Chat2PyViz: A Toolkit for Generating Data Visualization from Natural Language Prompts for Python

Henry Krukauskas and Almadi Shiryayev

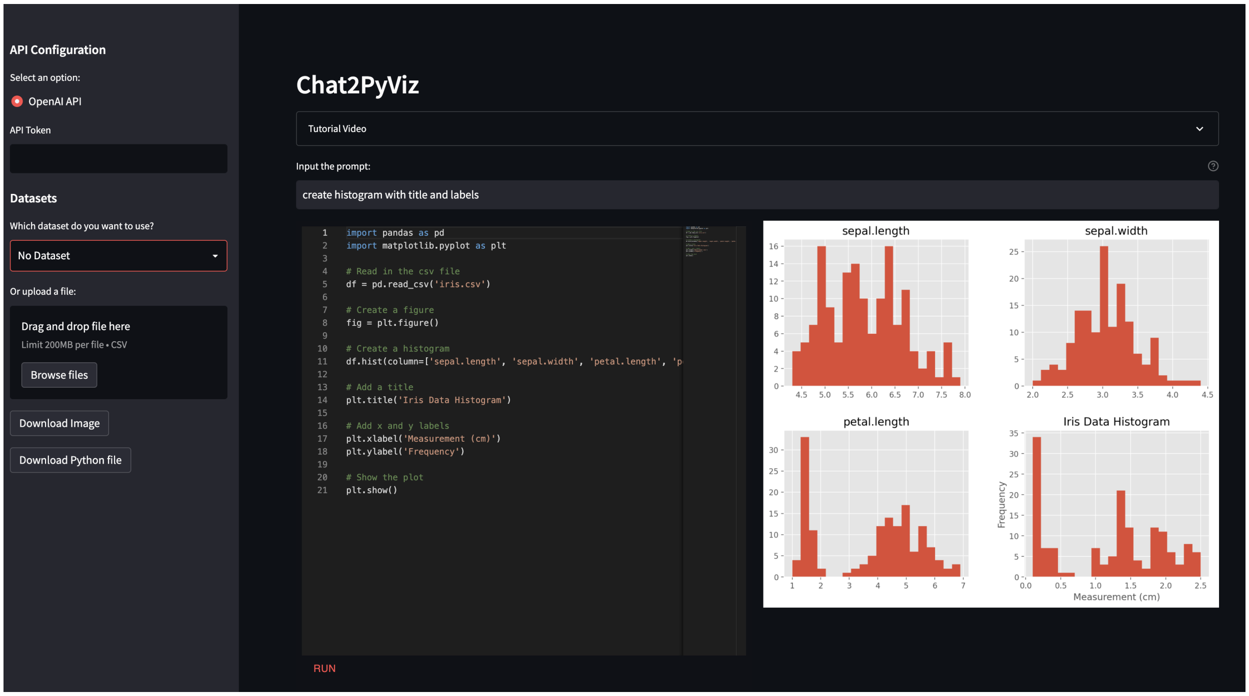

Fig. 1: Screenshot of Chat2PyViz Toolkit. First glance at the interface made for better user experience.

Abstract—The Chat2PyViz web toolkit presents a novel method of data visualization that incorporates both LLM and AI technologies into a user-friendly web-based interface. This platform facilitates the creation of tailored and interactive data visualizations, allowing users to make informed decisions based on their data. In addition, the toolkit offers an embedded code editor and the ability to rerun code with Streamlit, which is conducive to the continuous learning of prompt engineering and Python code implementation in the field of data visualization. Although the Chat2PyViz toolkit has some limitations, such as its dependence on the quality of user data and the restricted range of supported visualization techniques, these shortcomings can be addressed through further research and development. The authors intend to enhance the toolkit by incorporating advanced machine-learning models and expanding the repertoire of visualization techniques supported. Chat2PyViz represents a valuable contribution to the field of data visualization with its innovative integration of LLMs capable of generating Python code. The potential of Chat2PyViz to revolutionize data analysis and visualization tasks is substantial, and its continued development and refinement will be of significant interest to the research community.

Index Terms—Data Visualization, LLM, Prompt Engineering, Code Generation, Toolkit.

1 Introduction

In recent months, Generative AI has proven useful and relatively accurate technology to generate language-based responses, code scripts, and information retrieval. These models facilitate many daily tasks in a matter of seconds. Large Language Models (LLM), such as ChatGPT, GPT-3, and Codex, can generate code for data visualizations from user prompts, which usually require minor corrections [1, 2, 4]. However, currently developed toolkits need more functionalities to adapt the full capabilities of these language models within their user interface. Most of the time, the visualizations are limited to a small number of data visualization types, such as Vega [3, 5], and use only specific Python or Javascript modules [1, 2, 4], with no output code on the user's end.

In this research project, we aim to extend previous research on integrated data visualization toolkits with natural language prompting by employing a few large language models, such as OpenAI GPT-3.5-turbo and Codegen-6B-mono and by concentrating on user's experience for simple and advanced prompting. The proposed toolkit will allow users to input natural language prompts related to data analysis and visualization that would get back a generated code in Python, which will then be executed in the interpreter to produce the desired visualization. Moreover, the user will be able to continue prompting on top of their previously generated code or edit it locally in the interpreter to increase the accuracy of the desired output. The Feature Desk will enable users to decide which models to use for python code generation.

The toolkit supports OpenAI API and HuggingFace Inference API, which provides the user with more options than in other toolkits. Furthermore, the availability of the HuggingFace Inference API as a free

- *Henry Krukauskas is with New York University. E-mail: henry.krukauskas@nyu.edu.*
- *Almadi Shiryayev is with New York University. E-mail: almadi.shiryayev@nyu.edu.*

service enhances the inclusivity of this toolkit with respect to the utilization of open-source models. Implementing OpenAI's LLMs into this toolkit offers an advantage over traditional data analysis and visualization tools by reducing the time and effort required to write code and create visualizations, making data analysis more accessible to non-experts.

The main objective of this research project is to create a user-friendly and accessible toolkit for data analysis and visualization tasks by providing educational and insightful tips and guidance on how to prompt. By leveraging the power of natural language processing and machine learning, we aim to democratize data analysis, enabling more people to derive insights from complex datasets and learn how to implement and utilize Python for visualization tasks. This toolkit has the potential to serve as a valuable resource for researchers, analysts, and non-experts alike.

2 Related Works

The lack of essential features in existing toolkits has led to the further creation of multiple research papers proposing new iterations of them. Despite these attempts, the toolkits still fall short of providing an optimal user experience that is both efficient and educational. Many of these toolkits suffer from poor performance, limited functionality, and a lack of adaptability to different datasets and tasks. These limitations underscore the need for a more comprehensive approach to toolkit design that takes into account both user needs and the complexity of modern data analysis tasks. Moreover, it should be educational in terms of prompt engineering. By addressing these limitations and incorporating the latest advancements in machine learning and natural language processing, it may be possible to develop toolkits that are both efficient and effective in facilitating data analysis and exploration.

2.1 Vega-Lite Based Data Visualization Tools

Recently, there has been significant research towards developing natural language interfaces (NLIs) for data visualization. In Narechania et al. [3], the research project proposes a toolkit that can generate visualizations directly from natural language queries implemented with the Vega-Lite environment. The main novel contribution of the NL4DV toolkit is the development of a semantic parser that can translate natural language queries into structured analytic specifications for data visualization. The toolkit can also generate multiple candidate visualizations based on a single query and internally assign ranking scores for each candidate visualization. The NL4DV toolkit includes a number of built-in visualization templates, allowing for a variety of visualizations to be generated. However, this toolkit still lacks a lot of advanced functionalities that are provided in other toolkits integrated with LLMs.

From the NL4DV research paper, we identified a lot of useful functionalities in their user interface. Figure 2 shows the NL4DV's interface. The proposed toolkit incorporates additional functionalities to augment its overall appearance, a feature that captivated our attention. The main ideas from their research that we picked are the importance of seamless user experience and the ability to see the actual code for the provided data visualization, which promotes the user to learn the proposed technology's incorporated tools, such as Vega-Lite.

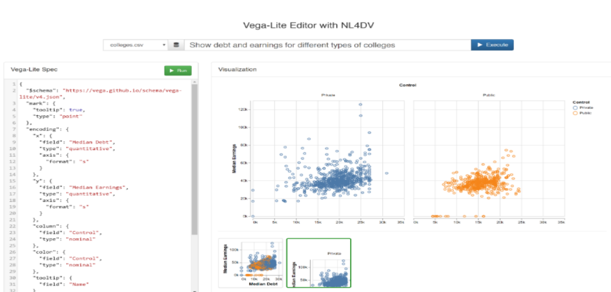

Fig. 2: A Vega-Lite editor that supports NL-based chart specification and presents design alternatives with their respective rankings. [3]

Another research paper proposed a similar toolkit called "VisFlow" which uses Natural Language Interface to generate interactive visualizations [5]. These toolkits are great and can generate many interactive graphs to describe the data. However, by integrating pre-trained LLMs to generate code that produces more advanced and easily tailored visualizations, further toolkits are more applicable to a wider variety of projects.

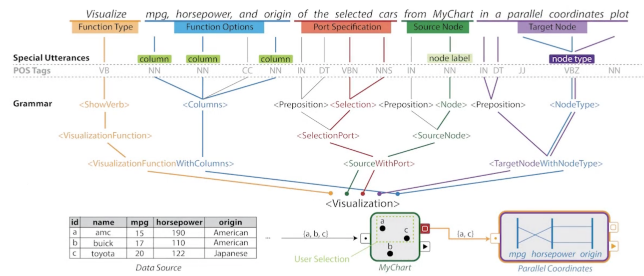

Fig. 3: NLI that helps to generate Vega-Lite-based visualizations with dataflow diagram editing in VisFlow. This figure displays a more detailed explanation of this NLI. [5]

2.2 Data Visualization Toolkits Integrated with LLMs

Furthermore, an existing toolkit we find essential to our research is the Chat2VIS system [1, 2]. The main novel contribution of the Chat2VIS system is leveraging the capabilities of pre-trained large language models (LLMs) such as ChatGPT, Codex, and GPT-3 to convert natural language queries into code for generating appropriate visualizations. By developing effective prompts for these models, Chat2VIS achieves better language understanding and more accurate end-to-end solutions than prior approaches. Additionally, the system offers a reliable approach to rendering visualizations from even highly unspecified and under-specified queries. However, Chat2VIS toolkit is limited since the user query's result is provided as a single graph without any further explanation or possibility to develop the prompt or visualization interaction. Thus, our research intends to provide a better user experience by adding an interpreter for the code, the ability to fix the code, and continue working with the query in real-time.

Figure 4 shows the logic behind the Chat2VIS system. The research provides a well documented prompt engineering strategy, which we intend to use in our project for case studies. Chat2VIS generates three visualizations during each iteration, which could be costly while using OpenAI API.

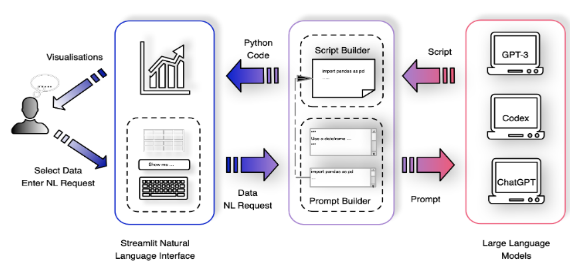

Fig. 4: The Chat2VIS architecture converting free-form query text into visualisations using the large language models GPT-3, Codex, and ChatGPT. [2]

Additionally, their research provides insight into how to engineer prompts to interact with LLMs and produce more accurate visualizations [1]. As part of our research, we will execute similar case studies to test our toolkit and learn how to further advance prompt engineering for data visualization.

The last toolkit with even more functionality than Chat2VIS is NLP2Chart [4]. This Python-based application makes the user experience more applicable and beneficial, as the ability to change the data features and attributes provide more control. The Feature Desk, which we intend to incorporate into our toolkit, has been inspired by the NLP2Chart tool. Additionally, this NLP2Chart is built using Streamlit, an open-source Python library used for building and deploying interactive data applications and dashboards. It allows data scientists and machine learning engineers to easily create and share their data projects, prototypes, and web applications with just a few lines of Python code. As Streamlit is a highly effective and user-friendly framework, we have chosen to adopt it rather than PyScript technology because PyScript is a fairly new framework that has a lot of performance issues. Our decision to build a web application from the ground up was motivated by a desire to maximize our control over the application's content display and functional capabilities. Figure 5 displays the NLP2Chart interface.

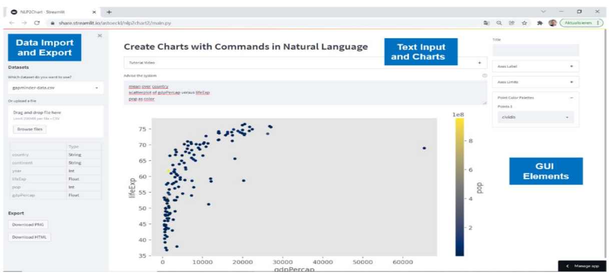

Fig. 5: NLP2Chart Interface [4]

2.3 Summary of Literature and Aims of the Research

The overview of the existing toolkits and their limitations provides a need for a more comprehensive approach to toolkit design that takes into account user needs and the complexity of modern data analysis tasks. Vega-Lite-based data visualization tools have many limitations, such as limited types of visualizations and models that could be produced, which highlight their inability to generate advanced and easily tailored visualizations, which can be addressed by integrating pre-trained large language models (LLMs).

Data visualization toolkits integrated with LLMs, such as Chat2VIS system or NLP2Chart, are essential toolkits for the research as it leverages the capabilities of pre-trained LLMs such as ChatGPT, Codex, and other LLMs to convert natural language queries into code for generating appropriate visualizations. However, these toolkits are limited in their ability to provide a satisfactory user experience as it provides the user's query result as a single graph without any further explanation or possibility of developing the prompt or visualization interaction. Therefore, the aim of the research is to provide a better user experience by adding an interpreter for the code, the ability to fix the code, and continue working with the query in real time.

Moreover, our research executed case studies similar to the ones provided produced in the Chat2Vis toolkit to test Chat2PyViz and learn how to further advance prompt engineering for data visualization. This research took into account the user's needs and the complexity of modern data analysis tasks while incorporating the latest advancements in machine learning and natural language processing to develop a comprehensive toolkit that is both efficient and effective in facilitating data analysis and exploration.

3 METHODOLOGY

In this study, we explored the ability of OpenAI *gpt-3.5-turbo* to generate Python scripts for visualizing data based on NL queries with explicit directions as to which types of graphs to generate. This model is the successor of OpenAI Codex, with the former released in 2023 as an advancement in programming AI. It is a more sophisticated version than its predecessor, offering better accuracy and faster processing times for code-generation tasks.

The model under consideration derives its name from its fundamental basis on the GPT-3 architecture. However, it is augmented with an extensive corpus of training data sourced from GitHub repositories in a similar manner to that of the OpenAI Codex model. This augmented design of the GPT-3 model uniquely positions it for natural language-to-code translation tasks, excelling in more than twelve programming languages, with Python serving as the language of interest in this specific study. As OpenAI models are considered to be state-of-art, we decided to concentrate solely on running our prompt engineering case studies on *gpt-3.5-turbo*.

3.1 Technologies Used to Create Chat2PyViz

The interface of our toolkit is made employing Streamlit and Streamlit Elements for a beautiful and interactive user experience. Additionally, Streamlit Elements provided us with access to Monaco Editor, which is a code editor that powers VS Code.

3.2 Natural Language Interface

The interface of Chat2PyViz is provided in Figure 1. The toolkit provides the user with the ability to choose whether to use OpenAI API or HuggingFace Inference API. After choosing the API service, the user is asked to provide the API key for that specific service and to input additional information, such as system prompt or necessary parameters for model inference.

The interface enables users to select or upload (as CSV) a dataset and enter free-form text describing their data visualization intent. An input box is provided for entering the NL free-format text. Additionally, the Monaco code editor and data visualization are generated each time NL free-format text is input into the input box and executed. After the initial results are presented, the user has the option to change the user prompt or fix code in the code editor and rerun it. When the desired visualization result is received, the user can download the Python code and the image of the specific visualization.

3.3 Prompt Engineering

In Maddigan et al. [1], the authors explain that the most effective method for obtaining the desired output from the LLM is to use the "show-and-tell" technique by supplying the system prompt an example together with instructions. The proposed system generates an LLM prompt consisting of two parts:

1. a system prompt built from a Python docstring and declared using triple double quotes """ at the beginning and the end of the definition;
2. a code prompt comprising of Python code statements that provide guidance and a starting point for the script.

Due to the necessity of a more general system prompt that would suit various Python visualizations, we decided not to provide LLMs with code prompts but rather stick to simple text prompts while still including the data type overview of the provided dataset.

Our general system prompt looks like this:

```
"""
    You are a data visualization bot that transforms
    the user's natural language instruction into the
    data visualization Python code.
    the code should include the import of CSV data
    file, which is {name_of_CSV_file}, the creation
    of a dataframe variable df, and the creation of
    a figure variable fig.
    The dataframe variable df includes these data
    columns and data types in them: {data_types_df}.
    Remember, you only need to return Python code.
"""
```

One of the examples of the variable `data_types_df` could be as shown below (this example is for iris.csv dataset included in our toolkit):

```
{'sepal.length': 'Float', 'sepal.width': 'Float',
'petal.length': 'Float', 'petal.width': 'Float',
'variety': 'String'}
```

This system prompt is simple and informative enough so the user is able to provide the user prompt with any level of ambiguity. Moreover, from case studies and continues usage of the toolkit, we derived that the best results for the visualizations can be achieved by providing the desired Python modules and frameworks to be used in the output code. The user should simply mention those Python libraries in its user prompt.

3.4 Chat2PyViz Evaluation

The capabilities of our Chat2PyViz toolkit to render visualizations based on Natural Language input via OpenAI API or HuggingFace Inference API are evaluated with 5 case studies where each case study will use various datasets, unique types of data visualization, and different prompt structure. It will help us to see the system's limitations, whether the system outputs proper code and visualization based on the user's query, and evaluate the system's usability for both simple and complex tasks.

Moreover, we will use System Usability Scale (SUS) to evaluate the toolkit's user experience and functionality. SUS is a widely used metric for evaluating the usability of a system. It consists of a 10-item questionnaire that measures the perceived usability of a system by asking users to rate their agreement with statements about the system on a five-point Likert scale (Strongly disagree, disagree, neither disagree nor agree, agree, strongly agree). The scores are then converted into a percentile ranging from 0 to 100, with scores above 68 indicating excellent usability. This metric will not only help us evaluate our toolkit but will also help us improve it based on the experience of actual users. The SUS survey will be available on our toolkit's demo website hosted on Streamlit Cloud.

4 Results

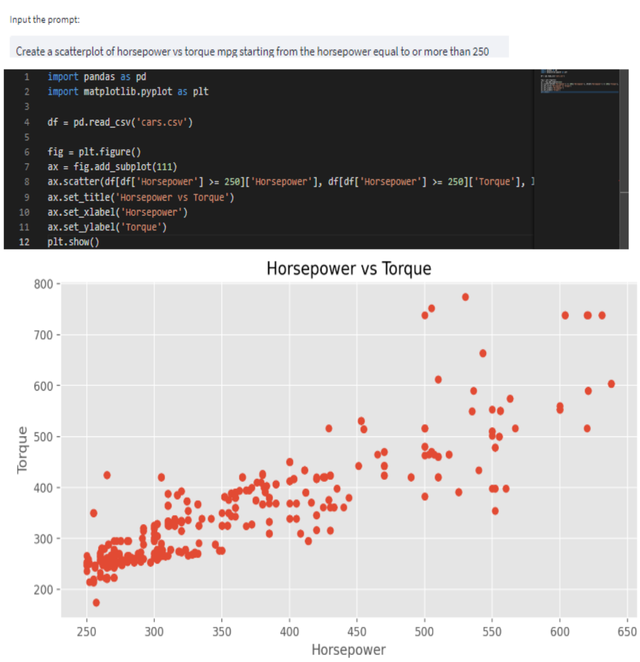

Fig. 6: Case Study 1: Cars Dataset

4.1 Case Study 1: Cars Dataset

The first case study is based on the *Cars* dataset that we initially uploaded to the toolkit. With the prompt *"Create a scatterplot with proper labels and title of horsepower vs torque mpg starting from the horsepower equal to or more than 250"*, *gpt-3.5-turbo* was able to generate the necessary code to create the visualization. It's important to note that the code generated by this model is only as accurate as the information provided to it, so it's always a good idea to double-check the code and make any necessary adjustments or corrections. The resulting visualization is shown in the Figure 6 above.

4.2 Case Study 2: Iris Dataset

As for the next case study, we used *Iris* dataset retrieved from the Kaggle. In this specific example, we wanted to see the performance of the toolkit when dealing with 3D visualization. The following prompt has been used in this example *"Create a 3D pca plot with proper labels and titles"*. PCA (Principal Component Analysis) is a useful technique for visualizing high-dimensional data in a lower-dimensional space. Thus, we decided to test it. Creating a 3D PCA plot of the Iris dataset can help visualize how the different features (sepal length, sepal width, petal length, and petal width) contribute to the variation in the data and how the different species of Iris (setosa, versicolor, and virginica) are separated in the lower-dimensional space. The toolkit's performance showed that it can easily make 3D visualization with the given data providing accurate Python code and visualization image. The resulting visualization is shown in the Figure 7 below.

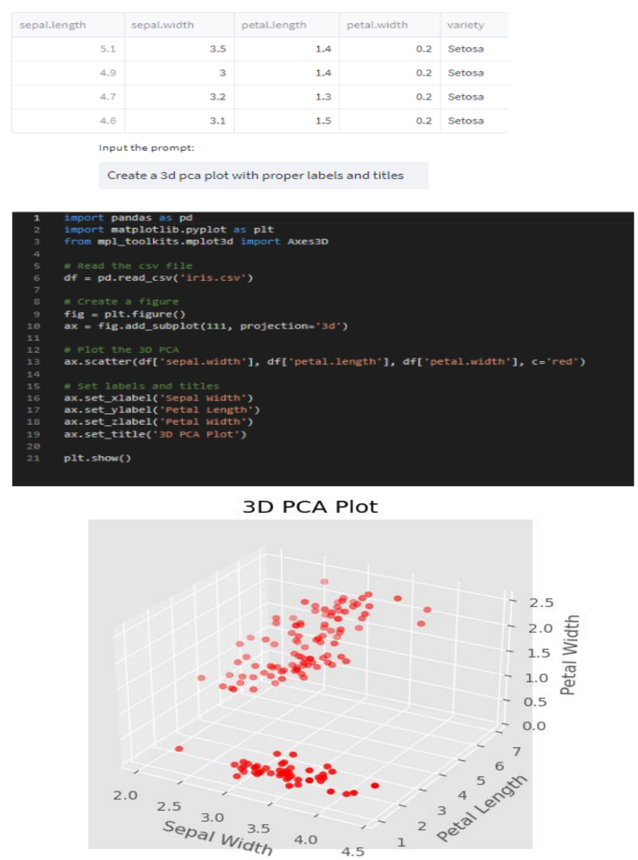

Fig. 7: Case Study 2: Iris Dataset 3D Visualization

4.3 Case Study 3: Iris Dataset Heatmap

In this case study, we also used *Iris* dataset to perform Heatmap Visualization. The following prompt has been used in this example *"Create a heatmap using seaborn"*. Heatmap is a great visualization tool for exploring the relationships between variables in a dataset because it allows you to quickly and easily identify patterns and clusters in the data. It helps to see where the data is concentrated and where it is more spread out. The Iris dataset contains measurements for four different features of three different species of Iris flowers. The four features are sepal length, sepal width, petal length, and petal width. A heatmap can be used to visualize the correlations between these features. That is why we decided that it is important to see how the toolkit performs when asked to create a heatmap. The resulting visualization is shown in the Figure 8 below.

Fig. 8: Case Study 3: Iris Dataset Heatmap

Fig. 9: Case Study 4: Cars Dataset Insights

4.4 Case Study 4: Cars Dataset Insights

In this Case Study, we used a different *Cars* dataset to see how the toolkit performs on general prompts. The following prompt has been used in this example *"show me some insights and create a visualization with proper labels and title"*. As a result, we get an accurate visualization with a legend shown in the top right corner. The resulting visualization was created without providing any instructions on what visualization type or Python library to use in the user prompt and shown in the Figure 9 below.

4.5 Case Study 5: Sales Dataset Insights Using Keyword

In our final Case Study, we used *Sales* dataset to see how the toolkit performs if we use a variable from the table as user prompt input. The following prompt has been used in this example *"price"* As a result, we get an accurate visualization with a legend shown in the left bottom corner showing *"price"* as an orange line. The entire visualization was created by providing just one keyword, which was a column's name *"price"*. Visualization is shown in the Figure 10 below.

5 Discussion

5.1 Study Limitations

Although our Streamlit web toolkit has shown promising results in creating data visualizations from user inputs and prompts, there are certain limitations that need to be acknowledged.

Firstly, one limitation of our toolkit is that it heavily relies on the quality and quantity of data provided by the user. If the data is incomplete, biased, or irrelevant, it may affect the accuracy and relevance of the generated visualizations. Therefore, it is important to encourage users to provide high-quality data that is relevant to the research question or problem at hand.

Secondly, the accuracy and effectiveness of the prompt engineering process heavily rely on the performance and reliability of the gpt-3.5-turbo model. Although this model has shown remarkable results in various natural language processing tasks, it may still produce errors or generate suboptimal prompts in certain cases. Therefore, it is important to keep in mind that the quality of the generated prompts may vary based on the input data and the specific research question.

Another limitation of our toolkit is the possibility of misinterpretation or miscommunication between the user and the generated code. Since the code is generated based on the user's input and prompt, it may not always accurately reflect the user's intentions or requirements. This may result in incorrect visualizations or suboptimal code that needs to be edited or re-run by the user.

Finally, the current version of our toolkit has limited support for different types of data and visualization techniques in Python. Due to this shortcoming, we are planning to add an advanced full prompting experience interface that will let users set the system requirements and user prompts as needed. Even with the limitations, Chat2PyViz produce remarkable code snippets for most types of visualizations available and is stable to run for long periods of time without any crashes.

5.2 Future work

In future work, we plan to integrate Vega-Altair with our Streamlit web toolkit for data visualization that utilizes the OpenAI API to access the gpt-3.5-turbo via prompt. This integration will enable users to not only analyze and interpret data with the help of AI but also to visualize the data in a powerful and interactive way.

To achieve this integration we will install and import the Vega and Altair libraries, load the dataset into the toolkit, and create Vega or Altair charts to visualize the data. We will then render these charts in the Streamlit app, which will be accessed through the OpenAI API to enable users to interact with and explore the data.

We believe that this integration will provide a valuable tool for researchers, data scientists, and other professionals who need to analyze and interpret complex datasets. By combining the power of AI with

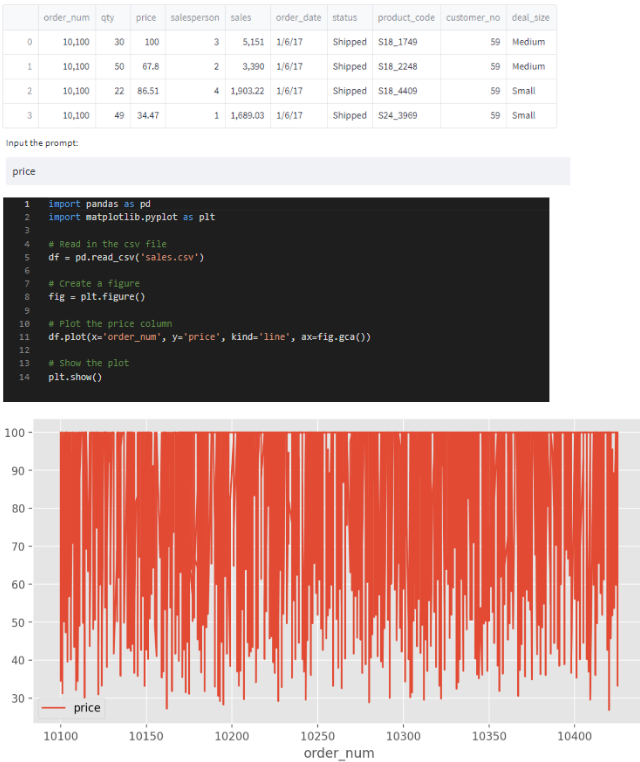

Fig. 10: Case Study 5: Sales Dataset Insights Using Keyword

interactive visualizations, we hope to enable users to gain new insights into their data and make better decisions based on these insights.

Furthermore, this integration can be extended to support real-time data analysis and visualization, allowing users to analyze streaming data and visualize it in real-time using Vega-Altair. This could have applications in various fields, such as finance, healthcare, and IoT, where real-time data analysis is critical.

We believe that the integration of Vega-Altair with Chat2PyViz represents an exciting avenue for future research and development in the field of data analysis and visualization.

Moreover, we plan to integrate data storytelling to our toolkit. Data storytelling is the process of using data visualization to tell a compelling story and communicate insights to stakeholders. We plan to explore ways to integrate data storytelling into our web toolkit to help users communicate their findings in a more effective way. This could involve providing users with templates for creating data-driven narratives, or incorporating features such as annotations and annotations to help users highlight key insights in their data.

Additional work may include implementation of collaborative features. Collaborative data analysis is becoming increasingly important in today's data-driven world. We plan to explore ways to implement collaborative features in our Streamlit web toolkit, allowing multiple users to work on the same dataset and visualization simultaneously. This will enable teams to work more efficiently and make better decisions by sharing insights and ideas.

And finally, we plan to provide users with a comprehensive history of their sessions. This will enable users to access their previous work and refer back to their previous prompts, code, and data visualization, which can be very helpful in many use cases. To implement this feature, we plan to introduce a history section that stores all the data visualizations that the user has created during the session. Each visualization will be accompanied by the associated code, which can be edited and rerun if the user needs to make any changes or fix any errors. The history section will allow users to quickly navigate to previous visualizations and related code, making it easier for them to revisit and build on their previous work. Additionally, this feature will help users to keep track of their progress and maintain a record of their analysis, which can be helpful for future reference.

5.3 Final Discussion

Another important measurement of our toolkit is the System Usability Scale (SUS), which we plan to evaluate after we collect an appropriate sample size of survey responses. The importance of this metric is to evaluate our toolkit's usability value but also to evaluate the current frameworks like Streamlit or Gradio (not implemented in this research). These technologies let ordinary users easily create interactive and beautiful web applications that do not require a deep understanding of front-end or back-end frameworks. This way, a lot of people are gaining more access to advanced programming technologies that are intuitive to use. The future progression of democratized technology is important; thus, this research aims to promote these awesome frameworks and evaluate their performance for more efficient improvement.

6 Conclusion

In conclusion, Chat2PyViz web toolkit represents an innovative approach to data visualization that integrates implementation of LLM and AI with a user-friendly web-based interface. By allowing users to input their own datasets and prompts, our toolkit enables the creation of customized and interactive data visualizations that can help researchers and practitioners make more informed decisions based on their data. The ability to use embedded code editor and rerunning the code with Streamlit extends the convenience of this toolkit.

While Chat2PyViz has several limitations that need to be taken into account, including its reliance on the quality of user data, the accuracy of prompt generation, and the limited range of supported visualization techniques, we believe that these limitations can be addressed through further research and development.

Moving forward, we plan to continue improving our toolkit by integrating more advanced machine learning models and expanding the range of supported visualization techniques. We also plan to explore ways to enhance the interpretability and accessibility of our generated code, which can help users better understand the underlying processes and make more informed decisions about their data.

Overall, we believe that Chat2PyViz represents a valuable contribution to the field of data visualization and has the potential to revolutionize the way researchers and practitioners approach their data analysis tasks. We look forward to continuing our research and development in this exciting field and welcome feedback and suggestions from the research community.

Supplemental Materials

In this section, we are including the links to the:

- GitHub repository of this research project (`https://github.com/Chat2PyViz/Chat2PyViz`).
- Chat2PyViz demo hosted on Streamlit Cloud ().

References

[1] P. Maddigan and T. Susnjak. Chat2vis: Fine-tuning data visualisations using multilingual natural language text and pre-trained large language models, 2023. 1, 2, 3

[2] P. Maddigan and T. Susnjak. Chat2vis: Generating data visualisations via natural language using chatgpt, codex and gpt-3 large language models, 2023. 1, 2

[3] A. Narechania, A. Srinivasan, and J. T. Stasko. NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries. *CoRR*, abs/2008.10723, 2020. 1, 2

[4] A. Stöckl. Natural language interface for data visualization with deep learning based language models. In *2022 26th International Conference Information Visualisation (IV)*, pp. 142–148, 2022. doi: 10.1109/IV56949.2022.00031 1, 3

[5] B. Yu and C. T. Silva. Flowsense: A natural language interface for visual data exploration within a dataflow system. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1–11, 2020. doi: 10.1109/TVCG.2019.2934668 1, 2