counting on a specific field “a”, the field to be encoded should be “count(a)”. **Third**, there are possibly optional operations for the resulting charts, such as color, sorting order, and bin width, etc. In this study, we primarily consider sorting by axis for simplicity.

After the six-step successive transformation process, adequate information is obtained to formulate a visualization specification. Indeed, chart generation extends far beyond the aforementioned steps. Other design alternatives encompass factors such as color, size, bandwidth, and orientation. The transformation involved in chart generation is not limited to simple filter conditions and aggregation functions as well. These alternatives can be realized through engineering extensions, i.e., introducing additional steps or options and expanding related datasets. In this study, as a proof-of-concept system, we only consider the six sub-tasks and several main design choices to simplify the problem and encourage more

design alternatives to be explored in future work.

2) Answer Template for Sub-tasks: After decomposing the problem into sub-tasks, the next step is to model each sub-task as a sequence-to-sequence problem solvable by the LLM.

As illustrated in Figure 2, the model processes each sub-task with a text sequence input, consisting of the utterance, table data, and answers to previous sub-tasks to enhance reasoning. The model will output a text sequence as the sub-task answer, expected to meet two criteria: (1) cover all mandatory information and (2) be well-formatted to enable accurate parsing and valid specification construction. To support this, we defined a corresponding template sequence for each sub-task, as illustrated in Figure 3, similar to Vega-Zero [2].

Specifically, the **selected columns** are represented by the data fields (denoted by ‘col’) separated by commas. **Filter** is an expression string comprised of conditions, each involving a data field with a predicate such as equal, greater than, and less than. These conditions can be logically connected by ‘and’ / ‘or’. **Aggregation** functions (denoted by ‘aggr’) can be applied to the selected columns as ‘aggr col’, including count, average, sum, max, and min. **Mark** specifies the chart type, including bar, pie, line, and scatter. **Encoding** maps the axes with selected columns (C) and aggregations (A). **Sort** indicates which axis (x/y) to sort and in which order (desc/asc).

The model outputs the answers to each sub-task and combines the answers of filter, mark, encoding, and sort to generate a Vega-Lite specification. Some sub-tasks, including filter, aggregation, color encoding, and sort, may not always be necessary, and the model will output ‘none’ in these cases.

Our system covers seven chart types commonly found in data analysis [65], [66]: bar, stacked bar, line, grouped line, scatter, grouped scatter, and pie. More complex chart types, such as radar charts and heat maps, are not included in the template. Filter and aggregation also contain design options that are beyond our scope. In our work, we only focus on the basic design alternatives for each sub-task to initially validate the potential of LLMs to reason about visualization design.

IV. CHARTGPT

This section describes the approach utilized to guide the LLM’s reasoning for answering each sub-task. We derived an

a Step Instruction	b Sequence Example	c Sequence Format
1 Select columns:	Name, Price, classify	[col]+ [C]+
2 Add filter:	Price < 100	[expression]
3 Add aggregations:	count Name	[aggr col]+ [A]+
4 Select chart type:	bar	[chart type]
5 Choose encoding:	x=Name, y=count Name,	x=[C/A], y=[C/A],
	color=classify	color=[C]
6 Add sort:	y desc	[axis] [direction]

Fig. 3. The template sequence for each sub-task.

abstract NL2VIS dataset to fine-tune a large language model and generate the answers through the model. The dataset was constructed through prompting GPT-3 [9]. The model¹ is released on Hugging Face. The dataset, prompts, and model input settings are provided in our supplementary materials.

A. Model Input

For a specific sub-task, since its answer is based on the input context and the answers of previous sub-tasks, the model input should comprise three pieces of information: (1) table data, (2) user utterance, and (3) answers to previous sub-tasks. However, due to the limited token length that LLMs can handle, it is not feasible to feed the entire table data into the model. Thus, we only incorporate the column names and the top two data rows into the model input. Moreover, to compensate for the possible model cognitive bias from including partial data only, we added the type of each data column to the input to provide a data overview.

B. Reasoning Prompt and Abstract Utterances

An effective way to use an LLM for a specific downstream task is to design a prompt that guides the model in understanding the task target. The prompt can comprise both instructions and examples. For instance, when a task is to classify the sentiment of Tweets, the prompt may include an instruction that states “decide whether a Tweet’s sentiment is positive, neutral, or negative,” along with a few examples such as “I loved the new Batman movie! =>positive”. The model should then be able to generate a response of “negative” for “I hate chocolate.” This technique of including examples in the prompt is called few-shot prompting [9]. Few-shot prompting can facilitate the model to understand the context and task, which motivated us to consider whether this technique can be applied to generate visualizations from utterances.

However, due to the flexibility of natural language, the user utterances can be abstract for different information and on different levels. For example, in terms of **information abstraction**, users may omit the chart type or refer to the data fields in vague terms, such as using “popular” to represent the column “rating” or “gross”. For **level abstraction**, users may concretely express their visualization intent, such as “A pie chart showing the number of faculty members for each rank.”, which directly specifies the selected columns (rank), aggregations (count) and chart type (pie chart). On the

other hand, they may also use more abstract queries, such as simply saying “show rank.” This omission of specifications can lead to multiple interpretations and reasoning paths for a particular sub-task. For instance, the choice of chart type can be determined by the selected columns (e.g., a scatter plot for two quantitative attributes) or the analytical intent of the user (e.g., a histogram for phrases like “distribution”).

The complexity of interpretation and reasoning paths makes it challenging to provide sufficient examples for each sub-task within a single prompt. To assist the model in gaining a more comprehensive understanding of the sub-task interpretations, we construct a dataset and fine-tune the model accordingly.

C. Dataset for Fine-tuning

1) **Dataset Requirements:** The dataset to fine-tune our model should consist of (data, utterance, chart) triplets. To provide the model with sufficient knowledge, the dataset should cover diverse interpretation and reasoning paths. Therefore, the dataset should meet several requirements:

Various domains and types of data and charts. Ensure diverse data sources across various domains to avoid overfitting to a single domain. If the domains are too concentrated, for example, if most tables are related to movies, the model may overfit this context, making it less adaptable to data from other domains. In addition, the data types and chart types involved should also be comprehensive and diverse.

Different levels of information for data analysis. The utterances should be abstract for different information and on different levels, as is mentioned in subsection IV-B. It should also cover various expressions, such as the way to describe selected columns (e.g., explicit or implicit) and phrasing (e.g., command, question, or query).

Previous work about NL2VIS datasets includes Quda [67], NLV Corpus [18] and nvBench [19]. Quda consists of 14,035 user utterance queries covering various analytical tasks. However, no associated charts are provided. NLV Corpus collected 893 utterances involving ten chart types and further analyzed the utterance features, spanning different expressions and abstractions. However, NLV Corpus is based on only three data tables, making it overly concentrated. The nvBench dataset is the closest to matching our requirements, with 25,750 (data, utterance, chart) triplets from 105 domains of table data. However, most utterances in nvBench are very explicit [56]. Therefore, we construct our dataset based on nvBench, which consists of utterances in different abstractions and expressions.

2) **Dataset Construction:** To construct a dataset based on nvBench, the main task is maintaining the diverse data tables and visual design and generating abstract utterances from the original triplets. To maintain the diversity, we randomly select part of the original triplets covering all domains and chart types, etc. To generate abstract utterances, we use GPT-3 (text-davinci-003) and involve four co-authors to verify their correctness. We produce the dataset in the following process:

Charts selection. We select charts from nvBench to align with our requirements. First, as nvBench contains some charts involving multiple tables (using the ‘join’ operation), we remove this part of the data. Second, nvBench consists of (data,

¹<https://huggingface.co/yuan-tian/chartgpt>

utterance, chart) triplets from various domains and chart types, categorized into four hardness levels: easy, medium, hard, and extra hard. These hardness levels reflect the complexity of chart generation. For instance, a chart that encodes three columns and requires filter, aggregation, and sort operations may be classified as extra hard. We select the charts randomly and ensure the selected data covers all domains, hardness levels, and chart types with a relatively balanced proportion.

Abstract utterance generation. After selecting the charts, we use GPT-3 to generate abstract utterances for each chart from its corresponding (data, utterance, chart) triplets. For each triplet, we manually design a prompt to guide GPT-3 to do this: First, we provide the top few lines of the CSV table data and describe a scenario in which we develop a tool to generate charts automatically based on user utterances and table data. Then, we give an original utterance from the triplet as an explicit utterance example. We tell the GPT-3 model that we need abstract utterances to test the tool’s performance, and require the model to generate abstract utterances based on the explicit original utterance and the table data. We also guide the model that the generated utterances should be more natural, vague, and incomplete and can be in various phrasings.

Moreover, we dynamically checked the diversity of generated utterances during the generation process. For example, at first, we observed that the results tended to use many polite and verbal expressions, such as “Can you show me” (e.g., “Can you show me the amount of matches for each competition on a graph?”) and “I want to see” (e.g., “I want to see a visualization of the number of cinemas in different locations, please.”). This may be attributed to GPT-3’s interpretation of “natural” as incorporating polite and verbal expressions. While these phrasings are commonly used, NLV Corpus demonstrates that short queries or commands are also very often in users’ utterances. Examples from NLV Corpus include “histogram for creative type” and “Plot IMDB rating against Rotten Tomatoes rating.” As NLV Corpus classified the majority of utterances into query, question, and command, we modified the prompts to accommodate a range of phrasings and obtained utterances without overly polite and verbal expressions such as “Budget creation trend” and “Plot capacity by opening year”. We retained the previously generated utterances and included them alongside the new additions in our final dataset.

Abstract utterance correction. The generated utterances should remain consistent with the original chart in the (data, utterance, chart) triplet from nvBench. In other words, the chart should be a reasonable answer to the utterance. As most generated abstract utterances remove or blur some information from the original utterance, some of them became inconsistent with the original charts. Specifically, for filters, compared to chart types and other settings that may still stay consistent with the original chart even after being removed, utterances that remove filter information are no longer consistent with the original chart. Generally, the inconsistent data were filtered manually through three co-authors before being reviewed by another co-author. Any disagreements in the correction of the data were resolved through discussion.

Step-by-step answer generation. As our model outputs consist of the answers to the intermediate sub-tasks, we need

to parse the chart configuration and extract the answers to each sub-task. We then combine the answers and the formatted template to construct the expected output of the model.

3) Dataset Statistics: Our constructed dataset consists of 1,916 (data, chart, utterance) triplets, including 236 data tables, 649 charts, and 1,916 utterances. Figure 4 illustrates the statistics of our dataset.

For data tables, our dataset contains 236 tables from 133 databases, with an average of 5 columns and 202 rows. Quantitative columns account for 47%, nominal columns 41%, and temporal columns 12%. **For charts**, our dataset covers seven chart types. Specifically, 79% of charts involve aggregation, 30% involve sorting, and 19% involve filtering operations.

For utterances, we retained the original utterances from nvBench. The final dataset comprises 1,916 utterances, with 1288 newly generated abstract utterances and 628 original ones. Furthermore, we compared the statistics between our dataset and the human-created dataset, NLV Corpus [18]. We quantified the frequency of explicit information related to selected columns, aggregations, and chart types mentioned in the utterances. **For selected columns**, we calculated the proportion of explicitly mentioned column names. For example, if a chart involved three columns, but the corresponding utterance only referred to two of them, the proportion would be 2/3. **For chart types and aggregations**, we examined the presence of explicit expressions, such as “bar”, “scatterplot” for chart types, and “number of”, “count” for aggregations.

The results indicate that among NLV Corpus utterances, selected columns are explicitly mentioned more frequently (79%), whereas chart types (49%) and aggregations (39%) are often omitted or vaguely expressed. The utterances from nvBench have a higher occurrence of explicit information across the board, particularly for aggregations (65%) and chart types (82%). However, after fusing with the abstract utterances generated with GPT-3, the resultant dataset exhibits a significant reduction in explicit information, particularly concerning aggregations and chart types, which closely resemble NLV Corpus. As a result, the constructed dataset looks natural and similar to the human-created ones to some extent.

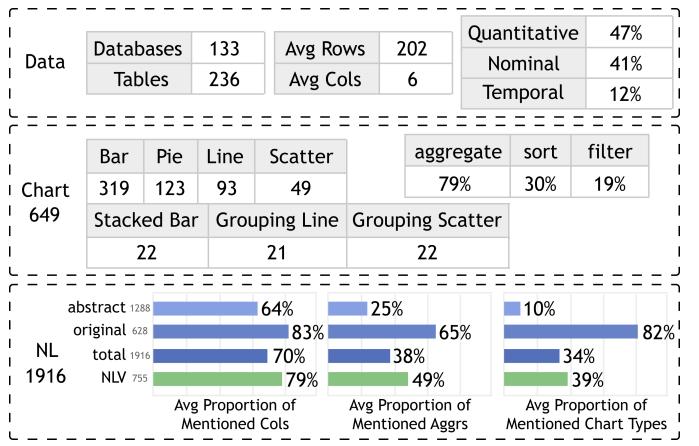


Fig. 4. The statistics of our constructed dataset. Specifically, “abstract” denotes our generated abstract utterances, “original” denotes the maintained original utterances from nvBench, and “total” denotes our total dataset, which includes the “abstract” and “original” utterances.



Fig. 5. The Turing test results between our generated utterances and NLV Corpus ones. (a) The rate of wrong judgment of each subject. (b) The average rate of the two sets that were judged as human-created.

In the analysis above, we focus on assessing the explicit mentions of columns, aggregations, and chart types, as they can be quantified with less ambiguity. However, quantifying the explicit mentions of encoding, filtering, and sorting can involve the subjective opinions of different people. To further measure the quality of our generated abstract utterances and assess their proximity to human-created utterances, we conducted a Turing test.

4) Turing Test: We recruited 14 subjects (7 males and 7 females, all of whom possessed experience in data analysis) to conduct a Turing test evaluating the quality of our generated utterances. We randomly selected 30 utterances from NLV Corpus across 3 tables and 30 utterances from our generated abstract utterances involving 8 tables with shuffled order. During the test, each subject was provided with an utterance alongside the corresponding table at a time. The scenario presented to the subjects was as follows: “Imagine a tool that automatically generates charts based on the table and users’ utterances. Which of the utterances below might be created by a real user?” We explicitly informed the subjects that some displayed utterances were human-created and some were not. Their task was to distinguish between the two categories based on two perspectives: (1) the naturalness of the phrasing and (2) the meaningfulness of the context. We hypothesized that the rate of the generated abstract utterances judged as human-created would be at the same level as the NLV Corpus. After the experiment, we compensated each subject with \$5.

Overall results. The results revealed an average error rate of 56% (Figure 5a), with the lowest error rate recorded at 33%, suggesting that it was hard for subjects to distinguish between the GPT-generated utterances and human-created ones. Additionally, we computed the average rate (α) at which each utterance was judged as human-created. The overall average α for all 60 samples was 0.73, indicating that subjects labeled most samples as human-created.

Comparison between generated utterances and human-created ones. Comparing the two sets, the average α values for our generated abstract utterances and NLV Corpus ones were 0.79 and 0.67, respectively (Figure 5b). The corresponding standard deviation (SD) values were 0.17 and 0.23.

To evaluate the disparity, we conducted a Mann-Whitney U test, which indicated a significant difference ($p = 0.03 < 0.05$). This result suggested that the generated utterances were even more likely to be perceived as human-created than the NLV Corpus ones. To understand this discrepancy, we examined the NLV Corpus samples with lower α values. One utterance stood out with a significantly low α of 0.14: “Sum(Sales) by Order Date split by Category render line asc”. This utterance is similar to captions in format, which is considered less natural

by most subjects. NLV Corpus also acknowledged that their collected utterances contain such samples whose phrasing was relatively infrequent.

D. Model Fine-tuning

We first divided our dataset into a training set consisting of 1,538 triplets for fine-tuning and a test set with 378 triplets (invisible to the model) for evaluation (4:1 split). Then, we fine-tuned the open-source FLAN-T5-XL model [20] with the AdamW optimizer [68] on the training set.

We selected Flan-T5 as it has undergone pre-training on various tasks and possesses strong reasoning capabilities. We employed a learning rate of 1e-4, a global batch size of 16, and trained for five epochs. Generally, the trained model obtains an evaluation loss of 0.10. These parameters are chosen based on the model document ², trial and error, and the capacity of our computational resources. We show the evaluation results in section VI.

E. Top-k Charts Generation

ChartGPT is designed to generate a set of top-k charts ($k=3$ by default) in response to a given utterance. We incorporate two strategies to produce the top-k valid charts efficiently.

To remove invalid candidates, for the candidates generated by the model, we identify and eliminate the invalid candidates that contain (1) column names not present in the data, (2) filter expressions that are grammatically wrong or can not be applied to the data, and (3) aggregation functions, chart types, encoding channels, and sort tokens falling outside our valid space. **To generate results efficiently**, we adopt the beam search [69] to retain the top combinations of the candidates based on cumulative probabilities. Finally, we return the top-k candidates to generate the charts.

V. INTERFACE

We developed an interface with three views: table view, chart view, and detail view. We present the features of our interface through a usage scenario based on a movie dataset.

To begin, the user uploads the CSV file (Figure 6b). The data table is displayed with the column types, including nominal, quantitative, or temporal. The user then quickly navigates through the columns, types, and relevant data. S/he notices that the table contains 10 columns and 709 rows, each row providing information about a particular movie.

The user wants to know “what kind of movies are the most popular?” and enters this question into the search box (Figure 6c). ChartGPT then returns the top three charts based on the input. The user observes that the first and the third charts display the number of movies by genre and creative types, respectively, and the second chart shows the average IMDB rating of each genre. The user is interested in the second one (Figure 6d) and understands that the movie genre with the highest average IMDB rating is Documentary.

²https://huggingface.co/docs/transformers/model_doc/t5

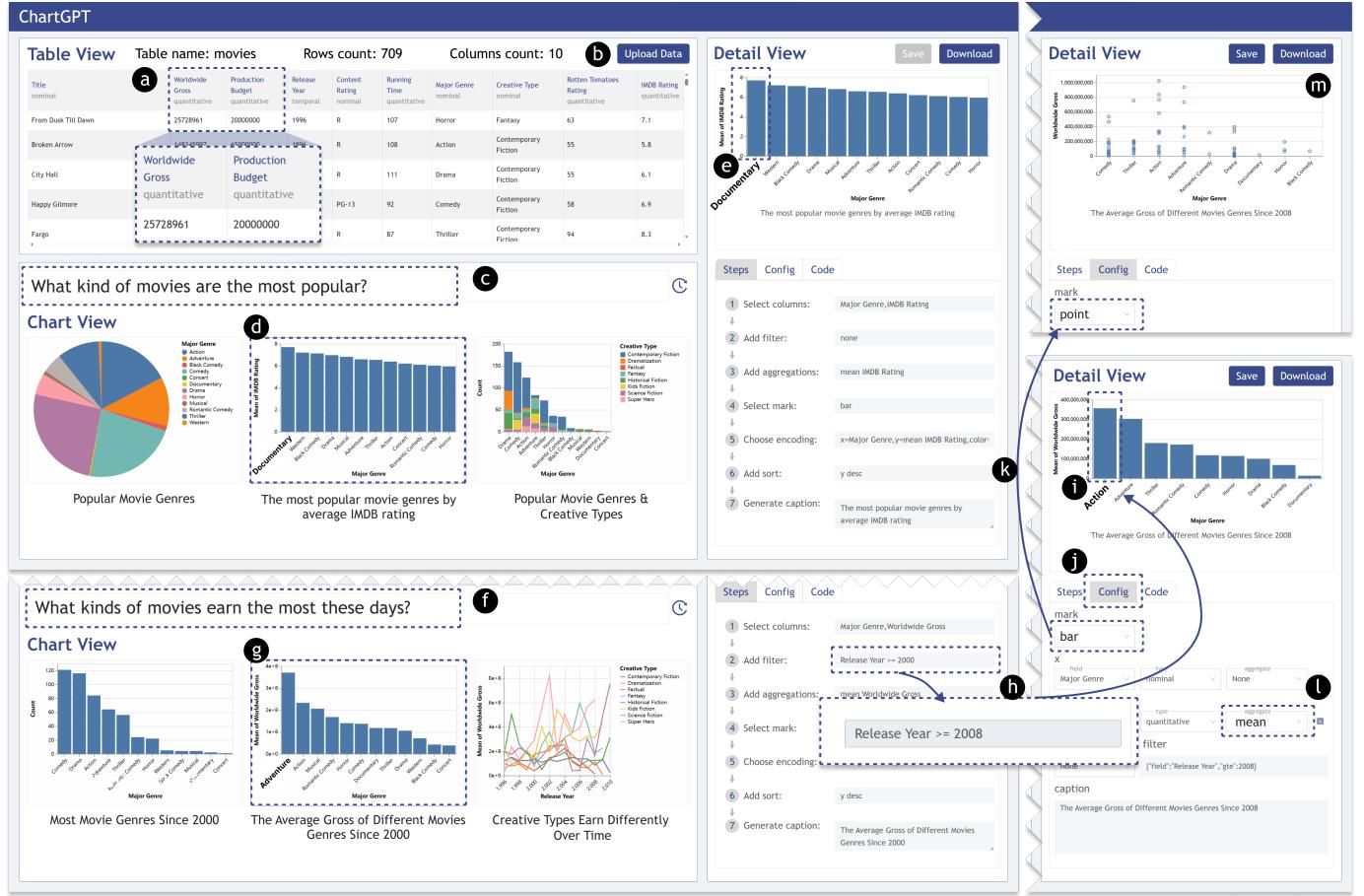


Fig. 6. ChartGPT interface consists of three views: table view (a-b), chart view (c-g), and detail view (h-m). Table view displays the data table and relevant data features. Chart view enables users to input their utterances and presents the generated charts. Detail view provides chart specifications and allows users to modify the results through interactions.

In addition to the count and ratings, the user further notes that the data contains information on gross and budget (Figure 6a). The user changes the input to “What kinds of movies earn the most these days?” (Figure 6f). The results update, and the second and the third charts are about worldwide gross. The user investigates the charts and checks them in the detail view. S/he observes that the second chart (Figure 6g) has a filter condition of “Release Year ≥ 2000 ”, which corresponds to the utterance “these days”.

The user is not fully satisfied with the filtering condition and expects more recent movies. S/he changes the condition to “Release Year ≥ 2008 ” and regenerates the result from step 3 (Figure 6h). After re-rendering, the user discovers that the genre with the highest average gross since 2008 is Action (Figure 6i). Furthermore, the user wants to see the distribution of movies for each genre. Therefore, the user switches to the config mode (Figure 6j) in the detail view. S/he changes the mark type to “point” (Figure 6k) and removes the aggregation of the y-axis (Figure 6l), resulting in a scatter plot that meets the needs (Figure 6m).

VI. EVALUATION

This section introduces the quantitative evaluation of ChartGPT with NL4DV and ncNet.

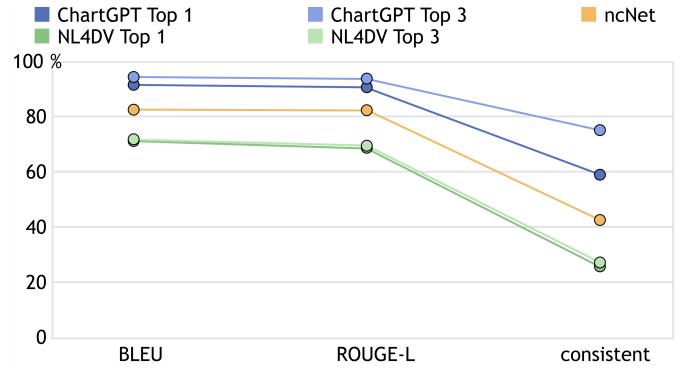


Fig. 7. The evaluation result shows the performance of ChartGPT, ncNet, and NL4DV on different metrics.

A. Evaluation Setup

We used our test set (different from the training set) derived in subsection IV-D to evaluate the performance of ChartGPT, NL4DV, and ncNet. As both our system and NL4DV can return more than one result, we reported the top-1 and top-3 results for these two methods and reported the top-1 result for ncNet. However, please note that the design spaces of the three methods are also slightly different from each other.

For example, NL4DV supports boxplots and tick charts but doesn't support pie charts. For fairness, we only compared results that can be produced by all methods. For the test data with configurations in our design space that NL4DV does not support, we didn't introduce them into the result statistics.

B. Evaluation Metrics

We measured two metrics: consistency and similarity. Consistency is used to count how many results are exactly the same as the ground truth. In addition, as abstract utterances may cause ambiguity, we further accounted for how similar the results are to the ground truth. We hypothesized that even if the utterance is ambiguous and can correspond to multiple correct answers, these answers are also similar to ground truth to some extent. Therefore, we used the degree of similarity to further measure the system's ability to handle abstract utterances.

Consistency Metrics. We define a result as “consistent” if the result is identical to the ground truth. In our scenario, “identical” means identical in all supported design alternatives, including mark, encoding, aggregation, sort, and filter. In addition, we consider two scatter plots with x and y reversed as consistent as well, as they still point to equivalent results [18].

Similarity Metrics. We define the “similarity” of a result as the degree to which it is similar to the ground truth in terms of the design alternatives. We converted the ground truth and results of different methods into equal-length word sequences, and then compared the similarity of the sequences. The format of the sequence is defined as an 8-word sequence, i.e., [mark] [x field] [x aggregation] [y field] [y aggregation] [color field] [filter] [sort], and each part is a single word. Then, we measured the ROUGH-L [70] and BLEU [71] metrics between the results and the ground truth sequences. ROUGH-L calculates the similarity based on the length of the longest common subsequence (LCS), which is affected by both the value and order of words. Under this metric, if the selected fields and aggregations in both the ground truth and the model result are the same but encoded on different axes, the score will reduce. We suppose that charts with the same selected fields and aggregations but mapped to different axes from the ground truth are still acceptable when compared to the ones with inappropriate encoding. Therefore, as a complement to ROUGE-L, we measured the BLEU score, which allows the model to switch the order of some encoded fields.

Specifically, before converting into a sequence, we validate whether the result is related to the data table and can be parsed properly into a Vega-Lite format. For example, if the result contains a column name that is not in the table, it is considered invalid. Parsing such results into Vega-Lite and displaying them will report errors or undefined displays because it cannot find the corresponding data. We marked the similarity and consistency of such results as zero.

C. Evaluation Results

Our evaluation results are presented in Figure 7, which showcases the top-1 and top-3 reviews of ChartGPT, ncNet, and NL4DV. The results indicate that ChartGPT outperforms the other two approaches in terms of both the consistency

metric and similarity metric, with its top-1 and top-3 reviews scoring higher than those of ncNet and NL4DV.

Comparison with the baselines. Looking through the tested cases, there are two key factors that account for the differences between the approaches: One is semantic understanding. ChartGPT has a better parsing of the semantic information of the columns and utterances. Examples include inferring the column “sex” from “male and female”, column “age” from “how old”, and inferring a temporal column and count aggregation from “when create the most departments”. The other is omitted information. Abstract utterances often omit information such as aggregation and chart types, which requires the reasoning capability of the visualization specifications. ChartGPT is based on Flan-T5, which is previously fine-tuned on chain-of-thought (CoT) reasoning tasks and is further fine-tuned by us on visualization datasets in a CoT way, so it may have a better reasoning ability of omitted information.

Metric difference analysis. The consistency metric is drastically lower when compared to the other two metrics, which is possibly due to two factors. First, ambiguity in abstract utterances often results in multiple reasonable answers. For instance, consider the abstract utterance “How many documents are at each location? ” from the original utterance “Show the number of documents for each location code in a pie chart”. This abstraction removes the chart-type information, making a bar chart also a reasonable response. Second, partial correct inferences occur when the model misses some subtle yet critical to chart expressiveness information. For example, the model may correctly extract the needed columns but give wrong aggregations, or miss the filter and sort conditions.

VII. USER STUDY

We derived a comparative study and a usability study to evaluate ChartGPT further. Through the user studies, we want to (1) compare the results of ChartGPT with the two baseline methods from users' perspective, and (2) evaluate the usability of ChartGPT.

A. Comparative Study

In this study, we recruited 12 subjects (6 males and 6 females, all of whom possessed experience in generating data visualization) to conduct a comparative study evaluating the quality of generated charts from different approaches (ChartGPT, ncNet, and NL4DV). None of them has the experience of using the approaches above.

Tasks and Data. We sampled 15 utterances from the test set, corresponding to 13 data tables and 42 charts generated from NL4DV (top-1), ncNet, and ChartGPT (top-1). Subjects were presented with the tables, utterances, and generated charts in random order and were required to compare and rank the quality of charts, deciding which charts were more reasonable for the table and utterances based on their preferences. If a chart makes no sense in their opinion, it won't be included in the ranking. The sampling is based on two steps. First, we selected the tables that are close to common sense to ensure that subjects can understand the context. Second, we

selected the abstract utterances from the selected tables and ensured that (1) the utterances are in various abstractions and phrasings and (2) the chart types are all included.

Procedure. The entire experiment lasted about 10-25 minutes. First, subjects were introduced to the background of NLIs for generating data visualizations for 3 minutes. Then, they began to compare and rank the quality of generated charts based on the provided data table and utterances. Subjects were required to ensure that they understood the data table and utterances before performing the actions. They were allowed to ask about the meaning of the data table, utterance, or a particular legend in the chart but had to rank the charts entirely according to their own preferences. After the experiment, we compensated each subject with \$5.

Results. We counted the ranking results of the subjects. Specifically, for the user's ranking of the charts corresponding to a particular utterance, we normalized the rankings into scores from 0 to 3, with the first ranking scored as 3 and the charts that did not appear in the ranking scored as 0. Additionally, we calculated the proportion that each approach was first-ranked. We used a Friedman Test to examine whether a significant difference exists across the approaches and a post hoc Wilcoxon Test to compare the pair-groups.

The results (Figure 8) showed both significant differences in the ranking score ($\chi^2=8.00$, $p < 0.05$) and first-ranked proportion ($\chi^2=17.64$, $p < 0.001$). Overall, ChartGPT had the best performance (i.e., the higher ranking score and first-ranked proportion on average, $p < 0.05$).

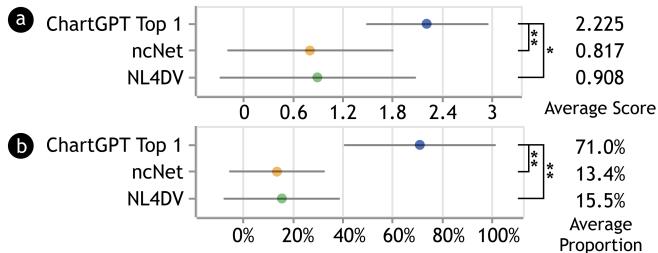


Fig. 8. Results of the comparative study with SD values (*: $p < 0.05$, **: $p < 0.01$), including (a) average ranking scores and (b) first-ranked proportions.

B. Usability Study

1) Experiment Settings: Participants. We recruited 12 subjects (S1-S12, 6 males and 6 females) from different departments, including Computer Science (3), Sports Science (2), Digital Media Design (2), Urban Informatics (1), Industrial Design (1), Geographic Information Science (1), Agricultural Engineering (1) and Corporate Finance (1). Most subjects were familiar with data visualization, with an average self-reported score of 3.4 on a 5-point Likert Scale. All subjects had experience using tools to author charts, including Microsoft Excel, Vega-Lite, D3.js, G2, ECharts, and Matplotlib. In addition, all of them had experience in using natural language interfaces (including ChatGPT) and scientific English writing.

Tasks and Data. Subjects were provided with two data tables (movies and cars) and were required to choose the one they were more familiar with or more interested in. They

were required to explore the selected data with ChartGPT and create at least four desired charts. During the creation process, if the default generated chart did not match their desires, subjects could rephrase their input, modify the step answers and regenerate the results, or modify the chart configuration directly. However, if subjects could not get the desired chart no matter what action they took, or if the desired chart was not supported, they could give up the intent and try to generate another one. In total, the created charts should contain at least two chart types and involve at least three different data columns. Both the movies and cars data come from NLV Corpus [18] and have more than 9 columns and 300 rows, involving all three types of values (temporal, nominal, and quantitative). We chose these two data tables as their context is close to common sense and is easy to understand.

Procedure. The entire experiment lasted about 20-35 minutes. Subjects were first presented with the movie and car datasets and were required to select one of them based on familiarity or interest. We ensured that subjects could understand the dataset before the next process. Subjects were then introduced to the interface and interactions of ChartGPT. During the introduction, we did not provide the subjects with any concrete input examples to avoid biasing their language organization. Instead, we encouraged users to formulate their own input and introduced the interface and interactions along the way. After the introduction, subjects began to create their desired charts with their selected data. All inputs and actions taken by subjects are recorded. Finally, we interviewed the subjects to collect their feedback about ChartGPT. After the experiment, we compensated each subject with \$10.

2) Quantitative Results: The results of the usability study are illustrated in Figure 9, and the corresponding statistics are presented in Figure 9b. A total of 53 historical logs were collected from the subjects, and 49 of them resulted in successfully generated charts. The other 4 failed logs indicated that the subjects could not obtain a satisfactory chart, thus giving up the input and began to generate a different chart. These successfully generated charts were further classified into three categories based on the actions that the subjects performed to obtain them: (i) obtained on the first attempt, (ii) obtained after adjusting step or config settings, and (iii) obtained after rephrasing the input utterance.

Nearly half of the charts (23 out of 49, 47%) were obtained on the first attempt. Besides, 13 cases involved step or config adjustments, and 13 cases involved input rephrasing. However, such adjustments did not necessarily imply that the system-generated results did not match their input. In fact, the input and the generated chart were consistent for all step or config adjustment cases and most rephrasing cases. Nonetheless, some subjects wanted to experiment with further adjustments after viewing the initial chart. We further counted the number of cases where the generated chart and user input matched among the rephrasing cases (10 out of 13). In the remaining three cases, the subjects attempted to rephrase their input once, twice, and thrice, respectively, until they obtained a result that matched their input.

Among the four failed inputs, three of them involved not supported data transformations or visual designs, such as

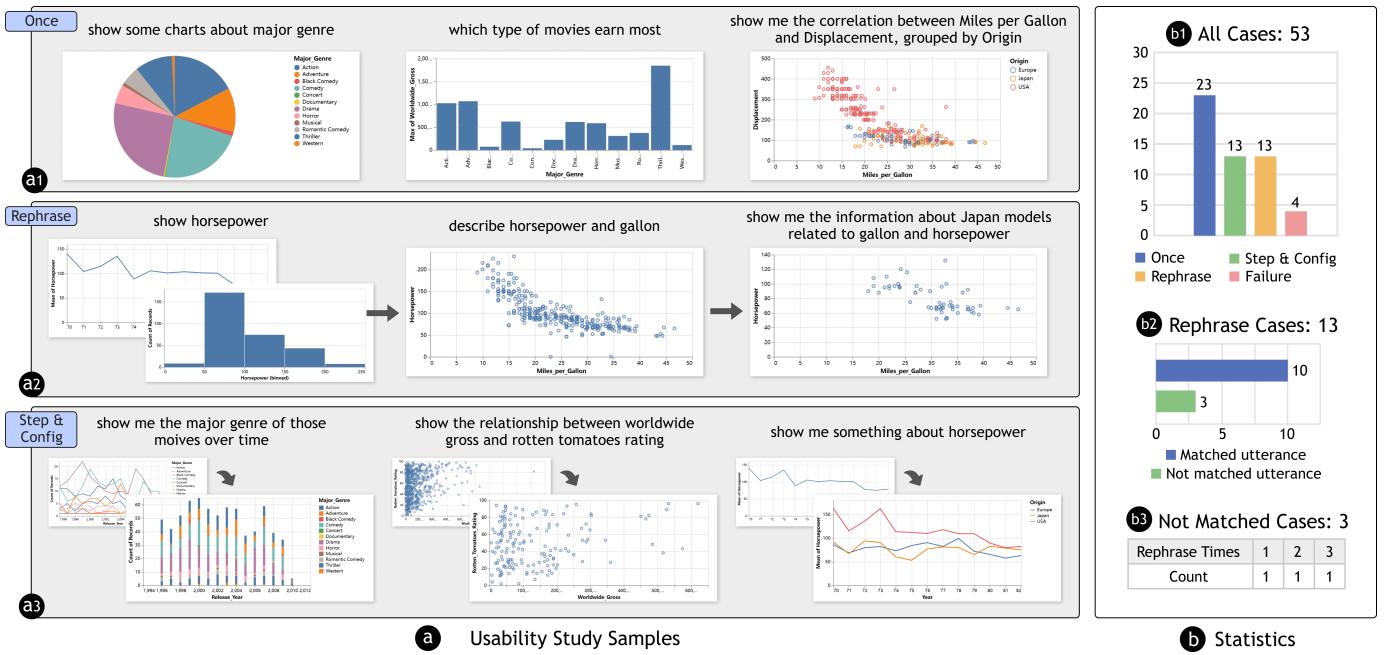


Fig. 9. Results of the usability study, including samples of generated charts from subjects (a) and quantitative statistics (b).

dividing gross by budget or displaying two bar charts side by side in a single chart. The remaining input could not produce a valid chart as the provided encoding was self-contradictory (attempting to encode two data fields on the x-axis).

3) Qualitative Feedback: The system's ability to respond to incomplete intent streamlines the thought process and enables users to explore data from the shallow to the deep. Most subjects involved some input with incomplete intent. Instead of referring to trends, distributions, or relationships, these inputs only indicate the data columns they are interested in, such as “show some charts about major genre” from S7 (left in Figure 9a1). S7 notes, “When I first started looking at the data, I only had an initial interest in certain data columns (e.g. major genre).” This allows them to give input once they have an initial idea and observe the system’s response. S6 further mentioned, “I only need to do one short step of thinking before viewing the results, while when using other tools, I often have to carefully define my intentions from vague to explicit.” In addition, some subjects used the results from incomplete input to understand the data, draw connections, and develop further intents. For example, as is illustrated in Figure 9a2, after seeing the results of “show horsepower”, S12 became interested in “miles per gallon” and entered “describe horsepower and gallon”. Further, she wanted to focus on Japanese cars and typed “show me the information about Japanese models related to gallon and horsepower” and obtained the desired result. As such, the system’s ability to answer abstract requests that don’t articulate a complete intention shortens the thought process needed for every single round of interaction, enabling users to explore the data from the shallow to the deep.

ChartGPT supports a semantic understanding of the visual intent, allowing users to express themselves flexibly and naturally. Some subjects involved inputs that can not match the corresponding data columns directly. For example,

when S6 entered “which type of movies earn most”, the system could understand the keywords ‘type’ and ‘earn’ and infer the Major Genre and Worldwide Gross columns (middle in Figure 9a1). Moreover, this semantic inference is not restricted to direct word-to-word mapping but is a general understanding. For example, on S8’s input of “number of movies over time,” the system could determine that the ‘Release Year’ column may be a more appropriate choice than ‘Running Time’. In this regard, about half of our subjects commented that the system is “smart” as it has some semantic inference ability and good support for natural language flexibility. Specifically, S2 praised its “flexible semantic associations” which alleviates his burden of perfecting their language to be more precise for the system. In general, our system’s semantic understanding of utterances facilitates a more user-friendly experience as it reduces the need to be exact in users’ phrasing.

The interaction to modify the results of intermediate steps can shorten the distance between the system-generated and user-desired results. Despite the majority of subjects recognizing ChartGPT’s ability to understand semantic natural language and produce accurate results, due to user preferences and the ambiguity of the user’s natural language, the generated results sometimes adhered to their expressions, yet did not yield their desired outcome in some parts. For instance, S3 initially entered “show the relationship between worldwide gross and rotten tomatoes rating” and obtained a scatter plot between the two mentioned columns. However, she thought that this chart had too many points and wanted to focus on comedy movies, so she added the condition ‘Major Genre = ‘Comedy’’ to the filter step and regenerated the results (middle in Figure 9a3). S3 commented, “I can regenerate results from the middle without reformulating my original input when I have a clear intent to target a particular step.” Overall, 10 of our 12 subjects have employed modifications of the steps

or configurations according to their preferences. S2 further pointed out that after seeing the initial results generated by the system, it is simple to determine if its details match his preferences, resulting in a “clear direction for modification”.

VIII. DISCUSSION

This section includes the implications, lessons learned, limitations, and future work of our system.

A. Implications

In terms of technique, our framework employs LLMs for generating charts from abstract utterances using a “decomposition and fine-tune” approach that involves a limited-size dataset. We demonstrate its effectiveness through both quantitative evaluations and a user study. In terms of evaluation, we contribute a dataset of abstract utterances and corresponding charts generated using LLMs. This dataset can serve as a benchmark for future research and training data for machine learning studies. Additionally, our method of constructing the dataset from LLMs and using it to fine-tune LLMs is significant. In terms of applicability, our framework’s applicability extends beyond NL2VIS generation, as it can be used to solve complex downstream tasks that LLMs cannot directly handle. For instance, long story writing can also be decomposed into several sub-modules, from planning the characters and outline to drafting and editing the story continuation [72]. The feedback from these experiments provides valuable insights into the potential applications of LLMs in generating visualizations, inspiring further research in this field.

B. Lessons Learned

Modification is important to suit different preferences. Users have varying preferences for chart design choices and may not always follow a consistent design rule. During our data collection, most of the data we collated tended to follow common design principles, such as using scatter plots for two quantitative data columns and line charts for displaying trends over time. However, our user study revealed that users’ preferences were not always consistent. For instance, when aggregation was not specified explicitly in the utterances, some subjects preferred to average data while others preferred to look at the maximum value. Additionally, during the free exploration task, some subjects switched from a scatter plot displaying two quantitative data columns to a line graph or from a line graph showing trends over time to a bar graph. This underscores the importance of providing users with interactions to modify or fine-tune the results in the authoring tool to facilitate human-in-the-loop, as the generated results are not guaranteed to always match everyone’s preferences.

C. Limitations and Future Work

Support for a larger scope. ChartGPT only supports some key chart components and design choices for chart generation, with an aim to demonstrate the usefulness of our framework. Future work could involve support for a larger scope. First, additional transformations and visualization parameters could

be considered. Currently, we have not considered operations that reshape data tables, such as pivot and mutate [73], [74]. We could add a transformation step before selecting the columns to support the transformations. Parameters, such as mark types and visual channels, can be extended by enlarging our dataset. Second, supporting follow-up utterances to modify the generated charts [75] is also an intuitive manner for human-LLM interaction. To achieve this, we can train an LLM using a dataset with existing specifications and modification commands as input and an updated specification as output. It requires the construction of the dataset, which can also be attained with the help of ChatGPT. Third, as an LLM for a specific domain, it is required to recognize out-of-domain queries and raise warnings. To do so, we can add an additional boolean token representing whether the utterance is related to the input data and visual analysis. Negative examples can be generated and mixed up with our proposed dataset.

Scalability for large input tables. The model input includes table headers, column types, two data rows, and utterances. Therefore, the number of columns in the table would affect the prompt length. Based on our dataset, we trained the model with a maximum prompt length of 580 input tokens. To accommodate large tables exceeding this size, there are two potential improvements: (a) Reconfigure the model input to reduce the token count. For example, enabling LLMs to selectively choose columns, followed by the system providing additional values and information, can help reduce the prompt length. Such prompt improvement also holds the potential to provide deeper data insights, such as data distribution, within the constraints of a limited prompt length. (b) Expand our training dataset and allocate additional computational resources to accommodate longer prompts. In addition, as a restricted prompt length would result in a less comprehensive inclusion of table information, further comparison with rule-based methods on large tables remains future work.

Comparison with the generic LLM-based methods. Researchers have explored using generic LLM-based methods for chart generation. For example, LIDA creates charts by generating and executing Python code. LIDA allows flexibility in selecting visualization libraries, such as Seaborn [76] and Altair [77]. Without a predefined design space, LIDA can accommodate diverse choices beyond our scope, posing challenges for applying our evaluation metrics. This limitation prompts the need for future research to compare such approaches comprehensively.

Despite the limitation, we tested LIDA on our test set. We observed 67 and 12 failed cases (raised errors while generating charts) for Seaborn and Altair, respectively (compared to 7 for ChartGPT). Many failures stem from calling nonexistent functions, revealing the inherent hallucination issues in generic LLMs. In addition, LIDA’s performance with Seaborn proves more stable than with Altair, possibly due to Seaborn’s prevalence in the GPT corpus. This underscores generic LLMs’ dependence on extensive existing corpora, exposing limitations in handling new schemas. Fine-tuning an LLM with specific data may compromise generalizability but can be valuable in scenarios requiring stability or new schemas.

Inspiration v.s. accuracy. ChartGPT aims to accurately

capture the intent from the user's abstract utterance and make reasonable inferences. Therefore, our system tends to prioritize the accuracy of the results by presenting the most relevant information first and providing optional charts later. Our dataset also reflects this tendency. When an utterance involves only a certain data column and lacks other intent, the ground truth is often to show the distribution of the column, which is the most closely related to the utterance's information.

Despite this emphasis on accuracy, user feedback has indicated that it is not always the primary concern. For example, during the comparing and ranking task, for the utterance "show something about origin," some of our subjects preferred the chart showing the origin and other data fields. Similarly, during the free exploring stage, three subjects suggested that they would like to see content that could inspire them beyond the scope of their utterances. Two of them emphasized that this requirement becomes more noticeable when seeking inspiration during data exploration. This feedback shows a tendency for desiring charts that cover a broader range of data columns while exploring data [30], [31]. In response to this feedback, we plan to propose an option for users to specify their desired level of inspiration (e.g., "high inspiration" versus "accuracy only") in their query in the future. This allows the system to match users' needs better and enhance their experience.

Flexibility v.s. certainty. Our system accommodated a wide range of user intentions, but limitations arose when users expressed intentions beyond the system's current capabilities. During our study, we observed two subjects attempting to explore data using intentions not supported by the system. One of the subjects expressed an intention that could not be drawn as a chart, while the other wanted to do a data transformation in which two columns in a table were computed, e.g., gross divided by budget. In such cases, our system still produced results, which unfortunately did not align with their intentions. However, it took the subjects quite some time to evaluate and finally realize that the system did not support their intentions after adjusting their inputs several times. While our design space could be expanded to accommodate more needs, the flexibility of natural language and the definite design space of the system mean that the system's capability is limited to support the full range of natural language expressions, leading to confusion for users about which inputs will lead to successful chart results. Future work could explore enhancing the system's recognition of inputs beyond its supported range. For example, instead of implementing all possible user intentions beyond the scope, one potential avenue is to integrate a preliminary step that identifies inputs exceeding the system's limits and issues a warning.

IX. CONCLUSION

This paper introduces ChartGPT, leveraging LLMs to generate charts from abstract utterances. We formulate the chart-generation problem as a sequential reasoning task and construct an abstract utterance dataset to fine-tune a language model for solving each task. Furthermore, we design an interactive interface for ChartGPT to enable users to examine and modify intermediate outputs. The effectiveness of the proposed system is evaluated through comparative and usability studies.

ACKNOWLEDGMENTS

This work was supported by NSFC (U22A2032) and Key "Pioneer" R&D Projects of Zhejiang Province (2023C01120).

REFERENCES

- [1] A. Narechania, A. Srinivasan, and J. Stasko, "NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 369–379, 2020.
- [2] Y. Luo, N. Tang, G. Li, J. Tang, C. Chai, and X. Qin, "Natural language to visualization by neural machine translation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 217–226, 2021.
- [3] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, "Vega-Lite: A grammar of interactive graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 341–350, 2016.
- [4] Tableau Software, "Tableau," <https://www.tableau.com/>, 2003.
- [5] D. E. Rose and D. Levinson, "Understanding user goals in web search," in *Proceedings of the International Conference on World Wide Web*, 2004, pp. 13–19.
- [6] V. Setlur, M. Tory, and A. Djalali, "Inferencing underspecified natural language utterances in visual analysis," in *Proceedings of the International Conference on Intelligent User Interfaces*, 2019, pp. 40–51.
- [7] A. Wu, Y. Wang, X. Shu, D. Moritz, W. Cui, H. Zhang, D. Zhang, and H. Qu, "AI4VIS: Survey on artificial intelligence approaches for data visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 12, pp. 5049–5070, 2022.
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Proceedings of Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.
- [10] OpenAI, "Introducing chatgpt," <https://openai.com/blog/chatgpt>, 2022.
- [11] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.
- [12] Y. Huang, Y. Zhou, R. Chen, C. Pan, X. Shu, D. Weng, and Y. Wu, "Interactive table synthesis with natural language," *IEEE Transactions on Visualization and Computer Graphics*, 2023.
- [13] J. J. Y. Chung, W. Kim, K. M. Yoo, H. Lee, E. Adar, and M. Chang, "TaleBrush: sketching stories with generative pretrained language models," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–19.
- [14] L. Ying, Y. Wang, H. Li, S. Dou, H. Zhang, X. Jiang, H. Qu, and Y. Wu, "Reviving static charts into live charts," *arXiv preprint arXiv:2309.02967*, 2023.
- [15] T. S. Kim, D. Choi, Y. Choi, and J. Kim, "Stylette: Styling the web with natural language," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–17.
- [16] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston, "Neural text generation with unlikelihood training," *arXiv preprint arXiv:1908.04319*, 2019.
- [17] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, O. Bousquet, Q. Le, and E. Chi, "Least-to-most prompting enables complex reasoning in large language models," *arXiv preprint arXiv:2205.10625*, 2022.
- [18] 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 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–10.
- [19] 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 International Conference on Management of Data*, 2021, pp. 1235–1247.

- [20] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei, “Scaling instruction-finetuned language models,” *arXiv preprint arXiv:2210.11416*, 2022.
- [21] X. Qin, Y. Luo, N. Tang, and G. Li, “Making data visualization more efficient and effective: a survey,” *The VLDB Journal*, vol. 29, pp. 93–117, 2020.
- [22] J. Lin, Y. Cai, X. Wu, and J. Lu, “Graph-based information block detection in infographic with gestalt organization principles,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 3, pp. 1705–1718, 2021.
- [23] L. Ying, X. Shu, D. Deng, Y. Yang, T. Tang, L. Yu, and Y. Wu, “Metaglyph: Automatic generation of metaphoric glyph-based visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 331–341, 2023.
- [24] Y. Wei, H. Mei, W. Huang, X. Wu, M. Xu, and W. Chen, “An evolutional model for operation-driven visualization design,” *Journal of Visualization*, pp. 1–16, 2022.
- [25] Y. Zhou, X. Meng, Y. Wu, T. Tang, Y. Wang, and Y. Wu, “An intelligent approach to automatically discovering visual insights,” *Journal of Visualization*, vol. 26, no. 3, pp. 705–722, 2023.
- [26] B. Saket, D. Moritz, H. Lin, V. Dibia, C. Demiralp, and J. Heer, “Beyond heuristics: Learning visualization design,” *arXiv preprint arXiv:1807.06641*, 2018.
- [27] J. Mackinlay, “Automating the design of graphical presentations of relational information,” *Acm Transactions On Graphics*, vol. 5, no. 2, pp. 110–141, 1986.
- [28] J. Mackinlay, P. Hanrahan, and C. Stolte, “Show Me: Automatic presentation for visual analysis,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1137–1144, 2007.
- [29] 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*, 2016, pp. 1–6.
- [30] Wongsuphasawat, Kanit and Moritz, Dominik and Anand, Anushka and Mackinlay, Jock and Howe, Bill and Heer, Jeffrey, “Voyager: Exploratory analysis via faceted browsing of visualization recommendations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 649–658, 2015.
- [31] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer, “Voyager 2: Augmenting visual analysis with partial view specifications,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2017, pp. 2648–2659.
- [32] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer, “Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 438–448, 2018.
- [33] J. Dong, H. Zhang, M. Cui, Y. Lin, H.-Y. Wu, and C. Bi, “Tcevis: Visual analytics of traffic congestion influencing factors based on explainable machine learning,” *Visual Informatics*, 2023.
- [34] Y. Sun, J. Li, S. Chen, G. Andrienko, N. Andrienko, and K. Zhang, “A learning-based approach for efficient visualization construction,” *Visual Informatics*, vol. 6, no. 1, pp. 14–25, 2022.
- [35] A. Jiang, M. A. Nacenta, and J. Ye, “Visualizations as intermediate representations (vlair): An approach for applying deep learning-based computer vision to non-image-based data,” *Visual Informatics*, vol. 6, no. 3, pp. 35–50, 2022.
- [36] Y. Luo, X. Qin, N. Tang, and G. Li, “DeepEye: Towards automatic data visualization,” in *Proceedings of IEEE International Conference on Data Engineering*, 2018, pp. 101–112.
- [37] V. Dibia and C. Demiralp, “Data2Vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks,” *IEEE Computer Graphics and Applications*, vol. 39, no. 5, pp. 33–46, 2019.
- [38] M. Zhou, Q. Li, X. He, Y. Li, Y. Liu, W. Ji, S. Han, Y. Chen, D. Jiang, and D. Zhang, “Table2Charts: recommending charts by learning shared table representations,” in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2389–2399.
- [39] H. Li, Y. Wang, S. Zhang, Y. Song, and H. Qu, “KG4Vis: A knowledge graph-based approach for visualization recommendation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 195–205, 2021.
- [40] H. Zhu, M. Zhu, Y. Feng, D. Cai, Y. Hu, S. Wu, X. Wu, and W. Chen, “Visualizing large-scale high-dimensional data via hierarchical embedding of knn graphs,” *Visual Informatics*, vol. 5, no. 2, pp. 51–59, 2021.
- [41] P. Soni, C. de Runz, F. Bouali, and G. Venturini, “A survey on automatic dashboard recommendation systems,” *Visual Informatics*, 2024.
- [42] A. Wu, Y. Wang, M. Zhou, X. He, H. Zhang, H. Qu, and D. Zhang, “MultiVision: Designing analytical dashboards with deep learning based recommendation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 162–172, 2022.
- [43] D. Deng, A. Wu, H. Qu, and Y. Wu, “DashBot: Insight-driven dashboard generation based on deep reinforcement learning,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 690–700, 2023.
- [44] 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*, vol. 29, no. 6, pp. 3121–3144, 2023.
- [45] H. Voigt, Ö. Alaca, M. Meuschke, K. Lawonn, and S. Zarrieß, “The why and the how: A survey on natural language interaction in visualization,” in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022, pp. 348–374.
- [46] R. Chen, X. Shu, J. Chen, D. Weng, J. Tang, S. Fu, and Y. Wu, “Nebula: A coordinating grammar of graphics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 12, pp. 4127–4140, 2021.
- [47] Y. Sun, J. Leigh, A. Johnson, and S. Lee, “Articulate: A semi-automated model for translating natural language queries into meaningful visualizations,” in *Proceedings of the International Symposium on Smart Graphics*, 2010, pp. 184–195.
- [48] T. Gao, M. Dontcheva, E. Adar, Z. Liu, and K. G. Karahalios, “DataTone: Managing ambiguity in natural language interfaces for data visualization,” in *Proceedings of the Annual Symposium on User Interface Software and Technology*, 2015, pp. 489–500.
- [49] B. Yu and C. T. Silva, “FlowSense: A natural language interface for visual data exploration within a dataflow system,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1–11, 2019.
- [50] V. Setlur, S. E. Battersby, M. Tory, R. Gossweiler, and A. X. Chang, “Eviza: A natural language interface for visual analysis,” in *Proceedings of the Annual Symposium on User Interface Software and Technology*, 2016, pp. 365–377.
- [51] E. Hoque, V. Setlur, M. Tory, and I. Dykeman, “Applying pragmatics principles for interaction with visual analytics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 309–318, 2017.
- [52] OpenAI, “Gpt-4 technical report,” 2023.
- [53] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, u. Rodriguez, A. Joulin, E. Grave, and G. Lample, “LLaMA: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.
- [54] P. Maddigan and T. Susnjak, “Chat2VIS: Generating data visualisations via natural language using chatgpt, codex and gpt-3 large language models,” *IEEE Access*, vol. 11, pp. 45 181–45 193, 2023.
- [55] V. Dibia, “LIDA: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models,” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2023.
- [56] L. Cheng, X. Li, and L. Bing, “Is gpt-4 a good data analyst?” *arXiv preprint arXiv:2305.15038*, 2023.
- [57] W. Zhang, Y. Shen, W. Lu, and Y. Zhuang, “Data-Copilot: Bridging billions of data and humans with autonomous workflow,” *arXiv preprint arXiv:2306.07209*, 2023.
- [58] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou, “Chain of thought prompting elicits reasoning in large language models,” *arXiv preprint arXiv:2201.11903*, 2022.
- [59] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, E. Grave, Y. LeCun, and T. Scialom, “Augmented language models: a survey,” *arXiv preprint arXiv:2302.07842*, 2023.
- [60] E. H.-h. Chi, “A taxonomy of visualization techniques using the data state reference model,” in *Proceedings of IEEE Symposium on Information Visualization*, 2000, pp. 69–75.
- [61] M. Card, *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.
- [62] L. Wilkinson, “The grammar of graphics: The ggplot2 package,” in *Handbook of computational statistics*. Springer, 2012, pp. 375–414.
- [63] R. L. Harris, *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press, USA, 1999.

- [64] J. Mackinlay, "Automating the design of graphical presentations of relational information," *ACM Transactions on Graphics*, vol. 5, no. 2, pp. 110–141, 1986.
- [65] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, and M. Stonebraker, "Beagle: Automated extraction and interpretation of visualizations from the web," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–8.
- [66] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis, "SeeDB: Efficient data-driven visualization recommendations to support visual analytics," in *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, vol. 8, no. 13, 2015, p. 2182.
- [67] S. Fu, K. Xiong, X. Ge, S. Tang, W. Chen, and Y. Wu, "Quada: natural language queries for visual data analytics," *arXiv preprint arXiv:2005.03257*, 2020.
- [68] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2019.
- [69] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 27, 2014.
- [70] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.
- [71] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.
- [72] K. Yang, Y. Tian, N. Peng, and D. Klein, "Re3: Generating longer stories with recursive reprompting and revision," *arXiv preprint arXiv:2210.06774*, 2022.
- [73] K. Xiong, Z. Luo, S. Fu, Y. Wang, M. Xu, and Y. Wu, "Revealing the semantics of data wrangling scripts with comantics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 117–127, 2022.
- [74] K. Xiong, S. Fu, G. Ding, Z. Luo, R. Yu, W. Chen, H. Bao, and Y. Wu, "Visualizing the scripts of data wrangling with somnus," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 6, pp. 2950–2964, 2023.
- [75] R. Mitra, A. Narechania, A. Endert, and J. Stasko, "Facilitating conversational interaction in natural language interfaces for visualization," in *IEEE Visualization and Visual Analytics*, 2022, pp. 6–10.
- [76] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.
- [77] J. VanderPlas, B. Granger, J. Heer, D. Moritz, K. Wongsuphasawat, A. Satyanarayan, E. Lees, I. Timofeev, B. Welsh, and S. Sievert, "Altair: Interactive statistical visualizations for python," *Journal of Open Source Software*, vol. 3, no. 32, p. 1057, 2018.



Yuan Tian received her B.S. degree in computer science from Zhejiang University in 2022. She is currently a Ph.D. student in the State Key Lab of CAD&CG, Zhejiang University. Her research interests include machine learning for visualization and visual analytics.



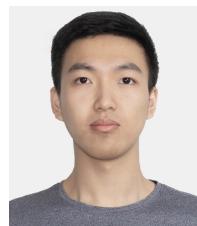
Weiwei Cui received the BS degree in computer science and technology from Tsinghua University, China, and the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology, Hong Kong. He is a principal researcher at Microsoft. His primary research interest is visualization, with the focuses on democratizing visualization and AI-assisted design. For more information, please visit <https://www.microsoft.com/en-us/research/people/weiweicu/>.



Dazhen Deng is currently a tenure-track assistant professor at the School of Software Technology, Zhejiang University. He received Ph.D. in Computer Science from Zhejiang University in 2023. His research interests mainly lie in machine learning for visual analytics. For more information, please visit <https://dengdazhen.github.io/>.



Xinjing Yi received her B.S degree in computer science from Wuhan University in 2022. She is currently a graduate student in Software Engineering, Zhejiang University. Her research interests mainly include visualization and visual analytics.



Yurun Yang received his B.S. degree in Software Engineering from University of Electronic Science and Technology of China in 2022. He is currently a graduate student in School of Software Technology, Zhejiang University. His research interests mainly include visualization and visual analytics.



Haidong Zhang received the PhD degree in Computer Science from Peking University, China. He is a Principal Architect at Microsoft. His research interests include visualization and human-computer interaction.



Yingcai Wu is a Professor at the State Key Lab of CAD&CG, Zhejiang University. His main research interests are in information visualization and visual analytics, with focuses on urban computing, sports science, immersive visualization, and narrative visualization. He received his Ph.D. degree in Computer Science from the Hong Kong University of Science and Technology. Prior to his current position, Dr. Wu was a postdoctoral researcher in the University of California, Davis from 2010 to 2012, a researcher in Microsoft Research Asia from 2012 to 2015, and a ZJU100 Young Professor at Zhejiang University from 2015 to 2020. For more information, please visit <http://www.ycwu.org>.