SmartMLVs: LLM-enabled Multiple Linked Views Generation for Interactive Visualization

Tian Qiu* — Fudan University

Fen Wang — Henan Institute of Advanced Technology, Zhengzhou University

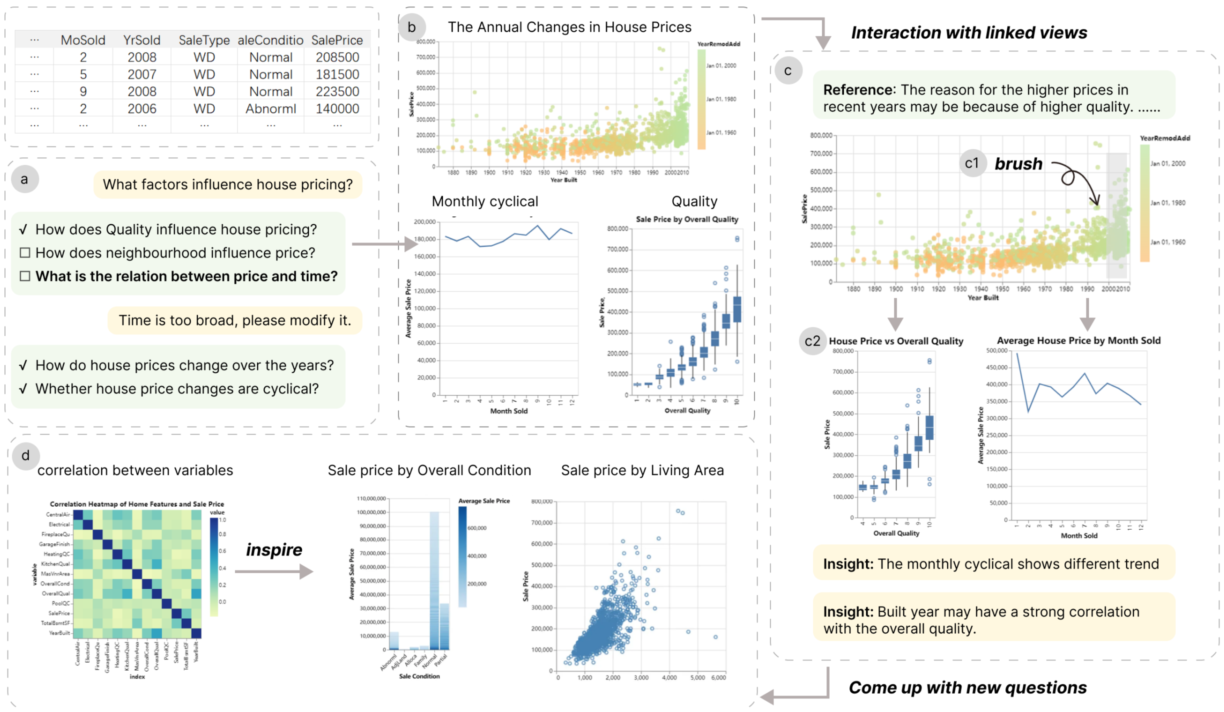
Shaohua Huang — Tongji University

Meng Guo — Fudan University

Yuheng Zhao — Fudan University

Jincheng Li — Beijing Normal University

Siming Chen† — Fudan University



···	MoSold	YrSold	SaleType	aleConditio	SalePrice
···	2	2008	WD	Normal	208500
···	5	2007	WD	Normal	181500
···	9	2008	WD	Normal	223500
···	2	2006	WD	Abnorml	140000
···	···	···	···	···	···

What factors influence house pricing?

✓ How does Quality influence house pricing?
☐ How does neighbourhood influence price?
☐ **What is the relation between price and time?**

Time is too broad, please modify it.

✓ How do house prices change over the years?
✓ Whether house price changes are cyclical?

Reference: The reason for the higher prices in recent years may be because of higher quality. ......

Insight: The monthly cyclical shows different trend

Insight: Built year may have a strong correlation with the overall quality.

a
b
The Annual Changes in House Prices
Monthly cyclical
Quality
Sale Price by Overall Quality
Average Sale Price
Month Sold
Overall Quality
Year Built
YearRemodAdd
Interaction with linked views
c
c1
brush
c2
House Price vs Overall Quality
Average House Price by Month Sold
Come up with new questions
d
correlation between variables
Correlation Heatmap of Home Features and Sale Price
inspire
Sale price by Overall Condition
Sale price by Living Area
Sale Condition

Figure 1: A usage scenario demonstrates how SmartMLVs operates: (a) The user begins by proposing the task, "What factors influence house pricing?" The system, with user involvement, decomposes this task. (b) It then automatically generates linked views. (c) The user interacts with these views and gains insights with the support of reference notes. (d) Finally, the user poses new questions, and the system provides responsive answers.

Abstract

Automating the generation of multiple linked view visualization is imperative for improving data analysis efficiency. Large Language Models (LLMs) offer substantial potential for enabling this automation, yet they encounter notable challenges in understanding complex queries and producing relevant interactive visualizations. To tackle these challenges, we introduce SmartMLVs, a system designed to harness LLMs for automatic interactive multiple linked views generation with human guidance. First, we analyze the challenges LLMs may encounter when designing visualizations in place of experts. To address these challenges, we gather the essential domain knowledge required for visual analysis process and propose a framework consisting of decomposition, visualization and linking. The decomposition process applies a human-AI interaction method to clarify user requirements. For each decomposed question, the generation process handles chart type selection, data processing and visualization generation. Finally, the linking process adds interactions for views and provides users with data insights. For better human-AI collaboration, we design a system for data exploration. Our system applies the entire framework, supporting users' interactive exploration with multiple linked views, and can iteratively generate linked views based on user feedback. We examine the effectiveness of our method through usage scenarios and evaluations.

Index Terms: Large Language Model, Visualization Generation

1 Introduction

Data visualization and visual analytics stand as significant instruments in presenting complex data clearly and supporting informed decision-making [31]. The process of manually designing and implementing visualizations is time consuming and highly specialized, which requires experts to possess a deep understanding of both data analysis and visualization techniques. The emergence of automated visualization generation methods efficiently solves these limitations.

*e-mail: tqiu24@m.fudan.edu.cn

†e-mail: simingchen@fudan.edu.cn, corresponding author

The natural language interface (NLI) methodology surpasses conventional visualization tools in terms of convenience and efficiency [39]. It empowers them to craft visualizations without resorting to traditional programming tools, thereby simplifying the data exploration and mining process.

Traditional machine learning algorithms have employed rule-based [32] and constraint-based [37] approaches to generate visualization charts. Recently, NLI algorithms based on LLMs have also made significant progress, outperforming traditional models in various aspects. For instance, LIDA [13] summarizes data using prompting and standardizes code generation formats, ensuring the correctness of visualization generation. ChartGPT [42] constructs a visualization generation dataset based on nvbench [27] and fine-tunes LLMs to accomplish NLI tasks. However, all these studies only generate non-interactive views. Users can only passively receive limited information, which reduces flexibility. The absence of filtering and screening functionality prevents the system from customizing the analysis to meet user-specific needs.

Interactive linked views enable users to explore multiple dimensions of data for a more comprehensive understanding, while interactions and view linkage enhance the exploration experience and discover potential insights. Therefore, our research aims to automate the process of interactive linked views generation. The research mainly faces three key challenges. Firstly, tasks proposed by users can be complex and ambiguous, leading to challenges in understanding them. Secondly, It is challenging to select targeted graphs tailored to diverse tasks, rather than being restricted to those commonly used in NLI (e.g. bar, line, pie and scatter). Lastly, Generating effective interactions and linked views that help users gain a deeper understanding of the data is challenging.

To tackle these challenges, we introduce *SmartMLVs*, a task-driven automatic visualization linked views generation system. Multiple linked views (MLVs) refer to several visual representations of a dataset linking by user interactions [33]. We utilize the LLMs in our system to interpret intent and generate code through prompt engineering. Specifically, we propose a method, which uses retrieval augmented generation (RAG) to inject visualization domain knowledge into LLMs, to support the generation of interactive visualization with multiple linked views. We first analyze the tasks and challenges involved in the automatic generation of linked views, and obtain the domain knowledge of visual analysis through investigation. Based on the challenges and knowledge, we propose a framework consisting of three key steps: decomposition, generation, and linking. The decomposition process is responsible for breaking down user-proposed tasks into visualization questions. The generation process manages chart type selection, data processing, and visualization creation, aiming to accurately produce diverse visualizations for each specific question. The linking process incorporates interactive linking between charts and generates chart descriptions, working collaboratively with users to uncover insights hidden within the data. Also, an interface is developed to support the collaboration of users and LLMs.

The main contributions of this study are reflected in three aspects:

- To the best of our knowledge, we are among the first to attempt automating the generation of interactive visualization with multiple linked views, reducing the cost of visual analytics and improving data analysis efficiency.

- We design a system to apply our framework, which supports users to tailor linked views according to their requirements and explore data in a code-free environment.

- We validate the effectiveness of our approach through a usage scenario and evaluation experiments, showcasing the effectiveness and generalizability of our framework.

2 Related Work

We review existing literature regarding NLI, task decomposition and LLMs based visualization generation.

2.1 Natural Language Interface

The integration of NLI with data visualization stands as one of the current research frontiers. Shen et al.'s comprehensive survey [39] systematically reviewed the integration of natural language with data visualization, pointing out directions for future research and practice. The research framework proposed by Arjun et al. [40]discussed the challenges and strategies in evaluating data visualization natural language interfaces. On the tooling front, Talk2Data [18] employed Bert to decompose tasks, automating the generation of multiple views to answer high-level questions proposed by users. Arpit et al. developed the NL4DV [32] toolkit, which generates data visualizations from natural language queries, providing developers with convenient development tools. The ADVISor [24] proposed by Liu et al. utilized pre-trained language models and deep learning techniques to achieve intelligent visualization and annotation of tabular data.

However, all these approaches employed a template based method to generate visualization, which lacks flexibility.

2.2 Large Language Model for Task Decomposition

Complex tasks typically demand a structured process for examining and addressing challenges. Task decomposition is a common way to help LLMs understand user needs and simplify problems.

Recent studies have proposed two strategies for task decomposition. Some studies decomposed the problem in processing order, which firstly separated the problem into several sub-problems and then solved each sub-problem in a specific order. For example, Least-to-Most prompting [54] broke an elaborate problem into many sub-problems and used the response to the previous sub-problem as the input to the next one. Creswell et al. proposed Selection-Interface [11], which divided each step of reasoning into two parts: selection and inference. This approach alternated between these two stages, creating a sequence of understandable and flexible reasoning steps that culminate in the ultimate conclusion. Plan-and-Solve prompting [48] formulated an initial plan at first and then decomposed the task into more manageable sub-problems. Some studies deconstructed complex problems by task types. Program of Thoughts Prompting [9]utilized pre-trained LLMs to decouple computations from reasoning tasks. ChartGPT also used this method to simplify the task.

However, these approaches do not perform task decomposition from a visualization perspective, making it challenging to derive multiple linked views from the resulting sub-problems. In our framework, we propose task decomposition strategies based on prompt engineering and self-consistency, enabling large models to generate more effective visualizations.

2.3 Large Language Models for Visualization Generation

Recently, large language models(LLMs), including GPT-4 [1], flan-T5 [10], LLAMA [43] and so on, perform well in understanding and generating natural language. Experts and scholars have applied LLMs in many fields and existing research results can prove that LLMs perform well in these areas, such as code generation [17], story generation [38] and solving problems [52].

Specifically, recent studies have applied LLMs to visualization generation. Some studies focus on prompting LLMs to generate visualization code. For instance, CHAT2VIS [28] generated Matplotlib [8] code in Python by prompting LLMs with column types and natural language guidance. Besides, LIDA [13] split the process into 4 steps and used richer and more sophisticated prompts to let LLMs generate Python code. Data Formulator [46] allowed users to

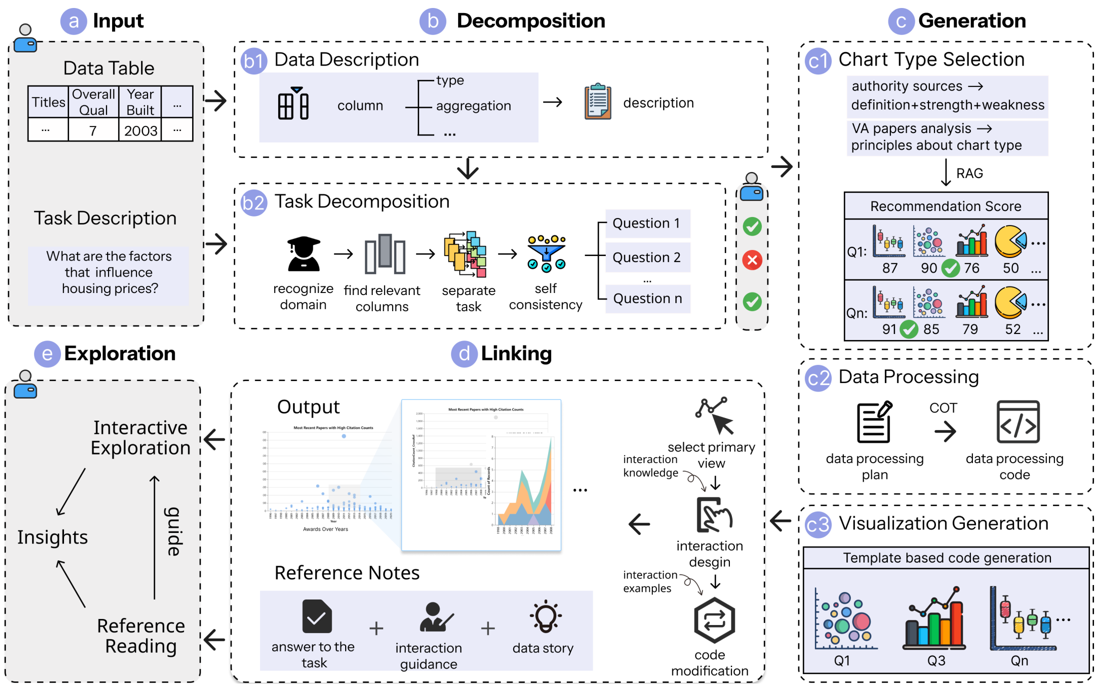


Figure 2: SmartMLVs framework: A framework including 3 stages is proposed based on the task, challenges and knowledge. In the decomposition module, LLMs separate a task into visualization questions. The generation process generates a chart for each question in three steps. The linking process adds interaction to charts and provides reference notes. Users gain data insights through interactive exploration and reference notes reading.

handle complex data transformation with the help of LLMs. Narrative Player [38] generated data stories based on data-rich documents and data. LEVA [53] used LLMs to enhance users' visual analytics workflow at multiple stages. Li et al. [22] comprehensively evaluated visualization generation models and proposed few-shot and one-shot methods to improve accuracy. Some studies focus on fine-tuning LLMs to generate visualization encodings. Chartgpt [42] fine-tuned open-source LLMs and adopted the chain-of-thought idea to split the task into 6 processes. Other studies focus on generating visualization and text in specific fields. For example, Data-Copilot [51] can automatically collect data from the economic field and generate visualizations according to users' queries. Tailor Mind [16] guided the application of fine-tuned LLMs to enhance visual interactions for domain-specific tasks.

However, existing works only generate a single visualization chart, which has a gap with the visualization requirements for multiple views. Therefore, we propose a framework that can generate linked views, aimed at bridging the gap in multi view visualization and assisting users in analyzing data more effectively and efficiently.

3 Problem Formulation

The goal of our system is to assist users in understanding, analyzing, and gaining insights from data by automatically generating linked views. We analyze the tasks that need to be accomplished and the challenges that may arise during the visual analysis process of LLMs. Then we gather the necessary knowledge for the generation process. Finally, we propose our framework based on the tasks, challenges and knowledge.

3.1 Tasks and Challenges

Through a review of numerous visual analytics surveys [36, 49], we have identified five common analysis steps. Visualization experts typically begin with analyzing visualization tasks, followed by data processing and visualization design. Next, users are invited to explore the data through the system, and the process concludes with the extraction of insights.

We envision how to use LLMs to replace visualization experts for these tasks and analyze the possible challenges. We will focus on addressing these issues when designing our framework.

Task: Visualization experts often clarify user requirements and system guidelines through user research and other related methods. We expect LLMs to decompose a complex task into multiple requirements by leveraging the data and tasks provided by the user, along with the background knowledge of the task's domain, to infer the user's intention. However, due to the lack of preliminary research and inspection, the task decomposition performed by large models may not align with the user's intent. Thus, addressing potential biases in how LLMs understand tasks presents a major challenge (**C1**). It can be a possible solution to let users make the final decision.

Data: When working with raw data, data analysts first need to develop a general understanding of the data, then perform data cleaning, and finally customize a model for data mining to prepare the data for designing visual charts [21, 26]. Given that the volume of data may be too large to provide directly to LLMs via prompts or assistants, we need to employ an appropriate data description method to convey as much relevant knowledge of the data to the model as possible in order to prevent hallucination (**C2**). As a result, significant features of the dataset should be extracted [47]. In addition, LLMs require the design of suitable algorithms and models to effectively mine the data (**C3**). Considering the complexity of the algorithm, a chain-of-thought method can be employed [50].

Visualization: Visual analytics systems often employ a variety of chart types to visually and aesthetically present data stories [45]. In our work, LLMs are employed to automatically select chart types. However, LLMs tend to have bias in chart type selection and often favour a limited range of charts. It remains challenging to get LLMs to recommend the most suitable chart types (**C4**).

Exploration: In the process of exploring a visual analytics system, users primarily analyze data through interaction, drawing conclusions by linking insights between different graphs. We aim for LLMs to automatically add interactions between graphs, facilitating seamless linking between them. However, it is hard for LLMs to understand the contextual linking between visualizations, making the process challenging (**C5**). How to instruct users to take full advantage of the interactions is also a big problem (**C6**), as a result of which, suitable guidance should be provided [23].

Insights: The ultimate goal of visual analytics systems is to assist users in gaining data insights and crafting data stories. We hope that the knowledge augmentation provided by LLMs will enable users to identify missing elements overlooked during the exploration process, leading to more comprehensive analysis results.

3.2 Requirements

Based on the above analysis, we formulate the following requirements:

R1 Proper data processing methods (C2, C3). Proper data preparation is the foundation of the visualization effort. In order to reduce the burden of LLMs, we need to design appropriate methods for data input, algorithm design, and code implementation.

R2 Visualization knowledge integration (C4, C5). In order to enable LLMs to generate linked views from the perspective of visual analysis experts, external knowledge should be integrated hoping that the system can complete the work of generating linked views from a more professional perspective.

R3 Involve users for pivotal points (C1, C6). Allowing users to participate in decision-making at key nodes can not only improve users' satisfaction with the results, but also improve users' sense of participation. Compared to having the user make all decisions directly, this human-machine collaboration reduces the user's meta cognition while improving the performance of our system.

3.3 External Knowledge

We have prepared external knowledge for LLMs from four aspects, including principles of chart type selection, basic information about charts, definitions and functions of interactions, as well as authoritative code examples for implementing interactions. We use RAG to integrate these knowledge into LLMs (**R2**).

Our primary goal is to understand how visualization experts select chart types when designing visualization systems. To this end, we conduct a quantitative analysis of visual analytics papers, examining how visualization charts are designed within their systems. Also, we refer to some visual analytics survey [12] to validate our conclusions. When selecting papers, we utilized the literature compiled in the VisPub dataset [19]. We screened all papers published at IEEE VIS conference over the past five years (altogether 641 papers). We identified those that focused on designing visual analysis systems and found 115 papers. Among these papers, we focused on work for application and removed those related to 3D visualization, scientific visualization, and deep learning interpretability. In total, we reviewed 52 papers, which are the latest work in their domain and represent a variety of data types, including spatio-temporal, network, high-dimensional, text and time-series. Our findings are presented in Figure 3. In the figure, we display the charts with the top 12 frequency.

The results show that the charts used in visual analytics systems are highly diverse, with charts such as force-directed charts, heatmaps, and parallel coordinates charts being used frequently, which LLMs may not typically generate. Additionally, the choice of chart types is strongly related to the data type. Also, some charts, like violin charts, are sometimes used in the visual analytics system to improve attractiveness. Our quantitative analysis will be integrated into LLMs to inform their decision-making when selecting chart types (**K1**).

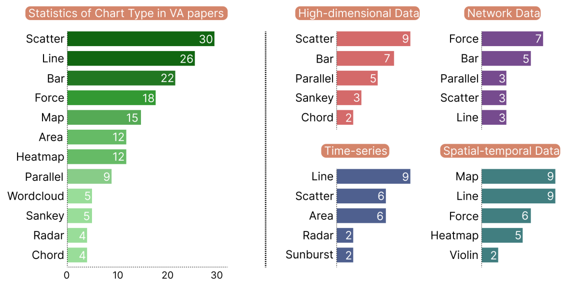


Figure 3: A quantitative analysis of the chart types used in visual analytics papers, showing the diversity of chart types and the strong relation between data types and chart selection strategies.

However, the tasks proposed by users may not always align with those typically addressed by visual analytics systems. We aim for LLMs to be generalizable in chart recommendation. LLMs are expected to not only understand which chart to suggest for a particular task, but also know why that chart is an effective choice. Therefore, we aim to equip LLMs with a deeper understanding of visualization charts. To ensure the reliability of this knowledge, we have compiled definitions, along with the strengths and weaknesses of various chart types, from authoritative sources such as reputable websites [41] and classic visualization textbooks [3, 14] (**K2**). During the chart selection process, we align the decomposed questions with these references to help LLMs make the most suitable choices.

With respect to interactions in visual analytics systems, we analyze the use of various types of interactions in visual analytics papers. The interactions mentioned in the papers we review primarily include selection, filtering, hovering, zooming, arrangement, and annotation. Among these, selection and filtering emphasize linking between charts, while hovering, zooming, and arrangement focus on interactions within a single chart. Annotation, on the other hand, is mainly used for recording insights. We gather the definitions and functions of these interactions from visual analytics survey (**K3**) [23].

Our focus is primarily on exploratory interactions, which are selection, filtering, hovering, zooming and arrangement. To create interactions, we download interactive visualization code examples from Altair [35, 44]. On one hand, these examples can help teach LLMs how to incorporate interactions into Altair code. On the other hand, they can enhance the LLMs' knowledge of the best interaction methods for each chart type (**K4**).

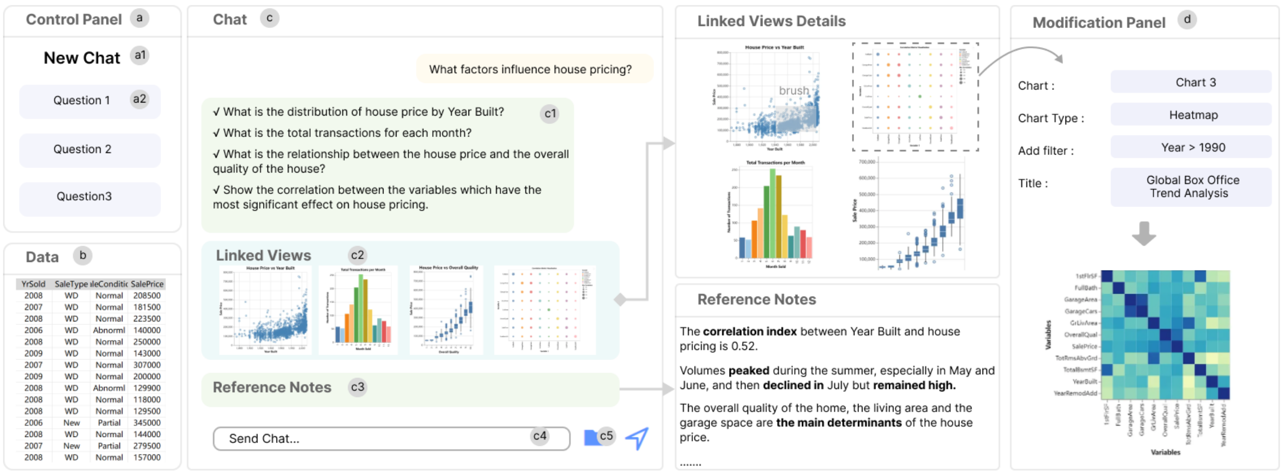
4 SmartMLVs

In this section, we introduce how our system is implemented and show how users interact with the system.

4.1 Framework

Based on the identified challenges and design requirements, we propose SmartMLVs. Our framework consists of three stages: Decomposition, Generation, and Linking. The user will upload the dataset and propose the questions in the beginning (Figure 2a), followed by LLMs generate interactive multiple linked views automatically, and finally users can get insights by interaction with linked views with the guidance of reference notes (Figure 2e).

Decomposition: To address the tasks and datasets provided by users, we aim to generate data descriptions that summarize the dataset (Figure 2-b1) and decompose the task into a series of questions. We instruct LLMs to first identify the domain of the task, then extract the relevant data from the dataset, and finally generate a list of decomposed questions needed to complete the task (Figure 2-b2).



YrSold	SaleType	ıleConditic	SalePrice
2008	WD	Normal	208500
2007	WD	Normal	181500
2008	WD	Normal	223500
2006	WD	Abnorml	140000
2008	WD	Normal	250000
2009	WD	Normal	143000
2007	WD	Normal	307000
2009	WD	Normal	200000
2008	WD	Abnorml	129900
2008	WD	Normal	118000
2008	WD	Normal	129500
2006	New	Partial	345000
2008	WD	Normal	144000
2007	New	Partial	279500
2008	WD	Normal	157000

What factors influence house pricing?

√ What is the distribution of house price by Year Built?
√ What is the total transactions for each month?
√ What is the relationship between the house price and the overall quality of the house?
√ Show the correlation between the variables which have the most significant effect on house pricing.

The **correlation index** between Year Built and house pricing is 0.52.

Volumes **peaked** during the summer, especially in May and June, and then **declined in** July but **remained high.**

The overall quality of the home, the living area and the garage space are **the main determinants** of the house price.

....

Chart :	Chart 3
Chart Type :	Heatmap
Add filter :	Year > 1990
Title :	Global Box Office Trend Analysis

Figure 4: An overview of our interface: (**a**) A control panel to start a new chat (a1) or refer to a specific history (a2). (**b**) Data preview window to help users get familiar with data. (**c**) The area users upload files (c5),propose tasks (c4), select questions (c1), view and interact with charts (c2), and understand reference notes (c3). (**d**) A modification panel to help users modify charts.

Additionally, a human-AI interaction method is employed to refine and polish these questions (**R3**).

Generation: For each question decomposed in the previous step, we aim to generate a suitable visualization chart. This process is completed from three perspectives: chart type, data, and visualization.

When selecting chart types, we first instruct LLMs to learn both the knowledge about visualization charts (**K2**) and the decision-making processes in real scenarios (**K1**) using RAG. Next, LLMs are asked to assign a score to each chart type, evaluating how suitable each one is for addressing the question. Ultimately, the chart type with the highest score is recommended (Figure 2-c1). When processing data, LLMs are instructed to employ a chain of thought method to solve the problem (**R1**). They may first generate a data processing plan including data cleaning and algorithm designing, and then generate the code according to the plan (Figure 2-c2). When generating visualizations, we first generate the visualization code, and our system automatically executes the code to display the charts. (Figure 2-c3). Finally, we generate multiple views according to the number of questions.

Linking: Our goal is to add interactions across multiple views and generate reference notes to describe the charts and insights, providing users with space for exploration (Figure 2d).

When achieving linking between charts, we first instruct LLMs to generate an interaction plan for these charts, which aims to choose which chart users interact with and how users interact with them, and which charts to respond to the interactions. In this step, we give LLMs the knowledge about the design principle of each interaction (**K3,K4**). Then, we instruct LLMs to add interactions based on the examples downloaded from Altair (**K4**). By executing the modified code, we generate multiple linked views based on proposed tasks. Additionally, we provide users with reference notes to assist them in exploring the linked views and gaining insights (**R3**).

In conclusion, SmartMLVs begin by receiving the task and dataset, then employ a human-computer interaction method to generate multiple linked views. Users are then guided to interact with the system through reference notes, which provide insights and prompt further exploration ideas.

4.2 Decomposition

The decomposition process starts with analyzing the data and aims to recommend a variety of questions to assist users in understanding various aspects of the task.

Initially, users upload their data and propose a task. We summarize and convert the data into a structured format, named data description. We extract the structure of the data table, along with detailed information for each column (**C2**).

The decomposition process then proceeds through three steps. First, LLMs are instructed to identify the domain of the task and data, then play the role of the domain expert. In this way, we aim to improve the profession of the result. Second, LLMs are tasked with selecting all columns related to the user's task, ensuring the completeness and integrity of the task decomposition. Third, LLMs decompose the task into specific questions based on the correlation between variables and the relationship between independent and target variables. In this step, we instruct LLMs to use the raw column names directly in the questions, avoiding misunderstanding in subsequent steps (**C1**). For example, for questions containing ambiguous expressions "What is the most popular product?", we automatically change the word 'popular' with a specific column name 'sale volume' and modify the question into "Which product has the highest sale volume?" to avoid hallucination. We repeat the third steps until all the related columns are considered.

Then, we apply a self-consistency mechanism for validation. Self-consistency is a prompt engineering method that involves instructing LLMs to generate multiple responses and then selecting the most frequently occurring answer. This approach helps reduce hallucinations and improves the overall quality of the generated answers. Any questions not relevant to the task, or those that cannot be addressed using visualization charts, are excluded. Additionally, questions that do not meet the specified requirements, such as those containing ambiguous expressions, are regenerated to ensure accuracy and clarity.

Finally, users are involved in selection or modification. Users may select various questions which meet their needs. If users are dissatisfied with the decomposition results, they may tell LLMs how to modify the questions or ask LLMs to decompose again.

4.3 Generation

For each question decomposed in the previous process, our goal is to generate a single visualization chart that ensures both accuracy and diversity. To achieve this, we must address model correctness (**C3**), ensure the diversity of charts (**C4**), and guarantee the executability of the visualization code. We tackle these challenges using the following methods.

Chart Type Selection: LLMs tend to show bias toward certain chart types, resulting in sub optimal performance in this area. However, in current research on natural language interfaces, visualization recommendations are typically restricted to a narrow range of chart types.

To solve the limitation, we enhance LLM performance by providing them with external knowledge, and we have selected Retrieval-Augmented Generation (RAG) as the solution (**C4**). The principle of RAG is to retrieve relevant background information related to the input through a retrieval module and provide this context to the generation module, thereby improving the accuracy and relevance of the generated results. Our background knowledge contains examples of how visualization experts select chart types the knowledge about different chart types. We select OpenAI Assistant to realize RAG, which provides a Vector Store to save files and applies various optimization methods (like query optimization, semantic searches and re-ranking) to improve retrieval performances.

Then, we use the concept of recommendation score ranging from 0 to 100 to estimate whether one type of chart is suitable to solve the question. The recommendation score is given by LLMs based on the question and knowledge. If the question fits the definition of the chart and falls within a specific strength interval, a higher recommendation score is obtained. Conversely, if it is clearly mentioned in the corpus that the chart is challenging in solving this type of problem, a lower recommendation score will be obtained. Finally, charts with the highest recommendation score will be selected.

Data Processing: We aim to filter relevant data and apply appropriate algorithms or models to process the data. We apply chain of thought (COT) to process the data. First, LLMs generate a detailed data processing plan which includes the algorithms and data processing process. Algorithms and mathematical models include machine learning algorithms such as dimensionality reduction, clustering, and classification. Data processing includes normalization, outlier handling, missing value handling, etc. The idea of code processing will be listed in as much detail as possible to avoid errors. Specifically, we will pay attention to time variables, which we require LLMs to use a standard format to show them. Second, we generate the data processing code based on the code template and data processing plan. The code template includes the necessary libraries, specifies the format of the input and output, and provides fixed areas for LLMs to fill. To ensure the executability of the code, LLMs can only modify the code in a specific region.

Visualization Generation: Visualization is automatically generated based on the question, chart type and data. We choose Python to generate visualization. Due to Python's strengths in data processing, we aim to maintain consistency across the various code generation stages to prevent potential issues during data transfer. For visualization libraries, we chose Altair and pyecharts for coding. Compared to other libraries like Matplotlib, these two libraries offer standardized code writing, are less prone to errors, and provide better support for interactive user exploration.

Existing work has been able to achieve accurate generation of single charts. In this work, we use lida [13] as our baseline, using an identical template-based method for visualization generation.

4.4 Linking

Users are hard to get sufficient insights from multiple non-interacted charts and now we aim to add linking and insights to these charts.

Linking: We incorporate linking interactions to enable users to explore data more deeply (**C5**). Through linking, we connect individual visualization charts to work together in addressing the task.

First, a primary view is selected for user to interact with, and the other views respond correspondingly to the user's actions. The primary view is selected based on the relation to the task [15], information complexity [6] and logic relation between views [34]. We hope the primary view to be task-relevant, informative and concise.

Second, we design interaction methods for the primary chart based on the knowledge. For example, in a scatter plot, we may implement a "brush" tool to allow users to select a region of data, while in a bar chart, we can design a "click" function that provides detailed information about a specific bar.

Third, we automatically generate linking interaction code according to the example. These interactions not only enable users to engage with individual charts but also allow for deeper data exploration in a multi-view environment by linking the charts together, enhancing the efficiency and effectiveness of the analysis.

After the process, users can interact freely and get real-time feedback.

Reference Notes: Text plays a significant role in helping users gain insight [2, 4]. In order to assist users in exploration and expand their insights derived from our system, we lead LLMs to generate reference notes that accompany the linked views. We mainly focuses on three aspects:

(1) Answering the Central Question. At the heart of every data visualization is a key question or set of questions we aim to address. This ensures that the primary objective of the data analysis is communicated.

(2) Interaction Guidance (**C6**). To ensure users fully understand and utilize the interactive features provided by the system, we need to provide clear operational guidance in our explanations.

(3) Extract stories behind data. We extract some insights that can be obtained from single visualization chart or linked views, including high-lighting trends, notable patterns, etc. We will also infer the underlying reasons behind these insights to present data stories.

Modification and Error Handling Process: In order to implement customization and improve the user experience, our system needs to support modifications and error handling. We input the code and the error information (or the comments made by users) into LLMs to instruct them to regenerate the code.

4.5 Interface

We design an interface to accommodate our framework and the system overview is presented in Figure 4. This section introduces the basic functions and design philosophy of the system.

The system overview section displays the framework and components of the entire system. As shown in Figure 4a, the control panel determines the content displayed in the chat interface. We provide a clear and concise method, allowing users to easily initiate new conversations or browse through historical chat records. By clicking the "New Chat" button (Figure 4-a1), users can quickly start new conversations. Meanwhile, selecting a specific chat record (Figure 4-a2) enables users to conveniently review past exchanges, thereby enhancing the efficiency of information search and browsing.

The data display area (Figure 4b) is designed as an intuitive interface for showing relevant data and information to the user. Once a file is uploaded via the "File Upload" button (Figure 4-c5), the data will be displayed in this area. This design aims to help data analysts become acquainted with the data and to facilitate easy data viewing during system interaction, thus better understanding and analyzing the information.

The chat interface (Figure 4c) is a key part of the system's interaction with the user. It allows users to select questions for decomposition (Figure 4-c1), view and interact with linked views (Figure 4-c2), and obtain in-depth explanations (Figure 4-c3). Also, a dialogue box (Figure 4-c4) is provided, enabling users to propose tasks and suggest modifications at any stage. This allows the system to adapt more flexibly to the needs of data analysts while fostering their deep understanding of the issues.

Finally, the modification panel (Figure 4d) serves as a supplementary part of the system, offering convenient references for users unsure of how to modify data or views. This enhances the system's user-friendliness by providing intuitive modification suggestions and guidance, helping data analysts more easily realize their intentions.

5 Usage Scenarios

In this section, we use a usage scenario to illustrate the effectiveness of the system.

In this usage scenario, we attempt to understand the factors affecting house prices. We employ the house pricing dataset [5], an EDA dataset from Kaggle, which includes over 60 factors that may influence house pricing. It is difficult for data analysts to find the most important variables in such a large-scale data set and make an accurate analysis. Therefore, employing SmartMLVs to show the different dimensions of the data and generate targeted interactive visualizations can efficiently and effectively aid in understanding the impact of various factors, allowing for interactive exploration of context-specific sales strategies under different scenarios.

After inputting the question "What are the factors influencing the house pricing?" and the associated data into SmartMLVs, the system breaks the task down into several questions (Figure 1a). However, considering that the sale date is too detailed, we provide feedback for modifications and receive the revised results. Taking into account all the output questions from the system, we choose three of them.

The system soon returns the result (Figure 1b). Analyzing the plot of house prices as a function of construction year reveals a general upward trend, with the rate of increase accelerating in recent years. Examining the change curve of the sales month indicates that house prices peak in September, with a trough occurring from April to June, showing no clear cyclical pattern. Additionally, the box plot of quality versus home price demonstrates that higher-quality homes tend to command higher prices, with the rate of increase appearing to be approximately exponential.

To uncover additional insights, interactions were used to explore the data further (Figure 1c). The reference note hypothesized that houses built in recent years tend to sell for higher prices due to superior quality. To investigate, the user filtered the data to include only houses built from 2000 onwards (Figure 1-c1), prompting the other charts to update in response to this interaction (Figure 1-c2). The analysis revealed that the comprehensive quality ratings of houses constructed after 2000 were 4 and above, supporting the conjecture proposed in the reference note. Additionally, the cyclical trends of house prices by month exhibited different patterns. Through this interactive exploration, users gained new insights and a deeper understanding of how these factors influence house prices.

Building on the insights gained above, the user sought to explore the associations between variables and investigate additional factors that might influence house prices. To address these questions, the system generated three additional charts (Figure 1d). From the heatmap, it was observed that the correlation coefficient between construction year and overall quality is approximately 0.5, indicating a moderate correlation. Furthermore, several other potential factors influencing house prices were identified through the analysis.

Overall, our system efficiently identified several factors influencing house prices (like built year, overall quality, living area, etc.), provided possible explanations through interactions and reference notes, and revealed associations between variables.

6 Evaluation

Two evaluation experiments were carried out to evaluate our system.

6.1 Ablation Study

To validate various design choices of our Generation process, we conducted an ablation study. We investigated the impacts of our proposed RAG based chart type selection and COT based data processing.

Table 1: Result of the ablation study. VG, CTS, DP refer to visualization generation, chart type selection and data processing respectively. The result proves the effectiveness of each module.

Model	ACC (%)
VG (lida, baseline)	63.2
VG+CTS	64.4
VG+DP	68.9
VG+CTS+DP (Generation)	70.1

6.1.1 Experiment Setup

We are unable to find an authoritative dataset for natural language generation of multiple linked views. However, the nvBench dataset [27] provides a mapping of natural language to a single visual chart, accomplishing a task similar to the Generation module in our framework. This dataset can be used to test the accuracy of that component. Therefore, we employ the nvBench dataset to conduct the ablation study. Tian et al. [42] constructed a dataset based on nvBench for NLI, containing a subset of 1,918 ¡NL, VIS¿ pairs, focusing on generating visualizations from a single table. We conduct our ablation studies on the test set which contains 378 cases.

In the ablation study, we removed the chart type selection module (CTS) and data processing (DP) module and used the visualization generation (VG) process, that is, lida [13], as our baseline, which just gives LLMs a template for code generation. We then added the chart type selection module and data processing module separately on the baseline. Finally, we combined both of these modules to update the scores.

6.1.2 Experiment Metrics

In the evaluation, we evaluated the accuracy of the Generation process. We leveraged the match accuracy between the ground truth and the results generated by our methods. We define the result as 'correct' if the result is exactly the same as the ground truth, including the selected data, chart type, sort type and data processing methods. Our accuracy rate is defined as: ACC = C / T * 100% where C stands for the number of correct results and T stands for the total number of results.

6.1.3 Experiment Results

The result is shown in Table 1. The accuracy rate of the baseline is 63.2%. Only with either the chart type selection module and data processing module, the performance rise to 64.4% and 68.9% respectively. When combining both of them, the accuracy rate increases to 70.1%, superior that of the baseline by 6.9%.

To analyze the effectiveness of RAG based chart type selection, we analyze the result generated by model 'VG+CTS'. It is surprising to find that some chart types that are not contained in the ground truth occur in our results, like sankey, word cloud and box plot, which can answer the question more intuitively than bar chart (ground truth). Despite their negative impacts on the accuracy rate, it is a good phenomenon proving that our method can recommend correct and diverse chart types. Therefore, though the accuracy rate does not improve remarkably, we can also show the effectiveness of this module.

We also validate COT based data processing process. Compared to the baseline, the accuracy rate improves 5.7%. We analyze the problems which are not solved by baseline and are solved by the model 'VG+DP'. Data processing module performs well in the following two aspects: 1.finding corresponding data columns and 2.using appropriate aggregation, filter and sort conditions, allowing our system to handle some hard problems.

Our Generation process performs much better than the baseline. However, it still encounters some problems when reading the data, handling components and selecting chart types in the dataset. 18

cases do not generate results because we fail to load the dataset using Pandas. 20 cases have problems handling components in the dataset, like 'Timestamp(1989-01-01)' which is recognized as a string in Python. 75 cases do not select the same chart types as the ground truth, but they do not influence users' understanding.

6.2 User Study

We conducted a user study to show the effectiveness of our system.

6.2.1 Experiment Setup

Participants: We invited 32 participants (18 males and 14 females, 25 postgraduate students and 7 undergraduate students) to participate in the user study. All the participants have data analysis experience and have data analysis needs in their work. All of them were not familiar with the dataset we use in the experiment before.

Procedure: Our user study contains two parts, a controlled experiment and a free exploration process. Both experiments use GPT4 to support our system.

In the controlled experiment, we divided the participants randomly into two groups with 16 participants each. Experiment group members used our system to explore the data. Control group members could use any data analysis tools, including but not limit to GPT-4, Python and Excel. We used house pricing dataset to carry out the experiment, which is the same dataset as the usage scenario. We gave participants a task for exploration: What factors influence house pricing and how these factors influence house pricing. Besides the task, users were encouraged to find more data insights. Time for exploration was limited to 20 minutes and users were interviewed about the conclusion they reached and the basis for their conclusions.

In the free exploration process, all participants could use our system to analyze their datasets. Then, a questionnaire and a review were conducted.

Measurement: For controlled experiment, we evaluated the result based on the quality of insights. We evaluated the insights from four perspectives: comprehensiveness, accuracy, depth and solidity. Comprehensiveness and accuracy evaluate whether the insights correctly and fully identify the factors influencing house prices. Depth assesses the extent of the user's understanding of how specific factors impact house prices. Solidity examines whether the user's conclusions are well-supported by evidence. To assist the evaluation process, we extracted answers to the question from the EDA notebook as ground truth.

For free exploration, we designed a questionnaire to gather user satisfaction levels with our system. Our questionnaire consisted of 8 questions, with 5 about the effectiveness of each component, 1 about the practicality of the framework and 2 about the usability of our system. We used a 7-point Likert scale to collect their response. Also, we prepared some questions for interview, including their feelings about using our system, the scenarios when they prefer to use our system and the pros and cons of our system.

In the whole process, we tested the time required to generate each module to evaluate the efficiency of our system.

6.2.2 Result Analysis

The results of controlled experiment are listed in Figure 5. We used mann-Whitney U test to judge the significance of the result. As is shown in the figure, our system significantly outperforms the control group in terms of depth and solidity. Examining the control group results reveals that most users chose to use ChatGPT directly for data analysis. While GPT provides answers by analyzing the relationships between variables and house prices and highlighting the top 10 factors with the highest correlation coefficients, it merely lists the variables without delving into the extent of their influence on house prices or presenting visual results unless explicitly requested, which still imposes a considerable analytical burden on the user. Users relying on Python for analysis must select variables individually for examination, and they often face the additional challenge of resolving repeated code errors. In contrast, SmartMLVs automatically assist users by generating insights from multiple dimensions—such as patterns, associations, trends, and anomalies—through linked views and reference notes tailored to data decomposition tasks. Furthermore, it provides supporting explanations, offering significant assistance to data analysts in understanding and interpreting data.

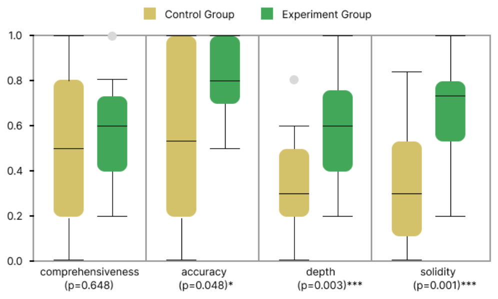


Figure 5: Results of the user study. The experiment group outperforms the control group significantly in depth and solidity. The p value stands for significance level with *,**,*** stands for p¡.05, .01 and .005, respectively.

During the free exploration process, participants utilized SmartMLVs to analyze datasets from diverse fields, including electricity, movies, soccer, stocks, and more. Additionally, Figure 6 summarizes the findings from the questionnaire survey. As is shown in the figure, the average satisfaction rate for our system is 5.75, indicating a generally favorable user perception. Particularly, users are highly satisfied with chart type selection and reference notes. However, users are not that satisfied with visualization charts generation because of its instability, which we will discuss in Section 7. When it comes to usability, the majority of participants think our system is easy to use and prefer to use our system.

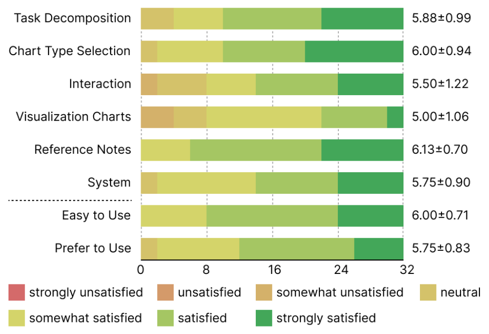


Figure 6: Results from the subjective questionnaires. The stack bars indicate feedback scores and the rightmost column shows Mean±STD.

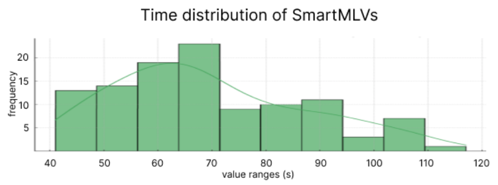
The result of the efficiency evaluation is shown in Figure 7. When using GPT4, the generation of multiple linked views takes approximately 68 seconds on average. Analyzing the efficiency of each module, we found that the majority of the time is spent on visualization generation and interaction generation, which take an average of 23 seconds and 25 seconds, respectively.

Participants estimate our system from the following perspectives.

Saving time and effort. Our system eliminates the tedious work of coding. All users who have utilized our system provided feedback indicating that our system is more efficient than manual data processing. In the comparative study, one participant in the control group first used ChatGPT to understand the data, but he thought

LLMs do not understand his data well, then he instructed LLMs to generate data processing code. However, he encountered problems of a coding error when using Python to analyze the data, preventing him from completing the task on time. Another control group participant encountered a similar problem. Faced with the complex dataset, he attempted to take into account as many factors as possible in a limited time, but fails. He told us, *the dataset is too large and it requires a significant amount of time to understand each column*. In the free exploration stage, both of the participants used our system to explore the house pricing dataset and evaluated our system to be efficient. He told us that he prefers to use our system when he faces an unfamiliar dataset, which can help him quickly know the data.

Deepen data understanding. Interactive linked views and reference notes allow for a more comprehensive and in-depth understanding of the task and data at hand. One participant uploaded a football dataset and want to explore the impact of age on the football players. Though he identified a negative correlation between age and value, some anomalies point catch his attention. He notes that some high-level players, despite being old, still maintain high value. *This is significant information. I may miss this detail if not using this system*, he further tells us. Another participant uploads a dataset about market risk management and wants to find high-risk business. Though our system does not solve his task because of the complexity, our system stills generates linked views about the relation between variables. He comments on our system, these charts play an important role in the actual market management, they are all valuable.



Time cost of each module

Module	decomposition	chart type selection	data processing	visualization generation	interaction generation	SmartMLVs
MEAN(s)	15.57	12.91	5.96	23.18	25.56	68.5
STD(s)	4.47	3.19	3.68	10.59	7.24	18.54

Figure 7: The efficiency of our method. It takes an average of 68 seconds to generate multiple linked views. The time distribution of each module is listed.

7 Discussion

In this section, we discuss the advantages and limitations of our work. On this basis, we look forward to our future work.

Generalizability. Our work is generalizable for the reason that our system can solve various problems proposed by users instead of a specific task. Also, users can quickly understand the dataset and obtain answers to their questions through our system. Our user studies prove the flexibility and efficiency of our work. However, our approach also has limitations in the following aspects:

(1) Data processing. In terms of the size of the dataset, since our system operates by generating and executing code, there are no inherent restrictions on the size of the data in the processing phase. The main limitation is the amount of data that a visualization library can support. In terms of the dimension, our method performs well in the house pricing datasets (which have over 60 features) and upper limit of the system needs more experimental verification. In terms of the complexity of algorithms, our system can only handle problems answerable from data or those can be solved by directly using some encapsulated machine learning algorithms (like PCA and linear regression), but it meets problems when it needs to design the structure of deep learning models, and automatically execute algorithms which requires training and validation. In the future, we can enhance the ability to design deep learning frameworks for large modes and develop frameworks that automate the training of deep learning models, thereby improving data processing capabilities.

(2) Visualization composition. Although we achieve diversity in chart selection as well as associations between views, we fail to create composition between views or design complex visualization charts. To address the problem, we need to extract patterns of chart compositions [12] and analyze the correlations between decomposed questions, trying to solve multiple problems in one chart.

(3) Customization. In our work, we achieve customization by giving the user a choice of question and by allowing the user to make changes to the chart. However, there is a certain lag in this way, and LLMs has a shallow understanding of some industry terms. In future research, we can achieve better customization by fine-tuning methods for a certain field.

The limitations brought by LLMs. Despite LLMs' excellent performance in natural language understanding and reasoning, we encounter problems from the following aspects:

(1) Stability in code generation. The code generated by LLMs may encounter issues, leading to failure in visualization generation or interaction. Although we have addressed some potential problems in data processing process (like time and splitting) and provided LLMs with the opportunities to correct errors in error handling process, LLMs may still experience hallucinations during execution, leading to a poor user experience. Some state-of-the-art work have proposed some methods to solve hallucination, like self reflection [20] and knowledge distillation [29]. In the future, we may combine LLMs and deep learning based methods to improve stability.

(2) Execution efficiency. In our experiment, we select GPT-4 to finish all the tasks. On average, a duration of 50 seconds is required to generate a single chart and the generation of linked views needs more time. The absence of real-time engagement or feedback mechanisms exacerbates the waiting period, making it feel even more tedious and laborious. It has been proven that models with fewer parameters generate results more quickly, but this comes at the cost of reduced quality [7]. Finding a balance between speed and the quality of generated responses is a topic worth further investigation. Besides, we can improve system efficiency with data caching [25] and prefetching technologies [30].

The limitations of ablation study. In our evaluation, we utilized the NVBench dataset, which is limited to testing the generation of individual charts and cannot evaluate the performance of our entire system. As a result, the necessity of decomposition and linking components cannot be quantitatively measured. Moreover, this dataset offers a limited selection of visualizations (only four types), and most tasks involve simple data processing. This limitation is insufficient to fully demonstrate the superiority of our method. In the future, we look forward to collecting datasets related to linked views from platforms like Tableau to establish benchmarks.

8 Conclusion

To conclude, we introduce SmartMLVs, a LLM-enhanced automatic interactive multiple linked views generation system designed for data analysts to comprehend data and gain insights. Our framework enhances the ability of LLMs by appropriate prompt engineering methods and injecting visual analytics domain knowledge, generates interactive linked views through a three-step process, and involves users into the generation and exploration process, allowing them to have a deep understanding of the task and dataset. The usage scenario and evaluation experiments prove the effectiveness and expandability of our method.

Acknowledgments

This work is supported by Natural Science Foundation of China (NSFC No.62472099 and No.62202105).

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] M. Alharbi and R. S. Laramee. Sos textvis: An extended survey of surveys on text visualization. *Computers*, 8(1):17, 2019.

[3] N. Andrienko, G. Andrienko, G. Fuchs, A. Slingsby, C. Turkay, and S. Wrobel. *Visual analytics for data scientists*. Springer, 2020.

[4] S. K. Badam, Z. Liu, and N. Elmqvist. Elastic documents: Coupling text and tables through contextual visualizations for enhanced document reading. *IEEE transactions on visualization and computer graphics*, 25(1):661–671, 2018.

[5] F. Basysyar and G. Dwilestari. House price prediction using exploratory data analysis and machine learning with feature selection. *Acadlore Trans. AI Mach. Learn*, 1(1):11–21, 2022.

[6] R. Borgo, A. Abdul-Rahman, F. Mohamed, P. W. Grant, I. Reppa, L. Floridi, and M. Chen. An empirical study on using visual embellishments in visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2759–2768, 2012.

[7] T. B. Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[8] T. A. Caswell, M. Droettboom, A. Lee, E. Sales de Andrade, J. Hunter, T. Hoffmann, E. Firing, J. Klymak, D. Stansby, N. Varoquaux, et al. matplotlib/matplotlib: Rel: v3. 4.0. *Zenodo*, 2023.

[9] W. Chen, X. Ma, X. Wang, and W. W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

[10] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[11] A. Creswell, M. Shanahan, and I. Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

[12] D. Deng, Y. Wu, X. Shu, J. Wu, S. Fu, W. Cui, and Y. Wu. Visimages: A fine-grained expert-annotated visualization dataset. *IEEE Transactions on Visualization and Computer Graphics*, 29(7):3298–3311, 2022.

[13] V. Dibia. Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. *arXiv preprint arXiv:2303.02927*, 2023.

[14] J. Dill, R. Earnshaw, D. Kasik, J. Vince, and P. C. Wong. *Expanding the frontiers of visual analytics and visualization*. Springer, 2012.

[15] M. Elias and A. Bezerianos. Exploration views: understanding dashboard creation and customization for visualization novices. In *Human-Computer Interaction–INTERACT 2011: 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part IV 13*, pp. 274–291. Springer, 2011.

[16] L. Gao, J. Lu, Z. Shao, Z. Lin, S. Yue, C. Leong, Y. Sun, R. J. Zauner, Z. Wei, and S. Chen. Fine-tuned large language model for visualization system: A study on self-regulated learning in education. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):514–524, 2025.

[17] W. Godoy, P. Valero-Lara, K. Teranishi, P. Balaprakash, and J. Vetter. Evaluation of openai codex for hpc parallel programming models kernel generation. In *Proceedings of the 52nd International Conference on Parallel Processing Workshops*, pp. 136–144, 2023.

[18] Y. Guo, D. Shi, M. Guo, Y. Wu, N. Cao, and Q. Chen. Talk2data: A natural language interface for exploratory visual analysis via question decomposition. *ACM Transactions on Interactive Intelligent Systems*, 14(2):1–24, 2024.

[19] H. Hao, Y. Cui, Z. Wang, and Y.-S. Kim. Thirty-two years of ieee vis: Authors, fields of study and citations. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1016–1025, 2022.

[20] Z. Ji, T. Yu, Y. Xu, N. Lee, E. Ishii, and P. Fung. Towards mitigating llm hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 1827–1843, 2023.

[21] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the sigchi conference on human factors in computing systems*, pp. 3363–3372, 2011.

[22] G. Li, X. Wang, G. Aodeng, S. Zheng, Y. Zhang, C. Ou, S. Wang, and C. H. Liu. Visualization generation with large language models: An evaluation. *arXiv preprint arXiv:2401.11255*, 2024.

[23] Y. Li, Y. Qi, Y. Shi, Q. Chen, N. Cao, and S. Chen. Diverse interaction recommendation for public users exploring multi-view visualization using deep learning. *IEEE transactions on visualization and computer graphics*, 29(1):95–105, 2022.

[24] C. Liu, Y. Han, R. Jiang, and X. Yuan. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pp. 11–20, 2021.

[25] L. Liu, X. Yuan, N. Zhang, D. Chen, K. Yu, and A. Taherkordi. Joint computation offloading and data caching in multi-access edge computing enabled internet of vehicles. *IEEE Transactions on Vehicular Technology*, 72(11):14939–14954, 2023.

[26] Z. Liu, J. Thompson, A. Wilson, M. Dontcheva, J. Delorey, S. Grigg, B. Kerr, and J. Stasko. Data illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2018.

[27] Y. Luo, J. Tang, and G. Li. nvbench: A large-scale synthesized dataset for cross-domain natural language to visualization task, 2021.

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

[29] D. McDonald, R. Papadopoulos, and L. Benningfield. Reducing llm hallucination using knowledge distillation: A case study with mistral large and mmlu benchmark. *Authorea Preprints*, 2024.

[30] S. Mostofi, H. Falahati, N. Mahani, P. Lotfi-Kamran, and H. Sarbazi-Azad. Snake: A variable-length chain-based prefetching for gpus. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 728–741, 2023.

[31] T. Munzner. *Visualization analysis and design*. CRC press, 2014.

[32] 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.

[33] C. North and B. Shneiderman. Snap-together visualization: can users construct and operate coordinated visualizations? *International Journal of Human-Computer Studies*, 53(5):715–739, 2000.

[34] J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Fifth international conference on coordinated and multiple views in exploratory visualization (CMV 2007)*, pp. 61–71. IEEE, 2007.

[35] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350, 2017.

[36] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE transactions on visualization and computer graphics*, 18(12):2431–2440, 2012.

[37] V. Setlur, M. Tory, and A. Djalali. Inferencing underspecified natural language utterances in visual analysis. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, p. 40–51. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3301275.3302270

[38] Z. Shao, L. Shen, H. Li, Y. Shan, H. Qu, Y. Wang, and S. Chen. Narrative player: Reviving data narratives with visuals. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–15, 2025.

[39] 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.

[40] A. Srinivasan and J. Stasko. How to ask what to say?: Strategies for evaluating natural language interfaces for data visualization. *IEEE Computer Graphics and Applications*, 40(4):96–103, 2020.

[41] P. Studio. tuzhidian. http://tuzhidian.com/.

[42] 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. *arXiv preprint arXiv:2311.01920*, 2023.

[43] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[44] J. VanderPlas, B. Granger, J. Heer, D. Moritz, K. Wongsuphasawat, A. Satyanarayan, E. Lees, I. Timofeev, B. Welsh, and S. Sievert. Altair: Interactive statistical visualizations for python. *Journal of Open Source Software*, 3(32):1057, 2018. doi: 10.21105/joss.01057

[45] P.-P. Vázquez. Are llms ready for visualization? In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pp. 343–352. IEEE, 2024.

[46] C. Wang, J. Thompson, and B. Lee. Data formulator: Ai-powered concept-driven visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[47] C. Wang, J. Thompson, and B. Lee. Data formulator: Ai-powered concept-driven visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1128–1138, 2024.

[48] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R. K.-W. Lee, and E.-P. Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*, 2023.

[49] X.-M. Wang, T.-Y. Zhang, Y.-X. Ma, J. Xia, and W. Chen. A survey of visual analytic pipelines. *Journal of Computer Science and Technology*, 31:787–804, 2016.

[50] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[51] W. Zhang, Y. Shen, W. Lu, and Y. Zhuang. Data-copilot: Bridging billions of data and humans with autonomous workflow. *arXiv preprint arXiv:2306.07209*, 2023.

[52] Y. Zhao, J. Wang, L. Xiang, X. Zhang, Z. Guo, C. Turkay, Y. Zhang, and S. Chen. Lightva: Lightweight visual analytics with llm agent-based task planning and execution. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–13, 2024.

[53] Y. Zhao, Y. Zhang, Y. Zhang, X. Zhao, J. Wang, Z. Shao, C. Turkay, and S. Chen. Leva: Using large language models to enhance visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 31(3):1830–1847, 2025.

[54] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.