Interactive Table Synthesis with Natural Language

Yanwei Huang, Yunfan Zhou, Ran Chen, Changhao Pan, Xinhuan Shu, Di Weng, Yingcai Wu

***Abstract*—Tables are a ubiquitous data format for insight communication. However, transforming data into consumable tabular views remains a challenging and time-consuming task. To lower the barrier of such a task, research efforts have been devoted to developing interactive approaches for data transformation, but many approaches still presume that their users have considerable knowledge of various data transformation concepts and functions. In this study, we leverage natural language (NL) as the primary interaction modality to improve the accessibility of average users to performing complex data transformation and facilitate intuitive table generation and editing. Designing an NL-driven data transformation approach introduces two challenges: a) NL-driven synthesis of interpretable pipelines and b) incremental refinement of synthesized tables. To address these challenges, we present NL2Rigel, an interactive tool that assists users in synthesizing and improving tables from semi-structured text with NL instructions. Based on a large language model and prompting techniques, NL2Rigel can interpret the given NL instructions into a table synthesis pipeline corresponding to Rigel specifications, a declarative language for tabular data transformation. An intuitive interface is designed to visualize the synthesis pipeline and the generated tables, helping users understand the transformation process and refine the results efficiently with targeted NL instructions. The comprehensiveness of NL2Rigel is demonstrated with an example gallery, and we further confirmed NL2Rigel's usability with a comparative user study by showing that the task completion time with NL2Rigel is significantly shorter than that with the original version of Rigel with comparable completion rates.**

***Index Terms*—Data transformation, Natural language interface, Large language model, Human-AI collaboration.**

I. Introduction

Data tables are a universal interchangeable format for data management [1], analysis [2], [3], and presentation [4]. One common practice of data practitioners is converting raw data, including relational tables and less structured formats like HTML, JSON, CSV, and PDF documents, into tables of different structures, as depicted in Fig. 1. While this transformation is crucial for effective communication and consumption of the insights buried under different data formats, the process is frequently ad-hoc, which can be challenging and time-consuming for average users without fluent scripting skills.

Interactive approaches have been proposed to democratize such complex data transformation. These approaches typically offer an imperative interface [5]–[7], allowing users to select and execute data operations, such as cross tabulation and transpose, using toolbars or menus. Nonetheless, it remains crucial for users to understand the implications of these data operations, which can be challenging if they lack familiarity with data transformation [8]. By-example approaches [8]–[11] have recently emerged as a promising technique for lowering the barrier of data transformation. These approaches enable users to provide a partial example of the transformation target, from which the possible transformation procedures can be inferred. However, these approaches are limited in their ability of synthesizing complex data transformation pipelines based solely on the input and partial output, particularly when the pipelines involve custom calculations like weighted average.

Natural language (NL) interfaces have shown great potential in enhancing the accessibility to complex tasks [12]–[15]. Several attempts [3], [16], [17] have been made to facilitate tabular data transformation by synthesizing code or formulas based on NL instructions. However, most approaches lack transparency and provide no accompanying explanation of the synthesized results. Other approaches, such as Nlyze [3] and Gridbook [16], directly present the obscure script-like formulae. This can leave users confused about the transformation process and struggled with debugging incorrect results [16]. Additionally, these approaches often lack fine-tuning operations, requiring users to blindly revise NL instructions until they achieve the desired outcome.

The limitations observed in the existing studies motivate us to propose a novel interactive data transformation approach that utilizes NL as the primary interaction modality and enables users to synthesize tables in an interpretable and incremental manner. Developing such an approach poses the following two challenges:

NL-driven synthesis of interpretable pipelines. The first step toward NL-driven data transformation is to understand the diverse NL instructions provided by the users and construct the interpretable transformation pipelines that align with these instructions. Inspired by the recent progress on large language models (LLMs) [18], [19], we exploit the NL understanding and generation capabilities of OpenAI GPT-3.5 [20] to parse the NL instructions and generate specifications of Rigel [11], a recently proposed declarative grammar of tabular data transformation that describes the transformation pipelines in an apprehensible manner. Compared to traditional NL parsing methods using context-free grammar and neural semantic parsing [21], LLMs are able to understand a wider range of NL input due to the knowledge injected, while also saving the needs of training tailored models on various tasks [22], [23]. For instance, they can identify the concepts "*quantity*" and "*total sales*" that are directly not clarified in the underlying data, as shown in Fig. 1. However, as a general model, it is still challenging to synthesize the complex Rigel specifications correctly with the GPT-3.5 model.

Incremental refinement of synthesized tables. Experimental studies have demonstrated that LLMs can produce

Y. Huang, Y. Zhou, R. Chen, C. Pan, D. Weng, and Y. Wu are with Zhejiang University, Hangzhou, China. E-mail: {huangyw, yunfzhou, chenran928, panch, dweng, ycwu}@zju.edu.cn. D. Weng is the corresponding author.

X. Shu is with University of Edinburgh, Edinburgh, UK. E-mail: xinhuan.shu@gmail.com.

Manuscript received Jul. 19, 2023; revised Oct. 06, 2023.

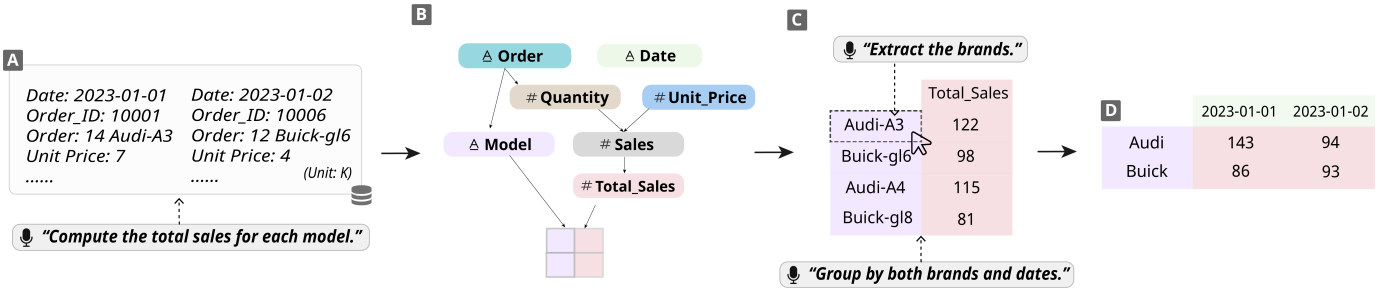


Fig. 1. NL2Rigel allows users to construct tables from semi-structured data through natural language instructions (A). It also assists users in understanding the conversion by providing visualizations of intermediate results (B) and enables interactive table refining through supplementary instructions (C, D).

inaccurate or unfaithful results in practice [23]–[25], including in scenarios related to data transformation [23]. As a consequence, the LLM-synthesized transformation pipelines may contain various errors. For example, certain parts of the instructions may be ignored, causing the desired cells to be left out of the generated tables. Moreover, the NL instructions provided to the LLMs may be misinterpreted, resulting in erroneous aggregations or calculations that can be challenging to identify. To alleviate such issues, it is essential to have an intuitive visual interface that explains how the tables are generated based on the complex transformation pipelines. Additionally, accessible interactions should be provided to allow users to quickly edit the pipelines and improve the transformation results.

In this paper, we propose NL2Rigel, an interactive system that facilitates table creation and refinement for data practitioners. NL2Rigel allows users to leverage NL instructions to construct desired tables from structured data (e.g., relational tables). It also supports a proportion of semi-structured data that has a regular format to be parsed into structured input. For the first challenge, we devise a multi-step prompt engineering approach to generate feasible Rigel specifications with GPT-3.5 based on users' instructions. For the second challenge, we develop an intuitive interface that provides NL explanations for different parts of the generated table, visualizes the complex Rigel specifications with legible data flow diagrams, and empowers users to incrementally refine the transformation results with targeted NL instructions. We show the comprehensiveness of NL2Rigel with a gallery of supported transformation cases that demonstrates its capability of completing various transformation tasks. A user study was also conducted to compare the affordance and accuracy of NL2Rigel with those of using Rigel directly. The findings show that transforming data with NL2Rigel was less demanding while maintaining comparable accuracy.

II. Related Work

Our work draws on and extends existing research on a series of foundational work in data transformation, interacting with data in NL, LLM-driven interfaces and the underlying declarative language for data transformation, Rigel [11].

A. Tools for Data Transformation

Numerous interfaces, software, and libraries have been developed for data transformation recently. However, given the versatility of transformations, the scopes they are mainly designed for are diverse, including **E**xtracting (i.e., parsing or extracting from raw data into more structured forms), **R**eshaping (i.e., modifying the data schema or records), **C**leaning (i.e., tidying or formatting dirty data), and **G**eneral. In the following, we discuss these tools along with their scopes.

In the early stage, data transformation is commonly performed using integrated functions from more general data processing tools. For instance, users can manually edit the target table using spreadsheet tools like Microsoft Excel (G), which obviously becomes time-consuming when confronted with complex tasks or a huge amount of data. Users with a professional background may perform data queries or management in database systems using structural languages (e.g., E: TSIMMIS [26]; R & C: SQL and its extensions [27], [28]), or writing custom scripts with data processing libraries [29]–[32] (G) to author sequences of operations. The well-designed functions and parameters provided by these tools are expressive enough to cover various scopes and classes of transformations. However, the learning curve of programming and mastering these functions also becomes an obstacle to data practitioners without a coding background [8], [33].

Consequently, a great many tailored interactive systems have been developed to promote efficient data transformation without programming. One of the most common practices is to put a compact list of functions and parameters in a menu that users can select and apply to specify desired transformations (E: [34]; R: [5]–[7], [35]; C: [36]). Despite the effectiveness, the imperative operations are often hard to be interpreted by average users, and can be demanding in complex data extracting and reshaping tasks due to the lengthy operation sequences [8]. To address this, modern systems incorporate interactions that encourage users to focus on "what the target table looks like" rather than "how to transform the data", hiding obscure details of the transformation process. For example, a number of systems allow users to provide an example pair of I/O tables and automatically synthesize transformation scripts (E: [37]–[39]; R: [8], [40]; C: [9], [10], [41]–[45]). However, examples are essentially hard to convey complex intents in table reshaping, particularly when the operations involve pivoting, nested aggregations or computations like the weighted average, which can hardly be inferred solely from input/output values. In particular, some systems incorporate declarative models that can directly generate tables based on

A1

Year	Name	Income
2020	Alice	37.5
2020	Bob	31.8
2021	Alice	45.5
2021	Bob	34.6

A2

Year	Name	Expense
2020	Alice	26.0
2020	Bob	11.7
2021	Alice	29.2
2021	Bob	12.3

B

	Alice	Bob
2020	37.5	31.8
2021	45.5	34.6

C

		Income	Expense
2020	Alice	37.5	26.0
2020	Bob	31.8	11.7
2021	Alice	45.5	29.2
2021	Bob	34.6	12.3

Fig. 2. Examples for applying the specifications of Rigel to given relational tables. (A) The input relational tables revealing the annual income (A1) and expense (A2) for each person. (B) The result table for the specification $(Year), (Name) \rightarrow (Income)$. (C) The result table for the specification $(Year \times Name), () \rightarrow (Income + Expense)$.

the user-constructed grammatical specification (R: [11], [46], [47]). Yet, they only save efforts for specifying the overall table structure, while detailed mutations on entities such as arithmetic operations still require manual configurations.

In recent years, users are further allowed to describe the intentional task with NL in several table reshaping tools, which is more intuitive and effortless to specify complex intents for average users. Nlyze [3] (R) presents an algorithm to parse NL into a list of candidate programs based on its designed DSL. GridBook [16] (R) further allows users to issue NL instructions in the spreadsheet grids that can be transformed to Excel formulas to generate tables. NL2Rigel contrasts these systems in two factors. First, it supports end-to-end table synthesis with a wider scope of table transformation classes such as creating cross-tabulations. Second, existing systems lack sufficient information to help users understand the results and flexible interactions to refine tables when necessary [16]. Although several NL interfaces in related areas (e.g., table query) provide explanation and debugging operations on the parsing results [48], [49], they depend on the specific technical representation of user requests (e.g., SQL and lambda DCS [50]) and cannot be directly applied to table synthesis. In contrast, NL2Rigel visualizes the intermediate results and supports incremental table refinement.

B. Interacting with Data in Natural Language

Since NL provides an intuitive and accessible way for users to interact with data and facilitate making complex or customized requests, NL interactions have been widely adopted in fields of data science including database [49], [51], [52] and information visualization [53]–[58]. Generally, to perform a NL-based interaction, there are two underlying steps: instruction parsing and execution. We classify existing research based on these two steps and discuss them below:

Instruction parsing. During the parsing process, many systems adopt parsers based on context-free grammar or the part-of-the-speech algorithm [59] to divide the NL instruction into semantic segments [14], [15], [53], [54], [56]. The rapid development of machine learning also promotes parsers based on neural networks for better accuracy [58], [60]. However, training models can be costly and require tailored datasets, which is often hard to seek in many domains including data transformation. In recent years, language models pre-trained with millions of parameters, or large language models (LLMs), have shown outstanding abilities in complex reasoning and code generation (e.g., PaLM [19], the GPT series [18]). Therefore, some researcher have been studying the potential of LLMs for textual wrangling problems [22], [23].

Execution. In the second step, the parsed NL instructions will be translated into usage-specific formats to carry out the desired tasks. Owing to the advances in natural language programming, most systems synthesize scripts of programming languages or domain-specific languages (DSLs, e.g., SQL, Vega-Lite [61]) based on the parsed results [53], [54], [58]. There are also a few systems transforming the parsed results to compact, tool-specific operations [60].

Accordingly, NL2Rigel adopts the GPT-3.5 model to parse NL instructions, enabling the comprehension of utterances with non-existent concepts in the underlying data or high-level intents. During the execution, the model also facilitates generating different formats for different sub-tasks within a single-model (e.g., JavaScript code for data extraction, and Rigel specifications for table construction), eliminating the need of training various models.

C. Leveraging LLMs in User Interfaces

The powerful capability of the state-of-the-art LLMs has attracted much attention in the HCI domain to apply them to interfaces for various tasks [62]–[66]. For instance, Jiang et al. [67] introduce an interface synthesizing code from NL instructions. However, applying this general model to domain-specific tasks often require extra efforts. One line of research directly encodes context information and provides task-specific examples in the *prompt* (i.e., the input to the model) [66]. Others fine-tune the model using customized task-related datasets to provide necessary information [62]. To improve the performance, some researchers further use tuning methods to optimize the prompts [64]. However, in cases where multi-step reasoning is demanded, the performance of LLMs can decrease [63]. Therefore, Wu et al. [63] introduces the approach of *chaining*, where a complex task is divided into multiple steps, each step handled by a separate LLM layer, and subsequent layers can take the output of previous ones as input. NL2Rigel draws inspiration from this work and introduces a tailored workflow with *chained* LLM layers for end-to-end table construction.

D. Rigel: A Declarative Language for Data Transformation

Taking one or more relational tables and a specification as input, Rigel [11] can directly derive the transformed target

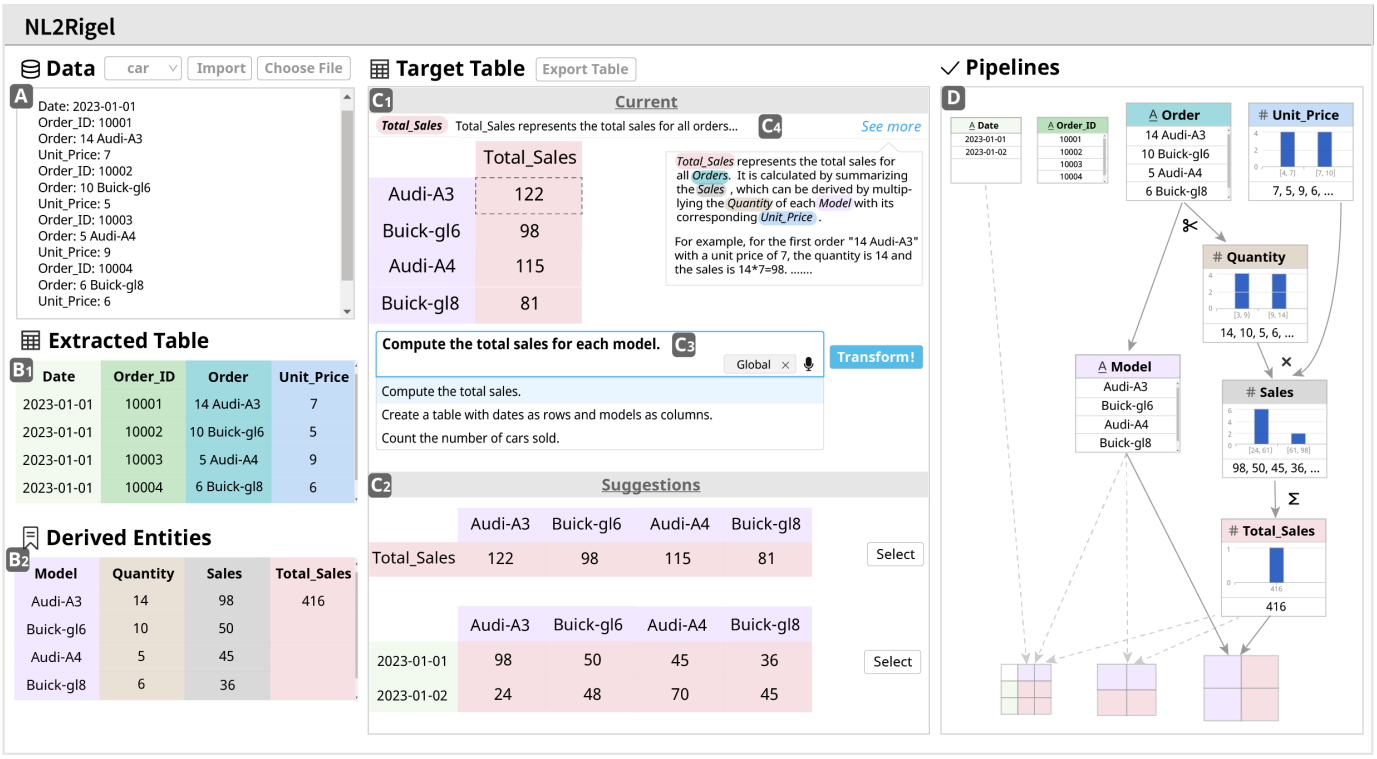


Fig. 3. The user interface of NL2Rigel. (A) The data view. (B) The extracted-table view consisting of the relational table extracted from the raw data (B1) and a table showing all entities that are derived during the conversion (B2). (C) The target-table view incorporates a generated table that is currently on focus (C1) and several tables as suggestions (C2). The user may enter NL instructions in the chatbox (C3) to generate or improve tables. A description of the corresponding entity is also offered when a cell is clicked (C4). (D) The pipeline view with a dataflow diagram visualizing the transformation process.

table corresponding to the specification. The main idea of Rigel is the declarative mapping approach, where a table is divided into 3 channels (row header, column header and cell) and entities can be mapped to these channels to construct tables. Specifically, the basic form of a valid Rigel specification is $(rowHeader), (columnHeader) \rightarrow (cell)$, where $rowHeader$, $columnHeader$ and $cell$ are entities. Initially, each column of the raw table is an entity. To support diverse transformation tasks, data functions can be applied to these entities to derive new ones. Besides, multiple entities can be mapped to a single channel, where the operator $+$ or $\times$ will be used to connect the entities respectively when the table channel is row/column header or cell.

Fig. 2 illustrates examples of applying the specifications of Rigel to given relational tables. For the two input relational tables of the annual income (Fig. 2(A1)) and expense (Fig. 2(A2)) by person, the specification $(Year), (Name) \rightarrow (Income)$ will map *years* to the row header channel, *names* to the column header and *income* to the cell respectively, which will consequently lead to the target table in Fig. 2(B). Similarly, for the specification $(Year \times Name), () \rightarrow (Income + Expense)$, the Cartesian product of *Year* and *Name* will be mapped to the row header channel, while *Income* and *Expense* are appended and positioned in cells of separated columns, as shown in Fig. 2(C).

III. Usage Scenario

Alice, a data analyst for a car enterprise, had been tasked with writing a sales report. She had been provided with the necessary order information, which included the date, order ID, order contents, and unit price for each product ordered. This information was currently stored in a plain-text document (Fig. 1(A)). To present the order information in a clear and informative manner in her report, Alice must transform the document into a legible table that would provide insights into the car sales data. Due to her lack of experience in writing tailored scripts, Alice attempted to use existing interactive transformation systems but found it difficult to process the plain-text document with these systems. As the result, she had decided to turn to NL2Rigel for assistance.

To begin the data transformation process, Alice imported raw data into the system (Fig. 3(A)). Her lack of expertise in data transformation left her unsure about which instructions to provide. Luckily, NL2Rigel offered suggestions in a drop-down list when she clicks on the input box (Fig. 3(C3)). After briefly reviewing the options, she found "*Compute the total sales*" to be a potentially useful instruction. With this in mind, she clicked on the microphone icon in the input box and spoke her instruction aloud: "*Compute the total sales for each model*". Based on this instruction, NL2Rigel automatically synthesized transformation pipelines based on Alice's intents and subsequently presented the results on the user interface. On the left side of the interface, NL2Rigel displayed the extracted (Fig. 3(B1)) and derived (Fig. 3(B2)) entities from the input data and presented them in corresponding relational tables. In the center of the interface, NL2Rigel presented the most probable target table (Fig. 3(C1)) along with two

alternative ones with different layouts (Fig. 3(C2)). On the right side of the interface, NL2Rigel illustrated the data flow diagram of the entire pipeline, which not only demonstrated the transformation path of the current target table but also the alternative ones in dashed gray lines.

Noticing that the target table included many numbers that were not present in the original data, Alice wondered whether these numbers were accurately calculated. After selecting the cell containing the value of *122*, NL2Rigel identified the *Total_Sales* entity that this cell was associated with and provided a succinct definition of this entity at the top of the target table (Fig. 3(C4)). Furthermore, a comprehensive textual explanation on how the entity was derived was presented when Alice hovered over the "*see more*" link to the right of the definition.

During her analysis of the target table, Alice noticed that many cars shared the same brand. To inspect sales by brand, she decided to alter the table layout. Instead of repeating the entire instruction, Alice simply clicked on "*Audi-A3*", one of the purple cells corresponding to the *Model* entity, and instructed NL2Rigel to "*extract the brands*" (Fig. 1(C)). In a few seconds, NL2Rigel updated the table to display the sales figures for each brand, including *Audi* and *Buick*. Impressed by the system's performance, Alice decided to create another table that expanded the total sales by date. Once again, she instructed NL2Rigel to "*group by both brands and dates*". NL2Rigel quickly presented Alice with the result (Fig. 1(D)). Overall, Alice found synthesizing and refining tables with NL2Rigel to be a smooth and effortless experience, and she successfully completed her report with the generated tables.

IV. NL2RIGEL

This section presents the design rationales, the detailed implementation and the interface of the NL2Rigel system.

A. Design Rationales and System Overview

To enable users to effectively synthesize and refine tables using NL instructions, we conducted a thorough literature survey on the design principles of NL interfaces [12], [14], [68], [69] and data engineering [49], [70]–[72]. We also collaborated over a period of three months with three experienced data practitioners, P1, P2, and P3. P1 was a senior visualization researcher with decades of experience in designing data-driven visual analytics systems. P2 worked as a researcher for a large corporation specialized in tabular data understanding and wrangling. We also invited P3, a Ph.D. student, who had over eight years of experience in writing data analysis scripts. During bi-weekly brainstorming sessions, we gathered their feedback on the preliminary design rationales and system prototypes. We also observed despite their expertise, they agreed with the intuitiveness and efficiency brought by NL instructions. They also appreciated the prototype in recommending diverse tabular views and solving ad-hoc and multi-step transformation tasks. Through this iterative design process, we arrived at the following three key design rationales.

Provide NL instruction recommendations. By integrating NL as the primary interaction modality, users can easily describe their desired data transformation. However, the cold start issue [14], [57], [60], [69], [73] can cause users to hesitate about what levels of abstraction the system can understand, making it difficult for them to express their goals in the first few attempts. To address this issue, the system should provide recommendations to assist users in giving NL instructions. Furthermore, NL instructions can vary greatly in granularity, ranging from very coarse (e.g., "*show me the sales report*") to very detailed (e.g., "*put product name in the first column and year in the first row, and then fill cells by multiplying unit price by quantity per product and year*"). To better adapt to users' preferences, the proposed system should offer recommendations at different levels of granularity.

Explain data transformation pipelines. To help ordinary users comprehend the underlying data transformation processes, the Rigel specifications generated by GPT-3.5 should be presented in an intuitive manner, allowing users to easily understand how NL2Rigel synthesizes tables. Previous NL-based systems [14], [49], [68] have demonstrated the effectiveness of using multi-modal explanations to improve comprehension of the results generated from NL instructions. Based on the identical rationale, we propose interpreting the pipeline from three perspectives: a) data-oriented interpretation, which presents the entities extracted and derived from the raw data in the form of relational tables; b) process-oriented interpretation, which illustrates the transformation process as a data flow diagram, showing the relationships among the entities; and c) result-oriented interpretation, which enables users to select entities directly in the table to obtain explanations.

Facilitate refinement with targeted NL instructions. Existing literature has demonstrated the importance of supporting the debugging of lengthy or nested queries to alleviate users from the tedious manual work of verifying each segment of the instruction [72]. As NL2Rigel is an end-to-end system that accepts complex NL instructions as input, it is crucial to devise an efficient strategy to enable users to refine the target tables without having to manually reorganize the instructions. To this end, NL2Rigel should permit users to provide additional NL instructions to perform mutations directly on the generated tables, which can assist users in providing more targeted utterances and reduce their trial-and-error attempts and efforts. Furthermore, NL2Rigel should offer recommendations and alternative transformation pipelines to aid users in exploring the space of potentially improved table layouts.

Based on these rationales, we designed NL2Rigel. NL2Rigel is mainly targeted at transforming structure data into new tabular views by utilizing NL instructions from users as input. To accommodate cases when initial structured data is unavailable, it also supports textual data that is semi-structured (e.g., HTML, JSON, CSV, or key-value pairs) as input. To support its features, NL2Rigel incorporates an interface to collect the user input and display the results, and an underlying computation module that leverages the language model to predict the desired pipelines.

As illustrated in Fig. 3, the interface of NL2Rigel consists of a data view displaying the raw data (Fig. 3(A)), an extracted-table view containing the extracted relational table and a derived-entities panel of entities generated throughout

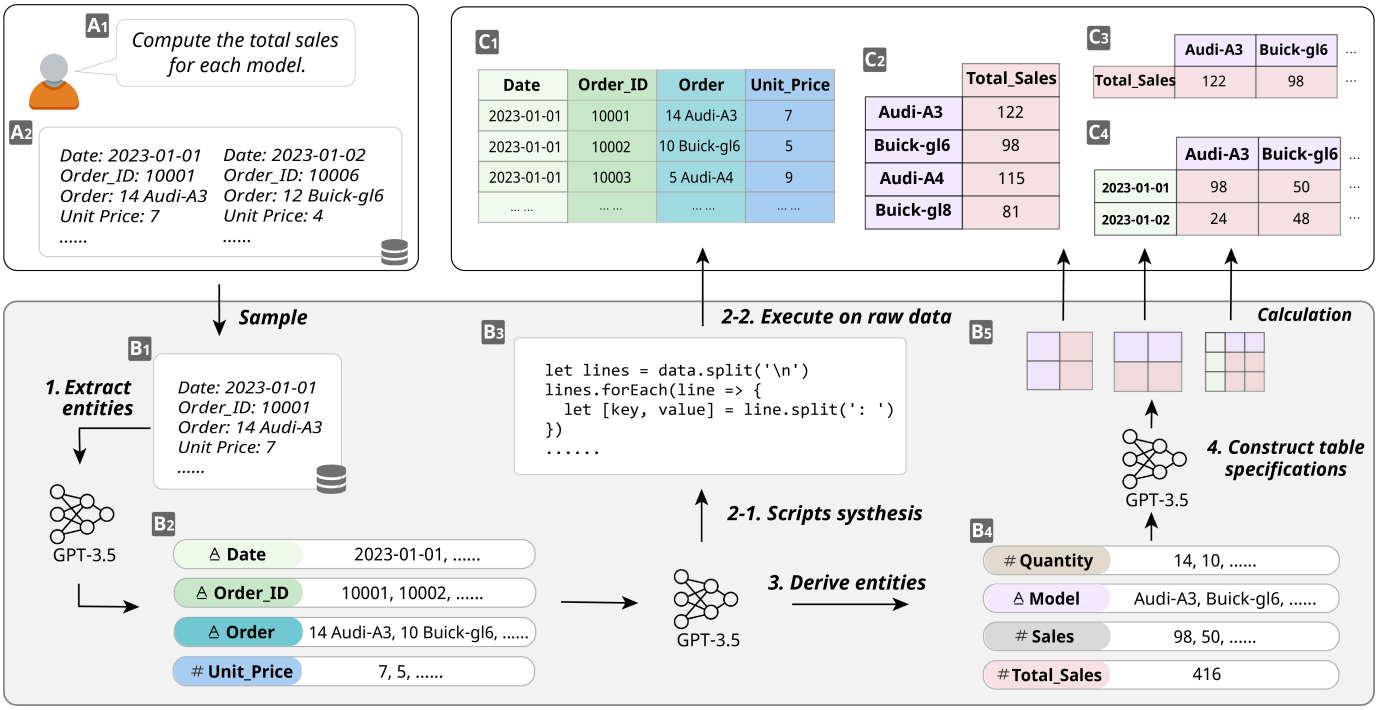


Fig. 4. The overall workflow of NL2Rigel. NL2Rigel takes the NL instruction from users (A1) and the raw data (A2) as input. The transformation process includes four steps: first, the raw data is sampled (B1) and used to extract relevant entities (B2) using the GPT-3.5 model. Then, the model uses the extracted entities to synthesize scripts (B3) and generate a relational table from the raw data (C1). Meanwhile, it also derives some new entities (B4), which are later leveraged to construct table specifications (B5). Target tables can be calculated from the specifications, with the most likely one taken as the current table (C2) and others as suggestions (C3, C4).

the conversion (Fig. 3(B1, B2)), a target-table view showing the transformation results and suggestions (Fig. 3(C1, C2)), and a pipeline view visualizing the transformation pipelines (Fig. 3(D)). To perform transformations, there are three general steps in the user workflow: first, the user provides an instruction to construct initial pipelines; second, the user may inspect the generated pipelines through interactions; third, supplementary instructions can be given to refine the target table. The last two procedures are iterated until the user acquires a desired result.

B. NL-Driven Synthesis of Data Transformation Pipelines

Having imported the textual data as input, the user can clarify their transformation intents through NL instructions in the chatbox below the target table (Fig. 3(C3)). NL2Rigel supports multiple input modalities, allowing users to either speak the instruction by clicking on the microphone icon or type it in plain text. To avoid the cold start problem for beginners, NL2Rigel leverages GPT-3.5 to proactively offer some potential instructions according to the raw data, as soon as the user clicks on the chatbox before giving an instruction. After choosing an instruction or revising it to their preference, the user may click on the *transform* button to start the transformation.

To construct initial pipelines based on the raw data and user instruction, we design a computational workflow empowered by GPT-3.5, a state-of-the-art LLM capable of solving numerous general tasks. As directly solving the problem can be too difficult for the model, we divide the problem into several manageable sub-tasks inspired by recent applications [63], [66]. Each sub-task will be handled by a separate GPT-3.5 layer, and the output of previous tasks is parsed and included in the input of subsequent tasks. An overview of the workflow is illustrated in Fig. 4. NL2Rigel takes the NL instruction from users (A1) and the raw data (A2) as input. Due to the character limit posed by GPT-3.5, NL2Rigel samples the raw data (Fig. 4(B1)) by truncating the data after the last newline within the first 1500 characters, aiming to fit as many complete records as possible in the prompts. After the sampling, it leverages a four-step workflow to synthesize Rigel specifications:

Extract entities. Initially, NL2Rigel identifies and extracts the entities from the sampled data that may be useful for the transformation with the assistance of GPT-3.5 (Step 1). For instance, taking the sampled data in Fig. 4(B1), it extracts four entities *Date*, *Order_ID*, *Order* and *Unit_Price* (Fig. 4(B2)). Relevant information such as example values, the source lines where the entity is extracted and whether the entity is quantitative or categorical is also recorded.

Generate the relational table. In this step, NL2Rigel transforms the raw data into a relational table. Since the prompt length of GPT-3.5 is constrained, a JavaScript program (Fig. 4(B3)) is synthesized based on the sampled data and the extracted entities (Step 2-1). It then generates the relational table (Fig. 4(C1)) by executing the program on the raw data (Step 2-2). Specially, in case the generated code has errors that prevent it from successfully running on the given data, the model will regenerate the code for up to 5 attempts before the

system sends an error message to the user. Meanwhile, users are also allowed to manually edit the generated relational table in the extracted-table view (Fig. 3(B1)) if data issues exist.

Derive entities. Having got the extracted entities, NL2Rigel further leverages GPT-3.5 to derive some new entities (Fig. 4(B4)) by combining extracted ones and the data functions of Rigel (Step 3). For example, *Quantity* and *Product* can be derived by splitting the values of the *Order* entity by the blank space. A unique name and a Rigel specification are generated and stored for each derived entity. To ensure the validness of the derived entities, the generated specifications will be tested by regular expressions, with up to 5 reruns of the derivation process when encountering errors.

Construct table specifications. After acquiring all the components of a Rigel specification, NL2Rigel then calls GPT-3.5 to map the extracted and derived entities to different table channels (Step 4) to construct candidate specifications for the transformation task (Fig. 4(B5)). The model is also employed to rank the results in descending order by the similarity to user intents. After undergoing syntactic evaluation similar to previous steps, these specifications can be eventually transferred to the calculation module of Rigel to produce the target tables, where the most likely specification will be used as the default current table in the interface (Fig. 4(C2)) while others will be taken as suggestions (Fig. 4(C3, C4)).

While applying the GPT-3.5 model to each step, we carefully engineer a prompt by combining the corresponding input and our compilation of context knowledge. Afterwards, the prompt is fed to the LLM and the generated contents will become the output for this step after some processing. By learning from the rationales of existing research on the prompting methods of LLMs [74]–[76], we design a prompt template and apply it throughout our workflow. Fig. 5 illustrates an example prompt for the *derive entities* step. Specifically, the prompt template comprises of the following six components:

- *Preamble* provides a brief introduction to Rigel by narrating the formalized definitions and rules that are mentioned in Section II-D.
- *Functions* offer information of the data functions in Rigel. Since we find understanding the usage and picking the functions require considerable reasoning and can be tricky for the LLMs, we provide a detailed list of the name, the valid format, a description of its usage and some examples for each function.
- *Task* describes the primary task the LLM is expected to perform. The user instruction is included in the statement if necessary.
- *Task input* embeds other fields (mostly with larger lengths) used for the performed task except the instructions. For example, the sampled data and extracted entities (with the names and some values written in the script form) are concatenated and positioned at the end of the prompt for the *derive entities* sub-task.
- *Task-specific examples* provide a few cases according to the task to be performed. For instance, we provide an example scenario and list the potential derived entities for the *derive entities* sub-task. Generally, we only include at most two examples here due to the prompt size limit and actual performance.
- *Type control* gives restrictions on the output type.

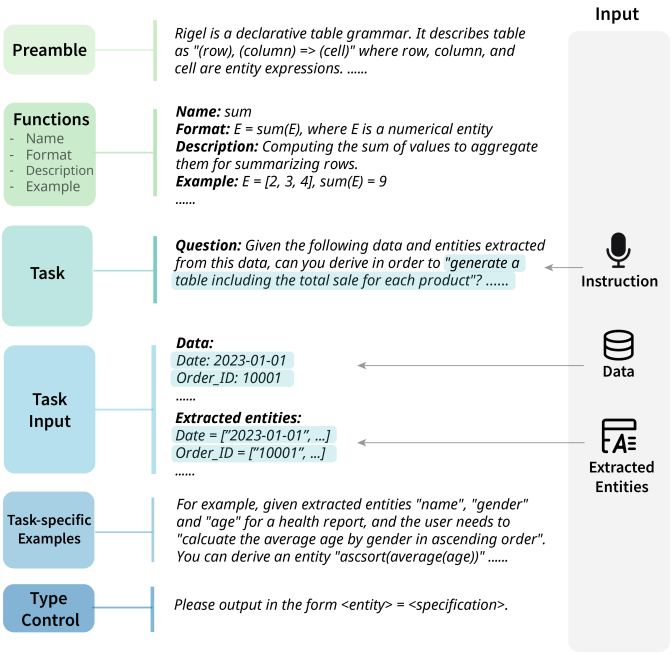


Fig. 5. An example prompt of NL2Rigel that includes 6 components (*preamble*, *functions*, *task*, *task input*, *task-specific examples* and *type control*). The input for this process (e.g., the user instruction, sampled data and extracted entities) is also embedded into the prompt.

Note that some components of the template are optional as they may bring unnecessary information for some sub-tasks. Table I shows an overview of the components that are included in the prompt for each sub-task. During the implementation, we also fine-tune the order of different components in some sub-tasks for better performance. We refer readers to the appendix for prompt examples. Meanwhile, our presented template is merely one feasible design and we encourage future researchers to adjust our solution or develop novel ones.

C. Inspection and Refinement of Transformation Results

When the conversion process is finished, the result tables recommended by the LLM (the *construct table specifications* sub-task in Section IV-B) will be shown in the target-table view (Fig. 3(C1, C2)), where the one identified by the model to satisfy the user intents most will be chosen as the current table. Users may click on the "select" buttons beside the suggested tables to switch to other candidates. Additionally, to help users understand the conversion process and inspect the generated table, NL2Rigel first provides details information about the transformation pipeline, which are displayed in three views:

a) As the intermediate output of the transformation process, the extracted-table view incorporates the relational table reorganized from the raw data (Fig. 3(B1)), where each column represents an *extracted entity* encoded with different colors. By exerting data functions on these entities, new entities are derived and placed in different columns of the table in the derived-entities panel (Fig. 3(B2)). Both the extracted and derived entities are later used to construct the target table.

b) To visualize the table construction process, the pipeline view includes a data flow diagram to facilitate user understanding of the pipelines (Fig. 3(D)). Specifically, this diagram visualizes the derivation relationship between entities and the mapping from entities to table channels. Each card represents an entity and includes a corresponding value list. For quantitative entities, an optional histogram is also provided. For each entity derivation, the cards of the source and target entities are connected by edges. To illustrate the mappings between entities and the target tables, we leverage a colored glyph to represent each target table, the color of whose grids indicate the source entities of the cells. We also connect each glyph with the cards of all entities used by the table. Besides, to highlight the current pipeline and reduce the space cost, the diagram follows the focus-and-context strategy [77] and scales the cards of all used entities and the table glyph for the current table. While the rest parts are diminished, users are still able to click on the table glyphs to switch to other pipelines.

To build the data flow diagram, we first identify the derivation relationships between entities by syntactically parsing the specifications of entities. Similarly, the component entities for each recommended table are also acquired by parsing the table specifications. A graph can be consequently built based on these relationships and rendered on the interface. We also utilize the Dagre [78] library to optimize the graph layout.

c) While the former views can provide a general overview of the transformation pipeline, there also exist scenarios where users are interested in the provenance of a specific cell or row/column. Therefore, NL2Rigel also provide an in-situ explanation to avoid the frequent focus shift between views. As soon as the user selects a cell in the current table, the corresponding entity and a short NL description of it are shown on the top of the current table (Fig. 3(C4)). Users can fetch the detailed version of the description by clicking on the *see more* link. The explanation is generated by prompting GPT-3.5 in a similar way to the synthesis of pipelines (Table I).

Through examining the synthesized pipeline, there can be cases where the tables generated by NL2Rigel fail to meet the users' demands. In this case, besides rephrasing their instructions, users are further allowed to provide targeted instructions in the chatbox as a supplement to previous utterances and specify the desired scope for this instruction. For example, if a user is to fine-tune the values (e.g., filtering, sorting, averaging, ...), he can first click on one of the cells to be altered. As such, the tag in the chatbox (Fig. 3(C3)) will contain the value of the selected cell, indicating the instruction will be performed exclusively on the entity that the cells belong to. After giving a local instruction such as "*remove the values below zero*", NL2Rigel will automatically compute the fine-tuned entity and update the current table by replacing the original entity with the new one. On the other hand, when the user prefers to alter the table schema, he can simply offer an instruction such as "*group by the dates*" without other interactions, leaving the tag in the chatbox as the default value *global*. In this case, NL2Rigel will recalculate appropriate table specifications to update the current table based on the instruction. Both refinement is powered by GPT-3.5 with its prompt components shown in Table I.

TABLE I
THE ADOPTED COMPONENTS FOR THE PROMPT OF EACH SUB-TASK IN OUR WORKFLOW. (P: *preamble*, F: *functions*, T: *task*, TI: *task input*, TE: *task-specific examples*, TC: *type control*.)

Sub-task	P	F	T	TI	TE	TC
Extract entities			✓	✓		✓
Generate the relational table			✓	✓	✓	
Derive entities		✓	✓	✓	✓	✓
Construct table specifications	✓		✓	✓	✓	
Input Suggestion			✓	✓		
Explain entities		✓	✓	✓		
Refine entities		✓	✓	✓	✓	✓
Refine specifications	✓		✓	✓	✓	✓

TABLE II
THE PROPOSED DESIGN SPACE FOR THE BASIC CORPUS.

Table structure	Channel	row-oriented	(A), () → (B)
		column-oriented	(), (A) → (B)
		cross-tabulation	(A), (B) → (C)
	Dimension	Unidimensional	(A), () → (B)
		Multidimensional	(A×B), () → (C)
Entity feature	Function	Nto1	concat
		1toN	split
		1to1	sum
	Nesting	Unnested	sum(A)
		Nested	sum(mul(A,B))

V. TASK GALLERY

To assess the expressiveness of NL2Rigel, we present a gallery of tasks to evaluate the coverage of our method in different contexts. We organize the gallery based on the complexity of the tasks.

Basic corpus. The tasks in the basic corpus are meant to show the comprehensiveness of tasks that NL2Rigel can manage to resolve. Unfortunately, the existing classification on tabular data transformation [79] is too general or high-level and can be inappropriate for evaluating the real-world performance of NL2Rigel, which tends to be easily influenced by task-specific factors such as function types and entity numbers. Therefore, we base the design of our task gallery on a design space that we summarize from the detailed grammatical features of Rigel [11].

An overview of the design space is provided in Table II. Since a Rigel specification is composed of entities mapped to different table channels, we divide tabular data transformation into *table structure* and *entity features*. Table structure indicates the fundamental shape of the table. Since it is determined whether the row/column header is empty and the maximum number of entities in the channels, we further classify it into the *channel* and *dimension*, respectively. For instance, a *row-oriented table* refers to a table without the column header and the data are grouped by the header of each row (e.g., Fig. 4(I)). Similarly, a *column-oriented table* has no row header (e.g., Fig. 4(J)) while a *cross-tabulation* has both the row header and column header(e.g., Fig. 4(K)). On the other hand, *entity features* focuses on different types of entities mapped to the the table channels. Specifically, we consider the types of functions by borrowing the taxonomy of [11] and their nesting in the composition of entities.

We refer readers to the appendix for more examples of the design space and the detailed tasks in the basic corpus.

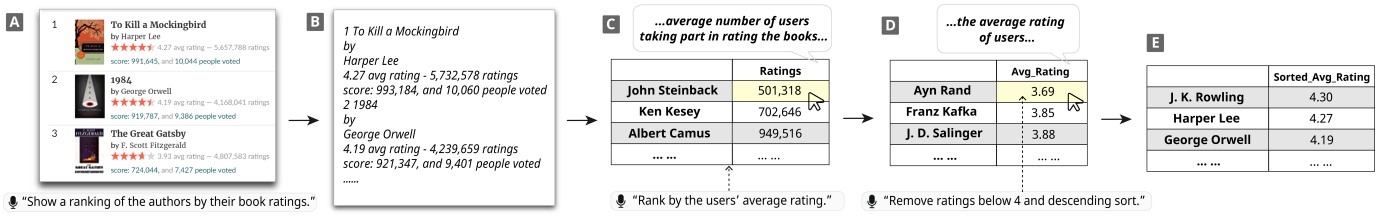


Fig. 6. A scenario of constructing tables from a web page. The user provides an initial NL instruction to convert the copied raw text (B) of the webpage (A) and perform two rounds of refinement (C, D) on the cells and the table structure to get the desired table (E).

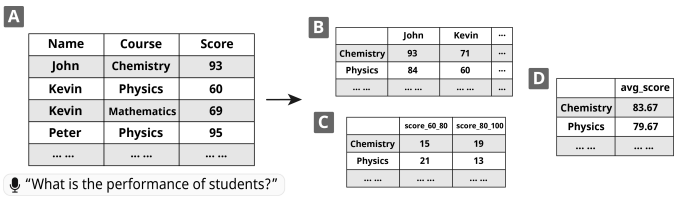


Fig. 7. A scenario of instructing NL2Rigel with a high-level utterance. (A) The input table and the user instruction. (B-D) The synthesized tables.

To summarize, NL2Rigel is able to support all of the 36 combinations of the conditions mentioned in the design space above, proving the expressiveness of our approach.

Advanced corpus. We further design an advanced corpus to demonstrate the generalizability of NL2Rigel by using cases of real-world scenarios. A full version of the cases can be found in the appendix. As one typical example, we illustrate how the journalist Mary constructs a table for her reports while browsing a book review website.

As shown in Fig. 6(A), the website displays a list of the best rated books from the 20th century. Mary wants to know which author has the best ratings, so she imports the copied texts Fig. 6(B) of the website to NL2Rigel while saying "*show a ranking of the authors by their book ratings*". While the system generates the target table for her in seconds, she finds that the ratings of the books are abnormally large (Fig. 6(C)). Wondering the source of this error, she clicks on an abnormal cell and subsequently gets informed that the system mistakes the ratings for the average number of participating users. She then clarifies her intents to the system with the instruction "*rank by users' average rating*", which results in the table in Fig. 6(D). When she checks for the explanation again, she learns that the mutated column now refers to the average user rating, which convinces her that her intents are correctly interpreted. Since Mary is only interested in the most popular authors, she further fine-tunes the table by keeping records with the rating above *4* and sorting the ratings, acquiring her desired table eventually (Fig. 6(E)).

Open-ended transformation. NL2Rigel is also capable of helping users explore potential tables and handling high-level user intents, as demonstrated in the following case. Suppose a teacher named Peter has exported a table (in the format of .CSV) of the students' scores on several courses from an online marking platform (Fig. 7(A)) and wants to perform an analysis on the academic performance of students. He imports the file into NL2Rigel with the instruction "*What is the performance of students?*" Fig. 7(B-D) shows the synthesized tables, which contains a cross-tabulation of scores over students and courses, the score distribution over all courses and their average score. Through NL2Rigel, Peter manages to generate a diverse list of multiple tables in parallel within seconds.

VI. User Study

A. Overview

To demonstrate the effectiveness and the usability of NL2Rigel, we conducted a user study comparing our system with the interface of Rigel [11].

Participants. We recruited 12 participants (6 males and 6 females, aged 25.5 years on average). The participants included 7 undergraduates and 5 Ph.D. students from various academic disciplines, including 6 from Computer Science, 3 from Art, 1 from Medicine, 1 from Biology, and 1 from Mathematics. Although some of the participants had experience of using data processing tools (e.g., Tableau, Excel, MATLAB), all of them reported an average familiarity with data transformation (M=2.92/5, SD=1.24 on a 5-point Likert scale) and had never used Rigel or NL2Rigel before. Besides, all participants were non-native English speakers, and they reported a moderate proficiency in English communication (M=3.42/5, SD=0.67).

Systems. The Rigel interface was chosen to be compared with NL2Rigel for the following reasons. First, both NL2Rigel and Rigel adopt the declarative grammar in the underlying models and are able to support a wide range of table transformations, including value-based changes (e.g., averaging, filtering, splitting) and tabular structural changes (e.g., transposing and building cross-tabulations). As such, they fairly reflect the actual user experience in various contexts. Second, Rigel has shown state-of-the-art usability where users can directly construct target tables mostly by drag-and-drop interactions, which is friendly to average users, making it an ideal comparison target.

Tasks. Table III shows 8 tasks (T1-T8) we designed in the study. T1-T4 are exactly the same as the tasks conducted in the user study of Rigel [11], while T5-T8 are newly proposed in this paper. We kept the following considerations in mind while designing these tasks. First, the tasks should be diverse enough to cover the design space in Table II and able to be performed by the two systems. To this end, the tasks took relational tables in the form of comma-separated values (csv) as the input data. Besides, we restricted the corresponding specifications to only contain the seven operations (*split*, *sum*, *average*, *filter*, *sort*, *concat*, and *count*) and at most one nesting

TABLE III
TASKS FOR THE USER STUDY WITH THEIR DESCRIPTIONS (ALSO USABLE AS EXAMPLE INSTRUCTIONS) AND RIGEL SPECIFICATIONS.
Entities that appear in the schema of the input table are in italics. The specifications of the target tables are shown in a template form where entities are replaced by capital letters (A, B, ...), and functions are represented as $f_{1:1}$, $f_{1:N}$ or $f_{N:1}$ depending on the type.

Task	Description (Example Instruction)	Rigel Specification Template
T1	Gather the *authors* for each *paper*	(A), () $\rightarrow$ (B)
T2	Generate a key for each record	$(A\times$B$)$, () $\rightarrow$ (C + $f_{N:1}(A, B)$)
T3	Remove empty values of *Type* & build a crosstab of the *Number* for each (*Type*, *Usage*) pair and *Name*	$(f_{1:1}(A)\ \times$B$)$, (C) $\rightarrow$ (D)
T4	Transpose the table	(), $(A\ \times$B$)\rightarrow$ (C)
T5	Summarize the *Scores* for each student and sort in descending order	(A), () $\rightarrow$ ($f_{1:1}(f_{1:1}(B))$)
T6	Split the *Name* into first name and last name	$(f_{1:N}(A))$, () $\rightarrow$ ($f_{1:N}(A)$)
T7	Calculate the average *GDP* by month and year	$(f_{1:N}(A))$, $(f_{1:N}(A))\rightarrow$ ($f_{1:1}(B)$)
T8	For each course, count the students who score lower and greater than *70*	(A), () $\rightarrow$ ($f_{1:1}(f_{1:1}(B))$ + $f_{1:1}(f_{1:1}(B))$)

due to the implementation limitation of the Rigel interface. Second, to balance cognitive load and complexity, we ensured that the raw data had at most 20 records and 4 entities similar to Rigel and a single task used at most 4 data functions. Third, we designed tasks with different complexity of descriptions, varying from intuitive ones (e.g., *"split a name into first name and last name"* in T6) to complicated ones (e.g., *"build a cross-tabulation with multiple entities"* in T3). We also conducted a pilot study to make sure that the tasks could be finished by both two systems and could be naturally expressed within 25 words in NL instructions.

Study procedure. We conducted a within-subject study with two sessions, where the participants were required to finish the same 8 tasks using Rigel and NL2Rigel respectively, one system per session. We shuffled the order of the two systems among the participants to balance the learning effect. At the beginning of each session, participants were shown a video tutorial introducing the basic functions of the system. They could freely use the system and ask questions. The study proceeded to finish eight tasks when the participants felt they had got familiar with the system. To avoid bringing bias to users' NL instructions, we omitted the textual descriptions and only gave input and output tables for each task. Specially, we provided non-NL hints for some tasks whose goals might be hard to identify at first glance, e.g., formulas for aggregations, color marks for group-bys, and arrows for sorting. Compared with explaining the tasks to users in words, we believe these hints could minimize the bias brought to users' instructions. Before performing tasks on the system, participants had to infer the transformation and confirmed it with the experiment moderator. They were then given 5 minutes to finish each task, during which their interactions were recorded. We provided a 2-minute break between the two sessions. Users were asked to finish post-questionnaires and report their feedback before the end of the study. The total duration of the study was about 90 minutes and all participants were paid $15 as compensation.

Measures. We utilized two classes of metrics to evaluate the performance of two systems. From the task perspective, we calculated the completion rate and time for each task. For NL2Rigel, We also measured the time spent on giving each instruction and the total number of instructions provided to assess how NL2Rigel assisted users in synthesizing transformation pipelines, understanding the pipelines and refining results. From the user perspective, we included five problem sets in the questionnaire. The first set used a 7-point Likert scale to inquire the preference of users to the two systems on the same six sub-scales as NASA-TLX [80]. It was then followed by four sets of 7-point NASA-TLX scales, with three sets regarding the user experience in instructing to synthesize the first table, understanding transformation pipelines and refining the results, respectively. The final set collected user's overall degree of satisfaction of the NL2Rigel system.

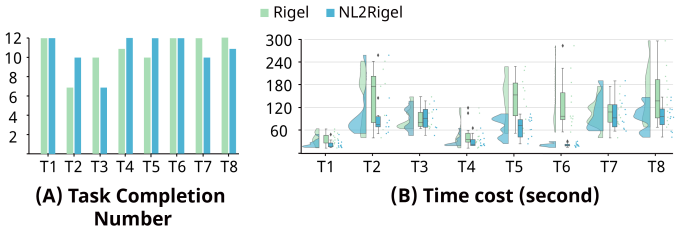


Fig. 8. The number of tasks completed (A) and the time cost for the successful tasks (B) of the participants when using Rigel and NL2Rigel in the user study. Generally, the total tasks complete numbers are the same while NL2Rigel shows a lower time cost in most tasks.

B. Qualitative Results

1) General Reactions: It is widely acknowledged that NL interactions are intuitive and natural to users in numerous fields, including data transformation [16]. Our study strengthens this result with a consensus over the participants on NL2Rigel's low learning cost and the benefits of its end-to-end table generation in saving users' mental efforts to some extents, such as decomposing table construction tasks into concrete operations or into a formal specification like Rigel. In addition, all participants appreciate NL2Rigel's support for various expression styles, which is friendly to novice users.

2) The effectiveness of multi-step table construction: NL2Rigel features a multi-step table construction approach where users can first directly construct a table through a general NL instruction and then refine it afterwards. Overall, most participants (10/12) found this method helpful in their data engineering practices. P5 thought the workflow of NL2Rigel resembles the "overview first, then details" rationale and naturally matched her thinking style. However, more than half of them (8/12) also mentioned the mental pressure brought by coming up with a general task expression, especially in complex tasks that are hard to describe and more suitable for a progressive workflow. Some participants also said this might be due to their suspect of the system's capabilities and

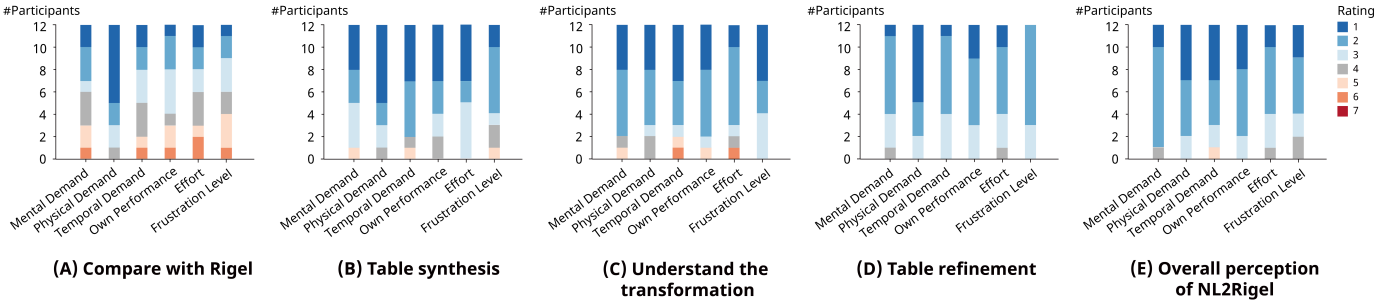


Fig. 9. Quantitative results of the user study. (A) Results of the 7-point Likert scale on participants' preference for Rigel and NL2Rigel on 6 criteria. (1: Strongly prefer NL2Rigel; 7: Strongly prefer Rigel) (B-E) Results of the 7-point NASA-TLX scale on participants' perception of NL2Rigel in terms of three individual steps during the workflow (table synthesis (B), understand the transformation (C) and table refinement (D)) and in general (E). (1: A low level of demand/effort/frustration perceived; 7: A high level of demand/effort/frustration perceived)

therefore they were more cautious when giving utterances. Nevertheless, P1, P3 and P7 claimed this style of table construction is "pedagogical", which might help users organize the ideas in mind before performing a task. In particular, P1 believed this merit is further emphasized when compared with Rigel. "When using NL2Rigel, I gradually formed a clear goal and got what I want. While in Rigel, I often finished tasks by coincidence through arbitrary drag-and-drops with my mind empty.(P1)" To conclude, the table construction method of NL2Rigel is effective and thought-provoking for various users and can get better with potential improvement in reducing expression efforts.

3) Refine results through NL: precision v.s flexibility: NL2Rigel allows users to use NL to refine the generated target table with different granularities. While most participants were satisfied with the performance, we also received some negative feedback. Four participants(P4, P6, P11, P12) expressed their wish that manual fine-tuning is also supported for some frequently-used operations such as sorting and deleting a column, which is more close to their habits and sometimes failed with NL. P9 and P11 also hoped the system becomes more intelligent so that the scope of refinement can be automatically inferred. On the other hand, there also exist users who appreciated the flexibility of NL during table refinement. P10 commented that she was excited to realize some customized minor changes through NL such as making a column listed as a row. In general, NL gives users a higher degree of freedom and flexibility that has the potential to support wide-ranging mutations, while manual interactions can complement NL in some common patches for their precision.

4) Beyond inspection: from reactive to proactive: As reflected by prior work, users tend to get confused about the transformation process without some explanatory information. To this end, NL2Rigel provides three divisions of information: detailed intermediate results, a dataflow diagram and NL descriptions for a given cell. Our observation is that although none of the participants checked all the provided information, most of them benefited from at least one function. In particular, seven participants expressed their preference for the dataflow diagram. (P4: "This graph was pretty straightforward and the color encoding was super distinctive.") Meanwhile, four participants pointed out the helpfulness of the extracted-table view and derived-entities panel. (P9: "I barely glanced at the pipeline graph because the (extracted) table and entities here were compact and clear enough.") Two participants also mentioned the potential usefulness of the NL description provided when a cell is clicked, with P6 further suggesting including formulas when the dataset is relatively small. To conclude, the strategies for explanation adopted by NL2Rigel are complementary and have the ability to meet the demands of various users.

On the other hand, there also exist some participants that require a more effective style of communication with the system. For instance, T4 requires users to transpose a given table. Because she had no idea about the terminology "*transpose*", P2 initially wrote "make the rows columns and columns rows" and ended up getting a cross-tabulation of the original table. "...From the pipeline diagram, you can easily grasp how the table is constructed...But I still can't get why the system got me wrong(P2)." Similarly, P6 gave the instruction "create a new column of... and put it after the third column" in T2. In this case, NL2Rigel did correctly create the column but put it in a wrong place. "Of course the system partially understands me. But what about the failed parts?(P6)" In a word, explanation for the transformation results should go one step further from visualizing the conversion process to resolving user-specific confusions. Future systems may offer more interactions to help users proactively seek needed information apart from understanding pre-generated contents, such as asking the "why" questions [16], [81] or requesting for explanation of how the system interprets specific phrases in their NL instructions.

C. Quantitative Results

Task completion rate. As depicted in Fig. 8(A), the participants manage to finish the same number of tasks using both Rigel and NL2Rigel (86 out of 96), which indicates that NL2Rigel has a comparative task success rate. In particular, half of the failed attempts (5 out of 10) in NL2Rigel origin from T3 and about one fourth come from T7 (2 out of 10), which both require users to build a complicated cross-tabulation and remove invalid values at the same time and is therefore hard to describe in a few words. Besides, a few users got frustrated after several times of rephrasing and gave up eventually (P1 on T3, P2 on T7 and P11 on T2). There is

also one occasional case due to the temporary failure of GPT-3.5 API (P9 on T8). In contrast, apart from 2 give-up cases (P1 on T2 and T3), failure cases in Rigel are due to the timeout mainly caused by user's misunderstanding of the underlying declarative grammar or improper use of data functions.

Consumed time. The total time cost of participants for the successful tasks is illustrated in Fig. 8(B). The results show that participants generally spent less time on the tasks in NL2Rigel than in Rigel. Specifically, this advantage was most prominent in T1, T4 and T6, all of which were easy enough to express within 10 words. However, for tasks that were more verbally challenging such as T3 and T7, we observed no significant decrease in the time cost compared to Rigel. Besides, there were slightly more outliers for NL2Rigel than Rigel (3 vs 1). We hypothesize from these statistics that the performance of NL2Rigel may vary depending on the expression abilities of individual users.

Workload. Fig. 9 shows the users' perception of workload while comparing two systems and their perception when using NL2Rigel. As for the comparison, our participants generally perceived a lower level of workload and spent less effort on the tasks in NL2Rigel (Fig. 9(A)). As for the perception of NL2Rigel, most participants were satisfied with the overall performance and individual performance of the workflow steps. They also showed no significant indication of demand and frustration, which demonstrates that our system is user-friendly and easy to use. However, We also observed a few users who claimed to be neutral or favored Rigel, mostly due to the frustration brought by the incorrect results of NL2Rigel and their struggle in articulating satisfactory NL expressions.

VII. Discussion

This section presents our reflections on the failure cases, the implications, the limitations, and the future work of NL2Rigel.

A. Failure cases

During the development and the user study of NL2Rigel, we also observed some failure cases that require additional research efforts to address appropriately. We discuss them from the following three major perspectives, i.e., *data-side*, *LLM-side*, and *user-side* failures.

Data-side failures. There are a few unsuccessful cases mainly caused by issues with the raw data. We have identified two typical classes of issues: (a) non-uniform or implicit data structure, and (b) poor data quality with missing values, empty lines, or irrelevant information. A concrete case of data-side failures is that when testing the example case in the advanced corpus of our task gallery in Section V, we initially used the HTML file of the webpage in Fig. 6(A) as the system input. In this case, NL2Rigel had difficulty generating the correct script to extract data from this file, because it was cluttered with irrelevant information such as figures and icons. Although manually cleaning this file is feasible, we ended up directly copying data from the webpage to a textual file (Fig. 6(B)) as the input for NL2Rigel. Besides, there is no significant evidence that data in specific formats are more or less likely to be extracted.

Currently, the failures are exposed to users in two ways. When the GPT-3.5 model cannot generate executable scripts within 5 re-runs, an error message is sent to the user. Alternatively, users may inspect the extracted relational table to detect errors. We have attempted to alleviate the problem by allowing users to manually edit the generated relational table in the extracted-table view (Fig. 3(B1)), which supports fixing certain problems such as error values. However, considering data cleaning is beyond the main focus of this paper and also a process that is arguably arduous and demanding in manual efforts [33], our current design is modest to some extent. Future work may enable users to edit the generated code for data extraction or embed semi-automatic approaches, such as offering metrics of data quality and usage [82], [83] or designing tailored LLM-based pipelines to recommend data cleaning operations.

LLM-side failures. We have also observed that a significant proportion of unsuccessful cases are caused by LLM's inability to generate valid results for a sub-task within several attempts (5 in our implementation) or misinterpretation of the user's intent, even when the NL instruction is clear and unambiguous. For example, when testing the system with the task in the usage scenario (Section III), we observed the following errors:

a) Misunderstand the task. When the user asks to compute the total sales, the LLM simply made an aggregation over the *Unit_Price* entity instead of the sales of each order.
b) Output invalid specifications. When calculating the sales for each order, the LLM directly multiplied *Order* and *Unit_Price* without extracting the *Quantity* by splitting the orders. We hypothesize that the LLM had difficulty identifying the implicit data types, such as numerical and categorical entities. Besides, there were also cases where the specifications generated by the LLM failed to meet the grammatical rules of Rigel or our restrictions on the output format even if they had been specified in the prompt.
c) Output values instead of specifications. When tested on a dataset with only a few records, the LLM directly output the concrete value of the total sales instead of following the instructions in the prompt. As a related note, this phenomenon was also observed when the given transformation task was easy enough to be resolved with fewer than 2 functions used in the Rigel specification.
d) Generate buggy scripts. The code for data extraction generated by the LLM led to unexpected results when executed on the raw data. In a typical case, the LLM generated a program where the number of iterations to extract the data was hard-coded based on the sampled data.

The effort NL2Rigel devotes to mitigating these issues is two-fold. From the engineering perspective, we perform syntactic checks and re-run relevant sub-tasks of the workflow in case of errors, which helps to reduce failures due to invalid results such as the problems b)-d). However, since no guidance is provided for further executions when an invalid result is found, there is high randomness in the regenerated results, and the number of reruns required may vary. We encourage future work to propose strategies for diagnosing runtime problems and studying the effect of dynamically altering the prompt

to handle identified problems. From the interface perspective, we propose a data flow diagram and fine-tuning interactions to facilitate understanding and refining the transformation pipeline. While it is helpful in most cases, it can be arduous for users to debug the transformation through NL instructions when the results deviate too much from their expectations. We suggest that a promising direction is to propose targeted datasets and fine-tune the LLMs for better performance.

User-side failures. These failures are mainly caused when users provided an ambiguous NL instruction that LLM misunderstood. Our user study has demonstrated that users tend to use colloquial descriptions (e.g., "*make the rows columns and columns rows*") on unfamiliar transformations, such as table transposing and folding, and therefore often underspecify the task. Besides, when the user desires a table with multiple transformations but provides a general NL instruction that lacks sufficient transformation details, the system performance can deteriorate. For instance, given an abstract query "*show me the sales of the best-performing car models as a table*", the system may fail to map the word "*best-performing*" to appropriate operators, i.e., *sort* and *filter*. It also tends to recommend basic tables as synthesized results, such as the tables in Fig. 7(B-D). We hypothesize one of the underlying reasons is our prescriptive style of prompting, and the issue is likely to be mitigated by including more examples with ambiguous or highly abstract utterances in the prompt or fine-tuning the model with datasets.

Meanwhile, although NL2Rigel is designed with no pre-requisite skills or knowledge in table synthesis, the lack of domain knowledge (e.g., table-related jargon) and familiarity with the system for users also contributes to these cases to some extent. One promising solution is to recommend revised NL instructions to users after they provide an utterance, where users may identify the ambiguity of the previous instruction and gradually pick up the skills for composing accurate and interpretable instructions [64], [84]. We plan to integrate this technique into the system in the future.

To conclude, we hypothesize that the system is better at handling concrete tasks, whereas its performance in processing abstract NL is limited by the degree of semantic ambiguity, the task domain and complexity, and the understanding abilities fostered by the examples provided in the prompts. A thorough investigation of the input limits may require a versatile tailored dataset and we leave it as future work.

B. Implications

This paper presents NL2Rigel, an end-to-end interactive system for synthesizing and improving table transformations with NL instructions. By leveraging NL, users have high flexibility and low learning costs in customizing the data transformation pipelines and refining the result tables. Furthermore, the system is able to process large-scale datasets theoretically, though in practice its performance may depend on factors like the intricacy of transformation tasks due to the time complexity of Rigel's computation module. We discuss a list of implications we gained as follows.

Table verification and debugging. Numerous research has shown tabular data is error-prone in the real world given common issues like missing values or wrong row/column headers [71]. In the context of automatic table generation, the boundary of such anomalies can be extent to unexpected results or system internal errors [16]. The lack of support for detecting and handling errors has long been a shortcoming of both end-to-end systems [58] and table construction [16]. In NL2Rigel, we address this limitation by providing intermediate results including an extracted relational table and derived entities, a diagram of transformation pipelines, and explanations of the selected cells in the end-to-end workflow. In addition, we allow users to interactively debug target tables by providing supplementary NL instructions. In this way, users can naturally reason about the transformation process and refine the result tables. Besides, since the generated tables are correct as long as the transformation process is verified, our table examination strategy avoids the inefficient inspection of table details and is scalable to large input data.

Application of LLMs. NL2Rigel is among the first studies that takes advantage of the reasoning abilities of LLMs in the data transformation community. Although there also exist some LLM-side applications in many other areas [62]–[64], [66], our work contributes to this emerging trend in the following factors. From the perspective of the interface, we showed a tailored NLI empowered with LLMs to solve complex content generation tasks like end-to-end table synthesis. This may serve as a preliminary glimpse into the future interfaces for more complex data-side tasks. In terms of the approach, NL2Rigel interprets users' NL instructions by *chaining* multiple LLM layers [63], which is applicable to other domains, such as visualization authoring which also generates domain-specific languages like Vega-Lite [61]. Moreover, we developed a new strategy for human-LLM collaboration, where LLMs are continuously used for generation, explanation, and iteration, lowering the barrier to interacting with the system.

C. Limitations and Future Work

We identify five limitations of NL2Rigel. First, considering that the system currently only samples the preamble of data and relies on the LLM to interpret its internal structure, our approach may be limited in handling data that is structured too implicitly or with quality issues. This can be addressed with more robust sampling strategies, such as sampling different parts of data in parallel. Another solution is providing information about data quality to help users examine the extracted table, especially when a large dataset is encountered. Second, the performance of NL2Rigel could be closely relevant to that of the underlying LLM and prompting method. Although we optimized the performance of LLM through prompt engineering, failure cases may still exist. We hope future research may contribute public NL-to-table datasets, which can be used to fine-tune the LLMs or train tailored models. Meanwhile, future work may further refine the prompts we used in our approach with more powerful LLMs. Third, the performance of NL2Rigel may also be restricted by the intrinsic drawbacks of declarative languages, which often excel at encoding the overall table structure rather than detailed value changes. Besides, the current implementation only supports a limited

number of data functions. One potential solution is redesigning the functions to enable the inclusion of simple embedded scripts as parameters, extending the expressiveness of the current function set with a lightweight mutation. Finally, the design of our user study has some limitations. We do not include open-ended tasks in the user study and the design of tasks are somewhat conservative to fit both systems. The usefulness of system-initiative recommendations for NL instructions is also not very well studied.

We have also identified some problems that deserve further systematic investigation. First, while we believe our system can be leveraged by users with various expertise, further empirical research is necessary to validate the extent of benefits experienced by different user groups. We plan to study these problems in detail in the future. Meanwhile, it is also unclear how our LLM-based approach fares compared to traditional machine learning methods that typically utilize grammar-based algorithms for parsing NL and feature task-specific models. Comparing and contrasting these approaches can contribute to a better understanding of the value brought by the LLM to our system. We encourage future researchers to conduct more thorough empirical studies on these issues.

VIII. Conclusion

This study presents NL2Rigel, an interactive system that helps users synthesize and enhance tables from semi-structured text using natural language instructions. During the conversion process, NL2Rigel takes advantage of the reasoning abilities of large language models and translates the user instructions into the specifications of Rigel, a recently developed declarative language for tabular data transformation. An intuitive interface is also developed to assist users in understanding this process and refine the tables through providing supplementary instructions. To evaluate NL2Rigel, we design a task gallery showing its expressiveness and perform a comparative user study with Rigel to demonstrate the usability and effectiveness in reducing user efforts.

Acknowledgments

The work was supported by National Key R&D Program of China (2022YFE0137800) and NSFC (U22A2032), and the Collaborative Innovation Center of Artificial Intelligence by MOE and Zhejiang Provincial Government (ZJU).

References

[1] C. Coronel and S. Morris, *Database systems: design, implementation, & management*, 12th ed. Boston: Cengage Learning, 2016.

[2] M. Zhou, W. Tao, P. Ji, H. Shi, and D. Zhang, "Table2analysis: Modeling and recommendation of common analysis patterns for multi-dimensional data," in *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2020, pp. 320–328.

[3] S. Gulwani and M. Marron, "NLyze: Interactive programming by natural language for spreadsheet data analysis and manipulation," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2014, pp. 803–814.

[4] 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*, 2021, pp. 2389–2399.

[5] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer, "Wrangler: interactive visual specification of data transformation scripts," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2011, pp. 3363–3372.

[6] P. J. Guo, S. Kandel, J. M. Hellerstein, and J. Heer, "Proactive wrangling: mixed-initiative end-user programming of data transformation scripts," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2011, pp. 65–74.

[7] Trifacta, Inc., "Trifacta." https://www.trifacta.com, accessed: 2023-03-31.

[8] Z. Jin, M. R. Anderson, M. J. Cafarella, and H. V. Jagadish, "Foofah: Transforming data by example," in *Proceedings of the ACM International Conference on Management of Data*, 2017, pp. 683–698.

[9] S. Gulwani, W. R. Harris, and R. Singh, "Spreadsheet data manipulation using examples," *Communications of the ACM*, vol. 55, no. 8, pp. 97–105, 2012.

[10] S. Gulwani, "Automating string processing in spreadsheets using input-output examples," in *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2011, pp. 317–330.

[11] R. Chen, D. Weng, Y. Huang, X. Shu, J. Zhou, G. Sun, and Y. Wu, "Rigel: Transforming tabular data by declarative mapping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 128–138, 2023.

[12] B. Shneiderman, C. Plaisant, M. S. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, *Designing the user interface: strategies for effective human-computer interaction*. Pearson, 2016.

[13] E. Fast, B. Chen, J. Mendelsohn, J. Bassen, and M. S. Bernstein, "Iris: A conversational agent for complex tasks," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[14] V. Setlur, S. E. Battersby, M. Tory, R. Gossweiler, and A. X. Chang, "Eviza: A natural language interface for visual analysis," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2016, pp. 365–377.

[15] T. Gao, M. Dontcheva, E. Adar, Z. Liu, and K. G. Karahalios, "Datatone: Managing ambiguity in natural language interfaces for data visualization," in *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 2015, pp. 489–500.

[16] S. Srinivasa Ragavan, Z. Hou, Y. Wang, A. D. Gordon, H. Zhang, and D. Zhang, "Gridbook: Natural language formulas for the spreadsheet grid," in *Proceedings of International Conference on Intelligent User Interfaces*, 2022, pp. 345–368.

[17] A. Wachtel, S. Weigelt, and W. F. Tichy, "Initial implementation of natural language turn-based dialog system," *Procedia Computer Science*, vol. 84, pp. 49–56, 2016.

[18] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Proceedings of Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[19] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "PaLM: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[20] OpenAI Inc., "Models - OpenAI API," https://platform.openai.com/docs/models/gpt-3-5, accessed: 2023-03-28.

[21] M. Gardner, P. Dasigi, S. Iyer, A. Suhr, and L. Zettlemoyer, "Neural semantic parsing," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2018, pp. 17–18.

[22] A. Narayan, I. Chami, L. J. Orr, and C. Ré, "Can foundation models wrangle your data?" *Proceedings of the VLDB Endowment*, vol. 16, no. 4, pp. 738–746, 2022.

[23] G. Jaimovitch-López, C. Ferri, J. Hernández-Orallo, F. Martínez-Plumed, and M. J. Ramírez-Quintana, "Can language models automate data wrangling?" *Machine Learning*, 2022.

[24] A. V. Miceli-Barone, F. Barez, I. Konstas, and S. B. Cohen, "The larger they are, the harder they fail: Language models do not recognize identifier swaps in python," *arXiv preprint arXiv:2305.15507*, 2023.

[25] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[26] J. Hammer, H. Garcia-Molina, S. Nestorov, R. Yerneni, M. Breunig, and V. Vassalos, "Template-based wrappers in the tsimmis system," in *Proceedings of the ACM SIGMOD international Conference on Management of Data*, 1997, pp. 532–535.

[27] R. Ahmed, P. D. Smedt, W. Du, W. Kent, M. A. Ketabchi, W. A. Litwin, A. Rafii, and M.-C. Shan, "The Pegasus heterogeneous multidatabase system," *Computer*, vol. 24, no. 12, pp. 19–27, 1991.

[28] L. V. S. Lakshmanan, F. Sadri, and S. N. Subramanian, "SchemaSQL: An extension to SQL for multidatabase interoperability," *ACM Transactions on Database Systems*, vol. 26, no. 4, pp. 476–519, 2001.

[29] "Arquero, a javascript library for query processing and transformation of data tables." https://github.com/uwdata/arquero, accessed: 2023-03-31.

[30] "pandas, a Python data analysis and manipulation tool." https://pandas.pydata.org, accessed: 2023-03-31.

[31] H. Wickham, D. Vaughan, and M. Girlich, *tidyr: Tidy Messy Data*, 2022, https://tidyr.tidyverse.org, https://github.com/tidyverse/tidyr.

[32] H. Wickham, R. François, L. Henry, and K. Müller, *dplyr: A Grammar of Data Manipulation*, 2022, https://dplyr.tidyverse.org,https://github.com/tidyverse/dplyr.

[33] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck, and P. Buono, "Research directions in data wrangling: Visuatizations and transformations for usable and credible data," *Information Visualization*, vol. 10, no. 4, pp. 271–288, oct 2011.

[34] S. E. Chasins, M. Mueller, and R. Bodik, "Rousillon: Scraping distributed hierarchical web data," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2018, p. 963–975.

[35] OpenRefine, Inc., "Open Refine." https://openrefine.org, accessed: 2023-03-31.

[36] V. Raman and J. M. Hellerstein, "Potter's wheel: An interactive data cleaning system," in *Proceedings of the VLDB Endowment*, vol. 1, 2001, pp. 381–390.

[37] M. Mayer, G. Soares, M. Grechkin, V. Le, M. Marron, O. Polozov, R. Singh, B. G. Zorn, and S. Gulwani, "User interaction models for disambiguation in programming by example," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2015, pp. 291–301.

[38] V. Le and S. Gulwani, "Flashextract: A framework for data extraction by examples," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2014, pp. 542–553.

[39] D. W. Barowy, S. Gulwani, T. Hart, and B. Zorn, "FlashRelate: extracting relational data from semi-structured spreadsheets using examples," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2015, pp. 218–228.

[40] C. Wang, Y. Feng, R. Bodik, I. Dillig, A. Cheung, and A. J. Ko, "Falx: Synthesis-powered visualization authoring," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2021, pp. 106:1–106:15.

[41] I. Drosos, T. Barik, P. J. Guo, R. DeLine, and S. Gulwani, "Wrex: A unified programming-by-example interaction for synthesizing readable code for data scientists," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–12.

[42] Z. Abedjan, J. Morcos, M. Gubanov, I. F. Ilyas, M. Stonebraker, P. Papotti, and M. Ouzzani, "DataXFormer: Leveraging the Web for semantic transformations," in *Proceedings of the Conference on Innovative Data Systems Research*, 2015.

[43] Z. Abedjan, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, and M. Stonebraker, "DataXFormer: A robust transformation discovery system," in *Proceedings of the IEEE International Conference on Data Engineering*, 2016, pp. 1134–1145.

[44] R. Singh, "Blinkfill: Semi-supervised programming by example for syntactic string transformations," *Proceedings of the VLDB Endowment*, vol. 9, no. 10, pp. 816–827, 2016.

[45] Y. He, X. Chu, K. Ganjam, Y. Zheng, V. Narasayya, and S. Chaudhuri, "Transform-data-by-example (TDE): an extensible search engine for data transformations," *Proceedings of the VLDB Endowment*, vol. 11, no. 10, pp. 1165–1177, Jun 2018.

[46] G. Li, R. Li, Z. Wang, C. H. Liu, M. Lu, and G. Wang, "Hitailor: Interactive transformation and visualization for hierarchical tabular data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 139–148, 2023.

[47] K. S.-P. Chang and B. A. Myers, "Using and exploring hierarchical data in spreadsheets," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2016, p. 2497–2507.

[48] J. Berant, D. Deutch, A. Globerson, T. Milo, and T. Wolfson, "Explaining queries over web tables to non-experts," in *Proceedings of the IEEE International Conference on Data Engineering*, 2019, pp. 1570–1573.

[49] A. Narechania, A. Fourney, B. Lee, and G. Ramos, "DIY: assessing the correctness of natural language to SQL systems," in *Proceedings of the International Conference on Intelligent User Interfaces*, 2021, pp. 597–607.

[50] P. Liang, "Lambda Dependency-Based Compositional Semantics," *arXiv e-prints*, Sep. 2013.

[51] H. Kim, B.-H. So, W.-S. Han, and H. Lee, "Natural language to SQL: Where are we today?" *Proceedings of the VLDB Endowment*, vol. 13, no. 10, pp. 1737–1750, 2021.

[52] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch, "Natural language interfaces to databases - 0an introduction," *Computer Research Repository*, 1995.

[53] A. Narechania, A. Srinivasan, and J. Stasko, "NL4DV: A Toolkit for generating Analytic Specifications for Data Visualization from Natural Language queries," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 369–379, 2021.

[54] J. Huang, Y. Xi, J. Hu, and J. Tao, "FlowNL: Asking the flow data in natural languages," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 1200–1210, 2023.

[55] A. Srinivasan and J. Stasko, "Orko: Facilitating multimodal interaction for visual exploration and analysis of networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 511–521, 2018.

[56] 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*, pp. 1–11, 2019.

[57] Z. Chen, Q. Yang, X. Xie, J. Beyer, H. Xia, Y. Wu, and H. Pfister, "Sporthesia: Augmenting sports videos using natural language," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 918–928, 2023.

[58] Y. Luo, N. Tang, G. Li, J. Tang, C. Chai, and X. Qin, "Natural language to visualization by neural machine translation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 217–226, 2022.

[59] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of the annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[60] Y. Wang, Z. Hou, L. Shen, T. Wu, J. Wang, H. Huang, H. Zhang, and D. Zhang, "Towards natural language-based visualization authoring," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 1222–1232, 2023.

[61] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, "Vega-lite: A grammar of interactive graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 341–350, 2017.

[62] J. J. Y. Chung, W. Kim, K. M. Yoo, H. Lee, E. Adar, and M. Chang, "TaleBrush: Sketching stories with generative pretrained language models," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–19.

[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 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–22.

[64] T. S. Kim, D. Choi, Y. Choi, and J. Kim, "Stylette: Styling the web with natural language," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–17.

[65] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. T. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K.-H. Lee, Y. Kuang, S. Jesmonth, N. J. Joshi, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu, "Do as I can, not as I say: Grounding language in robotic affordances," in *Proceedings of the Conference on Robot Learning*, 2023, pp. 287–318.

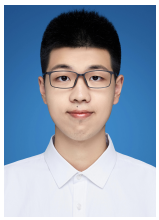
[66] B. Wang, G. Li, and Y. Li, "Enabling conversational interaction with mobile ui using large language models," 2022. [Online]. Available: https://arxiv.org/abs/2209.08655

[67] E. Jiang, E. Toh, A. Molina, K. Olson, C. Kayacik, A. Donsbach, C. J. Cai, and M. Terry, "Discovering the syntax and strategies of natural language programming with generative language models," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–19.
[68] Y. Feng, X. Wang, B. Pan, K. K. Wong, Y. Ren, S. Liu, Z. Yan, Y. Ma, H. Qu, and W. Chen, "XNLI: Explaining and diagnosing NLI-based visual data analysis," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–14, 2023.
[69] A. Srinivasan and V. Setlur, "Snowy: Recommending utterances for conversational visual analysis," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2021, pp. 864–880.
[70] A. Nazabal, C. K. Williams, G. Colavizza, C. R. Smith, and A. Williams, "Data engineering for data analytics: A classification of the issues, and case studies," *arXiv preprint arXiv:2004.12929*, 2020.
[71] D. Sukhobok, N. Nikolov, and D. Roman, "Tabular data anomaly patterns," in *Proceedings of International Conference on Big Data Innovations and Applications*. IEEE, 2017, pp. 25–34.
[72] S. Gathani, P. Lim, and L. Battle, "Debugging database queries: A survey of tools, techniques, and users," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–16.
[73] A. Srinivasan, M. Dontcheva, E. Adar, and S. Walker, "Discovering natural language commands in multimodal interfaces," in *Proceedings of the International Conference on Intelligent User Interfaces*, 2019, pp. 661–672.
[74] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2022.
[75] H. Dang, L. Mecke, F. Lehmann, S. Goller, and D. Buschek, "How to prompt? Opportunities and challenges of zero-and few-shot learning for human-AI interaction in creative applications of generative models," *arXiv preprint arXiv:2209.01390*, 2022.
[76] L. Reynolds and K. McDonell, "Prompt programming for large language models: Beyond the few-shot paradigm," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–7.
[77] P. Baudisch, N. Good, and P. Stewart, "Focus plus context screens: combining display technology with visualization techniques," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2001, pp. 31–40.
[78] "The dagre library." https://github.com/dagrejs/dagre/wiki, accessed: 2023-06-13.
[79] S. Kasica, C. Berret, and T. Munzner, "Table scraps: An actionable framework for multi-table data wrangling from an artifact study of computational journalism," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 957–966, 2021.
[80] S. G. Hart, "NASA-task load index (NASA-TLX); 20 years later," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 50, no. 9, pp. 904–908, 2006.
[81] A. J. Ko and B. A. Myers, "Designing the whyline: a debugging interface for asking questions about program behavior," in *Proceedings of the CHI conference on Human factors in computing systems*, 2004, pp. 151–158.
[82] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer, "Profiler: Integrated statistical analysis and visualization for data quality assessment," in *Proceedings of the International Working Conference on Advanced Visual Interfaces*, 2012, p. 547–554.
[83] A. Narechania, F. Du, A. R. Sinha, R. Rossi, J. Hoffswell, S. Guo, E. Koh, S. B. Navathe, and A. Endert, "Datapilot: Utilizing quality and usage information for subset selection during visual data preparation," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2023.
[84] M. X. Liu, A. Sarkar, C. Negreanu, B. Zorn, J. Williams, N. Toronto, and A. D. Gordon, ""What it wants me to say": Bridging the abstraction gap between end-user programmers and code-generating large language models," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2023.

Yanwei Huang received his B.Eng. degree from Zhejiang University in 2022. He is now pursuing a M.Eng. degree in Software Engineering in the State Key Lab of CAD&CG, Zhejiang University. His research interests include tools for visualization and data wrangling.

Yunfan Zhou is currently an undergraduate student who majors in Computer Science and Technology in Zhejiang University. His research interests include data mining, visual analytics, and human-computer interaction.

Ran Chen is currently a Ph.D. student in the State Key Lab of CAD&CG, Zhejiang University. He received the B.Eng. degree from Zhejiang University in 2018. His research interests mainly lie in tools and systems to author visualizations.



Changhao Pan is currently an undergraduate student who majors in Artificial Intelligence in Zhejiang University. His research interests mainly include machine learning and visualization authoring.

Xinhuan Shu is currently a Research Associate at Univeristy of Edinburgh. Her research probes the intersection of data visualization and human-computer interaction, where she aims to explore innovative approaches to facilitate human-data interaction. She also designs and develops interactive and automatic tools for both novices and experts to ease the creation of data visualizations, paving the way for data democratization.

Di Weng is a tenure-track assistant professor at School of Software Technology, Zhejiang University. His research interest lies in information visualization and visual analytics, focusing on interactive data transformation and spatiotemporal data analysis. For more information, please visit https://dwe.ng.

Yingcai Wu is a Professor at the State Key Lab of CAD&CG, Zhejiang University. His main research interests are information visualization and visual analytics, with focuses on urban computing, sports science, immersive visualization, and social media analysis. He received his Ph.D. degree in Computer Science from The Hong Kong University of Science and Technology. For more information, please visit http://www.ycwu.org.