Visualization-oriented Natural Language Interfaces for Grafana Dashboard

Bo-Yao Tong*, Ted T. Kuo†, and Chia-Yu Lin*
Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan
† *College of Artificial Intelligence, National Yang Ming Chiao Tung University, Tainan, Taiwan*
Corresponding Author: Chia-Yu Lin (sallylin0121@ncu.edu.tw)

2024 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan) | 979-8-3503-8684-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICCE-Taiwan62264.2024.10674281

***Abstract*—Real-time monitoring of individual node statuses is crucial in distributed systems to promptly identify and address anomalies and enable effective system management. Due to the less intuitive nature of traditional form-based methods for adding charts, there is a growing trend towards integrating large language models (LLMs) into visualization-oriented natural language interfaces (V-NLIs) to automate the entire front-end monitoring process. Although LLMs have advanced significantly, they still exhibit weaker capabilities in code generation, particularly when visualization platforms need LLMs to generate some specialized JSON formats for updating dashboards. To improve the code generation capability of general-purpose LLMs, we propose a system that allows users to create charts using arbitrary natural language. Additionally, concerning the limitations of LLMs in code generation, we aim to leverage the LangChain agent to develop a custom toolkit. The LangChain agent will enable language models to focus on text comprehension while our tool will handle code generation.**

***Index Terms*—Visualization-oriented natural language interfaces, large language model, Grafana, LangChain**

I. Introduction

Traditional DevOps teams commonly utilize various monitoring platforms, such as Prometheus [1] and Grafana [2], for storing metrics and visualizing data. Historically, they have added panels through platforms, often employing form-based methods. However, this approach typically necessitates manual filtering and confirmation of available options, resulting in a less straightforward and intuitive process. Consequently, there is a growing interest in integrating large language models (LLMs) into the front end to enable the generation of charts using natural language. Nevertheless, applying a general-purpose language model to generate monitoring graphs involves code generation, which may have limitations. These limitations could include susceptibility to issues like version updates or inaccuracies in data sources.

To tackle these challenges, Chat2vis [3] implemented prompt engineering techniques to guide LLMs in generating code. These guidelines encompassed explanations of various data tables and the necessary packages, aiming to prevent errors stemming from version updates and ensure accurate data interpretation by LLMs. Meanwhile, ChartGPT [4] adopted a strategy combining fine-tuning and prompt engineering. Their approach to prompt engineering differed in how they segmented user input into multiple segments to generate corresponding responses, thereby ensuring comprehensive coverage in tasks like data retrieval and chart generation. However, in the aforementioned methods, the design of prompts may encounter token limits, potentially hindering the complete expression of data meanings and desired code generation. [5] Grafana official is also working on integrating LLMs into dashboards, but the graphs generated remain constrained to time-series data source only from Prometheus.

In this paper, we propose a system to integrate LLMs into visualization-oriented natural language interfaces(V-NLIs), enabling users to create charts effortlessly using natural language. In our system, LLMs will no longer handle code generation; instead, they will capitalize on their proficiency in language understanding. The system determines the best approach by comparing tool descriptions with user needs and generates the appropriate code for the visualization platform. With our system, we substantially enhance the precision of language models in code generation scenarios.

II. Methods

Grafana and Prometheus are currently the most widely employed monitoring platforms. Mature teams usually have a reasonably comprehensive Grafana dashboard. A more user-friendly approach could involve incorporating a text box with LLMs into their existing dashboard when considering future maintenance and possible adjustments to particular charts. To address this need, we have developed a V-NLI that allows swift integration into existing dashboards, enabling the direct generation of graphs using natural language commands within the same dashboard.

A. Visual-oriented Natural Language Interfaces

Fig. 2 shows our interface, which consists of three main sections. Users can input their requests in the text box without any constraints, meaning they are not limited by predefined forms. This allows them to explore data through intuitive queries using their most familiar natural language, enabling them to understand the system's capabilities effortlessly. For example, a user can simply input "show me prometheus http requests total" without needing to remember the exact metric name "prometheus_http_requests_ total." The system identifies the most relevant metrics quickly and accurately using large language models (LLM), saving users time spent searching for precise names. The input is then sent to our servers via HTTP, where validation checks prompt users to provide missing required parameters or correct errors. Once validated,

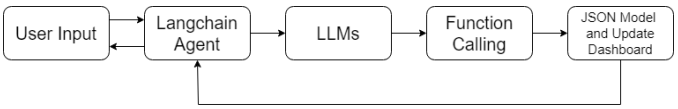

Fig. 1. The workflow of V-NLIs.

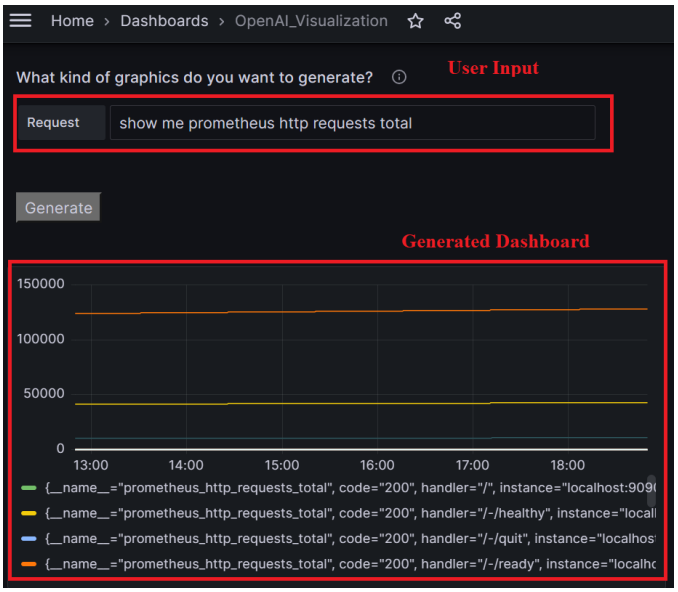

Fig. 2. V-NLIs with user input and generated chart

the query language used to access the desired metrics is displayed, and the graph is generated in the bottom section of the interface.

B. LangChain

LangChain [6] is a framework designed for building applications powered by language models. It supports context-aware and reasoning applications, providing libraries, templates, and tools for developing, testing, and deploying applications that leverage language models. LangChain simplifies the entire application lifecycle, and provides modules for interacting with language models, retrieving application-specific data, and agent-based decision-making.

C. Model and Workflow

The entire V-NLIs process operates as Fig. 1. To enhance the code generation capability of the GPT-3.5-turbo language model [7], we implement a LangChain agent between users and language models. LangChain provides a comprehensive OpenAI agent and bridges the user and the GPT-3.5-turbo language model. When a user asks a question, the LangChain agent processes the query and passes it to the language model. The agent will instruct the language model to determine which tools may be required for the users according to the query's intent and the description of tools. The tool description needs to align best with the query requirements. For example, our toolkit has a tool for drawing basic graphs, whose description might be something like "Useful for when users want to display metrics without any custom request." The language model successfully utilizes this tool through multiple experiments when a user query needs to display some metrics, and it will extract essential parameters, such as the metrics name required here. The parameters necessitate writing corresponding descriptions and their variable types so that the language model knows how to extract the critical parameters for this tool before using it. Our tool's content generates JSON format accepted by Grafana, setting the target property as the desired metrics name, and updates the results via the Grafana API on the dashboard for the user to view, as shown at the bottom of Fig. 2.

In the proposed V-NLIs, the LangChain agent interprets and presents the model's output in a user-friendly format, effectively mediating the interaction between the user and the language model. Furthermore, Users can communicate with the language model without being concerned about communicating with it, which can be quickly deployed on their existing dashboard.

III. Discussion

From Fig. 2, we can see that when the user inputs "show me prometheus http requests total," GPT-3.5 generates a line chart for the specified metric based on the user input and the default parameters for chart customization. We assume that if users do not specify the type of chart they want, we will generate the most commonly used line chart corresponding to the time series chart in Grafana.

IV. Conclusion

In this paper, we proposed a system to integrate LLMs into V-NLIs, enabling users to create charts effortlessly using natural language. The proposal suggests using the LangChain framework to improve code generation with language models. An agent selects the most suitable tool, and rule-based logic generates JSON models for the Grafana dashboard update. The proposed V-NLIs for the Grafana dashboard help users can rapidly deploy portable monitoring systems on Grafana with minimal costs.

Acknowledgements

This work is sponsored by the National Science and Technology Council (NSTC) under the projects NSTC 110-2222-E-008-008-MY3 and NSTC 112-2622-8-A49-021.

References

[1] "Prometheus," https://prometheus.io/.
[2] "Grafana," https://grafana.com/.
[3] Paula Maddigan and Teo Susnjak, "Chat2vis: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models," *IEEE Access*, vol. 11, pp. 45181–45193, 2023.
[4] Yuan Tian, Weiwei Cui, Dazhen Deng, Xinjing Yi, Yurun Yang, Haidong Zhang, and Yingcai Wu, "Chartgpt: Leveraging llms to generate charts from abstract natural language," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–15, 2024.
[5] "Grafana/Scenes," https://github.com/grafana/scenes.
[6] "LangChain," https://www.langchain.com/.
[7] OpenAI, "ChatGPT (3.5) [Large language model]," https://chat.openai.com, 2024.