VizAgent: Towards an Intelligent and Versatile Data Visualization Framework Powered by Large Language Models

Hue Luong Thi Minh, Vinh Nguyen The and Truong Quach Xuan

January 6, 2025

VizAgent: Towards an Intelligent and Versatile Data Visualization Framework Powered by Large Language Models

Hue, Luong-Thi-Minh [0009-0003-3737-7741] and Vinh, Nguyen-The [0000-0002-1300-3943] and Truong, Quach Xuan [0000-0003-1402-0803]

Thai Nguyen University of Information and Communication Technology, Vietnam
`lmhue@ictu.edu.vn`

Abstract. This study proposed VizAgent, a novel framework designed to address the challenges in data visualization. By leveraging the capability of Large Language Model (LLM), VizAgent automates describe data, guides users' intentions, automatically generates visualizations for specific tasks, and compares the quality of generated visualizations across different libraries. The results demonstrated that the matplotlib library outperformed other visualization libraries in terms of success rate, suggesting opportunities for further investigation and improvement, particularly in enhancing the performance of seaborn, plotly, and ggplot visualizations. The VizAgent framework presents a promising approach to intelligent data visualization, with several avenues for extending its capabilities. These include specialized handling for certain visualization libraries, empowering users to fine-tune parameters and styling, and incorporating advanced data analysis and feature engineering capabilities. VizAgent contributes to the ongoing efforts in data visualization by providing a valuable resource for researchers, practitioners, and individuals seeking data-driven decision-making.

Keywords: Data Visualization, Generative AI, LLM, Python, Streamlit.

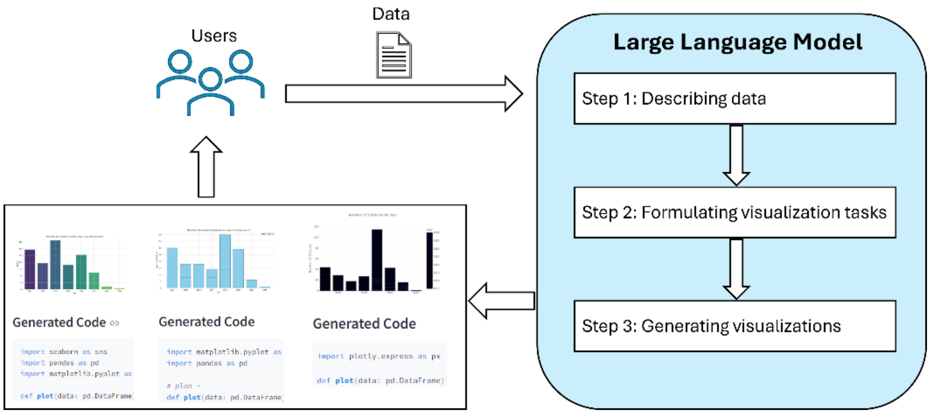
1 Introduction

Data visualization is crucial for conveying information and insights, as it involves intentionally rendered details by computer programs. It's used in various domains like business, education, and scientific research. However, there are many unsolved problems which often require expertise from creators in each domain [1, 2].

Researchers are actively conducting a wide range of experiments to address the challenges in data visualization, some of which have been integrated into commercial products like Microsoft Office and Google Analytics, while others remain in libraries, packages, and toolkits [2-4]. The evolution of data visualization generation can be classified into three periods. The first period focused on a rule-based system that tried to capture user intentions and build visualizations accordingly (e.g., Articulate, DataTone, Eviza, DeepEye, FlowSense, NL4DV) [3, 4]. However, these studies faced a common obstacle: limited rules to capture diverse user intent. The second period attempted to address the previous limitation by investigating the machine learning paradigm, with the goal of building a model capable of understanding user intention rather than memorizing user syntax. Notable works [2, 4] in this period include ADVISor, NL2DV, ncNet, RGVisNet, and Data2Vis. However, these approaches faced limitations in understanding complex natural language due to the lack of available training data and computational resources. The advent of generative AI models has ushered in the third period of data visualization, where visualizers can leverage large language models to overcome the existing issues. Pioneers in this emerging research include Chat2VIS [5], LIDA [6], and Data Formulator [7]. The shortage of experiments in this period can be partially attributed to the computational and resource requirements of these large models, as well as the limited availability of their APIs over time. Pioneer experiments have shown promising results, but face challenges such as Chat2VIS's requirement for users to understand data and ask questions, LIDA's step-by-step visualizations lacking simultaneous comparison, and Data Formulator's user involvement, creating barriers for beginners and requiring further investigation.

Our current study seeks to address the previous limitations by providing a novel framework. In other words, we try to answer the following questions: RQ1: How users' intention can be guided by the proposed framework based on user's dataset? RQ2: How visualizations can be generated based on visualization tasks? RQ3: What are the success rates of visualizations across libraries? By answering the above questions, our contributions to the body of knowledge are as follows: (1) We propose a framework that is capable of automatically generating users' intentions and visualizing them accordingly. (2) We demonstrate our proposed framework through an intelligent and versatile data visualization interface powered by Large Language Model. (3) We compare the success rates of generated visualizations among several libraries.

2 Materials and Methods



```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as

def plot(data: pd.DataFrame)
```

```
import matplotlib.pyplot as
import pandas as pd

# plan -
def plot(data: pd.DataFrame)
```

```
import plotly.express as px

def plot(data: pd.DataFrame)
```

Fig. 1. The proposed framework

2.1 Research Design

Fig. 1 Illustrates our proposed architecture in which users interact with the system initially by uploading the dataset. The system automatically processes the data by: (1) describing the data, (2) formulating visualization tasks, (3) generating visualizations. We describe each step in details as follows:

Step 1: Describing data

This step involves understanding the structure of data such as the number of columns, headers and data type (property). This information is crucial in the subsequent step as it provides context for the LLM. Similar to `DESCRIBE` function in SQL to describe table with column properties (e.g., varchar, int), we utilized `pandas.api.types.infer_dtype()` – a function in the pandas library that attempts to infer the data type of a given input - to infer our users' arbitrary data types. The output in Step 1 is an array `message` containing a number of objects (# of columns in the dataset). Each object is described as an example below:

```
{
  "column": "Year",
  "properties": {
    "dtype": "int64",
    "samples": [
      2018,
      2022,
      2017
    ],
  }
}
```

Step 2: Formulating visualization tasks

Upon having a description of data, we formulate visualization tasks through **prompt engineering** in which we communicate with LLM via a chat template [8]. The formulation of auto generated visualization tasks can be expressed as follows:

$$T = f(r, \mathcal{D}) \tag{1}$$

Where f is the AI agent or LLM, r is the user's request or requirements and $\mathcal{D}$ is the data description. As illustrated in the code snippet below, the `system` role (line 4 with truncated data) influences how the assistant behaves, and we describe the system message in approximately ~800 words. The `user` role (line 8) asks for the number of visualization tasks (`n` is set by users with default value of 5) based on data description (provided in Step 2). And finally, the `assistant` role (line 12) supports the system in the output format. The output format should be a list of Json objects with predefined properties. At this step, we expect to receive the visualization tasks in the form of possible research questions based on the provided data description.

```
1. messages = [
2.     {
3.         "role": "system",
        "content": " You are Savant from ICTU AI, a data analyst and visualizer. The user is in a conversation with you exploring their data ..…(truncated)"
    },
    {
        "role": "user",
        "content": "The number of TASKS to generate is {n}. The tasks should be based on the data summary below \n {summary}"
    },
    {
        "role": "assistant",
        "content": "THE RESULT SHOULD BE A CODE SNIPPET OF A LISTED JSON OBJECTS THAT IS VALID. THE FOLLOWING FORMAT MUST BE USE….(truncated)"
    }
]
```

Step 3: Generating visualizations

Given a list of visualization tasks (or possible research questions), when users select one research question of interest, the system will prepare a message like the previous ones with updated message contents. In terms of visualization rendering, we provided four templates from libraries/packages such as matplotlib, seaborn, plotly, and ggplot. We execute all generated visualization codes for comparison

```
code = f"""
import matplotlib.pyplot as plt
import pandas as pd
def plot(data: pd.DataFrame):
    <params>
    plt.title(f"{task.question}", wrap=True)
    return plt;
chart = plot(data)
"""
```

2.2 Experiment Setup

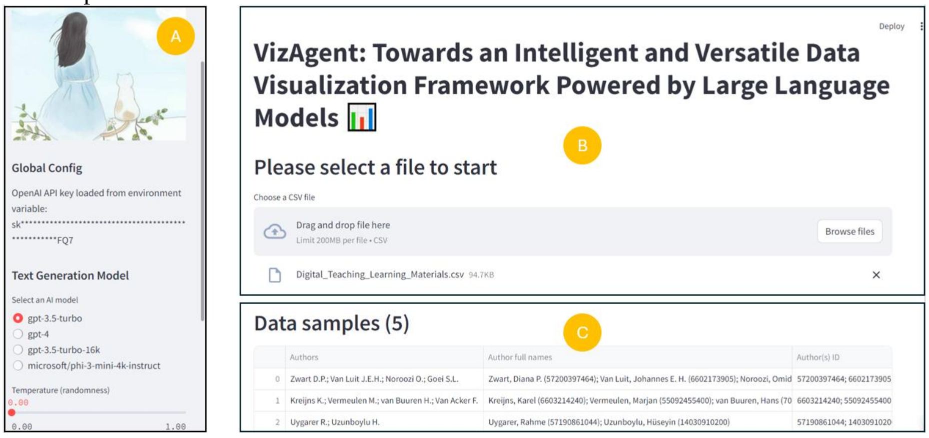
Model selection. Of the many available models on the internet such as `CodeT5`, `CodeLlama`, `gpt-4o`, `gpt-4-turbo`, `gpt-4`, `gpt-3.5-turbo`, `gemini` [9], our study focused on `gpt-3.5-turbo` model from OpenAI. This is due to our limited hardware constraints (deploy locally) and tokens paid. Furthermore, results from `gpt-3.5-turbo` would be served as baseline comparison in many other research settings where funding is limited. The model `temperate` is set to 0 for consistent response. **Framework deployment.** To implement our proposed framework, we utilize Streamlit - a free and open-source Python-based library designed for machine learning engineers to create machine learning web apps [10]. **Hardware specification.** Our study was carried out on Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz with 32.0 GB of RAM and 4GB of GPU. The operating system is Windows 10 Pro.

2.3 Evaluation

We evaluated the performance of our proposed model in three use cases and examined whether the visualizations can be corrected generated to fulfil the visualization tasks. The first use case explored dataset from publications extracted from Scopus. The second use case investigate `car` dataset. The final use case analyzes `iris` dataset [11]. While the 2nd and 3rd use cases were conducted in the popular dataset, the publication dataset may be more meaningful as it could help in writing scientific papers.

3 Results

Fig. 2 illustrates our VizAgent application. VizAgent consists of three components: A, B, C. Component A assists users in defining LLM's API, selecting model, and configuring temperature. Component B enables users to upload their own dataset for exploration. Component C is the automatically generated data description, visualization tasks and rendering visualizations. Users can adjust parameters in Component C, otherwhile default parameters are used.



Global Config

OpenAI API key loaded from environment variable:
sk***FQ7

Text Generation Model

Select an AI model
[✓ gpt-3.5-turbo]
gpt-4
gpt-3.5-turbo-16k
microsoft/phi-3-mini-4k-instruct

Temperature (randomness)
0.00
0.00 1.00

VizAgent: Towards an Intelligent and Versatile Data Visualization Framework Powered by Large Language Models 📊

Please select a file to start

Choose a CSV file

Limit 200MB per file • CSV

Digital_Teaching_Learning_Materials.csv 94.7KB

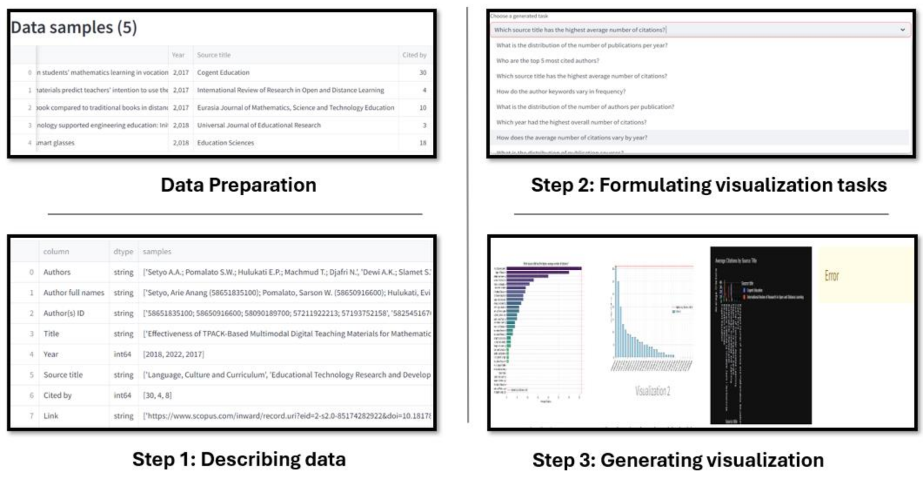
Data samples (5)

	Authors	Author full names	Author(s) ID
0	Zwart D.P.; Van Luit J.E.H.; Noroozi O.; Goei S.L.	Zwart, Diana P. (57200397464); Van Luit, Johannes E. H. (6602173905); Noroozi, Omid	57200397464; 6602173905
1	Kreijns K.; Vermeulen M.; van Buuren H.; Van Acker F.	Kreijns, Karel (6603214240); Vermeulen, Marjan (55092455400); van Buuren, Hans (70	6603214240; 55092455400
2	Uygarer R.; Uzunboylu H.	Uygarer, Rahme (57190861044); Uzunboylu, Hüseyin (14030910200)	57190861044; 1403091020

Fig. 2. The intelligent and versatile data visualization interface powered by LLM

3.1 RQ1: How users' intention can be guided by the proposed framework based on user's dataset?

Fig. 3 illustrates examples of flow from VizAgent. When users upload their data, Data Preparation shows the first five records for examination. In Step 1, data properties were extracted and stored in dtype (e.g., dtype of "Authors" is string, dtype of "Year" is int64, dtype of "Source Title" is string, dtype of "Cited by" is int64, etc.). Information from Step 1 will be concatenated with other contextual information (we manually engineered prompt) then sent to LLM API call. As a result, users' intention (visualization tasks/research questions) were presented in Step 2


Data Preparation
Step 2: Formulating visualization tasks
Step 1: Describing data
Step 3: Generating visualization
Visualization 2
Error

Data samples (5)

		Year	Source title	Cited by
0	n students' mathematics learning in vocation	2,017	Cogent Education	30
1	aterials predict teachers' intention to use the	2,017	International Review of Research in Open and Distance Learning	4
2	ook compared to traditional books in distan	2,017	Eurasia Journal of Mathematics, Science and Technology Education	10
3	nology supported engineering education: Ini	2,018	Universal Journal of Educational Research	3
4	mart glasses	2,018	Education Sciences	18

Choose a generated task

Which source title has the highest average number of citations?

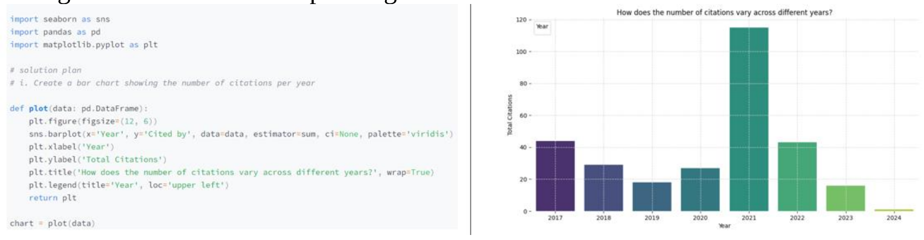
- What is the distribution of the number of publications per year?
- Who are the top 5 most cited authors?
- Which source title has the highest average number of citations?
- How do the author keywords vary in frequency?
- What is the distribution of the number of authors per publication?
- Which year had the highest overall number of citations?
- How does the average number of citations vary by year?
- What is the distribution of publication sources?

	column	dtype	samples
0	Authors	string	['Setyo A.A.; Pomalato S.W.; Hulukati E.P.; Machmud T.; Djafri N.', 'Dewi A.K.; Slamet S.
1	Author full names	string	['Setyo, Arie Anang (58651835100); Pomalato, Sarson W. (58650916600); Hulukati, Evi
2	Author(s) ID	string	['58651835100; 58650916600; 58090189700; 57211922213; 57193752158', '58254516
3	Title	string	['Effectiveness of TPACK-Based Multimodal Digital Teaching Materials for Mathematic
4	Year	int64	[2018, 2022, 2017]
5	Source title	string	['Language, Culture and Curriculum', 'Educational Technology Research and Develop
6	Cited by	int64	[30, 4, 8]
7	Link	string	['https://www.scopus.com/inward/record.uri?eid=2-s2.0-85174282922&doi=10.1817

Fig. 3. Example response from VizAgent

3.2 RQ2: How visualizations can be generated based on visualization tasks?

The Step 3 in **Fig. 3** shows the results of generated visualizations from different libraries such as matplotlib, seaborn, plotly, and ggplot. By providing code template to LLM API, the VizAgent responsibility is the execute the response. **Fig. 4** provides details of code generated and its corresponding visualization.



```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# solution plan
# i. Create a bar chart showing the number of citations per year

def plot(data: pd.DataFrame):
    plt.figure(figsize=(12, 6))
    sns.barplot(x='Year', y='Cited by', data=data, estimator=sum, ci=None, palette='viridis')
    plt.xlabel('Year')
    plt.ylabel('Total Citations')
    plt.title('How does the number of citations vary across different years?', wrap=True)
    plt.legend(title='Year', loc='upper left')
    return plt

chart = plot(data)
```

Fig. 4. (Left) Python code received from LLM, (Right) VizAgent renders the visualization

3.3 RQ3: What is the success rate of visualizations across libraries?

To evaluate our proposed framework and the response quality from LLM we conducted three case studies on different datasets including two popular ones (iris, cars) and one publication dataset we exported from Scopus. The results are described in **Table *1***.

Table 1. Performance of VizAgent across four python libraries

Datasets	sea-born	mat-plotlib	plotly	ggplot
Publications				
How does the number of citations vary across different years?	✓	✓	×	×
Which authors have the highest number of publications?	×	✓	✓	×
What are the most common author keywords used?	×	✓	×	×

How does the distribution of publication sources look like?	✓	✓	✓	✓
Is there a correlation between citations and the author(s)?	✗	✓	✓	✗
Cars				
What is the distribution of mpg in the dataset?	✓	✓	✓	✗
How does the number of cylinders (cyl) impact horse-power (hp)?	✓	✓	✓	✓
Which cars have the highest and lowest weight (wt)?	✓	✓	✗	✗
How does the transmission type (am) affect the acceleration time (qsec)?	✓	✓	✓	✓
What is the relationship between gear type (gear) and rear axle ratio (drat)?	✓	✓	✗	✗
Iris				
What is the distribution of SepalLengthCm?	✓	✓	✓	✓
How does PetalWidthCm vary across different species?	✓	✗	✓	✓
Which species has the highest average SepalWidthCm?	✓	✓	✓	✓
How does PetalLengthCm correlate with SepalLengthCm?	✗	✓	✓	✗
What is the spread of SepalWidthCm for each species?	✓	✓	✓	✓
Success rate (%)	73.3	**93.3**	73.3	46.7

The analysis of the VizAgent framework's performance across the four Python visualization libraries (seaborn, matplotlib, plotly, and ggplot) reveals that Matplotlib has the highest success rate at 93.3%, indicating that the framework is particularly well-suited for generating high-quality visualizations using this versatile library. Seaborn and plotly also performed reasonably well, with a 73.3% success rate, suggesting that the VizAgent framework can effectively leverage the capabilities of these libraries to create informative and interactive visualizations. The lower success rate with ggplot, at 46.7%, may indicate that the framework has more difficulty in translating research questions into the specific syntax and structure required by this grammar-of-graphics-based library.

4 Discussion

Overall, the VizAgent framework presents a promising approach to intelligent data visualization, and there are several avenues for extending its capabilities to meet the evolving needs of users and the growing complexity of data analysis tasks. First, the framework's lower success rate with the ggplot library suggests that certain visualization libraries may require more specialized handling or translation of user intentions and research questions. Second, while the current approach of providing code templates and relying on the LLM to generate the specific visualization code is effective, allowing users to fine-tune parameters, styling, or even the underlying algorithms used for visualization could further empower them to tailor the output to their specific needs. Finally, the VizAgent framework could be extended to provide more advanced data analysis and feature engineering capabilities such as data cleaning, feature selection, and even the generation of derived features or transformations.

5 Conclusion

The study introduced VizAgent, a new framework designed to address data visualization challenges. VizAgent guides users' intentions, automatically generates visualizations for specific tasks, and compares the quality of generated visualizations across different libraries. Our results demonstrated that the matplotlib library outperformed other visualization libraries like seaborn, plotly, and ggplot in terms of success rate. The findings suggest opportunities for further investigation and improvement, with new researchers exploring ways to enhance LLM performance for seaborn and plotly visualizations and experienced scientists focusing on ggplot. The VizAgent framework contributes to ongoing efforts in data visualization, providing a valuable resource for researchers, practitioners, and individuals seeking data-driven decision-making.

References

1. Nguyen, V.T., Jung, K., and Gupta, V.: 'Examining data visualization pitfalls in scientific publications', Visual Computing for Industry, Biomedicine, and Art, 2021, 4, pp. 1-15
2. Zhu, S., Sun, G., Jiang, Q., Zha, M., and Liang, R.: 'A survey on automatic infographics and visualization recommendations', Visual Informatics, 2020, 4, (3), pp. 24-40
3. Wang, C., and Han, J.: 'Dl4scivis: A state-of-the-art survey on deep learning for scientific visualization', IEEE transactions on visualization and computer graphics, 2022
4. Yang, W., Liu, M., Wang, Z., and Liu, S.: 'Foundation models meet visualizations: Challenges and opportunities', Computational Visual Media, 2024, pp. 1-26
5. Maddigan, P., and Susnjak, T.: 'Chat2vis: Generating data visualisations via natural language using chatgpt, codex and gpt-3 large language models', Ieee Access, 2023
6. Dibia, V.: 'LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models', in Editor (Ed.)^(Eds.): 'Book LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models' (2023, edn.), pp.
7. Wang, C., Thompson, J., and Lee, B.: 'Data Formulator: Ai-powered concept-driven visualization authoring', IEEE Transactions on Visualization and Computer Graphics, 2023
8. Marvin, G., Hellen, N., Jjingo, D., and Nakatumba-Nabende, J.: 'Prompt Engineering in Large Language Models', in Editor (Ed.)^(Eds.): 'Book Prompt Engineering in Large Language Models' (Springer, 2023, edn.), pp. 387-402
9. Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., and Wang, Y.: 'A survey on evaluation of large language models', ACM Transactions on Intelligent Systems and Technology, 2024, 15, (3), pp. 1-45
10. Khorasani, M., Abdou, M., and Fernández, J.H.: 'Web application development with streamlit: Develop and deploy secure and scalable web applications to the cloud using a pure Python framework' (Springer, 2022. 2022)
11. Omelina, L., Goga, J., Pavlovicova, J., Oravec, M., and Jansen, B.: 'A survey of iris datasets', Image and Vision Computing, 2021, 108, pp. 104109