LightVA: Lightweight Visual Analytics with LLM Agent-Based Task Planning and Execution

Yuheng Zhao, Junjie Wang, Linbin Xiang, Xiaowen Zhang, Zifei Guo, Cagatay Turkay, Yu Zhang and Siming Chen

Abstract—Visual analytics requires analysts to constantly plan and execute analysis tasks based on observations and generate visualizations and support analysis for gaining insights from data. Completing this process requires significant effort, emphasizing the necessity for a more intelligent and lightweight visual analytics process. Large language model agents have shown capabilities in logic inference and code generation, offering the potential to lower development costs and enhance the versatility of visual analytics. We propose LightVA, a lightweight visual analytics framework that completes task proposal, data modeling and interactive visual exploration with human-agent collaboration. Our method is grounded on the relationships among the goal, task, data, visualization, and insight. Specifically, we introduce an LLM-agent-based task-planning and execution method that supports task planning and execution, employing a recursive process involving a planner, executor, and controller, dynamically accommodating task complexity. The planner is responsible for task decomposition, the executor handles task execution, including visualization generation and data analysis, and the controller orchestrates the executor and planner. Based on the LightVA framework, we develop a system consisting of a task flow diagram and interactive visualization panel, which makes the workflow more controllable and transparent. We examine the effectiveness of our method through a usage scenario and an expert study.

Index Terms—Visual Analytics, Interface Agent, Task Planning, Large Language Model, Mixed-Initiative Interaction

1 Introduction

Visual analytics (VA) deciphers complex datasets with data mining and interactive visualizations [22], [49]. Building and using a VA system is a costly endeavor that encompasses several main stages: goal understanding, task decomposition, and implementing visualization and modeling to discover insights. Developers need to write codes for data mining and visualization to develop VA system for different tasks [60]. Once the system is built, analysts use it to explore data. However, navigating a vast exploration space from a goal to propose tasks can be complicated [7]. Consider a social media analysis scenario where the goal is to identify high-risk events from multi-faceted social media data. Developing a VA system for this goal is costly due to the need for diverse visualization and data modeling components for tasks such as trend analysis, sentiment analysis, and spatial analysis [8]. While analysts may use libraries or existing tools, most problems would require adaptations and modifications and rarely work directly off the shelf. This complexity highlights the need for intelligent task planning and code generation to reduce development costs and support interactive exploration efficiently.

Recent research focuses on data-driven or natural language-based visual data exploration, with an emphasis on automatic visualization generation [10], [66] or insight mining [9], [54]. Large Language Models (LLMs) present a potential for data analysis, supporting dynamic task planning and lower development costs across various scenarios [36]. They have reasoning abilities that enable autonomous planning and execution of analytical tasks [16], [46], [64], while their code generation skills support creating insightful analyses and visualizations efficiently [11], [29], [52]. Additionally, their broad knowledge base makes LLMs versatile tools capable of adapting to diverse data analysis contexts [33], [68]. LEVA [70] integrates LLMs in VA systems to recommend insights for a given task but still cannot generate visualization and data modeling methods adapted to tasks. There is a lack of approaches that support task planning, VA system development, and intelligent analysis with human-agent collaboration.

This paper, introduces *LightVA*, a lightweight VA framework with agent-based task planning. The term "lightweight" refers to the framework's focus to reduce the cost of development and the effort of using of VA systems. Using LLM agents to aid the task planning and execution process, it reduces the complexity and effort required from users. The framework builds upon the relationships among the goal, tasks, data, visualizations, and insights. Specifically, the framework is to involve employing a recursive process that includes a planner, executor, and controller, which accommodates task complexity dynamically. The planner is responsible for task decomposition, the executor handles task execution, including visualization generation and data analysis, and the controller orchestrates the executor and planner and manages whether tasks continue to be decomposed. We develop a system based on the framework that provides a chat view to support communication among users and agents, and a dynamically updated visualization view to demonstrate the analysis results, as well as a task flow view to visualize the process of task planning. Our main contributions are as follows:

- We propose a lightweight VA framework to involve LLM agent-based task planning and execution methods. The workflow includes developing of VA system and using VA system for data analysis based on human-agent collaboration.
- We develop a system that embodies our proposed framework, supporting users to analyze data with the assistance of agents and communicate with each other through the interactive interface.
- We demonstrate the effectiveness of the system through a usage scenario and an expert study.

2 Related Work

We review existing literature regarding visualization recommendations, task-driven data exploration, and LLM application in data exploration.

- *Yuheng Zhao, Junjie Wang, Linbin Xiang, Xiaowen Zhang, Zifei Guo, Siming Chen are with School of Data Science, Fudan University. E-mail: {yuhengzhao, simingchen, 19300290049}@fudan.edu.cn, {wangjj23, 23210980124, 21307110373}@m.fudan.edu.cn.*
- *Cagatay Turkay is with University of Warwick. E-mail: Cagatay.Turkay@warwick.ac.uk.*
- *Yu Zhang is with Department of Computer Science, University of Oxford. E-mail: yuzhang94@outlook.com.*

Manuscript received April 19, 2005; revised August 26, 2015.

2.1 Visualization Recommendation

Visual generation typically requires users to have professional visual knowledge and programming ability, which makes a lot of effort. In many analysis scenarios, tools like Tableau are frequently utilized, offering robust visual generation features. However, it only has visual generation capabilities and lacks the capability for overall task decomposition and VA development. A corpus of research has been proposed on automatic visualization recommendations [43], [53], [61]. These methods can be technically divided into rule-based and machine learning-based.

Rule-based methods, such as Voyager [58] and CompassQL [59], utilize the visualization principles to construct visual mapping and allow users to choose their interested data properties and visual encoding to create visualizations. For machine learning methods, Data2Vis [12] introduces an end-to-end trainable neural translation model for automatically generating visualizations from given datasets. VizML [19] learned visualization design choices from a corpus of data-visualization pairs. Table2Charts [72] recommends visualizations by learning patterns between tables and visualizations. ChartSeer [69] employs deep learning to recommend visualizations based on users' interactions. Unlike end-to-end deep learning methods that directly learn from datasets to generate visualizations, knowledge graph-based approaches, such as Adavis [67], KG4VIS [26], Lodestar [42], leverage structured information about data and relationships to recommend visualizations.

In addition, there are some works that consider multi-view generation. Dziban [27] is a visualization API using anchored recommendation and extending Draco [35] to reason about multiple views. DMiner [28] investigated the design rules of the single views and view-wise relationships from online notebooks to recommend multiple-view dashboards. MultiVision [62] and DashBot [10] recommend dashboards given an input dataset in an end-to-end manner. Shi et al. [48] optimize multi-view layouts by predicting the similarity of visual elements using Transformer-based models. However, the visualizations and insights provided by these systems are tool-specific, and the systems still do not support adaptive dashboard generation according to dynamic tasks, requiring considerable human effort.

2.2 Task-Driven Data Exploration

In addition to data attributes when recommending visualizations, some visualization recommendation systems are task-driven recommendation systems that consider one or more analytic tasks (e.g., correlate, analyze trend). Casner [6] presents one of the earliest examples of visualization systems that suggests charts based on a user's task (e.g., find direct flight routes or a table to see flight information). Saket et al. [44] conducted a study to assess the effectiveness of five canonical visualizations on ten low-level analytic tasks [1] and developed a recommendation engine based on their study's findings. Gotz and Wen [14] present a prototype system that observes interaction patterns (e.g., repeatedly changing filters or swapping attributes) to infer analytic tasks such as comparison or trend analysis and correspondingly recommends visualizations such as small multiples or line charts. VizAssist [5] enables its users to specify their data objectives in terms of analytic tasks (e.g., correlate, compare) and considers these tasks in combination with existing perceptual guidelines as input to a genetic algorithm for recommending visualizations. Foresight [9] uses tasks like distributions, outliers, and correlations to guide insight discovery and grouping recommendations. TaskVis [47] recommends visualizations under specific tasks through answer set programming.

In addition to visual generation based on a single task, Medley [38] recommends multi-view collections based on several analytic intents, and views and widgets can be selected to compose a variety of dashboards. However, in this work, the analytic intents or tasks are pre-specified, meaning that users must choose from a set of predefined tasks and visualizations. This limitation motivates the need for LightVA, which leverages human-agent collaboration to dynamically plan tasks based on existing findings.

2.3 Large Language Model in Data Exploration

LLM-based tools have been proposed for data exploration and visualization. For visualization generation and recommendation, LLM4Vis [52] and Chat2VIS [32] utilize LLMs for visualization recommendations and generating visualizations from natural language. Additionally, Li et al. [24] evaluate the capability of GPT-3.5 to generate visualization specifications from natural language queries, demonstrating its superiority over previous NL2VIS approaches. NL2Rigel [20] showcases the LLM's ability to convert instructions into comprehensive data visualizations and tables. For analytical task translation and automation, Hassan et al. [17] and Data-Copilot [68] concentrate on converting analytical goals and ambiguous queries into actionable data analysis tasks. Ma et al. [31] and JarviX [29] introduce systems that automate the data exploration process by identifying suitable analysis intents and generating insights. Text2Analysis [18] offers a framework for categorizing data analysis tasks, establishing a structured approach to tackling common analytical challenges ranging from basic operations to forecasting and chart generation. However, the tasks in these studies are still relatively simple and specific, do not involve the decomposition of complex tasks.

When using LLMs to solve complex tasks where multi-step reasoning is demanded, the performance of directly using LLMs tends to decrease. Recently, prior methods have utilized LLMs with input-output prompting, CoT [57], ToT [65] or GoT [4] to perform complex task planning and execution. These methods proved that LLMs are good at task planning but require appropriate prompting techniques. Another way is to integrate LLMs in the interface, allowing chaining multiple prompts to address a much wider range of human tasks. Wu et al. [63] introduce the chaining of AI models, where a complex task is divided into multiple steps. Talk2Data [15] presents a natural language interface that enables users to explore visual data through question decomposition. However, the linked visualization and more advanced data analysis methods are still limited. To compensate for this, we leverage LLMs and propose an agent-based autonomous task-planning strategy for adaptive VA system construction and exploration.

3 LIGHTVA FRAMEWORK

The development and usage of VA systems involve two types of stakeholders: developers and analysts. In LightVA, we aim to reduce the efforts for both developers and analysts. In the following, we will introduce the challenges and derive the design requirements for integrating LLM-based agents into the user's workflow. Finally, we introduce the conceptual framework of agent-based VA workflow.

3.1 Challenges

Through a review of visual analytics literature, we identify and dissect specific challenges that intensify the effort users must exert:

C1 Accommodating diverse analysis requirements: Data Analysts often face the immense challenge of navigating a vast exploration space, where the analytical process is dynamic and iterative [7]. Forming hypotheses and validating them through the continuous proposal of new tasks and insights is complex. They have to figure out the connections between tasks and insights in their mind and try to propose tasks in the next few steps until they achieve the goal.

C2 Developing visualization and data mining tools is time-consuming: Analysts often lack proficiency in developing and managing VA systems, which makes it challenging for them to select and apply appropriate data modeling and visualization techniques [38]. Additionally, when tasks involve multiple visualizations, these must be integrated into a linked view to enable more effective interactive exploration [55]. However, this process requires substantial knowledge of both visualization principles and coding skills, resulting in inefficiencies and delays [23].

3.2 Design Requirements

Based on these challenges, we have settled on a set of targeted design requirements.

R1 **Adaptive task planning:** Task proposals need to be tailored to users' analysis goals and the given dataset. This includes exploring in depth and breadth, with automatic planning and execution reducing user efforts. Moreover, as exploration results emerge, new tasks should contextually link to previous exploration results, adapting to the switching between tasks. (C1)

R2 **Flexible visualization generation:** The generation of visualizations should flexibly handle different tasks and data. This encompasses accurate data identification, transformation, and selection of visualization types, as well as adding highlights based on discovered insights to reduce cognitive load. Furthermore, generated views should support interaction, allowing for more immersive user analysis. (C2)

R3 **Automatic data mining:** To reduce development costs, the system should efficiently complete code-based data analyses for given tasks, providing visualizations and suggesting findings to facilitate hypothesis forming and validation. To lessen the cognitive burden, important parts of insights should be highlighted in rich text format in the output results. (C2)

R4 **Multiple view composition:** As new visualizations are added, older ones may become less relevant. To display the most recent results to the user, the visualization panel needs to be updated. However, it is crucial to avoid discarding previous results as they may be relevant to new insights. Users should be allowed to merge visualizations of interest even if they are not the latest. Within the merged views, users can focus their analysis on a smaller scope through interaction, reducing cognitive load among large sets. (C2)

R5 **Intuitive analysis process:** The relationship between tasks, visualization, insights, and analysis progress should be presented in a more intuitive form. The system should support interactions between human and agent to help users understand the agent's task planning and execution. (C1, C2)

3.3 Conceptual Framework

Based on the challenges and requirements discussed, we propose LightVA, a lightweight VA framework with agent-based task planning. The "lightweight" refers to the light expectations in developing VA systems and using the developed systems for analysis. We design the framework to involve a recursive task-solving process in which goals and data are inputs, and insights are outputs. The intermediate results are tasks, subtasks, visualization, data modeling (Figure 1). The LLM agents are integrated to support goal understanding and task decomposition (R1), data modeling and visualization codes generation (R2, R3), and linked view generation for interactive exploration (R4). Meanwhile, the users can guide the agent and refine the agent outputs.

3.3.1 Defining Primitives

According to the above descriptions of the framework, some key primitives need to be defined.

Goal and Dataset: We define a goal as an overall description or a vague utterance that expresses the purpose of the analysis. For instance, *"to analyze the influence factor on a car's fuel efficiency"*. The goal directs the focus of the analysis, and the data supplies the analysis evidence to meet the goal.

Task: Tasks are specific aspects derived from the goal and make the *goal* executable. We define them using four attributes:

$$task := \langle type, data\ variables, method, progress \rangle \tag{1}$$

The *type* indicates the nature or category of the analytical operation to be performed, such as finding extreme, outlier, change point, trend

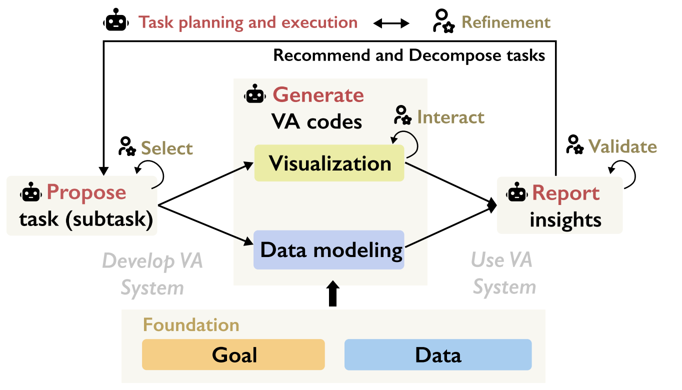


Fig. 1: The illustration of conceptual framework in LightVA. he agent creates the VA system by task planning and execution, while the user uses the created VA system and refines agent outputs.

analysis, etc. The *data variables* are the objects upon which the tasks will be applied. The *method* defines how the task will be solved, including data modeling and visualization methods, and the solved result is *insight*. If the task is complex, which means it covers multiple *data variables* and needs to use multiple *methods* to solve, the task can be decomposed into *subtasks*. To depict whether the task is completed, we define *progress*, which means the task is solved when its subtasks are solved as well. Given that the tasks can be complex and range from low to high abstraction, we describe them using natural language.

Insight: This denotes the expected or targeted knowledge or understanding that the task aims to achieve. We define insight using four attributes:

$$insight := \langle type, parameters, data\ variables, data\ values \rangle \tag{2}$$

The *insight type* can be discoveries, patterns, trends, or anomalies identified through analysis [13]. For example, if the task type is "compare", the insight type could be "difference" [54]. The *parameters* specific features of the insight, such as "increasing" or "decreasing" of "trend". The *data variables* for insights can be the original data columns or transformed variables, while the *data variables* for tasks are the original data columns. For example, if the task involves analyzing vehicle weight and fuel efficiency (MPG), the task's *data variables* would include "Weight_in_lbs" and "MPG". If the insight includes a derived metric such as "MPG per pound," this would be considered a transformed variable specific to the insight. The *data values* refer to particular values of *data variables*. For example, the maximum MPG value is "48".

Visualization: Refers to the visual representation required for the task that best conveys the data and insights.

$$visualization := \langle type, encoding, interaction, coordination \rangle \tag{3}$$

Our framework generates visualizations using Vega-Lite grammar [45], which supports the above four aspects. The types of *interactions* are, e.g., filtering, zooming, and hovering supported by the visualization. The *coordination* describes how this visualization interacts or synchronizes with other visualizations, such as a brush, to filter each other.

3.3.2 Workflow of Framework

Our framework involves building a **task flow**, represented as a directed compound graph $G = (V, E)$, where each *node* $v \in V$ represents a *task* and *edge* $e \in E$ represents the connection from one task to another task. Since the process of exploration involves both enlightening and in-depth thinking, we define two types of task edges: **recommendation** and **decomposition**. And we define the solving of the task as **execution**. The difference between recommendation and decomposition is that

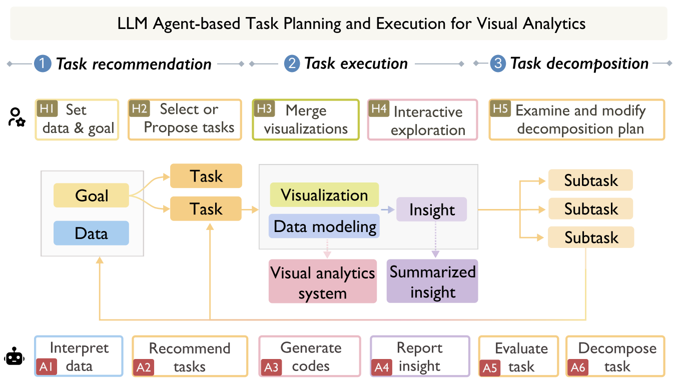


Fig. 2: The workflow of the LLM agent-based task planning and execution for visual analytics. The workflow characterizes the collaboration between users and the AI agent in three stages: task recommendation, task execution, and task decomposition. Users propose goals, select tasks, merge visualizations, and engage in interactive exploration (H1-H5). The agent interprets data, recommends tasks, generates codes, reports insights, evaluates tasks and decompose tasks (A1-A6).

recommendation is “goal-oriented”, and decomposition is “task-specific”. The recommendation aims to broaden the exploration scope, providing heuristic suggestions. The decomposition aims to ensure a detailed and logical approach to solving the task. The following sections will introduce the human-agent collaboration workflow of our framework (Figure 2).

Stage1: Task recommendation. In the beginning, the user uploads the data and inputs a goal (H1), and then the agent interprets data (A1) and transforms the goal into specific, actionable tasks (A2). The user can provide feedback and accept or modify these recommendations to ensure they stay aligned with their analysis needs (H2). Specifically, the interaction process involves two stages:

- **Initial stage:** To recommend a task, the agent should first interpret the data and propose tasks that make the goal executable by mapping it to the data. The principle is to identify the *data variables* and *task type*. For instance, for the goal of *“find high-risk events in a city”*, useful data aspects could include time, space, text, and sentiment. Thus, the agent needs to find *data variables* related to the goal. The agent also needs to distinguish the purpose of the analysis, such as *“finding outliers”*. Thus, different combinations of data variables and types can form different tasks under a goal.
- **Historical context stage:** As the analysis continues to accumulate in the recommendation process, the agent should recommend tasks considering previous tasks and the overall goal. First, when a task is completed, the agent evaluates if the overall goal has been achieved. If not, it should recommend new tasks aligned with the goal. Second, previously unexplored tasks should be re-evaluated and proposed again if they are still relevant to the goal.

Stage2: Task execution. In this stage, the agent generates visualization and modeling codes (A2) and executes the codes to report and summarize insights (A3). Having multiple visualizations and modeling approaches, users could select visualizations to merge a linked view (H1) and interactively explore it (H2). It comprises the following two-step approach:

- **Visualization and insight generation**: When the user selects an agent-proposed task or proposes a task by themselves, the agent will write codes to complete the analysis. The agent needs to choose appropriate data modeling and visualization *methods* for each *task* and run the codes to complete the analysis. The agent then needs to structure the *insights* into a structured format. Finally, the agent should summarize insights obtained from the decomposition of subtasks, providing a comprehensive overview of the analysis.
- **Multi-view linking**: Users can initiate *coordination* by selecting multiple visualizations they prefer while the agent generates codes to complete. The rationale why not directly merging visualizations from subtasks is that subtasks generated from task decomposition often share the same variables and visualization types. For example, several subtasks might use *data variables* like latitude and longitude to create maps exploring different spatial patterns. The linked view is often used to conduct multi-dimensional analysis with different visualizations. Thus, allowing users to choose their visualizations can provide a more flexible analysis.

Stage3: Task decomposition. We design the decomposition following a *“as-completed”* strategy. In other words, execution is prioritized, and decomposition is considered only if the task is not completed. This approach aims to ensure users can see initial results quickly in an interactive environment. In this stage, the agent is responsible for evaluating tasks (A5) and proposing a decomposition plan (A6), while the users can examine and modify agent output to override the agents (H5).

- **Results assessment**: Based on the initial execution's insights and the complexity of the task, the agent assesses the need for further analysis. According to the definition of *task*, the agent should verify the selection of *data variables* and the rationality of data modeling and visualization *methods*. The evaluation should be explained to make users understand the motivation of decomposition.
- **Sub-task generation**: If decomposition is necessary, the agent should formulate a plan outlining subtasks. Each subtask should have appropriate *data variables* and *methods* to address distinct aspects of the main task. The subtasks should have an execution order, e.g., in parallel or sequentially.

4 LIGHTVA SYSTEM

Guided by the design requirements summarized in Section 3.2, we design the pipeline of agent-based task planning and the interface to support interactive exploration with assistance.

4.1 Agent-based Task Planning

Based on the conceptual framework previously introduced, we propose an agent-based task-planning pipeline Figure 3. In this process, we have three kinds of modules. The *planner* has two types: *recommender* and *decomposer*, which are responsible for task recommendation and decomposition respectively. The *executor* handles task execution, including visualization generation and data analysis, and the *controller* is the recursive algorithm that bridges the *executor* and *planner* (Algorithm 1 is added in Appendix). The prompt examples and outputs are available in Appendix A.

4.1.1 Task recommendation

Task planning begins with heuristic approaches based on the given goal and dataset and aims to transform the goal into actionable tasks. The implementation follows the guidelines and consists of two stages: the initial stage and the historical context stage.

In the initial stage, the input is the goal and the data (Figure 3-a) and the objective is to convert the goal into actionable tasks. Based on the framework (Section 3.3.2), we guide the model with four principles. First, identify *data variables* relevant to the goal. Second, determine the *task type* based on the analysis purpose. Third, formulate a task description by combining data variables and task type. Finally, describe the task in a sentence in the output.

In the historical context stage (Figure 3-b), the generated tasks should be divided into two types: linking goals with discoveries to propose *new* tasks and tasks that might have been forgotten by the user needs to *review*. To achieve this goal, the model might needs to consider the following steps. First, evaluate completed tasks to check if the overall goal is achieved. Seoncd, propose new tasks based on the current context and previous tasks. Third, propose previously unexplored tasks if they remain relevant. In addition to guidelines, we should provide examples of output to guide models.

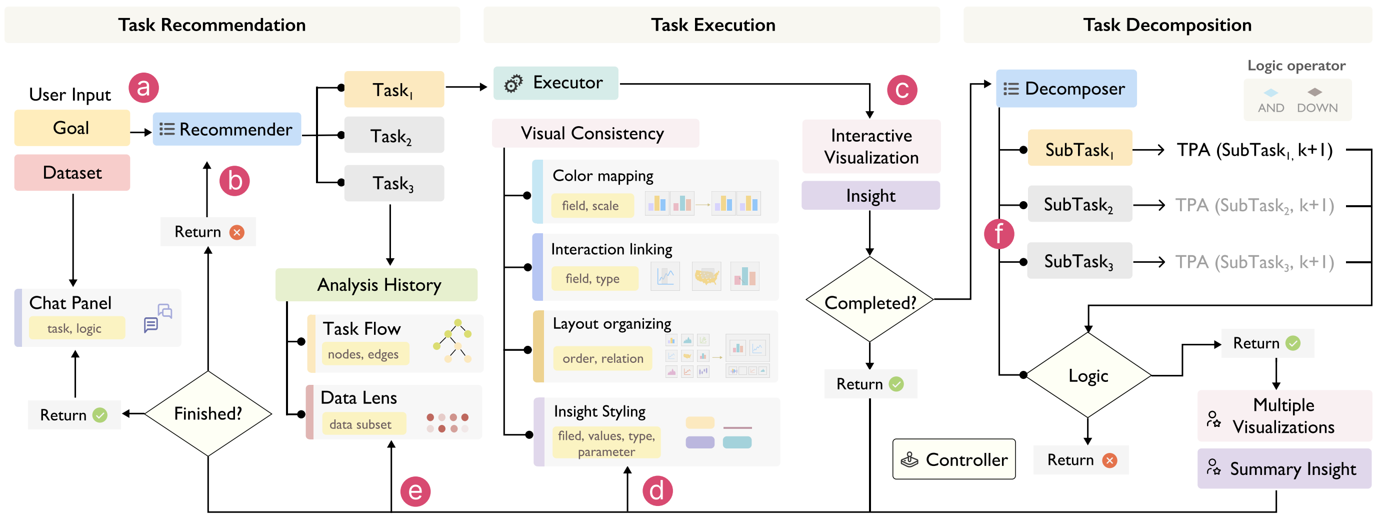


Fig. 3: The agent-based system architecture. The pipeline starts from the goal and data with an agent-based task planning strategy, including recommendation, execution, and decomposition. Specific components and interactions are labeled (a)-(f), where (a) is the initial stage to recommend tasks from the goal, (b) recommends tasks with historical contexts, and (c) is the execution of tasks generating visualization and insights. After that, a series of optimizations for visual consistency are performed (d). Historical records are managed, and the progress of tasks is updated in a timely manner (e). According to the analysis results, the agent evaluates whether the task needs to be decomposed (f).

Prompt Template 1 (Task Recommendation).

Initial stage

Input: {`goal, data`}
Instruction: You need to come up with a short plan based on the understanding of the data to help accomplish the goal. Please give a short summary of the dataset and recommend n exploratory tasks, including task description, type, and data variables and explanations. For task type, you can consider trend, correlation, category, distribution, etc. For each new task, you need to give an explanation of the proposal. For data_variables, list the original columns, even if transformation is required.
Indicator: {`An example in JSON format`}
Output: {`new tasks`}

Historical context

Input: {`goal, data, explored and unexplored tasks`}
Instruction: You need to supplement some new tasks for explored tasks by considering the results of tasks already explored if needed. Second, it is recommended that previous unexplored tasks be revisited if they are suitable for analysis at this stage by considering the explored tasks.
Indicator: {`An example in JSON format`}
Output: {`new tasks, existing tasks to review`}

4.1.2 Task Execution

After a task is proposed, the user can select a task to solve. According to the framework (Section 3.3.2, task decomposition ideally involves the second stage to detailed plan and then execute. However, to allow users to quickly see an initial overview, we prioritize initial execution, embodying the "decompose as-completed" concept. The inputs for task execution are a goal, data, and task, and the outputs are visualizations and insights for the selected task (Figure 3-c). We let the Executor write Python codes for visualization and insight generation. The execution includes two rounds of communication with the executor. The first round is to generate codes that include two snippets. The first snippet is for data analysis, and the following is for visualization generation. The second round is to write structured insight.

Prompt Template 2 (Task Execution).

Code generation

Input: {`task, data summary, code template`}
Instruction: You need to write Python codes to analyze the data to solve this task. After finishing the data analysis, you should continue to use Altair to generate an interactive visualization. Add a brush function, a tooltip, and a legend if different colors are used.
Indicator: {`A code template`}

```python
import altair as alt
import pandas as pd
def plot(data: pd.DataFrame):
    # Data preprocessing
        <codes>
    # Chart generation
    chart = alt.Chart().mark_bar().encode()
    return chart
```

Output:{`insight, visualization`}

Structured insight generation

Input: {`task, data, codes`}
Instruction: Run the `codes` to report an important insight for this task: `task`. You should output insight, including text, insight type, parameters, data variables, and data values. {`Definitions of insight attributes.`}
Indicator: {`A few examples in JSON format`}
Output: {`insight`}

Visualization generation: For the visualization part, we opt for Vega-Lite through Altair. The reason is that Vega-Lite is a high-level visualization grammar that provides a concise syntax for rapidly generating a wide array of visualization types. In addition, this declarative syntax makes it easy for users to modify visualizations. We let the Executor propose data modeling methods using Python codes, which can be connected with Altair. This offers flexibility for complex data transformation processes compared to directly producing grammar. Nevertheless, this enhancement, derived from Python's advanced and versatile capabilities, introduces a higher likelihood of errors during code parsing. To mitigate this issue and improve success rates, we incorporate a code template within the prompt.

Structured insight generation: After generating visualization and modeling codes, the executor needs to execute them to structure the results into insights. As the agent's coding environment[1] does

1. https://platform.openai.com/docs/assistants/tools/code-interpreter. Code Interpreter allows Assistants to write and run Python code in a sandboxed execution environment.

not support Altair, we use the system environment to execute the visualization part of the code. The agent runs the data modeling part, interprets the results, and derives insights. We provide a structured format example to guide the executor according to our definition of insight in the framework. This includes the insight text, the insight type, and the analyzed data variables and values. These attributes are used to highlight key elements in the insight text to improve readability.

Linked-view generation: Regarding the quantity of generation, we add rules to improve the visual consistency (Figure 3-d). Note that not all optimization is done by the model. After adding the interaction to the model, the color, layout, etc., need to be adjusted by the system. To improve the quality, we provide guidelines to LLMs.

- **Interaction linking:** We allow LLMs to add brush interaction to the given charts selected by the user. The method involves modifying multiple Python codes and then outputting a combined Python code. The guidelines are provided in the prompt, such as aggregating and transforming data as needed for each visualization and adding common selectors and filters as needed to enable interactions such as "click" and "brush" between charts. Ensure each chart includes consistent tooltips to facilitate interaction.
- **Layout organization:** When generating new visualizations, we set the charts to the same size and arrange them sequentially. If the user selects charts to merge multiple views, we allow selecting up to 6 charts to avoid excessive cognitive load. The LLMs should output the layout within a 2-row by 3-column configuration.
- **Color mapping:** We built a data-to-color mapping rule within the task space based on Qu et al.'s guidelines [41], where one color corresponds to one data dimension without reuse. For example, the same field should use the same quantitative color scale across different views, while different fields should use non-overlapping hues or palettes to avoid confusion.

Task progress calculation: In the analysis history, each node features a *progress* attribute that quantifies the task completion degree. Following each task execution, the agent evaluates the task's completion status. If further in-depth analysis is deemed necessary, indicating the task remains incomplete, the progress is set to 0%; otherwise, it is set to 100%. Upon determining a task as incomplete, the agent suggests a decomposition plan. Users can set the progress to 100% by denying the plan. Each recently executed task, represented as a leaf node, is assigned a progress value, initiating a bottom-up refresh of progress values for non-leaf nodes across the tree (Figure 3-e). This update process for internal nodes averages the progress of two specific child node categories: those derived from task decomposition and those recommendation nodes identified by users for inclusion.

4.1.3 Task Decomposition

Upon the completion of each task, the Decomposer evaluates the quality of completion. The output includes a score ranging from 0 to 10 and an explanation along with the scoring. If decomposition is needed, a detailed decomposition plan with the appropriate logic operators will be proposed (Figure 3-f). Conversely, if no further decomposition is required, the Recommender will propose new exploration tasks. According to the framework, the evaluation process can be guided by three main guidelines. The first is to determine if the initial solution adequately segments the data, ensuring that the analysis covers all relevant subsets. Second, assess if the task requires advanced statistical analysis (e.g., regression, clustering, hypothesis testing) to extract deeper insights. Third, evaluate if the current visualizations are appropriate for the insights being communicated and if different types might better reveal underlying patterns.

In cases where a task requires decomposition, a depth-first decomposition process is employed. This process is guided by a predefined maximum decomposition step depth (k_{max}), managing the depth (k) of decomposition (decompose when $k < k_{max}$). Each task can be decomposed based on two types of relationships:

Prompt Template 3 (Task Decomposition).

Is complete

Input: {`task, insight`}
Instruction: You need to judge whether the task requires further analysis. Ensure that the task aligns well with the overall goal. If the task appears tangential or incomplete, it may need to be refined or decomposed further. Please rate the task from 1-10 according to the degree of completion. {`Decomposition guidelines added here`}
Indicator: {`A output template in JSON format`}
Output: {`score, explanation`}

Decompose

Input: {`task, insight, score, explanation`}
Instruction: This task requires further analysis, including more detailed data segmentation or advanced statistical methods. Please generate no more than n subtasks and indicate the methods they use respectively based on this. (other guidelines added here.)
Indicator: {`An example in JSON format`}
Output: {`subtasks, execution order`}

- *AND*: Tasks are executed in parallel with multi-agents. For computational efficiency, further decomposition occurs only if a single node uncompleted. Multiple uncompleted indicate the failure of the task.
- *DOWN*: This is a particular *AND* case. Tasks will be executed in serial order with a single agent.

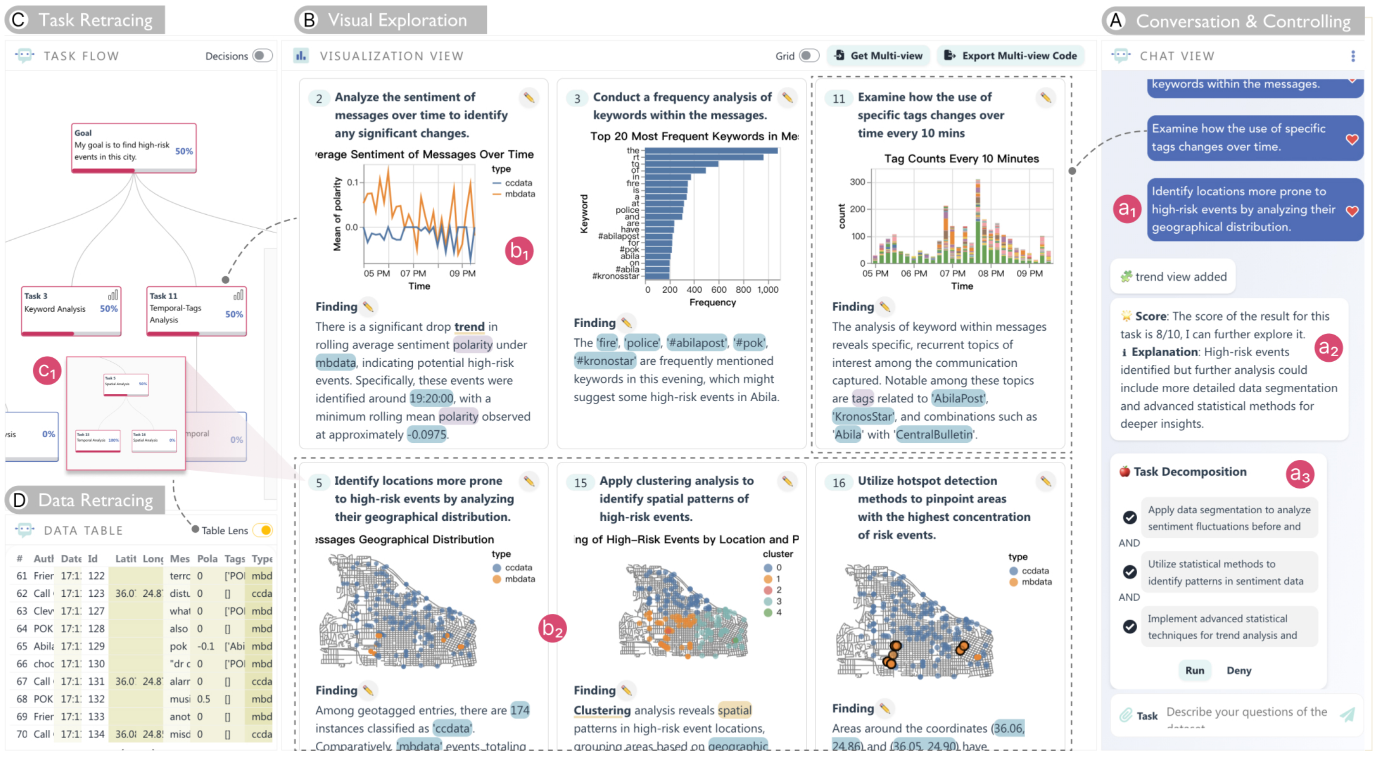
We use an example to illustrate the decomposition process. If the task of *analyzing vehicle weight and fuel efficiency* is completed with a score of 6/10, the agent might suggest the following decompositions: T1: *Segment the data into weight categories (light, medium, heavy) and analyze the fuel efficiency within each segment.* T2: *Within each weight category, conduct a clustering analysis to identify patterns or groupings that could further explain variations in fuel efficiency.* T3: *Perform a multiple regression analysis to control for additional variables like engine size and vehicle age*. The execution logic would be (T1 DOWN T2 AND T3).

According to the logic, the agent executes each subtask and summarizes the subtasks' insights to formulate an overall insight for the decomposed task. No further decomposition is required if all subtasks are completed or up to the max steps. Then, considering the goal and analysis results, the agent might propose revisiting previously proposed tasks or recommending new exploration tasks.

4.2 LightVA Interface

The agent-based interface includes several views to enable user-controlled visual exploration, as shown in Figure 4. To enable this, we provide four views: *Chat view*, *Visualization view*, *Task flow view*, *Data view*. The design of the interaction follows the design considerations of Section 3.2 and provides three modes of interaction.

Chat view: The exploration begins with the *Chat view* (Figure 4A), an LLM-based chat box that facilitates direct communication between users and agents through natural language interaction. The interaction between the user and agent can switch between delegate, guide, and discuss. In this interface, the user uploads data and inputs goals. The agent then responds with its understanding of the dataset and suggests new tasks in the form of buttons. Users can bookmark tasks of interest, and tasks that are not of interest will not be counted in the progress of exploration. The user can then execute a task by clicking on it. Additionally, users can enter their own tasks in the chat box. After submitting a task, the agent will provide a visualization of the production and insights obtained from the calculations in the *Visualization view* (Figure 4B). If the task needs to be decomposed, the task decomposition plan will appear. The user can modify the logical relationship between the task and the execution plan by switching the text button. The agent will analyze the task based on the user's instructions and return the results of multiple subtasks. During the exploration process, users can ask various questions and discuss them



2 Analyze the sentiment of messages over time to identify any significant changes.

Finding
There is a significant drop trend in rolling average sentiment polarity under mbdata, indicating potential high-risk events. Specifically, these events were identified around 19:20:00, with a minimum rolling mean polarity observed at approximately -0.0975.

3 Conduct a frequency analysis of keywords within the messages.

Finding
The 'fire', 'police', '#abilapost', '#pok', '#kronostar' are frequently mentioned keywords in this evening, which might suggest some high-risk events in Abila.

11 Examine how the use of specific tags changes over time every 10 mins

Finding
The analysis of keyword within messages reveals specific, recurrent topics of interest among the communication captured. Notable among these topics are tags related to 'AbilaPost', 'KronosStar', and combinations such as 'Abila' with 'CentralBulletin'.

5 Identify locations more prone to high-risk events by analyzing their geographical distribution.

Finding
Among geotagged entries, there are 174 instances classified as 'ccdata'. Comparatively, 'mbdata' events, totaling

15 Apply clustering analysis to identify spatial patterns of high-risk events.

Finding
Clustering analysis reveals spatial patterns in high-risk event locations, grouping areas based on geographic

16 Utilize hotspot detection methods to pinpoint areas with the highest concentration of risk events.

Finding
Areas around the coordinates (36.06, 24.86) and (36.05, 24.90) have

keywords within the messages.

Examine how the use of specific tags changes over time.

Identify locations more prone to high-risk events by analyzing their geographical distribution.

trend view added

Score: The score of the result for this task is 8/10, I can further explore it.
Explanation: High-risk events identified but further analysis could include more detailed data segmentation and advanced statistical methods for deeper insights.

Task Decomposition
Apply data segmentation to analyze sentiment fluctuations before and
AND
Utilize statistical methods to identify patterns in sentiment data
AND
Implement advanced statistical techniques for trend analysis and
Run Deny

Task Describe your questions of the dataset

#	Autl	Date	Id	Latit	Long	Mes	Pola	Tags	Type
61	Frier	17:1	122			terrc	0	['POI	mbd
62	Call	17:1	123	36.0'	24.8'	distu	0	[]	ccda
63	Clev	17:1	127			what	0	['POI	mbd
64	POK	17:1	128			also	0	[]	mbd
65	Abil	17:1	129			pok	-0.1	['Abi	mbd
66	choc	17:1	130			"dr c	0	['POI	mbd
67	Call	17:1	131	36.0'	24.8'	alarr	0	[]	ccda
68	POK	17:1	132			musi	0.5	[]	mbd
69	Frier	17:1	133			anot	0	[]	mbd
70	Call	17:1	134	36.0	24.8	misc	0	[]	ccda

Fig. 4: The LightVA system comprises four views. Users can communicate with LLMs and control the planning process in (A) Chat view by selecting the tasks or setting the decomposition plan; the generated visualization and insights from LLMs are updated in (B) Visualization view. Task flow view (C) explains the task planning process and allows users to control the analysis pipeline Task flow. When a task is completed, users check the data exploration situation in the Data table with table lens (D).

with LLM in the dialog box, such as the problem analysis method, understanding of the answer, and explanation for the recommendation.

Visualization view: This view presents visualizations and insights in the form of cards. Each card contains the task serial number, the task content, the interactive visualizations, and the insights in rich text format. Users can modify the Vega-Lite JSON codes and insight text. To address the issue of cognitive load for the user, we allow the user to select which visualization cards they want to view, and these selected cards are displayed in the main view while the remaining cards are moved to the candidate set below. Additionally, the user can merge the selected cards to create an interactive linked view. Moreover, users can modify and export the generated codes, which providing flexibility for those who need to customize their analyses further.

Task flow view: The task flow view updates as goals and tasks are added, providing the user with a clear analysis of status and progress (Figure 4C). Each node is a rectangular box showing the task id, task type, and percent progress. The original task text appears when the cursor is over a node. Clicking on a node will update the visualization layout in the *Visualization view*. Hovering over the edge can see the differences in data and task type between the associated nodes. In accordance with design considerations, we allow users to choose unexplored tasks to be executed and delegate them to the agent. Additionally, users can remove pending tasks from the flow to reduce workload. A clear exploration structure may inspire user's ideas.

Data table view: In addition to task visualization, we provide a data lens visualization in *Data table view* (Figure 4D) to guide users in the large exploration space. When a task is being completed, the user can observe the exploration of the accumulated data usage frequency in table lens mode. The color of the table cell represents the relative frequency of each cell being explored. This observation may inspire the user to discover regions of interest to propose new questions.

4.3 Error Handling

We refer to *errors* as issues that cause the system to crash or become unresponsive. Due to the inherent unpredictability of LLM outputs, errors may occur if the outputs cannot be parsed correctly, causing the system to crash or become unresponsive. To ensure the system operates correctly and prevents workflow disruptions, we implement an error-handling mechanism.

To develop an effective error-handling mechanism, we conducted a test using two datasets in expert evaluation with 20 agent-opposed tasks, employing both GPT-3.5-turbo and GPT-4-turbo. Each task was tested with code generation and insight annotation, utilizing two different prompting techniques. This resulted in a total of 160 initial tests. We classified the errors in the test into five categories: *(1) Unfamiliar dataset, (2) Data binding issues, (3) Serialization issues, (4) Data transformation issues, (5) Syntax errors*. More details about the test can be found in Appendix B. To address these types of errors effectively, we implemented specific error-handling strategies in three ways: before model generation with prompting techniques, within the system after generation, or delegated to users.

Prompting techniques: (1) Few-shot prompting: We provide examples to assist the model in better grasping the requirements of the outputs. For example, when making insight annotations, it is flexible to indicate key information from natural language. Examples can explain the components of insight to the model. (2) Chain-of-thought: We guide the model through a thought plan with step-by-step instruction, which is useful in reasoning processes, such as task execution.

Within system handling: (1) Self-correction: We allow the LLM to self-examine and correct if the codes are executed with an error. Based on our test results, the self-correction helped reduce around 40% initially identified errors, such as syntax errors, spelling mistakes, and logical inconsistencies. (2) Catching and feedback: Common syntax errors, such as matching quotes and parentheses, can be solved by rule-based solutions. We classify the common errors to highlight which steps in the data analysis process the LLM made mistakes rather than

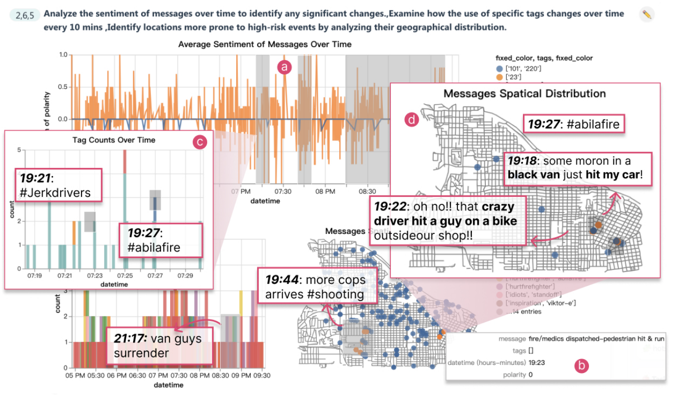

2,6,5 Analyze the sentiment of messages over time to identify any significant changes.,Examine how the use of specific tags changes over time every 10 mins ,Identify locations more prone to high-risk events by analyzing their geographical distribution.
Average Sentiment of Messages Over Time
Tag Counts Over Time
Messages Spatial Distribution
19:21: #Jerkdrivers
19:27: #abilafire
19:27: #abilafire
19:18: some moron in a black van just hit my car!
19:22: oh no!! that crazy driver hit a guy on a bike outsideour shop!!
19:44: more cops arrives #shooting
21:17: van guys surrender
message fire/medics dispatched--pedestrian hit & run
tags []
datetime (hours–minutes) 19:23
polarity 0
datetime
count
a
b
c
d

Fig. 5: A linked view example generated by LightVA for event analysis. The timeline, histograms, and map are selected by the user to generate a linked view. The charts can be brushed or clicked to filter each other (a). The polarity, tags, and message can be found in the tooltip (b) of each view to support detailed analysis. By interactively exploring the linked views (c, d), we find some dangerous events, such as "Hit and run", "Standoff", "Fire".

merely pointing out low-level errors.

User-side handling: Once errors are identified, the system notifies the user, and the user may edit the codes. Additionally, we maintain the analysis history, allowing for a rollback to the previous step if an error occurs during the execution of the current task. Users can choose a new task and remove the failed task from the task flow.

5 Usage Scenario: Event Analysis

To examine the effectiveness of our framework, we demonstrate the LightVA with IEEE VAST Challenge 2021 Mini-Challenge 3 [2] as a usage scenario. The challenge's goal is to detect events that happened in Abila City during the evening of January 23, 2014. The provided data include microblog records and emergency dispatch records from a call center. Our motivation for this scenario stems from two reasons. Firstly, it incorporates a representative blend of data types and corresponding visualizations, encompassing text, spatial, and temporal data, which is a typical VA scenario. Secondly, the VAST challenge has the ground truth to support an objective evaluation of our LightVA. We use the OpenAI GPT-4 model in our work. For a more detailed demonstration of this scenario, a video is added to the supplemental materials.

Initialization: In the beginning, we upload a dataset .csv file and a map outline data .json file at the *Chat view*. Then, we type a goal, "find some high-risk events in this city". The dataset is introduced briefly, and the system proposes four initial exploration tasks based on our goal: *sentiment analysis, keyword analysis, tags analysis, and spatial analysis* (Figure 4-a1). As the four tasks are aligned with our objectives, we bookmark them all in our progress. We then prioritize the exploration of the first task - *sentiment analysis*. Upon submission, the agent generates a time varying card at the *Visualization view* (Figure 4-B). The generated insight for sentiment analysis indicates that the microblog records has experienced a drop in average sentiment, especially with a minimum polarity in 19:20, indicating potentially dangerous events (Figure 4-b1).

Task recommendation: The agent not only generates the visualization and findings but also evaluates them. For this *sentiment analysis* task, the agent gives a score of 8/10 and explains that the generated results have basically completed the task, but further statistical analysis can be performed (Figure 4-a3). As we prefer to start with a broad overview, we decide not to decompose. The agent thus reminds us to revisit our previous collection tasks in time. Thus, we choose the second task (*keyword analysis*), which involves generating a preliminary keyword visualization. The generated finding summarizes the highly frequent words such as "fire", "police", and "POK". In addition to the agent's finding, clicking over the stacked histogram for task 11 shows the time range of each word. For example, the "shooting" from 19:40 until 19:50 and "abilafire" from 19:00 until 21:30. These findings suggest high-risk incidents like shootings and fires at first glance. The agent then recommends *spatial analysis*. From the *Data table* with lens, we can notice that there is indeed no latitude and longitude explored so far, which indicates that agent recommendation can pay attention to the data coverage (Figure 4-D). After execution, the system generates a map with shapes and messages, and the decomposition structure is updated on the Task flow (Figure 4-c1). We observe a higher concentration of microblogs in two specific locations, which might indicate incidents affecting public safety in those areas (Figure 4-b2).

Task decomposition: Based on the assessment of tasks and preliminary results, the agent recommends further decomposition with three subtasks: *clustering analysis, hotspots detection, and prediction modeling*, with an AND and DOWN logic (Figure 4-a3). To conduct a retrospective analysis, we choose the first two subtasks. The cluster analysis combines location and polarity to divide messages on the map into five categories. We find that two places on the left and right sides of the map have a distinct sentiment distribution. The second sub-task, through analyzing message quantity, highlights hotspots, visually indicating areas with dense messages.

Linked-view analysis: Based on the suggestion of the agent, we can form the analytical logic from multi-variables to concrete details. To find events more clearly, we select three tasks from the task flow view. The tasks include sentiment evolution charts, tag evolution charts, and a map to form an interactive multi-view (Figure 5). Through interaction on line chart (Figure 5-a), we observe the histogram and map and discover that around 19:20, a "hit-and-run" event occurs: a "black van" first collides with a small car and then a cyclist, sparking a lot of discussions (Figure 5-b, c). Additionally, we find that the fire occurred near the hit-and-run site. Later, we find that the black van heads west by 19:44, and a shooting occurs with the police, marking the period with the highest discussion intensity. Finally, at 21:17, the van guys surrender. By observing the table and task flow, we find that the progress of the goal reached 100%, and the data is basically covered, "type" "latitude" and "longitude" were explored frequently.

To conclude, in this scenario, we find significant events such as Fire, Hit and Run, and Stand-off without manually coding and designing. With task planning, we generate eight views with auto-summarized findings based on statistical analysis and a linked view to solve the goal rapidly. Compared to our submission to the VAST Challenge [39], where we spent 2 people, 2 days, and 6 hours on data preprocessing (including tagging, categorizing, and removing spam messages) plus 2 people, 7 days, and 6 hours on interface construction and analysis, totaling around 108 hours. In LightVA, we upload the preprocessed data. Further construction and analysis take roughly 1 hour. This represents an efficiency increase of approximately 5 times, as the current effort is 25 out of 108 hours. Moreover, it is worth noting that data wrangling still require human involvement, especially in complex scenarios. One potential avenue would be to integrate agent-assisted data annotation and cleaning into LightVA's workflow. In addition, there may be a lack of finesse and aesthetic appeal in interface visualizations and interactions when compared to manual designs. Despite this, the overall cost-effectiveness has been enhanced. To address this concern, we offer support for exporting the code, which can be further modified as needed.

6 Expert Study

To examine the effectiveness of LightVA in VA system construction and task recommendation, we invited two VA experts (denoted as E1 and E2) and a domain expert (denoted as E3) to participate in the

2. https://vast-challenge.github.io/2021/MC3.html

study. These experts have experience in data analysis and are familiar with using VA systems.

Participants background: E1 is a VA expert specializing in VA system development, topics around digital humanities, text-based data, road data for autonomous driving, spatiotemporal datasets, and tabular data. E1 often uses D3.js and Vue.js to develop VA systems. E2 is a VA expert with experience in autonomous driving and social media, as well as work experience in business analysis. In daily work, E2 chooses to use tools like Tableau within the company to analyze quantitative data. The frequency of using data analysis and visualization tools is about weekly. E3 is a domain expert analyzing data for fast-moving consumer goods and supply chains. E3 frequently uses Excel and Power BI and occasionally uses Python for in-depth analysis. The use of data analysis and visualization tools occurs on a daily basis.

Procedure: The study consists of three sessions. First, we spent 15 minutes to know the experts' backgrounds in data analysis and introduced our work by showing the video demonstration. Then, in the exploration phase, the experts spend about 30 minutes exploring the system using the think-aloud method [51]. Considering E3's daily work requirements and aligning with their domain expertise, we opted for a dataset that mirrors their routine tasks. Due to data confidentiality concerns, we substituted the original dataset with a comparable one related to sales for E3's use. Meanwhile, the automotive dataset was designated for exploration by E1 and E2, fitting their respective areas of expertise. During the system usage stage, we observed and recorded how they interacted with the LightVA system. In the final stage of the interview, we spent approximately 30 minutes gathering the experts' feedback on their usage experience of the system.

6.1 Visual Analytics Expert Evaluation: Cars Dataset

In this study, we evaluate LightVA with E1 and E2 using a dataset about cars. The dataset has 406 records and 9 attributes, including brand, model, performance indicators (such as horsepower and cylinders), and the year of manufacture and place of origin.

During the exploration process, E1 is more concerned with the construction of the VA system. At the beginning, E1 sets the analytical goal to *identify the factors influencing fuel efficiency*. E1 chooses one of the initially proposed tasks for execution. After completing that task, the agent presents a detailed decomposition plan. E1 then follows the agent's guidance to continue selecting subsequent sub-tasks for further decomposition and in-depth analysis. In the generated visualization (Figure 6-a1), E1 discovers the differences in the distribution of the number of vehicles produced by different origins across various performance ranges in the bar chart on the right, by utilizing the area brushing feature on the scatter plot. E1 also discovers the performance differences between cars with different numbers of cylinders through the filter functionality between linked view (Figure 6-a2).

Another expert E2, demonstrates a focused interest in the data analysis, pursuing a clear analytical goal that was refined throughout the analysis process. Initially, E2's goal was *to identify vehicles that excel both in fuel efficiency and performance* (Figure 6-b1). However, recommended insights revealed a strong negative correlation between fuel efficiency (measured in MPG) and the majority of performance metrics, indicating a trade-off between high fuel efficiency and superior performance capabilities. This insight led E2 to quickly adjust the analytical goal towards *prioritizing fuel efficiency while ensuring performance was not significantly compromised* (Figure 6-b2). Based on the plan proposed in task decomposition, E2 explored the relationship between various vehicle performance metrics and fuel efficiency with five charts. Finally, E2 combined these charts to generate linked view (Figure 6-b3). By filtering points across views, E2 found several vehicle models meeting the revised goal.

6.2 Domain Expert Evaluation: Sales Dataset

To further evaluate LightVA, we conducted another study with a domain expert. E3 was interested in a sales dataset for a large store

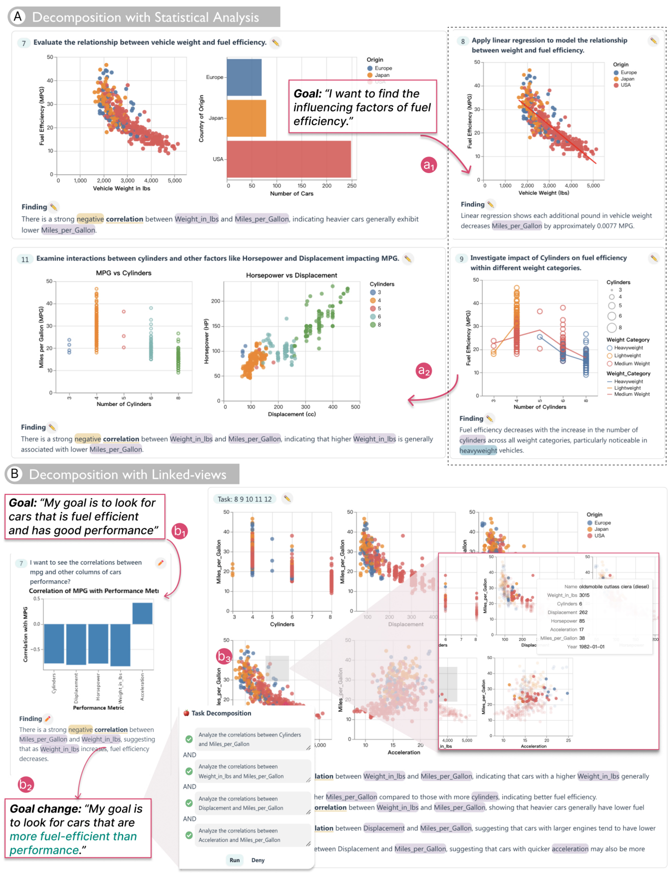

A Decomposition with Statistical Analysis
Evaluate the relationship between vehicle weight and fuel efficiency.
Goal: "I want to find the influencing factors of fuel efficiency."
a1
Apply linear regression to model the relationship between weight and fuel efficiency.
Finding
There is a strong negative correlation between Weight_in_lbs and Miles_per_Gallon, indicating heavier cars generally exhibit lower Miles_per_Gallon.
Linear regression shows each additional pound in vehicle weight decreases Miles_per_Gallon by approximately 0.0077 MPG.
Examine interactions between cylinders and other factors like Horsepower and Displacement impacting MPG.
MPG vs Cylinders
Horsepower vs Displacement
a2
Investigate impact of Cylinders on fuel efficiency within different weight categories.
Fuel efficiency decreases with the increase in the number of cylinders across all weight categories, particularly noticeable in heavyweight vehicles.
B Decomposition with Linked-views
Goal: "My goal is to look for cars that is fuel efficient and has good performance"
b1
I want to see the correlations between mpg and other columns of cars performance?
Correlation of MPG with Performance Metrics
b3
Task Decomposition
Analyze the correlations between Cylinders and Miles_per_Gallon
AND
Analyze the correlations between Weight_in_lbs and Miles_per_Gallon
AND
Analyze the correlations between Displacement and Miles_per_Gallon
AND
Analyze the correlations between Acceleration and Miles_per_Gallon
Run
Deny
b2
Goal change: "My goal is to look for cars that are more fuel-efficient than performance."

Fig. 6: The illustrations of the expert study using our framework with cars dataset. Two VA experts show different analysis preferences during the exploration process. (A) E1 focuses on the creation of VA, progressively decomposing the goal and employing statistical methods to complete the exploration. (B) E2 has a clear goal during the analysis process, and continuously optimizes the goal with task decomposition, and leverages the linked-view interactions to find a satisfactory answer.

that provides detailed transaction information. The dataset contains 3,312 records with 21 fields, e.g., sales, discount, profit, and category.

At the beginning, E3 mentioned the high-time sensitivity of sales data for products in real-world scenarios. Therefore, in the initial tasks, E3 opted to explore the temporal trends of product sales volumes. In the histogram and insight (Figure 7-a1) suggested by the agent, E3 found that the overall sales of the item differ significantly from month to month. In the next step of the decomposition plan, E3 categorized the products to explore the temporal trends of different categories (Figure 7-a2). It was found that the temporal trends of the different categories were broadly similar, but the total sales volume of the technology category in November was the highest. Moreover, following the agent's decomposition into another task: *"Analyze sales trends of top-selling and bottom-selling products"*, E3 noted that in the domain scenario, *"finding out how well products are selling can optimize inventory to prevent stockouts and excess inventory."* Therefore, E3 selected a task previously proposed but not executed, recommended for review by the agent, to rank the sales volumes of different categories and subcategories (Figure 7-a3). Based on the generated visualization, E3 discovered that the "phone" subcategory within the "Technology" category has the best sales performance.

After finding the answer to the previous task, E3 was interested in *"analyze the correlation between sales and profit."* Upon receiving the specific scatter plot (Figure 7-b1), E3 discovered that high sales do not always mean higher profit. The expert speculated that this may be due to the impact of discounts, as items with high sales often have significant discounts. E3 confirmed this hypothesis through interaction with and analysis of the scatter plots of sales and profit, with and without discount (Figure 7-b2). To find the optimal discount rate for products, E3 chose to analyze the relationship between discount and profit across different product categories. According to Figure 7-b3, the differences between product categories are minor, with all reaching maximum profit at a 10% discount.

6.3 Results and Analysis

This section discusses the expert's exploration process and feedback on the system construction and task planning.

Exploration process analysis: The exploration behaviors of the three experts differ significantly. VA expert E1 created the most visualization charts, with half including interactive features, emphasizing the generation of visualizations and the exploratory use of linked views. Furthermore, E1 delved deeper into task decomposition to employ more complex statistical methods, such as linear regression and polynomial regression. E2, with profound data analysis experience, did not strictly follow the agent's recommendations during the exploration process. Instead, E2 refined and improved the analysis goals and exploration directions based on the results of previous tasks, proposing new questions and flexibly employing interactive VA to complete the goal. E3, as a domain expert, possessed an understanding of the characteristics of certain data attributes within the dataset, such as the relationships between discounts, sales, and profit. E3's analysis was grounded in real-world scenarios, selecting different analysis methods as needed to integrate data insights into practical applications. This suggests our system supports a degree of flexibility, responding effectively to users' individual needs.

Feedback on system construction: E2 and E3 both expressed that interactive analysis through linked views facilitates a better understanding of the data. From the perspective of a VA expert, E1 commented that *"basic charts and interactions can be generated, which are sufficient for simple data analysis problems, but modifications might be necessary for more complex issues."* E2 and E3 agreed that the system can effectively generate insights, enhancing the accuracy of analysis. E2 suggested that generating insights based on awaring of user behavior would enhance the execution of tasks. Additionally, E3 noted the significant reduction in manual effort due to AI generation, stating, *"Previously, using PowerBI required a lot of time and operations, but now it only takes a sentence."*

Feedback on task planning: E3 observed that *"the proposed tasks are quite good, indicating the system has a certain understanding of the dataset, and the language used is very standard."* E1 valued the system's ability to decompose tasks as *"the most useful part,"* which can provide deeper analysis and reveal certain characteristics of the data, especially for those lacking domain-specific expertise. However, domain expert E3 cautioned that the decomposed tasks are not always reliable, stating, "The process of task decomposition needs to be explained." Regarding task recommendations, E1 remarked that the recommended tasks might be divergent for specific analysis goals. Conversely, E3 pointed out that in actual domain scenarios, exploring a dataset is not limited to one aspect. Supplementing the original question from different perspectives can provide enlightening insights. We discuss this needs for personalization accoding to different user profiling in Section 7

7 DISCUSSION

This section outlines the limitations of our work and discuss the implications learned from the research with the future directions.

Generalizability of the framework: Our framework is generalizable in three aspects. First, our conceptual framework unifies the development and usage of VA systems, based on the connections between goals, tasks, visualizations, and insights. Secondly, the expert study indicates that our system saving more effort than manual

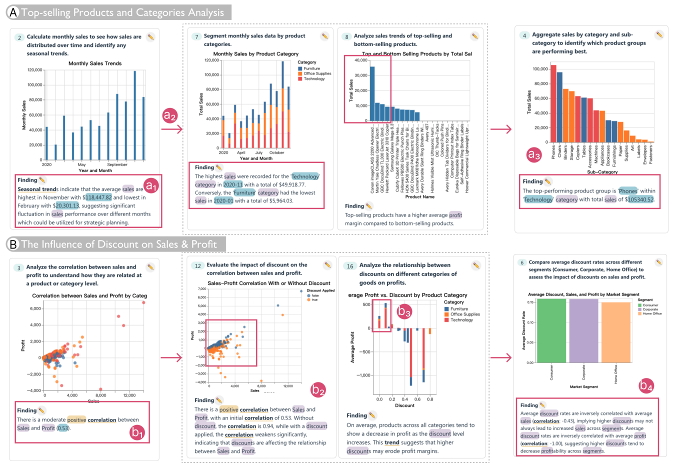

A Top-selling Products and Categories Analysis
Calculate monthly sales to see how sales are distributed over time and identify any seasonal trends.
Monthly Sales Trends
Segment monthly sales data by product categories.
Monthly Sales by Product Category
Analyze sales trends of top-selling and bottom-selling products.
Top and Bottom Selling Products by Total Sal
Aggregate sales by category and sub-category to identify which product groups are performing best.
a1
a2
a3
Finding
Seasonal trend: Indicate that the average sales are highest in November with $118,447.82 and lowest in February with $20,301.13, suggesting significant fluctuation in sales performance over different months which could be utilized for strategic planning.
The highest sales were recorded for the Technology category in 2020-11 with a total of $49,918.77. Conversely, the Furniture category had the lowest sales in 2020-01 with a total of $5,964.03.
Top-selling products have a higher average profit margin compared to bottom-selling products.
The top-performing product group is 'Phones' within Technology category with total sales of $105,340.52.
B The Influence of Discount on Sales & Profit
Analyze the correlation between sales and profit to understand how they are related at a product or category level.
Correlation between Sales and Profit by Categ
Evaluate the impact of discount on the correlation between sales and profit.
Sales–Profit Correlation With or Without Discount
Analyze the relationship between discounts on different categories of goods on profits.
erage Profit vs. Discount by Product Category
Compare average discount rates across different segments (Consumer, Corporate, Home Office) to assess the impact of discounts on sales and profit.
b1
b2
b3
b4
There is a moderate positive correlation between Sales and Profit (0.53).
There is a positive correlation between Sales and Profit, with an initial correlation of 0.53. Without discount, the correlation is 0.94, while with a discount applied, the correlation weakens significantly, indicating that discounts are affecting the relationship between Sales and Profit.
On average, products across all categories tend to show a decrease in profit as the discount level increases. This trend suggests that higher discounts may erode profit margins.
Average discount rates are inversely correlated with average sales (correlation: -0.43), implying higher discount may not always lead to increased sales across segments. Average discount rates are inversely correlated with average profit (correlation: -1.00), suggesting higher discounts tend to decrease profitability across segments.

Fig. 7: The example results of the domain expert study with a superstore sales dataset. E3 implements analysis from overview to details and beyond with two rounds of task decomposition. (A) The top-selling products and specific categories are found for inventory formulation. (B) The system effectively pointed out the impact of discounts on sales and profits, promoting hypothesis generation and verification.

tools like Tableau or PowerBI. Comparing with existing visualization software, e.g, Tableau (with AI version), LightVA offers several distinct advantages. While both LightVA and Tableau support natural language interfaces, task recommendation and single visualization generation, only LightVA provides task decomposition, multi-view visualization generation, and iterative human-in-the-loop task completion evaluation. In summary, Tableau (with AI version) supports basic NLI functions, task recommendation, and single visualization generation. However, it lacks support for task completion, task decomposition, and automatic multi-view visualization generation, which are the focuses of LightVA.

However, there are some limitations to our framework. First, the framework may require more detailed guidelines for complex and specific domains, particularly in task customization, visualization methods, and model selection. A more comprehensive grammar for tasks and insights would improve model output evaluation and refinement. This necessitates extensive research and documentation, categorizing different tasks and domains to enhance the framework's applicability. Second, the relatively small number of participants in the expert study limits the generalizability of the results. Conducting a larger crowdsourced user study would be beneficial to further verify the quality of the automated output and to investigate the factors that contribute to users' perceived quality.

The performance of LLMs in VA tasks: While LLM exhibits potentials, some of the limitations of LLM requires careful handling.

- **Output stability and accuracy**: The output of LLMs can be unstable. During our evaluation, we encountered instances where the model failed to follow instructions accurately, leading to parsing and execution failures in the generated code. In order to address this problem, we propose an error-handling mechanism. Although this mechanism helps to avoid errors making system crashes, additional LLM errors about incorrect facts should be detected and corrected. In the future, we can incorporate LLMs' self-reflection to solve hallucination [21] and provide insights on errors so that users can work together to fix these errors.
- **Response speed**: From our test, GPT-3.5-turbo completes tasks in significantly less time than GPT-4-turbo. To increase the running speed, we use a multi-agent parallel computing strategy and error handling for time exceeding a certain threshold. A future direction could be to utilize caching mechanisms such as GPTCache [2] to explore acceleration strategies in data analysis scenarios.

- **Problem-solving ability**: Our test results indicate that LLMs can struggle with complex computational tasks, such as predictive modeling, leading to errors. This highlights the need for a broader evaluation of model capabilities across various complex domain-specific problems to derive guidelines or evaluation metrics for task planning and execution. Additionally, we found that LLMs can handle data transformation and statistical modeling analysis, but they struggle with issues like missing values and inconsistent punctuation. From the implementation side, we can provide additional tools and APIs and teach agents to use them effectively [40].
- **Domain knowledge**: In some scenarios, LLMs may not have sufficient domain knowledge. To address this limitation, future research can focus on combining retrieval augmented generation (RAG) and fine-tuning for LLMs to solve specific domain tasks [56].

LLM-generated code versus human-written code: Through study results, we find that the integration of LLMs within the LightVA framework underscores a significant shift towards more efficient and cost-effective VA system development. While recognizing the limitations in the quality and customization of code produced by LLMs, we enable code modification and exportation. Moving forward, the balance between automation and human expertise will remain a critical consideration. Future research can integrate LLM capabilities to support co-coding [34] for specific scenarios.

Injecting visualization design knowledge: Design knowledge is important in designing VA systems. In our implementation, we constrain color, interaction, and layout. However, these constraints are not exhaustive and we only consider them as preliminary. Different application scenarios may require different visual designs. Further research may incorporate more design guidelines into the LLM with additional prompting techniques or combining vision models such as GPT-4V [71] to perceive and evaluate the effectiveness. In addition, the inclusion of a memory module [37] can be integrated to support automatic visual consistency.

Automation versus personalization: From the expert study, we found that users from different backgrounds have different needs when using the system. Some users have a clear idea of what they want and need the agent to follow orders, but some others want the agent to act as the lead in a more automated way. We currently allow users to implement both ways. In the future, we can conduct further qualitative experiments to test the difference in decision-making paths with the agent's asisstantce with experts from visualization backgrounds [3], [30]. We can use visualization and graph algorithms to visually compare their task flows. This experiment could help us build a preference knowledge base for different types of users in the form of fine-tuning or external knowledge retrieval, allowing LLM to provide more personalized assistance. In addition to assistance, we can explore different roles of collaboration, such as Human-creator with AI-optimizer [25]. To support personalized creation, combining multiple interaction ways allows flexible customization beyond natural languages, such as generating dynamic visualization widgets [50].

8 Conclusion

In this paper, we aim to reduce the complexities and technical demands of current visual analytics practices. Therefore, We introduce LightVA, a lightweight visual analytics framework that supports task planning, insight analysis, and linked visualization generation based on human-agent collaboration. Our framework utilizes LLM agents for task planning and execution. The framework employs a recursive approach in which the agents recommend tasks, break down complex tasks into subtasks, and generate visualization and data modeling codes to solve tasks. We develop a system that embodies our proposed framework, supporting users to analyze data based on the communication with LLM agents and use the task-driven generated VA system. A usage scenario and an expert study suggest that LightVA not only reduced the manual effort required but also provided new opportunities to leverage LLMs to facilitate visual data exploration.

Supplemental Materials

To supplement the main text, we provide several materials. Appendix A.1 introduces prompt examples guiding LLMs. Appendix A.2 describes an example data introduction to let LLMs understand the dataset for code generation. Appendix A.3 describes the analysis history data structure. Appendix B describes the test results of LLM performance for error analysis.

References

[1] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *Proceddings of IEEE Symposium on Information Visualization*, pp. 111–117. IEEE, 2005. doi: 10.1109/INFVIS.2005.1532136

[2] F. Bang. GPTCache: An open-source semantic cache for llm applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS)*, pp. 212–218. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.nlposs-1.24

[3] L. Battle and J. Heer. Characterizing exploratory visual analysis: A literature review and evaluation of analytic provenance in tableau. *Computer graphics forum*, 38(3):145–159, 2019. doi: 10.1111/cgf.13678

[4] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, J. Gajda, T. Lehmann, M. Podstawski, H. Niewiadomski, P. Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023. doi: 10.1609/aaai.v38i16.29720

[5] F. Bouali, A. Guettala, and G. Venturini. Vizassist: an interactive user assistant for visual data mining. *The Visual Computer*, 32:1447–1463, 2016. doi: 10.1007/s00371-015-1132-9

[6] S. M. Casner. Task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics (ToG)*, 10(2):111–151, 1991. doi: 10.1145/108360.108361

[7] D. Ceneda, T. Gschwandtner, T. May, S. Miksch, H.-J. Schulz, M. Streit, and C. Tominski. Characterizing guidance in visual analytics. *IEEE transactions on visualization and computer graphics*, 23(1):111–120, 2016. doi: 10.1109/TVCG.2016.2598468

[8] S. Chen, L. Lin, and X. Yuan. Social media visual analytics. *Computer Graphics Forum*, 36(3):563–587, 25 pages, 2017. doi: 10.1111/cgf.13211

[9] C. Demiralp, P. J. Haas, S. Parthasarathy, and T. Pedapati. Foresight: Recommending visual insights. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, 10(12), 2017. doi: 10.48550/arXiv.1707.03877

[10] D. Deng, A. Wu, H. Qu, and Y. Wu. DashBot: Insight-driven dashboard generation based on deep reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):690–700, 2023. doi: 10.1109/TVCG.2022.3209468

[11] V. Dibia. LIDA: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. In D. Bollegala, R. Huang, and A. Ritter, eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 113–126. Association for Computational Linguistics, Toronto, Canada, July 2023. doi: 10.18653/v1/2023.acl-demo.11

[12] V. Dibia and Ç. Demiralp. Data2Vis: Automatic generation of data visualizations using sequence to sequence recurrent neural networks. *IEEE computer graphics and applications*, 39(5):33–46, 2019. doi: 10.1109/MCG.2019.2924636

[13] R. Ding, S. Han, Y. Xu, H. Zhang, and D. Zhang. QuickInsights: Quick and automatic discovery of insights from multi-dimensional data. In *Proceedings of the International Conference on Management of Data*, SIGMOD '19, 16 pages, pp. 317–332. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3299869.3314037

[14] D. Gotz and Z. Wen. Behavior-driven visualization recommendation. In *Proceedings of the 14th international conference on Intelligent user interfaces*, IUI '09, 10 pages, pp. 315–324. Association for Computing Machinery, New York, NY, USA, 2009. doi: 10.1145/1502650.1502695

[15] 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*, 2021. doi: 10.1145/3643894

[16] S. Hao, Y. Gu, H. Ma, J. Hong, Z. Wang, D. Wang, and Z. Hu. Reasoning with language model is planning with world model. In H. Bouamor, J. Pino, and K. Bali, eds., *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173. Association for Computational Linguistics, Singapore, Dec. 2023. doi: 10.18653/v1/2023.emnlp-main.507

[17] M. M. Hassan, A. Knipper, and S. K. K. Santu. Chatgpt as your personal data scientist. *arXiv preprint arXiv:2305.13657*, 2023. doi: 10.48550/arXiv.2305.13657

[18] X. He, M. Zhou, X. Xu, X. Ma, R. Ding, L. Du, Y. Gao, R. Jia, X. Chen, S. Han, et al. Text2Analysis: A benchmark of table question answering with advanced data analysis and unclear queries. *arXiv preprint arXiv:2312.13671*, 2023. doi: 10.1609/aaai.v38i16.29779

[19] K. Hu, M. A. Bakker, S. Li, T. Kraska, and C. Hidalgo. VizML: A machine learning approach to visualization recommendation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2019. doi: 10.1145/3290605.3300358

[20] Y. Huang, Y. Zhou, R. Chen, C. Pan, X. Shu, D. Weng, and Y. Wu. Interactive table synthesis with natural language. *IEEE Transactions on Visualization and Computer Graphics*, 2023. doi: 10.1109/TVCG.2023.3329120

[21] 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.

[22] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In *Information Visualization: Human-Centered Issues and Perspectives*, vol. 4950, pp. 154–175. Springer, Berlin, Heidelberg, Germany, 2008. doi: 10.1007/978-3-540-70956-5_7

[23] S. Khan, P. H. Nguyen, A. Abdul-Rahman, E. Freeman, C. Turkay, and M. Chen. Rapid development of a data visualization service in an emergency response. *IEEE Transactions on Services Computing*, 15(3):1251–1264, 2022.

[24] 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.

[25] H. Li, Y. Wang, and H. Qu. Where are we so far? understanding data storytelling tools from the perspective of human-ai collaboration. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, article no. 845, 19 pages. Association for Computing Machinery, New York, NY, USA, 2024. doi: 10.1145/3613904.3642726

[26] H. Li, Y. Wang, S. Zhang, Y. Song, and H. Qu. KG4Vis: A knowledge graph-based approach for visualization recommendation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):195–205, 2021. doi: 10.1109/TVCG.2021.3114863

[27] H. Lin, D. Moritz, and J. Heer. Dziban: Balancing agency & automation in visualization design via anchored recommendations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '20, pp. 1–12. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3313831.3376880

[28] Y. Lin, H. Li, A. Wu, Y. Wang, and H. Qu. DMiner: Dashboard design mining and recommendation. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–15, 2023. doi: 10.1109/TVCG.2023.3251344

[29] S.-C. Liu, S. Wang, T. Chang, W. Lin, C.-W. Hsiung, Y.-C. Hsieh, Y.-P. Cheng, S.-H. Luo, and J. Zhang. JarviX: A llm no code platform for tabular data analysis and optimization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 622–630, 2023. doi: 10.18653/v1/2023.emnlp-industry.59

[30] Y. Liu, T. Althoff, and J. Heer. Paths Explored, Paths Omitted, Paths Obscured: Decision points & selective reporting in end-to-end data analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '20, 14 pages, pp. 1–14. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3313831.3376533

[31] P. Ma, R. Ding, S. Wang, S. Han, and D. Zhang. Demonstration of insightpilot: An llm-empowered automated data exploration system. *arXiv preprint arXiv:2304.00477*, 2023. doi: 10.48550/arXiv.2304.00477

[32] P. Maddigan and T. Susnjak. Chat2vis: Generating data visualisations via natural language using chatgpt, codex and gpt-3 large language models. *IEEE Access*, 2023. doi: 10.1109/ACCESS.2023.3274199

[33] R. Mao, G. Chen, X. Zhang, F. Guerin, and E. Cambria. Gpteval: A survey on assessments of chatgpt and gpt-4. *arXiv preprint arXiv:2308.12488*, 2023. doi: 10.48550/arXiv.2308.12488

[34] A. M. McNutt, C. Wang, R. A. Deline, and S. M. Drucker. On the design of ai-powered code assistants for notebooks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1–16, 2023. doi: 10.1145/3544548.3580940

[35] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):438–448, 2018. doi: 10.1109/TVCG.2018.2865240

[36] OpenAI. GPT-4 technical report. *arxiv.2303.08774*, 2023. doi: 10.48550/arXiv.2303.08774

[37] C. Packer, V. Fang, S. G. Patil, K. Lin, S. Wooders, and J. E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. doi: 10.48550/arXiv.2310.08560

[38] A. Pandey, A. Srinivasan, and V. Setlur. Medley: Intent-based recommendations to support dashboard composition. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1135–1145, 2022. doi: 10.1109/TVCG.2022.3209421

[39] L. Peng, Y. Zhao, Y. Hou, Q. Wang, S. Shen, X. Lai, J. Gao, J. Dong, Z. Lin, and S. Chen. Mixed-initiative visual exploration of social media text and events. In *Proceedings of the IEEE Conference on Visualization and Visual Analytics*, 2021.

[40] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, R. Tian, R. Xie, J. Zhou, M. Gerstein, D. Li, Z. Liu, and M. Sun. ToolLLM: Facilitating large language models to master 16000+ real-world apis. *arxiv.2307.16789*, 2023. doi: 10.48550/arXiv.2307.16789

[41] Z. Qu and J. Hullman. Keeping multiple views consistent: Constraints, validations, and exceptions in visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):468–477, 2017. doi: 10.1109/tvcg.2017.2744198

[42] D. Raghunandan, Z. Cui, K. Krishnan, S. Tirfe, S. Shi, T. D. Shrestha, L. Battle, and N. Elmqvist. Lodestar: Supporting independent learning and rapid experimentation through data-driven analysis recommendations. *arXiv preprint arXiv:2204.07876*, 2022. doi: 10.48550/arXiv.2204.07876

[43] P. Ren, Y. Wang, and F. Zhao. Re-understanding of data storytelling tools from a narrative perspective. *Visual Intelligence*, 1(1):11, 2023.

[44] B. Saket, A. Endert, and Ç. Demiralp. Task-based effectiveness of basic visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25(7):2505–2512, 2018. doi: 10.1109/TVCG.2018.2829750

[45] 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, 10 pages, jan 2017. doi: 10.1109/TVCG.2016.2599030

[46] S. Sharan, F. Pittaluga, M. Chandraker, et al. LLM-Assist: Enhancing closed-loop planning with language-based reasoning. *arXiv preprint arXiv:2401.00125*, 2023. doi: 10.48550/arXiv.2401.00125

[47] L. Shen, E. Shen, Z. Tai, Y. Song, and J. Wang. TaskVis: Task-oriented visualization recommendation. In *EuroVis*, 2021. doi: 10.2312/evs.20211061

[48] D. Shi, W. Cui, D. Huang, H. Zhang, and N. Cao. Reverse-engineering information presentations: Recovering hierarchical grouping from layouts of visual elements. *Visual Intelligence*, 1(1):9, 2023.

[49] J. J. Thomas and K. A. Cook. *Illuminating the Path: An R&D Agenda for Visual Analytics*, pp. 69–104. IEEE Press, 2005.

[50] P. Vaithilingam, E. L. Glassman, J. P. Inala, and C. Wang. DynaVis: Dynamically synthesized ui widgets for visualization editing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, article no. 985, 17 pages. Association for Computing Machinery, New York, NY, USA, 2024. doi: 10.1145/3613904.3642639

[51] M. Van Someren, Y. F. Barnard, and J. Sandberg. The think aloud method: A practical approach to modelling cognitive. *London: AcademicPress*, 11(6), 1994. doi: 10.1016/0306-4573(95)90031-4

[52] L. Wang, S. Zhang, Y. Wang, E.-P. Lim, and Y. Wang. LLM4Vis: Explainable visualization recommendation using chatgpt. *arXiv preprint arXiv:2310.07652*, 2023. doi: 10.48550/arXiv.2310.07652

[53] Q. Wang, Z. Chen, Y. Wang, and H. Qu. A survey on ML4VIS: Applying machine learning advances to data visualization. *IEEE transactions on visualization and computer graphics*, 28(12):5134–5153, 2021.

[54] Y. Wang, Z. Sun, H. Zhang, W. Cui, K. Xu, X. Ma, and D. Zhang. DataShot: Automatic generation of fact sheets from tabular data. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):895–905, 2020. doi: 10.1109/TVCG.2019.2934398

[55] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, pp. 110–119. Association for Computing Machinery, New York, NY, USA, 2000. doi: 10.1145/345513.345271

[56] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. 2022. doi: 10.48550/arXiv.2401.01614

[57] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. 35:24824–24837, 2022.

[58] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of

visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):649–658, 2015. doi: 10.1109/TVCG.2015.2467191

[59] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Towards a general-purpose query language for visualization recommendation. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pp. 1–6, 2016. doi: 10.1145/2939502.2939506

[60] A. Wu, D. Deng, F. Cheng, Y. Wu, S. Liu, and H. Qu. In defence of visual analytics systems: Replies to critics. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1026–1036, 2023. doi: 10.1109/TVCG.2022.3209360

[61] A. Wu, Y. Wang, X. Shu, D. Moritz, W. Cui, H. Zhang, D. Zhang, and H. Qu. AI4VIS: Survey on artificial intelligence approaches for data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):5049–5070, 2021.

[62] A. Wu, Y. Wang, M. Zhou, X. He, H. Zhang, H. Qu, and D. Zhang. MultiVision: Designing analytical dashboards with deep learning based recommendation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):162–172, 2021. doi: 10.1109/TVCG.2021.3114826

[63] T. Wu, M. Terry, and C. J. Cai. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *Proceedings of the SIGCHI Conference on human factors in computing systems*, CHI '22, 22 pages, pp. 1–22. Association for Computing Machinery, New York, NY, USA, 2022. doi: 10.1145/3491102.3517582

[64] J. Yang, A. Prabhakar, K. Narasimhan, and S. Yao. InterCode: Standardizing and benchmarking interactive coding with execution feedback. 36:23826–23854, 2023.

[65] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. 36:11809–11822, 2023.

[66] 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, 2019. doi: 10.1109/tvcg.2019.2934668

[67] S. Zhang, Y. Wang, H. Li, and H. Qu. Adavis: Adaptive and explainable visualization recommendation for tabular data. *IEEE Transactions on Visualization and Computer Graphics*, 2023. doi: 10.1109/TVCG.2023.3316469

[68] 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. doi: 10.48550/arXiv.2306.07209

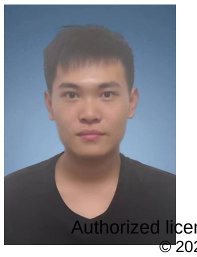
[69] J. Zhao, M. Fan, and M. Feng. ChartSeer: Interactive steering exploratory visual analysis with machine intelligence. *IEEE Transactions on Visualization and Computer Graphics*, 28(3):1500–1513, 2020. doi: 10.1109/TVCG.2020.3018724

[70] 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*, pp. 1–17, 2024. doi: 10.1109/TVCG.2024.3368060

[71] B. Zheng, B. Gou, J. Kil, H. Sun, and Y. Su. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024. doi: 10.48550/arXiv.2401.01614

[72] M. Zhou, Q. Li, X. He, Y. Li, Y. Liu, W. Ji, S. Han, Y. Chen, D. Jiang, and D. Zhang. Table2Charts: Recommending charts by learning shared table representations. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2389–2399, 2021. doi: 10.1145/3447548.3467279

Yuheng Zhao is currently a Ph.D student at the School of Data Science, Fudan University. Her primary research interests are visualization and visual analytics, with an emphasis on foundation model-powered intelligent visual analytics. For more information, please visit https://www.yuhengzhao.me/.



Junjie Wang is currently a master's student at the School of Data Science, Fudan University. His primary research interests are visualization and visual analytics, with a specific focus on intelligent and adaptive visual analytics.

Linbing Xiang received a B.S. in Software Engineering from Fudan University, she is currently pursuing her master's degree in Big Data Technology from the Hong Kong University of Science and Technology. Her primary research interests focus on applied artificial intelligence, specifically emphasizing data visualization and visual analytics within the realm of generative AI.

Xiaowen Zhang is currently a master's student at the School of Data Science, Fudan University. Her primary research interests lie in visualization and visual analytics, with a particular focus on visual analytics in the field of fintech.

Zifei Guo is currently an undergraduate at the School of Mathematical Sciences, Fudan University. Her primary research interests are data science, visual analytics, and machine learning application.

Cagatay Turkay is a Professor at the Centre for Interdisciplinary Methodologies at the University of Warwick, UK and a Turing Fellow at the Alan Turing Institute, London, UK. His research investigates the interactions between data, algorithms and people, and explores the role of interactive visualisation and other interaction mediums such as natural language at this intersection. He frequently publishes his research on visualisation journals such as IEEE TVCG, CGF, and IEEE CG&A, as well as journals in machine learning and data mining, and also recently co-authored a coursebook titled "Visual Analytics for Data Scientists'. He has been awarded the EuroVis Young Researcher 2019 award and named a EuroGraphics Junior Fellow in 2019.

Yu Zhang received a B.S. degree in Intelligence Science and Technology from Peking University in 2017. Since then, he has been pursuing a Ph.D. at the Department of Computer Science, University of Oxford. His research focuses on intelligent user interfaces in the field of human-computer interaction.

Siming Chen is an Associate Professor at School of Data Science, Fudan University. Prior to this, he was a Research Scientist at Fraunhofer Institute IAIS in Germany. He received his Ph.D. in computer science Peking University. His research interests are visualization and visual analytics, with the emphasis on Human-AI Collaboration, including LLM-driven visual analytics, social media and autonomous driving visual analytics. He has published 100 papers and more than 40 in top conferences and journals, including IEEE VIS, IEEE TVCG, EuroVis, ACM CHI, UIST, CSCW, etc. He served as multiple organizing chairs, committees and reviewers. He was awarded 10+ best paper/poster awards and honorable mentioned awards in multiple conferences For more information, please visit http://simingchen.me.