Accepted to appear in IEEE VIS 2025 (Poster Track)

NL4DV-Stylist: Styling Data Visualizations Using Natural Language and Example Charts

Tenghao Ji*
University of Michigan

Arpit Narechania†
The Hong Kong University of Science and Technology

Abstract

Natural language interfaces (NLIs) enable flexible authoring and interaction with visualizations (VIS) using natural language (NL). However, current NLIs predominantly focus on identifying the data attributes and analytic tasks from the user's query and give less emphasis to customizing the design of the output visualizations, which is an important consideration to satisfy personal preferences, meet branding requirements, and/or ensure accessibility. We present **`NL4DV-Stylist`**, a Python toolkit that utilizes example charts and NL instructions as design guidance, enhancing current NL to VIS authoring workflows to produce more design-informed visualizations. For instance, if a user generally prefers horizontal legends or wants to re-use the blue color scheme they saw in a scatterplot, they can supply these as inputs to the toolkit, which will incorporate these design preferences when generating visualizations. NL4DV-Stylist is available as open-source software at **`https://nl4dv.github.io`**.

Index Terms: Natural language interface, natural language, data visualization, visualization, styling, open-source, toolkit.

1 Introduction and Background

Natural language (NL) interfaces (NLIs) have transformed visual data analysis by enabling people to flexibly specify and interact with visualizations using NL [10]. Additionally, there exist open-source libraries and toolkits [8, 6, 7] that have enabled visualization developers–who may not have a background in natural language processing (NLP)–to create new visualization NLIs or incorporate NL input within their existing systems.

However, while current NLIs understand *what* to visualize, they often fall short in addressing *how* to style the resulting visualizations. Visual design elements–such as color schemes, typography, and layout–play a crucial role in effective data communication, e.g., satisfying personal preferences, meeting branding requirements, and ensuring accessibility. Recent work has begun addressing this gap through structured *editing actions* [12] (e.g., "Turn the dashed lines black"), yet remains limited to textual descriptions.

Recent advances in Vision Language Models (VLMs) have opened new possibilities for understanding and manipulating visual content. Empirical evaluations [1, 2, 4, 5] already demonstrate that VLMs can somewhat effectively interpret visualizations and respond to NL-based questions about the associated content. In addition, there are new approaches to design extraction and reuse, with systems like Brickify [11] and CreativeConnect [3] decomposing designs into reusable tokens that can be creatively recombined.

Building on these advances, we present **`NL4DV-Stylist`**, a Python toolkit that leverages VLMs to author *design-informed* visualizations using a combination of example charts and NL. Our approach addresses current NLI limitations by streamlining the process of reusing and remixing design inspiration from multiple sources to inform the design of the target visualization, as illustrated in Figure 1. Our primary contribution is:

1. An open-source Python toolkit, **NL4DV-Stylist**, comprising:
(a) A prompt template crafted to extract visual design elements from an image-based chart into a Vega-Lite specification.
(b) A prompt template crafted to appropriately apply the extracted design elements into an output visualization generated as part of an NL to visualization authoring workflow.
(c) An extensible API for developers to prototype effective NLIs.

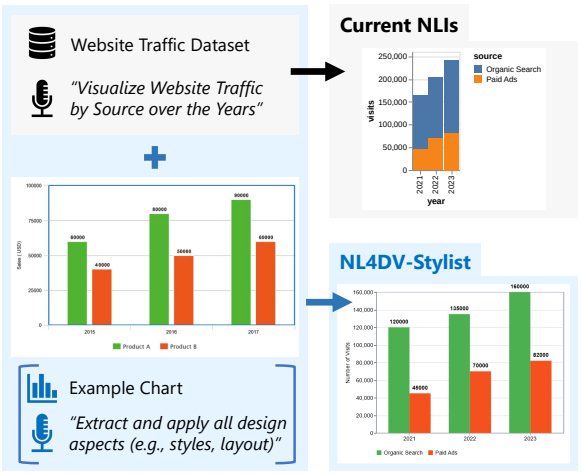

Figure 1: Given a dataset on *website traffic* and a natural language (NL) query, *"Visualize Website Traffic by Source over the Years,"* current natural language interfaces (NLIs) generate a relevant visualization but with a default layout and style, e.g., a stacked bar chart with blue and orange colors. With NL4DV-Stylist, users can additionally input one or more example charts, e.g., a grouped bar chart with green and red colors, along with corresponding NL instructions to extract and apply its design into the output visualization.

2 NL4DV-Stylist

2.1 Design Principles

Our approach is built using two core design principles:

D1. Expressivity: Enable users to author design-informed visualizations by combining natural language instructions with example charts, allowing them to express complex design intents through references to visual elements in the examples.

D2. Modularity: Support style multiplexing by allowing users to select and combine design elements–such as color schemes, typography, and layouts–from multiple charts, enabling coherent integration of these aspects into a single visualization.

2.2 Architecture

NL4DV-Stylist extends an existing Python toolkit, NL4DV. NL4DV processes a tabular dataset and a natural language (NL) query about the dataset to generate a complete analytic specification

*e-mail: jimji@umich.edu
†e-mail: arpit@ust.hk

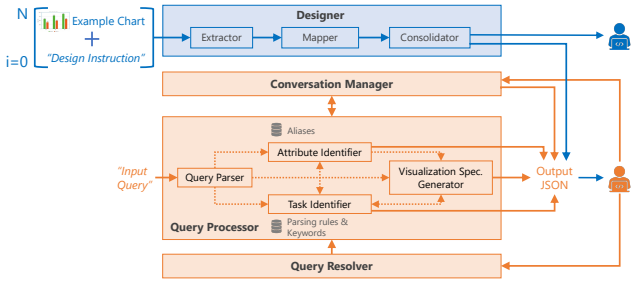

Figure 2: NL4DV-Stylist's architecture (in blue) extended from NL4DV's architecture (in orange) to support the new visual styling capabilities. The arrows indicate the flow of information.

comprising data attributes, analytic tasks, and relevant visualizations (as Vega-Lite), all modeled as a single JSON object. NL4DV's architecture (Figure 2, in orange) includes a Query Processor module that translates NL queries into visualizations, a Conversation Manager module to facilitate multi-turn dialogs, and a Query Resolver module to handle ambiguities and errors during query translation. NL4DV-Stylist builds on this foundation by introducing a new **Designer** module (Figure 2, in blue), consisting of three sequential components, as described below:

1. **Extractor:** Given one or more example charts and user design instructions (**D1**), this component utilizes a VLM and a crafted prompt template–where the design instructions are inserted as parameters–to extract design elements (e.g., colors).
2. **Mapper:** Next, as part of the same VLM call as the Extractor, this component maps the extracted design elements into a structured Vega-Lite [9] specification, for downstream use.
3. **Consolidator:** Lastly, this component utilizes a VLM and another crafted prompt template–into which the Vega-Lite specifications from the Mapper are inserted–to generate a consolidated specification (**D2**), resolving conflicting instructions.

2.3 Implementation and Usage

Implementation-wise, NL4DV-Stylist is integrated into NL4DV (as a major version upgrade). Currently, given a tabular dataset and an NL query about the dataset, with a single function call–`analyze_query(query)`–NL4DV outputs a JSON object comprising an `attributeMap` composed of the inferred dataset attributes, a `taskMap` composed of the inferred analytic tasks, and a `visList`, a list of visualizations relevant to the input query. NL4DV-Stylist, in addition to supporting these functionalities, introduces a new `design_config` parameter–that takes as input pairs of example charts and corresponding NL design instructions–and an `lm_config` parameter–that takes as input corresponding VLM configuration (e.g., GPT-4o). These inputs are processed and an appropriate design-informed visualization is output. Listing 1 illustrates the corresponding Python source code for Figure 1.

Listing 1 Sample Python code to use NL4DV-Stylist.

```
from nl4dv import NL4DV
data_url="Website-Traffic.csv"
processing_mode = "language-model"
lm_config = { "model": "gpt-4o", "api_key": "..." } # model

design_config = [
  {"type": "image_url", "image_url": {"url": "Example.png"}},
  {"type": "text", "text": "Apply the design of Example.png"}
] # styling object

nl4dv_instance = NL4DV(data_url, processing_mode, lm_config,
↪  design_config)
query = "Visualize Website Traffic by Source over the Years"
output = nl4dv_instance.analyze_query(query)
```

2.4 Example Applications

1. **Learning Vega-Lite syntax.** Users who are less familiar with Vega-Lite can learn the syntax by inputing a chart and specifying which design elements they want to extract. NL4DV-Stylist will output a JSON object that will show, e.g., how to set the background color (`{"background":"red"}`), change the font size of axis label (`{"axis":{"labelFontSize":16}}`), or reorient the legend (`{"legend":{"orient":"bottom"}}`).
2. **Expressively authoring custom charts or customizing existing charts.** Users who are less satisfied with the default chart designs of NL to VIS authoring toolkits, can input example charts (consisting of target styles) and provide design instructions using natural language. NL4DV-Stylist will extract, map, and apply these styles, into a unified, conflict-free Vega-Lite specification, enabling both novice and experts to expressively create visually coherent and bespoke charts.

3 Ongoing Work

We next plan to benchmark NL4DV-Stylist's performance on multiple Vision Language Models (e.g., GPT-4o, Claude) studying (1) to what extent styles are recoverable from charts, (2) how often and how successful are the subsequent style transfers, and (3) if performance varies for simple versus complex charts, among other factors. We also plan to conduct user studies to assess the toolkit's efficacy and usability to inform subsequent features and refinements.

References

[1] A. Bendeck and J. Stasko. An Empirical Evaluation of the GPT-4 Multimodal Language Model on Visualization Literacy Tasks. *IEEE Trans. Vis. Comput. Graph.*, 31(1), 2025. 1

[2] 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 Trans. Vis. Comput. Graph.*, 31(1), 2025. 1

[3] D. Choi, S. Hong, J. Park, J. J. Y. Chung, and J. Kim. CreativeConnect: Supporting Reference Recombination for Graphic Design Ideation with Generative AI. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024. 1

[4] L. Dong and A. Crisan. Probing the visualization literacy of vision language models: the good, the bad, and the ugly. *arXiv preprint arXiv:2504.05445*, 2025. 1

[5] 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 Trans. Vis. Comput. Graph.*, 2025. 1

[6] R. Mitra, A. Narechania, A. Endert, and J. Stasko. Facilitating Conversational Interaction in Natural Language Interfaces for Visualization. *IEEE VIS (Short Papers)*, 2022. 1

[7] A. Narechania, A. Srinivasan, and J. Stasko. NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries. *IEEE Trans. Vis. Comput. Graph.*, 2021. 1

[8] S. Sah, R. Mitra, A. Narechania, A. Endert, J. Stasko, and W. Dou. Generating Analytic Specifications for Data Visualization from Natural Language Queries using Large Language Models. *NLVIZ Workshop at IEEE VIS*, 2024. 1

[9] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Trans. Vis. Comput. Graph.*, 23(1), 2016. 2

[10] 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 Trans. Vis. Comput. Graph.*, 29(6), 2023. 1

[11] X. Shi, Y. Wang, R. Rossi, and J. Zhao. Brickify: Enabling Expressive Design Intent Specification through Direct Manipulation on Design Tokens. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 2025. 1

[12] Y. Wang, Z. Hou, L. Shen, T. Wu, J. Wang, H. Huang, H. Zhang, and D. Zhang. Towards Natural Language-based Visualization Authoring. *IEEE Trans. Vis. Comput. Graph.*, 29(1), 2023. 1