Analyzing the Sensitivity of Prompt Engineering Techniques in Natural Language Interfaces for 2.5D Software Visualization

Daniel Atzberger
daniel.atzberger@hpi.de
Hasso Plattner Institute
Digital Engineering Faculty
University of Potsdam
Potsdam, Brandenburg, Germany

Adrian Jobst
adrian.jobst@hpi.de
Hasso Plattner Institute
Digital Engineering Faculty
University of Potsdam
Potsdam, Brandenburg, Germany

Mariia Tytarenko
m.Tytarenko@cgv.tugraz.at
Graz University of Technology
Graz, Styria, Austria

Willy Scheibel
willy.scheibel@hpi.de
Hasso Plattner Institute
Digital Engineering Faculty
University of Potsdam
Potsdam, Brandenburg, Germany

Jürgen Döllner
juergen.doellner@hpi.de
Hasso Plattner Institute
Digital Engineering Faculty
University of Potsdam
Potsdam, Brandenburg, Germany

Tobias Schreck
tobias.schreck@cgv.tugraz.at
Graz University of Technology
Graz, Styria, Austria

Abstract

Natural Language Interfaces (NLIs) backed by Large Language Models (LLMs) are used to interact with visualizations through natural language queries. Using the specific example of 2.5D treemaps, the Delphi tool was recently presented, introducing an interactive 2.5D visualization with an accompanying chat interface, where the LLM can react to user input and adapt the visualization at its own discretion. While Delphi has demonstrated effectiveness, the authors have not included an evaluation of the LLM's performance with respect to its prompt and specific task types. In this study, we systematically evaluate the impact of prompt engineering on Delphi's ability to answer factual questions related to data and visualization. Specifically, we investigate the effect of the Chain-of-Thought prompting technique by employing a questionnaire comprising 40 questions across ten low-level analytic tasks. Our findings aim to refine prompt design methodologies and enhance the usability and effectiveness of NLIs in advanced visualization systems.

CCS Concepts

• **Human-centered computing → Natural language interfaces**; **Information visualization**; *Visualization techniques*; Empirical studies in visualization.

Keywords

Natural Language Interfaces, Chart Question Answering, Prompt Sensitivity, Chain-of-Thought Technique

ACM Reference Format:

Daniel Atzberger, Adrian Jobst, Mariia Tytarenko, Willy Scheibel, Jürgen Döllner, and Tobias Schreck. 2025. Analyzing the Sensitivity of Prompt Engineering Techniques in Natural Language Interfaces for 2.5D Software Visualization. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3701716.3717813

This work is licensed under a Creative Commons Attribution 4.0 International License.
WWW Companion '25, April 28-May 2, 2025, Sydney, NSW, Australia
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1331-6/2025/04
https://doi.org/10.1145/3701716.3717813

1 Introduction

Software visualizations represent software artifacts using geometric artifacts to support stakeholders in program comprehension tasks [8]. Among these visualizations, treemaps are a popular visualization technique, leveraging layouts that reflect the hierarchical structure of files within a software project and visual variables to convey aspects related to complexity and quality [17]. Extending traditional 2D treemaps using 3D cuboids enhances their expressive power by introducing additional visual variables and providing greater flexibility for stakeholders during visual exploration [12]. However, fully utilizing the potential of these 2.5D treemaps requires users to understand both the visualization and the domain.

To assist users in their exploration process, Jobst et al. introduced *Delphi*, an NLI extension for 2.5D treemaps, as shown in Figure 1 [10][1]. Delphi enables users to ask questions about the visualization through a *Natural Language Interface* (NLI), which are processed by an underlying *Large Language Model* (LLM). The LLM generates responses displayed to the user within the NLI and might initiate adjustments to the visual mapping. The prototype can be accessed via https://hpicgs.github.io/llm-treemaps/.

Delphi requires the visualization designer to write an instruction prompt, i.e., a textual description of the tasks and context. While Jobst et al. demonstrated Delphi's effectiveness using a fixed prompt, LLMs are known to be highly sensitive to prompt variations [3, 18, 26], and tailored prompting techniques can significantly enhance performance [14, 24]. In this study, we conduct a sensitivity analysis of Delphi's instruction prompt, focusing on its ability to answer factual questions about the data and visualization. Specifically, we examine the impact of the *Chain-of-Thought* prompting technique, which encourages step-by-step reasoning in LLM responses [23]. To evaluate this, we propose a questionnaire comprising 40 questions spanning ten low-level analytics

[1]author's version:
https://drive.google.com/file/d/18rIDAQiBx_gw7SQYkV7aR5bCywNiKMrf/view?usp=sharing

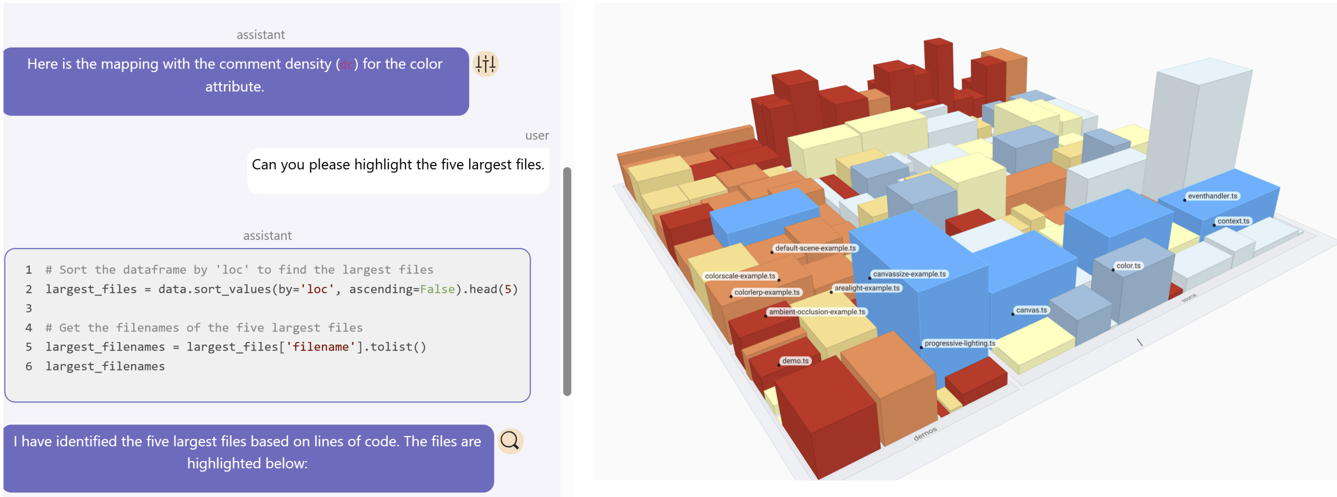

Figure 1: Illustration of Delphi. (Right) A 2.5D treemap visualizing the *webgl-operate* (https://github.com/cginternals/webgl-operate) project from GitHub, a TypeScript-based WebGL rendering framework. The height of each cuboid represents the number of functions in a file, and color indicates comment density. The five largest files by lines of code are highlighted. (Left) The NLI interface displays the textual interaction between the user and the system, where the user queries the LLM to highlight the five largest files.

tasks, each requiring an understanding of visual mappings and data column semantics.

2 Related Work

Shen et al. define NLIs as systems that "interpret a user's natural language queries as input and output appropriate visualizations" [19]. In this section, we review related work that aligns with this definition, with a particular emphasis on systems leveraging LLMs. Additionally, we discuss existing studies that examine the sensitivity of prompts and their impact on system performance.

2.1 Natural Language Interfaces

Shen et al.'s definition of NLIs includes not only components embedded within a user interface (UI), but also methods that generate visualizations from natural language inputs and *Chart Question-Answering* (CQA) systems, i.e., techniques that "take a chart and a natural language question as input and automatically generate the answer to facilitate visual data analysis" [9]. Choe et al. developed a system, that is conceptually comparable to Delphi, designed to support visualization novices in learning visualization techniques by enabling modifications. In both cases the visualizations are predesigned by a visualization expert and the underlying LLM might adapt to the user's question [5].

Since LLMs can generate source code based on natural language specifications, by integrating libraries such as *matplotlib*, they can also create visualizations without a pre-given design [4, 21]. However, studies have highlighted weaknesses in these visualizations, including deviations from established design guidelines [22]. An alternative approach is to divide the visualization generation process into multiple sub-steps. For example, *LIDA* first generates a summary of the given dataset, proposes a list of potential questions, and then produces the source code for graphical representations based on these questions [7]. Similar pipeline-based approaches have been proposed by Tian et al. [20] and Cui et al [6].

Previous research on the use of LLMs for visualizations has largely overlooked the impact of prompt design on the outcomes. The generation of prompts has typically relied on experimental trial-and-error approaches, thus lacking a systematic evaluation framework. In this study, we address this gap by systematically examining the influence of various prompt engineering techniques on the results. Our goal is to propose a generalizable framework for evaluating and improving prompt effectiveness in the context of visualization tasks.

2.2 Prompt Sensitivity Analysis

LLMs excel in natural language understanding and generation but exhibit high sensitivity to their input prompts. Sclar et al. demonstrated that prompt formatting significantly affects LLM performance in downstream tasks, with this influence varying across models [18]. To enable fair comparisons, they developed an evaluation framework that systematically explores and assesses different prompt formats. Similar frameworks have been employed by Zhuo et al.[26] and Chatterjee et al.[3]. Another strategy for improving output quality involves employing dedicated prompt engineering techniques. For example, Lu et al. showed that example-based prompts allow LLMs to outperform specialized fine-tuned models, even without additional fine-tuning [14].

Wu et al. introduced a similar method called *Chain-of-Charts* which provides LLMs with examples of questions paired with their corresponding answers[24]. Based on a benchmark comprising over 22,000 question-answer pairs, the authors showed that this prompting technique improves the performance of multimodal LLMs in low-level analytics tasks for CQA. Our methodology builds on the

```
CONTEXT
You are the backbone of a visual analytics application. You use a knowledge
base to answer analytics-related questions and control part of the visual
analytics app if necessary. The analytics platform uses a treemap to visualize
hierarchical data.

Your knowledge base is software analytic data of Git repositories.
Data is stored in csv files which have the following columns:
- filename: name of the file
- loc: lines of code
- noc: number of comments (comment blocks)
- cloc: number of comment lines
- dc: comment density; ratio of comment lines to all lines
- nof: number of functions

TASKS
As you are the backbone of the visual analytics application, you mainly do two
things. You provide explanations for human users, and you control parts of the
application, mainly a treemap visualization. That we can use your responses
properly, your response for controlling the application has to be valid JSON
format. You append the json at the end of your user message as a separate
message. There should be no sign that a message contains a configuration object,
for instance never use wording like "Here is the configuration for ...". Just
use JSON for easier parsing at the end of the message.

[ Explain your answer stepwise, i.e., apply the Chain-of-Thought technique. ]

Here is more information about your core functionality:
1: You answer analytic related questions about the provided knowledge base and
provide reasoning about the actions you take when you control the app. Keep
your answers as brief as possible, also don't use too much text styling.
2: You create the visual mapping of the data columns for the treemap
visualization. The treemap uses three visual attributes. The area of a bar, the
height of a bar and the color of a bar. Per default the treemap displays the
number of lines of code (loc) as area. You can choose the mapping of the other
two visual attributes based on what you think makes most sense, or on what the
user specifies. To speed up the user interaction, you never ask for confirmation
when you create a mapping. The mapping object will configure the treemap
component of the system, therefore it will be in the JSON response object. The
format is either { mapping: { height: columnName, colors: columnName } } or {
mapping: null }.
3: Whenever appropriate, you can highlight single or multiple columns. A column
represents a single file in the knowledge base. When you want to highlight a
column, you respond with the "filename" of the item in the knowledge base. So
the format is either { highlight: [filename] } or { highlight: null }.
```

Figure 2: Instruction Prompt of Delphi taken from Jobst et al. [10]. In case, where the Chain-of-Thought technique is integrated the sentence "Explain your answer stepwise, i.e., apply the Chain-of-Thought technique" is added at the end of the first paragraph of the tasks section.

work of Wu et al. but focuses specifically on a single visualization type with known data and a predefined visualization specification.

3 Prompt Sensitivity Analysis

Jobst et al. showcased the advantages of incorporating an NLI into their 2.5D treemap visualization by employing a variety of questions. In this study, we evaluate how the prompt influences Delphi's visualization literacy, defined as its "ability and skill to read and interpret visually represented data and to extract information from data visualizations" [11]. Throughout our experiments, we rely on the GPT-4o model and the visualization setting shown in Figure 1.

3.1 Prompt Design

Delphi requires the visualization designer to provide an instruction prompt. The original prompt from Jobst et al. is depicted in Figure 2. It begins with a description of the application context and an explanation of the dataset variables, which are stored as columns. The prompt leverages the *persona-adoption technique*, instructing the LLM to adopt a specific persona to guide its role [15]. Following this, the instruction prompt outlines the tasks and details the methods for responding to queries. These methods include providing textual answers, modifying the visual mapping, or highlighting specific objects. To support this, the visualization designer must explain the available visual variables—in our case, the height and color of the cuboids (with the area reserved for the lines of code).

Finally, the instruction prompt specifies the required output format, which is JSON. To introduce variation, we incorporate the *chain-of-thought technique* into the prompt to analyze its impact. This technique encourages the LLM to provide a step-by-step explanation of its reasoning process, enabling a more structured and transparent approach to answering questions [23]. By comparing the results with and without this technique, we aim to evaluate its influence on the LLM's performance and visualization literacy.

3.2 Quantifying Delphi's Visualization Literacy

Various methods have been developed to measure an individual's visualization literacy, with the most notable being the *Visualization Literacy Assessment Test* (VLAT) [11] and its simplified subset, the Mini-VLAT [16]. The VLAT consists of 52 questions in 12 two-dimensional chart types, each question targeting a specific low-level analytical task. Bendeck and Stasko applied the VLAT to assess the visualization literacy of multimodal LLMs [2]. However, in our case of a fixed visualization, the VLAT is not applicable. Additionally, the VLAT comprises only non-visual questions that do not require the LLM to interpret the visual mapping; instead, it focuses solely on understanding the meaning of the data columns. In our study, we build on the approaches of Xu and Wall [25] and Wu et al. [24], whose evaluations encompassed ten low-level analytical tasks outlined in Amar et al.'s taxonomy [1]. Additionally, we include both visual and non-visual questions to provide a more comprehensive assessment. Overall, our questionnaire consists of 40 questions, as detailed in Table 1. According to Hoque's taxonomy of CQA tasks, our questionnaire includes factual questions that are both visual and non-visual, encompassing both simple and compositional types [9].

3.3 Results

The results of our experiment are presented in Table 1. We manually input the questions into Delphi and verified the accuracy of the answers either by fact-checking (e.g., for value retrieval) or by subjective judgment (e.g., for clustering tasks). The answers were marked as either correct (✓) or incorrect (✗). In some cases, the two prompts produced different outputs that were both considered correct (✓).

In the case of the baseline prompt, in no instance were both a visual question and its non-visual counterpart answered incorrectly. Question 9 was marked incorrect because its approximation of the value was less accurate compared to question 11. Similarly, question 19 was deemed incorrect because the answer failed to mention the endpoint of the sorted list. For question 20, the LLM erroneously stated that ordering based on color made no sense, which is incorrect in our context and was correctly addressed in question 18. Questions 23 and 24 only identified the minimum and maximum values without providing the range, resulting in them being marked as incorrect. Additionally, the answers to questions

Table 1: In our evaluation of Delphi's visualization literacy, we consider several questions. Each question corresponds to a specific low-level analytic task and may or may not include a reference to the visual mapping. The answers were marked as correct (✓) or incorrect (✗), with some cases where both prompts produced differing but correct outputs (✓).

ID	Task Type	Visual (V)/Non-Visual(NV)	Question / Stem	Baseline	Chain-of-Thought
1	Retrieve Value	NV	"What is the number of comments of the file cornellbox.ts?"	✓	✓
2	Retrieve Value	NV	"What is the number of lines of code of the file cornellbox.ts?"	✓	✓
3	Retrieve Value	V	"What is the size of the base area of the cuboid representing the file cornellbox.ts?"	✓	✓
4	Retrieve Value	V	"What is the height of the cuboid representing the file cornellbox.ts?"	✓	✓
5	Filter	NV	"How many files have more than 400 lines of code?"	✓	✓
6	Filter	NV	"How many files have a comment density larger than 10 percent?"	✓	✓
7	Filter	V	"How many cuboids have a base area larger than 400?"	✓	✓
8	Filter	V	"How many cuboids have a hight larger than 20?"	✓	✓
9	Compute Derived Value	NV	"What is the average size of source code files?"	✗	✗
10	Compute Derived Value	NV	"What is the average number of functions?"	✓	✓
11	Compute Derived Value	V	"What is the average area of the bases of the cuboids?"	✓	✗
12	Compute Derived Value	V	"What is the average height of the cuboids?"	✓	✓
13	Find Extremum	NV	"What is the largest file?"	✓	✓
14	Find Extremum	NV	"Which file has the most functions?"	✓	✓
15	Find Extremum	V	"What cuboid has the largest base area?"	✓	✓
16	Find Extremum	V	"What cuboid has the highest height?"	✓	✓
17	Sort	NV	"Sort the files according to their lines of code in descending order."	✓	✗
18	Sort	NV	"Can you order the files according to their comment density?"	✓	✗
19	Sort	V	"Sort the cuboids according to their sizes of their bases area in descending order."	✗	✗
20	Sort	V	"Can you sort the cuboids according to their color?"	✗	✗
21	Determine Range	NV	"What is the range in the lines of code?"	✓	✓
22	Determine Range	NV	"Describe the range of the number of functions."	✓	✓
23	Determine Range	V	"What is the range in the bases area?"	✗	✓
24	Determine Range	V	"Describe the range in the height of the cuboids."	✗	✓
25	Characterize Distribution	NV	"How would you characterize the distribution of the lines of code?"	✓	✓
26	Characterize Distribution	NV	"Please describe the distribution of the number of functions."	✓	✓
27	Characterize Distribution	V	"How would you characterize the sizes of the bases of the cuboids?"	✗	✓
28	Characterize Distribution	V	"Please describe the distribution of height of the cuboids."	✗	✓
29	Find Anomalies	NV	"Are there anomalies in the number of lines of code among the files?"	✓	✓
30	Find Anomalies	NV	"Are there files with an unusual comment density?"	✓	✓
31	Find Anomalies	V	"Are there anomalies in the sizes of the base areas of the cuboids?"	✓	✓
32	Find Anomalies	V	"Are there anomalies in the colors of the cuboids?"	✓	✓
33	Cluster	NV	"Are there clusters within the files?"	✓	✓
34	Cluster	NV	"Are there groups of files with similar characteristics?"	✓	✓
35	Cluster	V	"Are there clusters within the cuboids, i.e., do their shapes seem to be similar?"	✓	✓
36	Cluster	V	"Are there groups of cuboids with similar geometric characteristics?"	✓	✓
37	Correlate	NV	"Is there a strong positive correlation between the size and the number of functions?"	✓	✓
38	Correlate	NV	"Is there a strong positive correlation between the size and the comment density?"	✓	✓
39	Correlate	V	"Is there a strong positive correlation between the base area and the height of the cuboids?"	✓	✓
40	Correlate	V	"Is there a strong positive correlation between the base area and the color of the cuboids?"	✓	✓

27 and 28 were less detailed compared to those for questions 25 and 26, leading to the same conclusion.

In cases where the Chain-of-Thought technique was applied, questions 9 and 11 produced approximated results, similar to question 9 when using the baseline prompt. We marked all answers for the task sort as incorrect because the LLM failed to provide the minimum.

For the tasks *Characterize Distribution*, *Find Anomalies*, and *Cluster*, the two prompts led to different definitions and algorithms applied by the LLM. In all cases, the LLM generated Python code that was executed, with the results displayed to the user. For Characterize Distribution, the LLM provided a textual explanation with the baseline prompt, whereas it produced a highly structured output with the Chain-of-Thought prompt. Additionally, the LLM defined anomalies differently in the two cases: either as points differing by 2 standard deviations from the mean or by 1.5 standard deviations. For the Cluster task, the LLM employed two distinct clustering algorithms, resulting in different outputs.

4 Discussion

From the results of our evaluation, we derive our main findings. However, our evaluation is subject to threats to validity.

4.1 Main Findings

Notably, the LLM demonstrated an ability to understand visual mappings, as seen in the Clustering task, where it referenced related non-visual answers. However, the outputs were highly sensitive to prompt variations, with only one additional sentence leading to different definitions and algorithms. While the Chain-of-Thought technique produced distinct results, it offered no consistent advantage. Overall, crafting effective instruction prompts remains a trial-and-error process.

4.2 Threats to Validity

We identified two major threats to validity in our experiments. First, due to repeated server connection failures, we had to restart Delphi multiple times and reintroduce the visual mapping. Instead of repeating all prior questions, we resumed from the point of interruption. Since LLMs maintain context within a single chat, this loss of query history may have influenced the results. Second, we fixed the temperature parameter throughout our experiments. The temperature setting controls the randomness of an LLM's output: lower values yield more deterministic responses, while higher values increase variability. Previous studies have shown that prompt sensitivity can be affected by variations in this parameter [13]. By keeping the temperature constant, we reduced randomness and focused on analyzing prompt-related sensitivity more effectively.

5 Conclusions & Future Work

Delphi combines an LLM-backed NLI with a 2.5D treemap for software visualization. Through its NLI, users can interact with the visualization using text, significantly enhancing its accessibility. However, the visualization designer must provide the LLM with context in the form of an instruction prompt. This study investigates the sensitivity of Delphi's visualization literacy to variations in its instruction prompt. We posed 40 questions derived from 10 analytics tasks, encompassing both visual and non-visual queries. To explore the effect of different prompts, we compared the baseline instruction with a second prompt, enhanced using the Chain-of-Thought technique. Our findings reveal that Delphi's outputs are sensitive to changes in the prompt. Delphi showed an understanding of the visual mapping despite the absence of a graphical representation. The Chain-of-Thought technique did not offer measurable benefits.

As future work, we aim to develop a comprehensive benchmark based on the methodology used in this study to evaluate NLIs for visualization literacy. Our work provides a foundational framework for assessing the visualization literacy of LLMs and NLIs, contributing to the advancement of accessible and intelligent user interfaces.

Acknowledgments

This work is part of the "SPUR-OPT" project (grant 16KN113522) funded by the Federal Ministry for Economic Affairs and Climate Action of Germany. The work of Mariia Tytarenko was funded by the FWF as part of the project 'Human-Centered Interactive Adaptive Visual Approaches in High-Quality Health Information' (A^+CHIS; Grant No. FG 11-B).

References

[1] R. Amar, J. Eagan, and J. Stasko. 2005. Low-level components of analytic activity in information visualization. In *2005 IEEE Symposium on Information Visualization (INFOVIS '05)*. IEEE, 111–117. https://doi.org/10.1109/INFVIS.2005.1532136

[2] Alexander Bendeck and John Stasko. 2025. An Empirical Evaluation of the GPT-4 Multimodal Language Model on Visualization Literacy Tasks. *IEEE Transactions on Visualization and Computer Graphics* 31, 1 (2025), 1105–1115. https://doi.org/10.1109/TVCG.2024.3456155

[3] Anwoy Chatterjee, H S V N S Kowndinya Renduchintala, Sumit Bhatia, and Tanmoy Chakraborty. 2024. POSIX: A Prompt Sensitivity Index For Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. ACL, 14550–14565. https://doi.org/10.18653/v1/2024.findings-emnlp.852

[4] Nan Chen, Yuge Zhang, Jiahang Xu, Kan Ren, and Yuqing Yang. 2025. VisEval: A Benchmark for Data Visualization in the Era of Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* 31, 1 (2025), 1301–1311. https://doi.org/10.1109/TVCG.2024.3456320

[5] Kiroong Choe, Chaerin Lee, Soohyun Lee, Jiwon Song, Aeri Cho, Nam Wook Kim, and Jinwook Seo. 2024. Enhancing Data Literacy On-demand: LLMs as Guides for Novices in Chart Interpretation. *IEEE Transactions on Visualization and Computer Graphics* (2024), 17 pages. https://doi.org/10.1109/TVCG.2024.3413195 Early Access.

[6] Yuan Cui, Lily W. Ge, Yiren Ding, Lane Harrison, Fumeng Yang, and Matthew Kay. 2024. Promises and Pitfalls: Using Large Language Models to Generate Visualization Items. *IEEE Transactions on Visualization and Computer Graphics* (2024), 11 pages. https://doi.org/10.1109/TVCG.2024.3456309 Early Access.

[7] Victor Dibia. 2023. LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models. In *Proc. 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. ACL, 113–126. https://doi.org/10.18653/v1/2023.acl-demo.11

[8] Stephan Diehl. 2007. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer Science & Business Media. https://doi.org/10.1007/978-3-540-46505-8

[9] Enamul Hoque, Parsa Kavehzadeh, and Ahmed Masry. 2022. Chart question answering: State of the art and future directions. *EG Computer Graphics Forum* 41, 3 (2022), 555–572. https://doi.org/10.1111/cgf.14573

[10] Adrian Jobst, Daniel Atzberger, Willy Scheibel, Jürgen Döllner, and Tobias Schreck. 2025. Delphi: A Natural Language Interface for 2.5D Treemap Visualization of Source Code. In *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications – Volume 1: GRAPP, HUCAPP and IVAPP (IVAPP '25)*. INSTICC, SciTePress. in press.

[11] Sukwon Lee, Sung-Hee Kim, and Bum Chul Kwon. 2017. VLAT: Development of a Visualization Literacy Assessment Test. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 551–560. https://doi.org/10.1109/TVCG.2016.2598920

[12] Daniel Limberger, Willy Scheibel, Jürgen Döllner, and Matthias Trapp. 2022. Visual Variables and Configuration of Software Maps. *Springer Journal of Visualization* 26 (2022), 249–274. https://doi.org/10.1007/s12650-022-00868-1

[13] Manikanta Loya, Divya Sinha, and Richard Futrell. 2023. Exploring the Sensitivity of LLMs' Decision-Making Capabilities: Insights from Prompt Variations and Hyperparameters. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. ACL, 3711–3716. https://doi.org/10.18653/v1/2023.findings-emnlp.241

[14] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. ACL, 8086–8098. https://doi.org/10.18653/v1/2022.acl-long.556

[15] OpenAI. 2024. Prompt Engineering. URL: https://platform.openai.com/docs/guides/prompt-engineering.

[16] Saugat Pandey and Alvitta Ottley. 2023. Mini-VLAT: A Short and Effective Measure of Visualization Literacy. *EG Computer Graphics Forum* 42, 3 (2023), 1–11. https://doi.org/10.1111/cgf.14809

[17] Willy Scheibel, Matthias Trapp, Daniel Limberger, and Jürgen Döllner. 2020. A Taxonomy of Treemap Visualization Techniques. In *Proc. 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications – Volume 3: IVAPP (IVAPP '20)*. INSTICC, SciTePress, 273–280. https://doi.org/10.5220/0009153902730280

[18] Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324* (2023).

[19] Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2023. Towards Natural Language Interfaces for Data Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics* 29, 6 (2023), 3121–3144. https://doi.org/10.1109/TVCG.2022.3148007

[20] Yuan Tian, Weiwei Cui, Dazhen Deng, Xinjing Yi, Yurun Yang, Haidong Zhang, and Yingcai Wu. 2024. ChartGPT: Leveraging LLMs to Generate Charts from Abstract Natural Language. *IEEE Transactions on Visualization and Computer Graphics* (2024), 15 pages. https://doi.org/10.1109/TVCG.2024.3368621 Early Access.

[21] Pere-Pau Vázquez. 2024. Are LLMs ready for Visualization?. In *2024 IEEE 17th Pacific Visualization Conference (PacificVis '24)*. IEEE, 343–352. https://doi.org/10.1109/PacificVis60374.2024.00049

[22] Huichen Will Wang, Mitchell Gordon, Leilani Battle, and Jeffrey Heer. 2025. DracoGPT: Extracting Visualization Design Preferences from Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* 31 (2025), 710–720. https://doi.org/10.1109/TVCG.2024.3456350

[23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS '22, Vol. 35)*. Curran Associates, Inc., 24824–24837.

[24] Yifan Wu, Lutao Yan, Leixian Shen, Yunhai Wang, Nan Tang, and Yuyu Luo. 2024. ChartInsights: Evaluating Multimodal Large Language Models for Low-Level Chart Question Answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. ACL, 12174–12200. https://doi.org/10.18653/v1/2024.findings-emnlp.710

[25] Zhongzheng Xu and Emily Wall. 2024. Exploring the Capability of LLMs in Performing Low-Level Visual Analytic Tasks on SVG Data Visualizations. *arXiv preprint arXiv:2404.19097* (2024).

[26] Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. ProSA: Assessing and Understanding the Prompt Sensitivity of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. ACL, 1950–1976. https://doi.org/10.18653/v1/2024.findings-emnlp.108