# Integrating Natural Language Interfaces into Data Visualizations with Trustworthiness Scores

Adrian Jobst* Hasso Plattner Institute, University of Potsdam
Daniel Atzberger† Hasso Plattner Institute, University of Potsdam
Willy Scheibel‡ Hasso Plattner Institute, University of Potsdam
Jürgen Döllner§ Hasso Plattner Institute, University of Potsdam
Tobias Schreck¶ Graz University of Technology

## Abstract

We present a framework for integrating Natural Language Interfaces (NLIs) backed by Large Language Models (LLMs) into data visualizations to enhance accessibility for users with limited visualization literacy. Given a visualization specified in a structured JSON schema and its underlying data, the framework enables users to interact through natural language queries. The LLM interprets the query, decides whether to respond in text or modify the visualization by updating the specification, and generates answers by writing and executing code that analyzes the data. To enhance the reliability, the framework assigns a trustworthiness score to each response, derived from the LLM's performance on comparable tasks from the revised Visualization Literacy Assessment Test (Mini-VLAT). Our implementation supports both OpenAI and local models. We further discuss the pros and cons of these two alternatives concerning their use in NLIs. Our framework is intended for visualization designers working with charts of moderate complexity, such as those used in data journalism or embedded in websites. The system is available on GitHub at https://github.com/hpicgs/lumos-trustworthiness.

**Index Terms:** Natural Language Interfaces, Chart Question Answering, Visualization Literacy, Large Language Models.

## 1 Introduction

User interfaces and interaction design for visualizations often follow established principles, such as the *Visual Information-Seeking Mantra*: *"overview first, zoom and filter, then details-on-demand"* [32]. *Natural Language Interfaces* (NLIs) provide a complementary approach to interacting with visualizations, providing an intuitive alternative in particular for users with limited visualization literacy that can better express their need using natural language [9, 12]. Recently, *Large Language Models* (LLMs) have become popular as backbones of NLIs, as they enable seamless integration without the need to specify rules, and further provide additional functionalities, such as logical reasoning, factual knowledge, and the capability to perform computations by writing and executing code [15].

This work presents a software framework that enables users to integrate an LLM-backed NLI into their visualizations. Figure 1 illustrates an example of a visualization extended by our framework. Our system is conceptualized for scenarios where a visualization of limited complexity has been created for diverse audiences with varying levels of visualization literacy. Typical use cases include public agencies reporting on disease outbreaks [3], news organizations covering political or economic trends [2], and NGOs communicating initiatives [7]. Building on the concept introduced by Jobst et al. [18], our framework requires the visualization designer to provide two files: (1) the dataset underlying the visualization, and (2) an instruction prompt for the LLM that embeds the visualization specification together with task descriptions and context. Given a user's query, the LLM can generate Python code that analyzes the dataset and executes it to derive an answer, which is then displayed within the user interface. Additionally, the LLM can modify the visualization specification to present the answer visually.

A limitation of LLMs is that they are prone to *hallucination*, i.e., *"[...] instances where the AI system generates factually incorrect, irrelevant, or nonsensical responses"* [19]. To mitigate this, our framework incorporates a *trustworthiness score* that quantifies LLM performance across a set of nine analytics tasks (e.g., determining value ranges, finding extrema, deriving values). When a user submits a query, the LLM first classifies it into one of these nine task types and then provides both the answer and its corresponding trustworthiness score directly in the user interface. The trustworthiness score is derived from the LLM's historical performance on the *Visualization Literacy Assessment Test* (VLAT) [21]. In summary, we make the following contributions:

1. A framework for integrating LLM-based NLIs in visualizations with available input data and specifications. Our system supports both local LLMs and cloud-based models from OpenAI. The framework can be accessed on GitHub: https://github.com/hpicgs/lumos-trustworthiness.
2. The introduction of a trustworthiness score that quantifies the reliability of the output based on historical results on a visualization literacy assessment test. In particular our test can be used in further studies.
3. A discussion on the performance of local LLMs and GPT models based on our own set of experiments.

## 2 Related Work

*Chart Question Answering* (CQA) systems, as defined by Hoque et al., *"[. . . ] take a chart and a natural language question as input and automatically generate the answer to facilitate visual data analysis"* [13]. Within this broader category, NLIs are systems that *"[. . . ] interpret a user's natural language queries as input and output appropriate visualizations"* [31]. This definition encompasses a range of techniques, particularly those that generate visualizations from scratch, e.g., by writing code and executing it. For instance, Dibia proposed a three-stage pipeline that incrementally produces visualizations using libraries such as *Matplotlib* [8]. Tian et al. proposed a similar pipeline for generating visualizations using GPT models [36]. Although LLMs already show good performance in generating visualizations, their outputs usually require manual editing by the designer and are limited to basic visualizations without customized optics [34, 38].

Our work addresses a different scenario, in which the visualization is already provided rather than generated from scratch. In this setting, the NLI functions as part of the UI, enabling the user to pose questions about the existing visualization. For this use case, *Multimodal Large Language Models* (MLLMs) have been applied as CQA systems [1, 11, 23, 41, 42]. MLLMs possess built-in capabilities for extracting visual features from charts and responding to

*e-mail: adrian.jobst@hpi.uni-potsdam.de
†e-mail: daniel.atzberger@hpi.uni-potsdam.de
‡e-mail: willy.scheibel@hpi.uni-potsdam.de
§e-mail: doellner@uni-potsdam.de
¶e-mail: tobias.schreck@cgv.tugraz.at

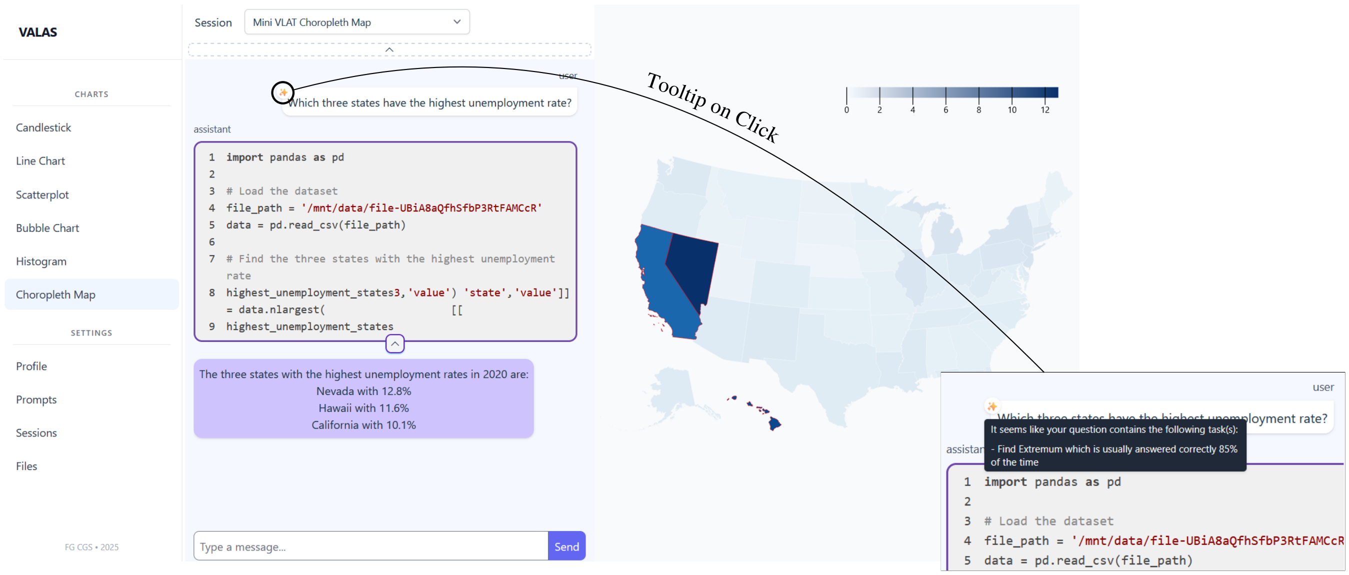


Figure 1: Illustration of our NLI integration. Users can query the choropleth map visualization of U.S. state unemployment rates (right) directly through the chat interface (middle). In this example, the LLM identifies the states with the highest unemployment rate by generating and executing Python code, then returning the result. The three states are automatically highlighted in the visualization. By clicking the star icon, the user can view the model's performance score for the Find Extremum task.

related questions. However, they currently cannot modify the visualization and often produce results of insufficient quality, limiting reliable deployment.

To overcome the limitations of MLLM-based approaches, Bursztyn et al. proposed using a formal specification of visualizations rather than image-based inputs [4]. In a study comparing the performance of LLMs and MLLMs in explanation generation and question answering, their method demonstrated competitive results. Narechania et al. introduced *NL4DV*, a system that interprets natural language queries by extracting relevant data attributes and analytic tasks, and then generates a corresponding *Vega-Lite* specification [27]. Choe et al. introduced a system that also relies on a visualization specification and the underlying data, enabling it to modify the visual output to help users learn new visualizations [6]. Similarly, Jobst et al. adopted this architecture for a 2.5D treemap in the software visualization domain [17], and later proposed a concept for an architecture to support visualization grammars in general [18]. Our system builds on this line of work and requires the visualization designer to provide a visualization specification as well as the underlying data.

Commercial systems also integrate NLIs into their visualization environments [10, 20, 24–26, 30, 33, 35, 37]. While the exact integration methods are not disclosed, the limited flexibility in visualization options and reliance on predefined selection possibilities suggest an approach similar to ours. Such restrictions likely serve to reduce errors and ensure reliability, aligning with prior observations about constraining model outputs in visualization generation.

## 3 Concept

Our system enables textual and visual outputs, e.g., highlighting areas of interest, covering the full spectrum of CQA tasks in the case of given data [13]. To foster user trust, we introduce a trustworthiness score derived from historical performance.

### 3.1 Unified API & System Architecture

We expose a provider-agnostic chat API that routes requests to *OpenAI* or local *Ollama* models. If supported by the underlying model, a code interpreter can execute Python code to analyze uploaded datasets. The UI is responsible for parsing control information of the response message and apply visualization updates, and displays answers with their trustworthiness score.

### 3.2 Instruction Prompt & Visualization Specification

The instruction prompt consists of three main components: (1) Overall context, (2) Task classification with integration of the trustworthiness score, and (3) Output specification with a JSON schema. The instruction prompt for our choropleth map example is shown in Figure 2 and Fig. 3. The first component establishes the overall context for the LLM. It explains the underlying dataset columns and, in our example, also provides a sample data point. Additionally, the visual mapping is described to help the LLM understand how data attributes correspond to visual elements. Questions are categorized into nine task types defined by the VLAT. Instead of employing a dedicated text classifier, we rely on the LLM to perform this downstream classification. We provide classification examples using the *few-shot learning* prompting technique to support this process [39]. Finally, the JSON schema enforces a consistent response structure, specifying both visual mappings and task classification.

### 3.3 Trustworthiness Score

Standardized tests such as the VLAT [21] and its shortened version, the *Mini-VLAT* [28] assess visualization literacy, i.e., *"[...] the ability and skill to read and interpret visually represented data and to extract information from data visualizations"*. These multiple-choice tests cover 12 chart types and tasks (e.g., retrieving values, finding extrema, determining ranges). The VLAT variants have also been applied to LLMs [1, 11, 23, 29]. To help users quantify the reliability of an LLM's response, the system displays the VLAT performance associated with the relevant analytical task alongside each answer, as illustrated in Figure 1.

To benchmark our metric, we used the Mini-VLAT, as the full VLAT dataset is unavailable. We extended the Mini-VLAT[1] by constructing additional items to match the original test length, yielding an "alternative VLAT" of 53 questions. GPT-4o was evaluated on these items, with each question repeated three times to reduce variance [1] using the evaluation framework of Jobst et

[1] https://osf.io/46rt8/files/osfstorage and GitHub [40]

**Overall Task Section**
```
You are the reasoning engine behind a natural language
interface for a visualization system that displays a
choropleth map. You are provided with a CSV file named
'Mini-VLAT-ChoroplethDataset.csv', which is visualized
as a choropleth map. Your task is to interpret user
questions, analyze the chart and its underlying data,
and respond with accurate, concise answers. You may
also output control information that affects how the
chart is rendered or interpreted.
```

**Contextual Priming**
```
The dataset 'Mini-VLAT-ChoroplethDataset.csv' contains
the unemployment rate for states in the US for the
year 2020. It has four columns: 'state', 'value',
'code' and 'id'. For example, the unemployment rate for
Nebraska in 2020 was 4.2.
```

**Visualization Section**
```
The chart is a choropleth map configured as follows:
- The color scale represents 'value'.
```

**Output Section**
```
Respond to user questions with clear, concise answers
grounded in the chart and data. Do not fabricate
information. Only respond based on what can be inferred
from the chart or data.
```

**Task Classification Section**
```
You must classify every user message into one or more of
the following predefined visualization reasoning tasks:
- Retrieve Value
- Find Extremum
- Determine Range
- Characterize Distribution
- Find Anomalies
- Find Clusters
- Find Correlations/Trends
- Make Comparisons
- Identify the Hierarchical Structure
If the message does not match any of these task types,
assign "None".
```

**Classification Output Format**
```
Include the classification in the "task_classification"
field of the control information JSON object. It must
be an array of one or more task names, even if only one
is assigned.
- Always classify every message.
- Do not explain or justify your classification.
- The classification must appear only in the JSON block,
  using valid JSON syntax.
```

**Reasoning Guidelines**
```
- Classify based on semantic intent, not keyword
  matching.
- Multiple tasks may apply to a single message.
- Use reasoning about the user's goals and expectations.
- Always output valid JSON.
- Do not add markdown, comments, or explanations around
  the JSON.
```

**Classification Examples**

| User Message | Task Classification |
|---|---|
| What was the price of oil in May 2020? | ["Retrieve Value"] |
| When was the oil price highest in 2020? | ["Find Extremum"] |
| What was the range of oil prices in 2020? | ["Determine Range"] |
| How much did the oil price fall between March and April? | ["Make Comparisons"] |
| Which month looks like an outlier in the trend? | ["Find Anomalies"] |
| Did oil prices form any natural clusters? | ["Find Clusters"] |
| Can I export the data file? | ["None"] |

Figure 2: Part of the prompt which contains the general system description, visualization and dataset information as well as the task classification description with few shot learning examples.

al. [16]. Results were then aggregated by task type. These scores form the trustworthiness values reported by our system.

## 4 Implementation

The framework is implemented as a client–server application. The backend is built with *Node.js* and the *Express* framework, using *Mongoose* to manage connections to a *MongoDB* database. It encapsulates the API layer, which provides endpoints for uploading datasets and prompts, creating chat sessions, and generating message responses. The frontend is implemented as a *React*-based web application. It provides a reusable set of components, including forms for session creation and a chat controller that manages interactions with the language models.

Integration with LLMs is realized through a unified API layer. This layer defines a general interface and common data structures for chat interactions and internally translates requests to the respective provider APIs, e.g., OpenAI or Ollama. Currently, only a session-based chat interface is supported. For Ollama models, the full conversation history is persisted, whereas for OpenAI models only the session parameters are stored and the dialogue text is retrieved live from the API to avoid duplication. Prompts are managed as persistent entities in the system. They must be uploaded in advance and can then be used in chat sessions. While not yet modular, this design ensures consistency across sessions and simplifies evaluation of different prompt configurations. Since Ollama does not provide built-in code execution, we integrate the `llm sandbox`[2] project. This component executes Python code in an isolated *Docker* container with controlled access to the required data files. Code execution is only available for Ollama models that support tool calls. This mechanism enables the LLM to perform data analysis tasks securely and return computational results alongside textual or visual outputs. The NLI allows the LLM to modify the visualization by generating an updated specification according to the schema given in the prompt seen in Figure 3. This specification is returned as part of the model's response, extracted and parsed by the session chat controller in the frontend, and passed to parent components via callback. The parent component then decides whether to update the visualization or provide alternative feedback.

The system is publicly available on GitHub as a Docker Compose setup. An automated initialization routine configures all required artifacts, including Mini-VLAT resources and sample prompts, enabling a minimal working deployment with a single command.

## 5 Discussion

Our system enables interaction with different LLMs through a unified API. The API is session-based and preserves a persistent message history, with built-in code interpreter integration, in a manner conceptually similar to the OpenAI Assistant. In our experiments, we observed clear performance differences between OpenAI's models and local Ollama models. OpenAI's models consistently showed higher performance concerning visualization literacy. In contrast, Ollama models offer a cost-effective alternative and, in some cases, provide faster responses. Based on these findings, we recommend using OpenAI's models in operation, where accuracy is essential, while relying on local models for testing and development, where cost-efficiency and rapid iteration are of greater importance.

We want to point out that our trustworthiness score is based on an extended version of the Mini-VLAT. While this provides a useful proxy for assessing model reliability, it remains unclear how well the scores generalize to broader analytic tasks or more complex visualizations.

[2]https://github.com/vndee/llm-sandbox/

```
Control Section
You may output control information in a JSON object
to influence the display of the choropleth map.
This control information must appear at the end
of your textual response and must be enclosed in a
markdown-style JSON code block:

```json
{ "fill": "value", "highlight": [],
  "task_classification": ["Retrieve Value"] }
```

When to Use Control Output
Proactively include control information if it supports
or enhances your answer.  For example, highlight
outliers when asked about anomalies, or confirm axis
mappings when reasoning about trends.
Do not explicitly state that you are providing control
output.  Avoid phrases like "Here is the configuration"
or "I will now provide the control information."

Control JSON Schema
{ "$schema":
    "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "fill": {
      "type": "string",
      "description": "Data variable mapped to
    fill color of data points",
      "default": "value"
    },
    "highlight": {
      "type": "array",
      "description": "Indices of highlighted
    observations in the chart. If empty, all
    observations will be displayed normally.",
      "items": { "type": "number" },
      "default": []
    },
    "task_classification": {
      "type": "array",
      "items": {
        "type": "string",
        "enum": [
          "Retrieve Value",
          "Find Extremum",
          "Determine Range",
          "Characterize Distribution",
          "Find Anomalies",
          "Find Clusters",
          "Find Correlations/Trends",
          "Make Comparisons",
          "Identify the Hierarchical Structure",
          "None"
        ]
      },
      "description": "The task(s) identified
    from the user's message"
    }
  },
  "required": [ "fill", "task_classification" ],
  "additionalProperties": false }
```

Figure 3: Control section part of the prompt which describes how the LLM can interact with the UI. In our choropleth map, the LLM can change the dataset variable used as the reference for the color gradient and highlight states.

## 6 Conclusions

NLIs offer a complementary way of interacting with visualizations. They can be particularly valuable for users with limited visualization literacy, who may find it easier to communicate through natural language. In this work, we presented a system that enables visualization designers to integrate an LLM-powered NLI into their visualizations, given the underlying dataset and an instruction prompt that includes a visualization specification. Our implementation is available on GitHub, and the provided example instruction prompt serves as a blueprint for further customization thus requiring only little effort in the setup.

As future work, we plan to expand the capabilities of our system, e.g., by incorporating additional hallucination identifiers [14, 22] and by integrating LLMs from a wider range of providers. Moreover, evaluating LLM performance on additional benchmarks, e.g., the *CALVI*, would enable our system to quantify the reliability of the output for a broader set of tasks [5,28]. We intend to explore uncertainty visualizations as a means of conveying the trustworthiness score more effectively. In addition, integrating human feedback to update trustworthiness scores could help improve reliability over time. Finally, we aim to streamline integration with data providers like *Statista* by automatically extracting both the visualization specification and the underlying data directly from HTML files.

## Acknowledgments

Part of this research work is supported by a PhD grant from the HPI Research School for Service-Oriented Systems Engineering at the Hasso Plattner Institute for Digital Engineering, University of Potsdam. Further, this project is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag (FK 16KN113522 "SPUR-OPT" and 16KN117820 "INSTA.KI"). The work of Tobias Schreck 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] A. Bendeck and J. Stasko. An empirical evaluation of the GPT-4 multimodal language model on visualization literacy tasks. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1105–1115, 2025. doi: 10.1109/TVCG.2024.3456155 1, 2

[2] Bloomberg. Bloomberg graphics, 2024. URL: https://www.bloomberg.com/graphics/infographics/. 1

[3] Bundesministerium für Gesundheit. Infektionsradar, 2024. URL: https://infektionsradar.gesund.bund.de/de. 1

[4] V. S. Bursztyn, J. Hoffswell, E. Koh, and S. Guo. Representing charts as text for language models: An in-depth study of question answering for bar charts. In *2024 IEEE Visualization and Visual Analytics*, VIS '24. IEEE, 2024. doi: 10.1109/VIS55277.2024.00061 2

[5] N. Chen, Y. Zhang, J. Xu, K. Ren, and Y. Yang. VisEval: A benchmark for data visualization in the era of large language models. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1301–1311, 2024. doi: 10.1109/TVCG.2024.3456320 4

[6] K. Choe, C. Lee, S. Lee, J. Song, A. Cho, N. W. Kim, and J. Seo. Enhancing data literacy on-demand: LLMs as guides for novices in chart interpretation. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):503–513, 2025. doi: 10.1109/TVCG.2024.3413195 2

[7] Climate Analytics. Climate action tracker, 2024. URL: https://climateactiontracker.org/. 1

[8] V. Dibia. LIDA: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 113–126. ACL, 2023. doi: 10.18653/v1/2023.acl-demo.11 1

[9] J. Gorniak, Y. Kim, D. Wei, and N. W. Kim. VizAbility: Enhancing chart accessibility with LLM-based conversational interaction. In *Pro-*

*ceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, UIST '24. ACM, 2024. doi: 10.1145/3654777.3676414 1
[10] Helical Insight. Open source NLP based data visualization, 2024. URL: https://www.helicalinsight.com/open-source-nlp-based-data-visualization. 2
[11] J. Hong, C. Seto, A. Fan, and R. Maciejewski. Do LLMs have visualization literacy? an evaluation on modified visualizations to test generalization in data interpretation. *IEEE Transactions on Visualization and Computer Graphics*, 31(10):7004–7018, 2025. doi: 10.1109/TVCG.2025.3536358 1, 2
[12] E. Hoque. NLP4Vis: Natural language processing for information visualization – half-day tutorial at IEEE VIS conference 2023, 2023. URL: https://nlp4vis.github.io/IEEEVis-2023/index.html. 1
[13] E. Hoque, P. Kavehzadeh, and A. Masry. Chart question answering: State of the art and future directions. *Computer Graphics Forum*, 41(3):555–572, 2022. doi: 10.1111/cgf.14573 1, 2
[14] M. N. Hoque, T. Mashiat, B. Ghai, C. D. Shelton, F. Chevalier, K. Kraus, and N. Elmqvist. The HaLLMark effect: Supporting provenance and transparent use of large language models in writing with interactive visualization. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24. ACM, 2024. doi: 10.1145/3613904.3641895 4
[15] M. Hutchinson, R. Jianu, A. Slingsby, and P. Madhyastha. LLM-assisted visual analytics: Opportunities and challenges. In *Computer Graphics and Visual Computing*, CGVC '24. EG, 2024. doi: 10.2312/cgvc.20241237 1
[16] A. Jobst, D. Atzberger, W. Scheibel, and J. Döllner. Towards a software framework for evaluating the visualization literacy of large language models. In *Proceedings of the 27th EG Conference on Visualization – Posters*, EuroVis Posters '25. EG, 2025. doi: 10.2312/evp.20251134 3
[17] A. Jobst, D. Atzberger, W. Scheibel, J. Döllner, and T. Schreck. 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*, IVAPP '25, pp. 867–874. SciTePress, 2025. doi: 10.5220/0013119600003912 2
[18] A. Jobst, D. Atzberger, M. Tytarenko, W. Scheibel, J. Döllner, and T. Schreck. A concept for integrating an LLM-based natural language interface for visualizations grammars. In *Proceedings of the 2nd Workshop on Prompt Engineering for Pre-Trained Language Models*, PromptEng '25. ACM, 2025. doi: 10.1145/3701716.3717812 1, 2
[19] I. Kaate, J. Salminen, S.-G. Jung, T. T. T. Xuan, E. Häyhänen, J. Y. Azem, and B. J. Jansen. "You Always Get an Answer": Analyzing users' interaction with AI-generated personas given unanswerable questions and risk of hallucination. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*, IUI '25, pp. 1624–1638. ACM, 2025. doi: 10.1145/3708359.3712160 1
[20] Knowi. Natural language BI, 2025. URL: https://www.knowi.com/natural-language-bi. 2
[21] S. Lee, S.-H. Kim, and B. C. Kwon. VLAT: Development of a visualization literacy assessment test. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):551–560, 2017. doi: 10.1109/TVCG.2016.2598920 1, 2
[22] F. Leiser, S. Eckhardt, V. Leuthe, M. Knaeble, A. Mädche, G. Schwabe, and A. Sunyaev. HILL: A hallucination identifier for large language models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24. ACM, 2024. doi: 10.1145/3613904.3642428 4
[23] Z. Li, H. Miao, V. Pascucci, and S. Liu. Visualization literacy of multimodal large language models: A comparative study. *arXiv CoRR*, arXiv:2407.10996, 2024. – Preprint. doi: 10.48550/arXiv.2407.10996 1, 2
[24] Microsoft. Introduction: Use natural language to explore data with Power BI Q&A, 2024. URL: https://learn.microsoft.com/en-us/power-bi/natural-language/q-and-a-intro. 2
[25] Microsoft. Analyze data in Excel, 2025. URL: https://support.microsoft.com/en-us/office/analyze-data-in-excel-3223aab8-f543-4fda-85ed-76bb0295ffc4. 2
[26] Microsoft. Create charts with Copilot in Excel, 2025. URL: https://support.microsoft.com/en-us/topic/create-charts-with-copilot-in-excel-0c07842d-4492-4c2b-8c3a-fb3114653ad7. 2
[27] 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*, 27(2):369–379, 2021. doi: 10.1109/TVCG.2020.3030378 2
[28] S. Pandey and A. Ottley. Mini-VLAT: A short and effective measure of visualization literacy. *Computer Graphics Forum*, 42(3):1–11, 2023. doi: 10.1111/cgf.14809 2, 4
[29] S. Pandey and A. Ottley. Benchmarking visual language models on standardized visualization literacy tests. *Computer Graphics Forum*, 44(3), 2025. doi: 10.1111/cgf.70137 2
[30] V. Setlur. Preparing data for natural language interaction in Ask Data, 2019. URL: https://www.tableau.com/learn/whitepapers/preparing-data-nlp-in-ask-data. 2
[31] 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*, 29(6):3121–3144, 2023. doi: 10.1109/TVCG.2022.3148007 1
[32] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The Craft of Information Visualization*, pp. 364–371. Elsevier, 2003. doi: 10.1016/B978-155860915-0/50046-9 1
[33] Spotfire. Take advantage of the latest advancements in AI and ML with Spotfire, 2025. URL: https://www.spotfire.com/overview/ai. 2
[34] M. Ströbel, K. Eckert, and T. Nagel. Hey ChatGPT, can you visualize my data?–a multi-dimensional study on using an LLM for constructing data visualizations. In *Proceedings of the EuroVis – Posters*. EG, 2024. doi: 10.2312/evp.20241083 1
[35] ThoughtSpot. Introducing spotter: The AI analyst for everyone, 2025. URL: https://www.thoughtspot.com/product/ai-analyst. 2
[36] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang, and Y. Wu. ChartGPT: Leveraging LLMs to generate charts from abstract natural language. *IEEE Transactions on Visualization and Computer Graphics*, 31(3):1731–1745, 2025. doi: 10.1109/TVCG.2024.3368621 1
[37] veezoo. Uncover insights like never before with search, 2024. URL: https://veezoo.com/search-and-visualize/. 2
[38] P.-P. Vázquez. Are LLMs ready for visualization? In *Proceedings of the 2024 IEEE 17th Pacific Visualization Conference*, PacificVis '24, pp. 343–352. IEEE, 2024. doi: 10.1109/PacificVis60374.2024.00049 1
[39] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53(3), 2020. doi: 10.1145/3386252 2
[40] washuvis. Mini-VLAT, 2023. URL: https://github.com/washuvis/Mini-VLAT. 2
[41] Z. Xu and E. Wall. Exploring the capability of LLMs in performing low-level visual analytic tasks on SVG data visualizations. In *Proceedings of the 2024 IEEE Visualization and Visual Analytics*, VIS '24, pp. 126–130. IEEE, 2024. doi: 10.1109/VIS55277.2024.00033 1
[42] X. Zeng, H. Lin, Y. Ye, and W. Zeng. Advancing multimodal large language models in chart question answering with visualization-referenced instruction tuning. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):525–535, 2025. doi: 10.1109/TVCG.2024.3456159 1