Natural Language Interaction for Editing Visual Knowledge Graphs

Reza Shahriari
University of Florida
Gainesville, Florida, USA
rshahriari@ufl.edu

Eric D. Ragan
University of Florida
Gainesville, Florida, USA
eragan@ufl.edu

Jaime Ruiz
University of Florida
Gainesville, Florida, USA
jaime.ruiz@ufl.edu

ABSTRACT

Knowledge graphs are often visualized using node-link diagrams that reveal relationships and structure. In many applications using graphs, it is desirable to allow users to edit graphs to ensure data accuracy or provides updates. Commonly in graph visualization, users can interact directly with the visual elements by clicking and typing updates to specific items through traditional interaction methods in the graphical user interface. However, it can become tedious to make many updates due to the need to individually select and change numerous items in a graph. Our research investigates natural language input as an alternative method for editing network graphs. We present a user study comparing GUI graph editing with two natural language alternatives to contribute novel empirical data of the trade-offs of the different interaction methods. The findings show natural language methods to be significantly more effective than traditional GUI interaction.

CCS CONCEPTS

• **Human-centered computing → Empirical studies in HCI**; **Natural language interfaces**.

KEYWORDS

Knowledge Capture and Interaction, Natural Language Interfaces

ACM Reference Format:
Reza Shahriari, Eric D. Ragan, and Jaime Ruiz. 2025. Natural Language Interaction for Editing Visual Knowledge Graphs. In *Knowledge Capture Conference 2025 (K-CAP '25), December 10–12, 2025, Dayton, OH, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3731443.3771344

1 INTRODUCTION

A knowledge graph organizes information by connecting entities and their attributes within a structured, semantic framework. To support meaningful reasoning and interpretation, knowledge graphs are often visualized using node-link diagrams that reveal relationships and structure. Maintaining accurate data is essential for applications relying on network data, especially in dynamic environments where real-time data impacts outcomes [24, 25]. Graph updates involve changes in items (nodes) or relationships (links), and changes to additional properties are also common for knowledge graphs. Regular interaction with these structures, such as editing, expanding, and correcting, ensures they remain reliable. Editing graph relationships is especially useful in human-in-the-loop (HITL) applications, such as correcting or enriching data [1, 9, 17].

While node-link graph visualizations are excellent for easy interpretation of small graphs [3], common forms of interaction for visual graph editing can be tedious and time-consuming for a large number of edits. Typically, interactions with graphs involve using visual interfaces where users select individual nodes or edges to make changes directly within the graph [16]. This process includes actions like clicking nodes to view or edit properties, dragging to explore relationships, and drawing or deleting edges to modify connections. These methods tend to be easily understood and align well with the familiar interaction philosophy of *direct manipulation* [28]. However, direct GUI interactions in graphs also typically require execution of each individual action. Consequently, due to the importance of connectivity in network data, basic updates often require multiple actions. As graph size or complexity grows, GUI interactions become more cumbersome and time-consuming, increasing user effort.

To address the limitations of traditional GUI-based interaction, our research studies alternative methods for editing visual graph representations. We investigate how natural-language processing (NLP) techniques can increase the efficiency of interacting with graphs using text-based inputs. We demonstrate interaction methods that allow users to describe multiple graph relationships and properties through language instead of a sequence of individual edit operations. To evaluate these methods, we compare natural language (NL) inputs with structured textual commands, which offer precision but less accessibility for non-technical users. This comparison highlights trade-offs between flexibility and efficiency in graph editing tasks. Also, we conducted an experiment comparing three interaction types: GUI Interaction, Textual Command, and Natural Language. The study involved user updates to labeled knowledge graphs with varied size and expected changes. Results show that natural language enables effective data manipulation, offering advantages over traditional command formats and supporting more accessible, efficient interaction.

2 BACKGROUND

2.1 Knowledge Graph Visualization

Knowledge graphs illustrate the connections between entities, enabling an understanding of relationships. Various visualization formats support interpretation, with node-link diagrams being the most common, using points (e.g., circles or boxes) for entities and

K-CAP '25, December 10–12, 2025, Dayton, OH, USA

ACM ISBN 978-1-4503-XXXX-X/2018/06…$15.00
https://doi.org/10.1145/3731443.3771344

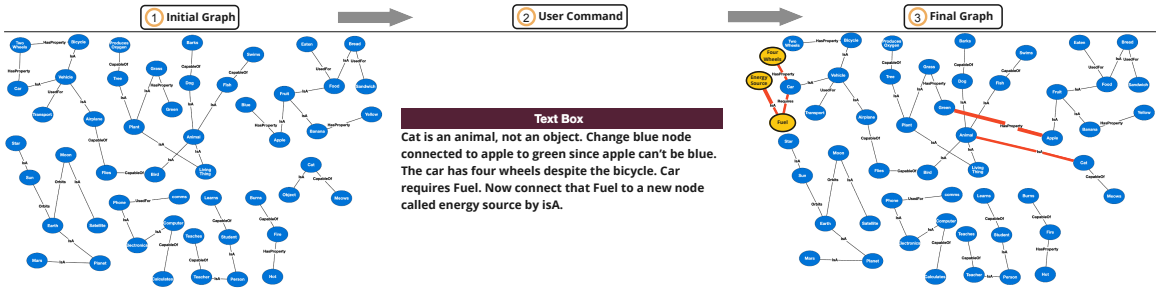

Figure 1: Screenshot of the interface visualizing relationships from the ConceptNet [29] dataset.

lines for connections [18]. Another common representation is adjacency matrix [18], where nodes are arranged along the vertical and horizontal edges of a square, and connections are indicated by flagging the cell at the intersection of two nodes. Moreover, multiple tools [6, 11] combined these representations to allow a hybrid representation of network data that can take advantage of multiple representations. For instance, NodeTrix [11] combines node-link diagrams and adjacency matrices into a single, cohesive representation to ease interpretation. Similarly, GraphTrail [6] integrates various visualizations, such as bar charts, tag clouds, and node-link diagrams, to analyze large multivariate networks. By allowing users to pivot between different aggregates of nodes and edges based on their attributes, GraphTrail supports dynamic exploration of heterogeneous networks.

Different studies have applied various techniques [32, 33] to achieve this balance. For example, Tominski et al. [32] presented fisheye tree views for exploring hierarchical trees as well composite lenses combined with several lens techniques to facilitate the exploration of local graph information. They found that these tools provided enhance navigation of complex graph structures, with their potential in visualizing clustered graphs. While past work improved navigation and understanding of large graphs, more effective methods are still needed for editing, especially in knowledge graphs with rich semantic information.

2.2 Editing Network Data

After visualizing a network data structure, it is important to keep it current and accurate by incorporating new information and correcting any errors. One approach is using *direct manipulation*, that involves actions such as clicking, dragging, and dropping. These interactions provide immediate feedback and enhance the user's sense of control [28]. The goal is to create interfaces that minimize the need for complex command languages and intermediary steps [28]. Optimizing interaction through *direct manipulation* required further exploration of various aspects to understand the reasons behind the usability or challenges that users encountered with these interfaces. For example, Hutchins et al. [13] studied the cognitive aspects of direct manipulation to understand why these interfaces feel natural and engaging to users and to identify potential issues. They found that reducing the cognitive distance between what users intend to do and how they interact with the system, along with providing immediate feedback, are key factors in the effectiveness of direct manipulation interfaces.

While direct manipulation relies on interfaces and immediate feedback, natural language interaction (NLI) enables more conversational interactions [15]. Voice-activated commands and chatbots allow users to perform tasks through natural language, improving accessibility and reducing the cognitive load of traditional graphical user interfaces (GUIs) [26]. By combining direct manipulation and natural language processing, Cohen et al. [5] developed the Shoptalk system, a prototype for information and decision making. It allowed users to interact with databases and simulations using language alongside graphical actions like pointing and menus, enabling object description, temporal reasoning, and large-scale operations with immediate visual feedback.

2.3 Natural Language Interaction

While we mentioned different ways of editing network data and the use of NLP, the next step is to analyze how these natural language interactions occur within systems. One of the main benefits of NLI is that it enables individuals to engage with machines using conversational language, eliminating the need to learn commands or interfaces [19]. By taking advantage of NLI technology, researchers examined data visualization interaction models based on natural language [27, 34, 38]. For example, Yu et al. [38] developed a natural language interface called FlowSense, enabling users to construct and modify dataflow diagrams using natural language commands.

While studies such as those by Wang et al. [36] and Power et al. [23] have demonstrated the overall efficiency of NLP in various domains, it is essential to acknowledge that NLP techniques are not without limitations and challenges. For instance, NLP models often struggle with context ambiguity since the same phrase can have multiple meanings. In particular, when dealing with complex network graph data, these shortcomings can become even more obvious due to the inherently interconnected nature of graphs.

To contribute to the existing body of literature on visualization and interaction techniques for networked data with a focus on node-link diagrams, we integrated and compared direct manipulation of graphs through click-based details on demand with more advanced methods like natural language interaction by utilizing LLMs to study the trade-offs and efficacy of each method. By comparing these forms of natural language interaction with traditional GUI-based methods, we can better assess which approach is more effective for different scenarios, such as making minor corrections or managing multiple data entries.

3 EXPERIMENT

3.1 Research Goals

Existing literature has demonstrated the efficiency of natural language interaction across various fields [34, 38], highlighting its potential to simplify complex tasks. Building on this foundation, our research aims to explore how natural language techniques can enhance interaction within node-link or graph structures by comparing them to interaction through common GUI designs involving cursor selection, menus, and individually editing items. The specific research goals are as follows:

- **RQ1**: How does natural language input compare to GUI interaction for graph editing?
- **RQ2**: How does method performance vary with graph size and extent of changes?
- **RQ3**: How do command-style (e.g., "rename X to Y") vs. conversational text inputs (e.g., "it's not an X, it's a Y") affect editing?

To form our hypotheses, we considered the strengths and limitations of each interaction technique. We expected natural language to outperform GUI methods due to its lower learning curve and ability to handle multiple edits in a single input, making it efficient for complex or contextual updates. However, we also predicted that GUI interaction might be more effective when only minimal edits are needed, as it allows for direct, precise adjustments without requiring detailed instructions.

Furthermore, we recognize that natural language can be used in multiple ways. Users might either describe a scene more narratively (e.g., "a person is holding a bottle") or issue concise commands (e.g., "add a person and a bottle node"). We expected that when users encounter minor issues in the graph, they will likely prefer command-like natural language inputs to fix specific elements (e.g., renaming a node or adjusting its properties) rather than describing the entire scene again. This approach enables quicker, more targeted corrections, combining the flexibility of natural language with the precision of command-based interactions.

3.2 Graph Editing Task

To address our research goals through a user-study experiment, we needed a graph-editing task that involved various types of graph modifications at both the node and edge levels. Knowledge graphs were chosen because they offer structured representations of graph data, where entities (nodes) and their relationships (edges) are clearly labeled and interconnected. These graphs are further enhanced with features like metadata labels, hierarchical structures, and contextual information, making them ideal for modeling complex relationships and data attributes. Knowledge can capture rich semantic relationships that apply to a wide range of real-world scenarios [10, 21, 31]. Interaction with knowledge graphs for data review also enables users to address errors, fill in missing information, and better connect related data, enhancing scalability and supporting the addition of new features [22].

As the basis for graph editing in the experiment, the experimental design required a way to encourage participants to make changes to a graph. However, we also sought to avoid textual instructions for changes to prevent influencing the participants' use of natural language input by adapting the given text from instructions. Thus, we designed the experimental task for graph editing based on an image prompt (Figure 1). To this end, we utilized the *GQA Dataset* [12], a highly regarded resource for visual reasoning that includes real-world images alongside their corresponding knowledge graphs that represent the objects and their spatial relationships. Figure 3 shows an example from this dataset within the study interface. Participants reviewed each image and its graph, then updated the graph by adding or correcting connections based on the image content. While they had flexibility in the level of detail, the image simplicity encouraged consistent editing around primary objects.

3.3 Interaction Methods

The experiment compared three different interaction methods for graph editing: *GUI Interaction*, *Natural Language*, and *Textual Command*. For all versions, the study used the same visual design in the study application for displaying the graphs (Figure 3). To provide feedback to users after each input and help users easily see changes, the most recent modification to the graph—such as adding nodes or edges—were visually highlighted in red. However, the GUI condition provided additional sub-task feedback (such as context pop-up menus or node highlights to show selected nodes during node-linking actions) to aid operations requiring multiple selections (Illustrated in Figure 2). The following subsections describe the three interaction methods and explain their differences.

3.3.1 GUI Interaction. We implemented the GUI Interaction interaction method (shown in Figure 2) as a standard method for interacting with a graph by cursor selection and contextual pop-up menus to select operations (i.e., add, remove, or rename) on the selected nodes or edges. Users could also click on the white space of the box to create and name a new node. Keyboard input was used for naming or editing the textual labels of nodes or edges. For this method, the interface also included a "Remove all" button to delete all current nodes and edges in the graph. Additionally, an "Undo" button was provided to revert the last action, whether it involved adding, renaming, or deleting a node or edge.

3.3.2 Textual Command. While GUI interaction relies on interfaces and immediate feedback, natural language processing (NLP) enables more conversational input [15]. We developed a technique that lets users modify graphs through structured text commands for various operations. For example, users can enter commands like "Rename X to Y" or "Connect X to Y by Z" to rename nodes or add edges (Figure 2). This method is limited to structured edits and excludes scene descriptions, which are handled by the natural language approach in Section 3.3.3. Commands are parsed to apply predefined functions for adding, removing, or renaming graph elements.

To study the potential of this approach, we leveraged the ChatGPT API [20], specifically the "gpt-3.5-turbo" model, to implement these features. Detailed instructions are then sent to the ChatGPT API [20], explaining how to update the graph based on the user's input. These prompts break down all possible operations (e.g. add, rename, delete, etc.) and include examples for each through few-shot prompting [2], thereby enhancing the model's performance and accuracy in understanding and executing user commands.

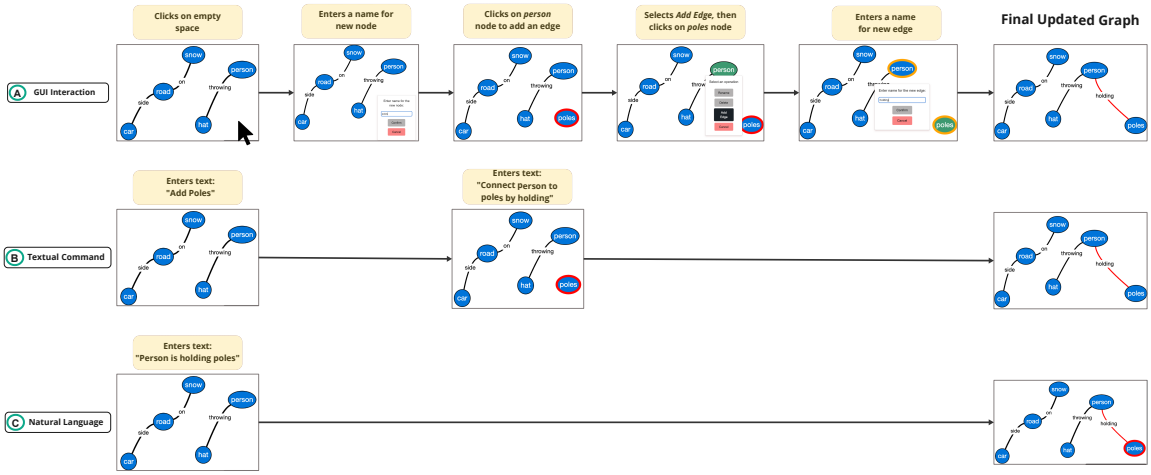

Figure 2: Screenshots of the interface before and after user commands: (A) GUI Interaction via clicking and menus, (B) Textual Command using structured input, (C) Natural Language using free-form input.

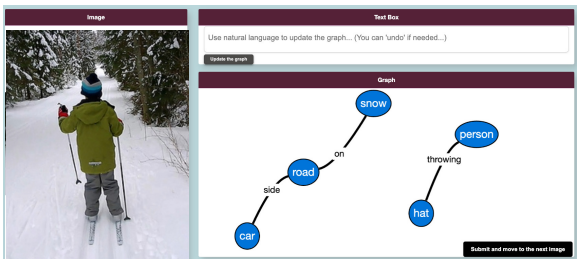

Figure 3: Screenshots of the study interface, including the text box for the Natural Language method. In each trial, participants updated the graph to match the image on the left.

To validate the model output before running the study, we conducted multiple pilot studies and sessions with researchers who tested the system using challenging and unconventional commands. Through several iterations, the model demonstrated high accuracy by effectively handling nearly all complex cases and natural language inputs. Importantly, the analysis of the data logged for Natural language conditions (51 out of 76 participants) showed that only a small number of trials (2.8%) had to issue *undo* commands to revert the changes, indicating the system's reliability and consistency.

In terms of performance metrics, the average processing time for both natural language and textual commands was 2.43 seconds. This rapid response time ensures a near-seamless user experience, minimizing wait times to prevent potential user frustration or disengagement. As described in *data processing* Section 3.6, we subtracted the processing time for each trial to allow a solid comparison without biasing the results, ensuring that performance metrics are directly tied to the specific model's capabilities.

3.3.3 Natural Language. This method is an expanded version of the previously described Textual Command technique from Section above (3.3.2). The Natural Language technique not only supports all the command types, but it also allows users to describe entire scenes using either a single command or multiple separate commands. For example, users can issue commands like "person is holding poles", "person is wearing a hat", and "person has a green jacket" either sequentially or combined in a single statement, and the graph will be updated accordingly (as shown in Figure 2). This approach can accurately process any type of human-understandable text input to reflect changes in the graph. To implement these features, we leveraged models from the OpenAI API [20] (Section 3.3.2).

3.4 Experimental Design

In evaluating the different techniques for updating knowledge graphs, we also aimed to assess their efficiency across different scenarios and take into account variations in graph size and setup. By comparing the number of changes made per unit of time or the number of operations required to implement a number of changes, we provide findings regarding the effectiveness of different methods for updating graphs as well as their trade-offs. Moreover, participants reviewed an image alongside its corresponding knowledge graph (Figure 3) and were tasked with adding or correcting information based on their assigned experimental condition.

To further analyze the impact of each interaction method on the graph, we divided the trials into two size groups for each participant, which we refer to as *small* and *large*. This division allows us to observe the effects of interaction methods on graphs of different complexities, helping to identify if certain methods are more effective based on the graph size. Graphs in the *small* group had 4 or 5 nodes and 3 or 4 edges, while graphs in the *large* group had 7 or 8 nodes and 6 or 7 edges.

Additionally, to evaluate the effectiveness of these methods in scenarios with varying numbers of inaccuracies, we categorized the trials into three graph setup: *Major* (50-60% inaccuracies), where the task involves correcting a significant number of inaccuracies; *Minor* (20-30% inaccuracies), where the task involves addressing a relatively small number of inaccuracies; and *Empty*, where the task requires starting from scratch and adding substantial new data to an initially empty graph. It is important to note that we manually

randomized all within-subject trials to minimize potential random effects and ensure a balanced distribution of conditions. This setup allows us to assess how different interaction methods perform in diverse contexts, from minor corrections to building a graph from scratch. Therefore, the experiment followed a 3x2x3 mixed design with three independent variables: 1) *Interaction Method* (between subjects) and 2) *Graph Size* (within subjects), and 3) *Graph Setup* (within subjects) as shown in Table 1.

Table 1: Independent variables and their levels for the 3×2×3 mixed-design user study

Independent Variables	Levels
Interaction Method *(between-subjects)*	1. GUI Interaction 2. Textual Command 3. Natural Language
Graph Size *(within-subjects)*	1. Small (4–5 nodes & 3–4 edges), 12 trials 2. Large (7–8 nodes & 6–7 edges), 12 trials
Graph Setup *(within-subjects)*	1. Major (50–60% inaccuracies), 8 trials 2. Minor (20–30% inaccuracies), 8 trials 3. Empty (start with an empty graph), 8 trials

3.5 Procedure and Participants

The study was conducted online using a web application, allowing participants to work at their own pace without researcher intervention. Participation was voluntary, with extra credit offered. The study was approved by the Institutional Review Board (IRB). Participants first provided informed consent and completed a demographic questionnaire (age, gender, education). They were then randomly assigned to one of three experimental conditions (Table 1). After a tutorial and example trials, participants completed 24 image-graph tasks, with unlimited time to finish. They were required to make at least one change per graph to proceed. Interaction logs were collected anonymously, and data quality was ensured by excluding outliers and those who made few changes despite taking excessive time. Eighty-nine participants took part, but after two rounds of quality checks (Section 3.6.1), 76 were included in the analysis: GUI Interaction (25), Textual Command (25), and Natural Language (26). All participants were students in computer science or human-centered computing courses, aged 18–40 (median 20). The final group included 45 males, 29 females, and 2 non-binary individuals.

3.6 Measures

During the online experiment, we logged participants' interactions to evaluate each interaction method. For each image reviewed, we recorded 1) *user response time*, 2) *final version of the graph*, and 3) *graph operation details*. In addition to the final edited version of the graph submitted by each participant, we also logged all the *graph operation details* for each action. For the direct manipulation method, this includes *operations' name* such as adding, renaming, deleting, or removing nodes or edges. For the natural language methods, it includes the *user commands or text* used to edit the graph, along with the *processing time* for each operation, which indicates the time taken on the backend to process the results.

3.6.1 Data Processing. To fairly compare interaction methods, particularly the time taken to make graph changes, we subtracted backend processing time from the response time in natural language conditions. This adjustment accounts for system delays unrelated to user performance and ensures comparability with direct manipulation, which has no backend delay. Backend time was excluded because it varies by model complexity. Also, we applied two quality control steps: (1) participants with below 80% accuracy across trials 1, 8, 16, and 24 were excluded to ensure consistent engagement (8 excluded); (2) trials with response time Z-scores above 2 were removed (4.2% of trials), and participants with more than three such trials were also excluded (5 more excluded), resulting in 13 total exclusions from the 89 participants.

3.7 Metrics

After logging and filtering the data, we calculate the number of changes made to each graph. This helps assess the speed and quality of edits across interaction methods and graph types. We define two metrics based on these changes to present our results.

3.7.1 Graph Changes Per Time. For each image reviewed, we captured 1) *User completion time*, 2) *Final version of the graph*, and 3) *Operation Details* as explained in Section 3.6. Then, we detected differences between the initial graph and final graph to determine the number of changes each participant made for each image. The *graph changes per time* metric was calculated by dividing the *number of changes* by the *response time* for each trial or image. This metric allows us to compare the advantages and trade-offs of each interaction method by quantifying how quickly and effectively users can modify the graph.

3.7.2 Graph Changes Per Action. We used the *graph changes heuristic* to determine the number of changes each participant made for each image by comparing the initial and final graphs. The *graph changes per action* metric was calculated by dividing the *number of changes* by the *number of actions* for each trial or image. This metric enables us to compare the advantages and trade-offs each interaction method by determining whether one method allows users to achieve their goals with fewer or more precise actions.

4 RESULTS

The experiment evaluated graph manipulation efficiency across interaction methods, graph sizes, and setups. As the data violated normality and homogeneity assumptions, we used ARTool [37] for nonparametric factorial analysis via aligned rank transformation. A three-way mixed ANOVA tested the effects of interaction method, graph size, and setup. For significant main effects, we performed post-hoc paired t-tests with Tukey correction using ARTool [7], and report results with partial eta squared (η_p^2) as the effect size.

4.1 Changes Per Time

The study's results highlight findings regarding how different interaction methods for editing graphs impact the average changes per time for the varied sizes and expected amount of changes in each graph (Figure 4, Table 5). The test detected a significant main effect of the *interaction method* on *changes per time*. Post-hoc tests revealed that Natural Language resulted in significantly higher changes per time compared to both GUI Interaction ($p < 0.001$, Cohen's $d = 2.31$) and Textual Command ($p < 0.05$, Cohen's $d = 0.9$), indicating it is a more effective interaction method for making faster

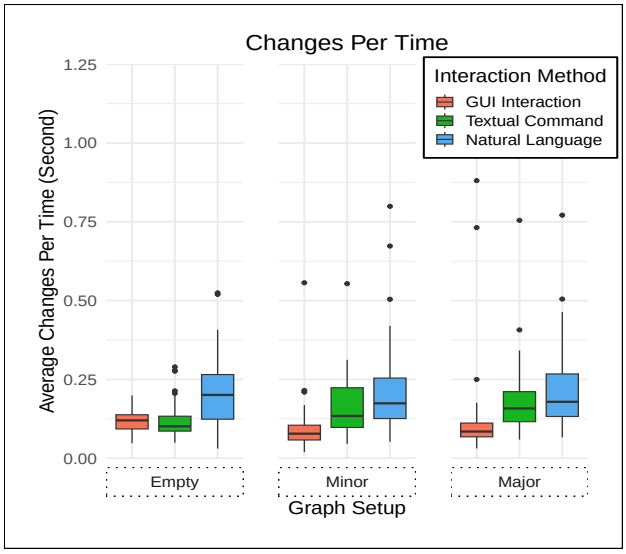

Figure 4: Average changes per time. Natural Language is significantly faster than all methods, and Textual Command outperforms GUI Interaction except in empty graph.

Changes Per Time	p-value	Effect Size
Main Effect of Interaction Method		
$F(2, 73) = 21.61$	< 0.001*	$\eta_p^2 = 0.37$
Post-hoc results		
Natural Language > GUI Interaction	< 0.001*	$d = 2.31$
Textual Command > GUI Interaction	< 0.001*	$d = 1.41$
Natural Language > Textual Command	0.03*	$d = 0.90$
Main Effect of Graph Setup		
$F(2, 146) = 1.08$	0.34	$\eta_p^2 = 0.01$
Post-hoc results		
Major > Empty	0.31	$d = 0.19$
Minor > Empty	0.80	$d = 0.08$
Major > Minor	0.69	$d = 0.11$
Main Effect of Graph Size		
$F(1, 73) = 18.90$	< 0.001*	$\eta_p^2 = 0.21$
Post-hoc results		
Large > Small	< 0.001*	$d = 0.45$
Interaction Effects		
Interaction Method × Graph Setup		
$F(4, 146) = 15.66$	< 0.001*	$\eta_p^2 = 0.30$
Interaction Method × Graph Size		
$F(2, 73) = 10.14$	< 0.001*	$\eta_p^2 = 0.22$
Graph Setup × Graph Size		
$F(2, 146) = 37.88$	< 0.001*	$\eta_p^2 = 0.34$
Interaction Method × Graph Setup × Graph Size		
$F(4, 146) = 4.64$	< 0.01*	$\eta_p^2 = 0.11$

Figure 5: ANOVA and Posthoc Tukey HSD test results for Interaction Method differences on changes per time (* indicates a statistically significant difference at $p < 0.05$)

changes. Also, Textual Command had significant higher *changes per time* than GUI Interaction ($p < 0.001$, Cohen's $d = 1.41$).

Additionally, the results also show a significant interaction effect between *interaction method* and *graph setup* (Figure 5), indicating that the impact of interaction methods on performance varies with the type of graph setup. For example, in the case of an empty *graph setup*, the test did not detect a significant interaction effect between the GUI Interaction and Textual Command. However, in the case of minor and major *graph setup*, the interaction effect was significant. Under these conditions, the Textual Command method significantly outperformed GUI Interaction, showing an advantage for relying on text commands when addressing major graph inaccuracies. Specifically, for both the small *graph size* ($p < 0.001$, $d = 1.46$) and the large *graph size* ($p < 0.01$, $d = 1.86$) in the major inaccuracies *graph setup*, Textual Command yielded significantly higher *changes per time*.

Although no significant difference was found between Natural Language and Textual Command in the minor and major *graph setup* with large *graph size*, a significant difference emerged for the empty *graph setup*, where Natural Language showed higher changes per time ($p < 0.01$, $d = 1.63$).

This interaction effect highlights the advantages of each method depending on the nature of the task. GUI Interaction proves more efficient for adding new information to the graph, as it allows for quicker visual adjustments without the need for multiple text commands, which can be cumbersome when dealing with repetitive inputs. However, for correcting major inaccuracies or modifying existing details, Textual Commands enable faster edits by letting users describe changes directly, avoiding manual selection, pop-up menus, and switching between keyboard and mouse.

4.2 Changes Per Action

The results for *changes per action* (Figures 6 and 7) show a significant main effect of *interaction method*, indicating clear differences in the number of changes achieved per action across methods. Post-hoc tests revealed that Natural Language resulted in significantly higher changes per action compared to both GUI Interaction ($p < 0.001$, Cohen's $d = 5.48$) and Textual Command ($p < 0.001$, Cohen's $d = 2.15$). This suggests that the Natural Language method enables users to accomplish more modifications per action, likely due to the ability to describe or command multiple changes at once rather than performing single, discrete actions as required in GUI Interaction. Furthermore, the Textual Command method also showed higher changes per action than the GUI Interaction method ($p < 0.001$, Cohen's $d = 3.33$). This indicates that Textual Command allows users to input several commands to modify the graph, whereas GUI Interaction typically involves more granular, individual actions.

Additionally, the analysis for *changes per action* found a significant interaction effect between *interaction method* and *graph setup* ($F(4, 365) = 23.38, \eta_p^2 = 0.2$). This effect indicates that the influence of interaction methods on performance varies depending on the type of graph setup. This is similar as what described for *changes per time* with the fact that it can be seen both for minor inaccuracies *graph setup* ($p < 0.001$, $d = 2.99$) as well as major inaccuracies ($p < 0.001$, $d = 2.81$) *graph setup* but only for large *graph size*.

5 DISCUSSION

5.1 Interpretation of Results

The study aimed to evaluate the trade-offs of different interaction methods with graph data across various graph complexity and graph setups. The results revealed that the Natural Language method consistently outperformed other approaches, particularly in terms of speed and the efficiency of implementing changes over

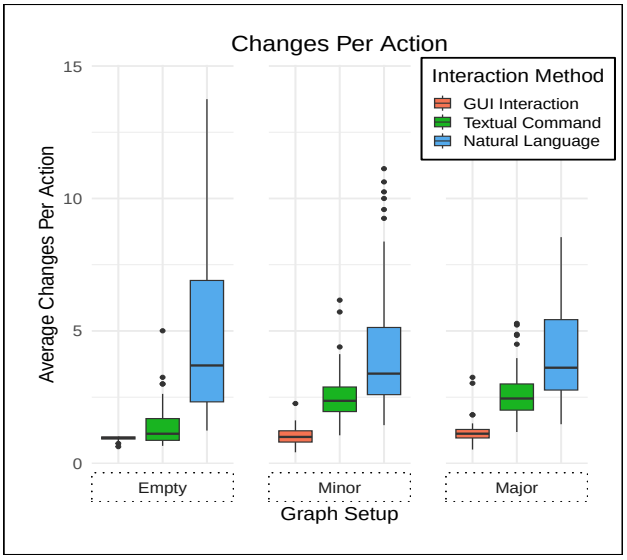

Figure 6: Average changes per action. Natural Language requires less effort, and Textual Command achieves more changes per action than GUI Interaction in all cases. Higher values reflect more graph changes per action, such as renaming in GUI or text inputs in other methods.

Changes Per Action	p-value	Effect Size
Main Effect of Interaction Method		
$F(2, 73) = 135.80$	< 0.001*	$\eta_p^2 = 0.79$
Post-hoc results		
Natural Language > GUI Interaction	< 0.001*	$d = 5.48$
Textual Command > GUI Interaction	< 0.001*	$d = 3.33$
Natural Language > Textual Command	< 0.001*	$d = 2.15$
Main Effect of Graph Setup		
$F(2, 146) = 9.93$	< 0.001*	$\eta_p^2 = 0.12$
Post-hoc results		
Empty < Major	< 0.001*	$d = 0.62$
Empty < Minor	< 0.001*	$d = 0.61$
Major < Minor	0.99	$d = 0.02$
Main Effect of Graph Size		
$F(1, 73) = 49.02$	< 0.001*	$\eta_p^2 = 0.40$
Post-hoc results		
Large > Small	< 0.001*	$d = 0.69$
Interaction Effects		
Interaction Method × Graph Setup		
$F(4, 146) = 17.46$	< 0.001*	$\eta_p^2 = 0.32$
Interaction Method × Graph Size		
$F(2, 73) = 14.65$	< 0.001*	$\eta_p^2 = 0.29$
Graph Setup × Graph Size		
$F(2, 146) = 58.89$	< 0.001*	$\eta_p^2 = 0.45$
Interaction Method × Graph Setup × Graph Size		
$F(4, 146) = 20.50$	< 0.001*	$\eta_p^2 = 0.36$

Figure 7: ANOVA and Tukey HSD results for interaction method differences in changes per action (* indicates $p < 0.05$). GUI actions include each operation (e.g., add node/edge); for other methods, actions refer to each text input submitted.

time. It is important to note that, although this was an online study, we conducted several iterations of quality checks and outlier removal, as detailed in Section 3.6.1, to ensure that the recorded time was logical and minimally affected by the inherent interruptions of online studies. As shown in Figure 4 and 5, which present the average number of changes over time, the Natural Language method enabled significantly faster modifications compared to other methods. This was especially evident when the graph was empty, indicating scenarios where new information needed to be added. Additionally, for graphs with minor and major inaccuracies, where the goal was to edit errors in addition to adding new information, the Natural Language method still achieved a higher rate of changes over time. However, the performance gap between Natural Language and other methods was narrower in these cases.

Figure 6 and 7, which display the average number of changes per action, highlight the relative difficulty users encounter when modifying the graph by taking advantage of multiple actions such as clicking or submitting text entry. The Natural Language and Textual Command particularly excelled in graphs with minor and major inaccuracies due to the efficiency of implementing multiple changes in each input. For example, a single input can include multiple operations and many details, which is not possible with a single execution through GUI Interaction. In contrast, for *empty graphs*, where users begin from scratch, no significant difference was observed between GUI Interaction and Textual Command. Nonetheless, the Natural Language method consistently outperformed both GUI Interaction and Textual Command across all scenarios.

Notably, after analysis of user inputs for the Natural Language method based on a manual review of all inputs logged from participants, we found that an average of 51% of inputs were text-based commands. Participants, despite having the flexibility to use various types of text input, frequently opted for Textual Commands when tasks involved editing information. Figures 4 and 6 illustrate this pattern, showing that while Natural Language usage is significantly higher than Textual Commands in the *empty* graph setup, the gap narrows in both the *minor* and *major graph setup*, reinforcing the observation. This supports use of text-based commands for editing purposes. The preference for Textual Commands likely stems from allowing precise edits without re-describing the entire scene. On the other hand, Natural Language allows them to describe their thoughts with the freedom of language they are comfortable with, resulting in an advantage when combined with textual commands.

5.2 Natural Language Use Cases in Different Application Domains

Considering the strong performance of natural language for editing knowledge graphs, this method has potential for broader applications. Integrating natural language interaction into data visualization [27, 35] marks a shift from traditional GUI-based approaches. While tools like [6, 11] enable flexible exploration, they often require navigating complex menus, switching visual forms, and mastering techniques like zooming or filtering. Natural language offers a simplified alternative by letting users issue commands such as "Show me a bar chart of sales data grouped by region" or "Highlight the most connected nodes." This reduces the effort needed to generate visualizations, making them more accessible to users. Additionally, natural language supports features like interactive annotations, allowing users to label visual elements with contextual information [30]. These annotations can also be interactive, enabling dynamic exploration and reinforcing feedback loops during data updates.

This aligns with our research goal of streamlining data interaction through natural and intuitive modalities.

Natural language interfaces are especially valuable in HITL applications, where users provide feedback to improve model performance [4, 8, 14]. These interfaces offer a way for users to update data or provide input without navigating complex interfaces. For example, users can request dataset updates or corrections as new information becomes available. Since our study used graph data from GQA dataset [12], commonly used in visual reasoning, it demonstrates how natural language interaction can support HITL tasks like data correction and annotation. Similar approaches can be extended to other data sources or AI-powered systems.

6 CONCLUSION

This paper demonstrates that natural language input offers an effective alternative to GUI-based graph editing, enabling flexible interaction with node-link visualizations. A user study confirms the benefits of supporting descriptive input. However, one limitation is that NL efficiency may be inflated due to design differences, as users could perform combined actions in a single input, unlike the step-by-step process required in the GUI (Figure 2). Also, the study was limited in the range of graph sizes and editing tasks due to the constraints of an online setting. Future work should explore larger graphs and diverse visual representations in addition to node-link diagrams.

ACKNOWLEDGMENTS

This was supported by the DARPA ECOLE Program HR00112390063.

REFERENCES

[1] Tyler Bikaun, Michael Stewart, and Wei Liu. 2024. CleanGraph: Human-in-the-loop Knowledge Graph Refinement and Completion. *arXiv preprint arXiv:2405.03932* (2024).

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[3] Michael Burch, Kiet Bennema Ten Brinke, Adrien Castella, Ghassen Karray Sebastiaan Peters, Vasil Shteriyanov, and Rinse Vlasvinkel. 2021. Dynamic graph exploration by interactively linked node-link diagrams and matrix visualizations. *Visual Computing for Industry, Biomedicine, and Art* 4 (2021), 1–14.

[4] Angelica Chen, Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Samuel R Bowman, Kyunghyun Cho, and Ethan Perez. 2024. Learning from Natural Language Feedback. *Transactions on Machine Learning Research* (2024).

[5] Philip R Cohen. 1992. The role of natural language in a multimodal interface. In *Proceedings of the 5th annual ACM symposium on User interface software and technology*. 143–149.

[6] Cody Dunne, Nathalie Henry Riche, Bongshin Lee, Ronald Metoyer, and George Robertson. 2012. GraphTrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1663–1672.

[7] Lisa A Elkin, Matthew Kay, James J Higgins, and Jacob O Wobbrock. 2021. An aligned rank transform procedure for multifactor contrast tests. In *The 34th annual ACM symposium on user interface software and technology*. 754–768.

[8] Alex Endert, M Shahriar Hossain, Naren Ramakrishnan, Chris North, Patrick Fiaux, and Christopher Andrews. 2014. The human is the loop: new directions for visual analytics. *Journal of intelligent information systems* 43 (2014), 411–435.

[9] Zuohui Fu, Yikun Xian, Yaxin Zhu, Shuyuan Xu, Zelong Li, Gerard De Melo, and Yongfeng Zhang. 2021. Hoops: Human-in-the-loop graph reasoning for conversational recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2415–2421.

[10] Nicolas Heist, Sven Hertling, Daniel Ringler, and Heiko Paulheim. 2020. Knowledge Graphs on the Web-An Overview. *Knowledge Graphs for eXplainable Artificial Intelligence* (2020), 3–22.

[11] Nathalie Henry, Jean-Daniel Fekete, and Michael J McGuffin. 2007. Nodetrix: a hybrid visualization of social networks. *IEEE transactions on visualization and computer graphics* 13, 6 (2007), 1302–1309.

[12] Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 6700–6709.

[13] Edwin L Hutchins, James D Hollan, and Donald A Norman. 1985. Direct manipulation interfaces. *Human–computer interaction* 1, 4 (1985), 311–338.

[14] Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016. Dialogue learning with human-in-the-loop. *arXiv preprint arXiv:1611.09823* (2016).

[15] Elizabeth D Liddy. 2001. Natural language processing. (2001).

[16] Michael J McGuffin and Igor Jurisica. 2009. Interaction techniques for selecting and manipulating subgraphs in network visualizations. *IEEE transactions on visualization and computer graphics* 15, 6 (2009), 937–944.

[17] Robert Munro Monarch. 2021. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster.

[18] Tamara Munzner. 2014. *Visualization analysis and design*. CRC press.

[19] William C Ogden and Philip Bernick. 1997. Using natural language interfaces. In *Handbook of human-computer interaction*. Elsevier, 137–161.

[20] OpenAI. 2024. ChatGPT. https://www.openai.com Large language model.

[21] Fabrizio Orlandi, Jeremy Debattista, Islam A Hassan, Clare Conran, Majid Latifi, Matthew Nicholson, Fahim A Salim, Daniel Turner, Owen Conlan, Declan O'sullivan, et al. 2018. Leveraging knowledge graphs of movies and their content for web-scale analysis. In *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, 609–616.

[22] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web* 8, 3 (2017), 489–508.

[23] Richard Power, Donia Scott, and Roger Evans. 1998. What you see is what you meant: direct knowledge editing with natural language feedback.. In *ECAI*, Vol. 98. 677–681.

[24] Krithi Ramamritham. 1993. Real-time databases. *Distributed and parallel databases* 1 (1993), 199–226.

[25] Krithi Ramamritham, Sang H Son, and Lisa Cingiser DiPippo. 2004. Real-time databases and data services. *Real-time systems* 28 (2004), 179–215.

[26] Johanna Schmidhuber, Stephan Schlögl, and Christian Ploder. 2021. Cognitive load and productivity implications in human-chatbot interaction. In *2021 IEEE 2nd International Conference on Human-Machine Systems (ICHMS)*. IEEE, 1–6.

[27] Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2022. Towards natural language interfaces for data visualization: A survey. *IEEE transactions on visualization and computer graphics* 29, 6 (2022), 3121–3144.

[28] Ben Shneiderman. 1983. Direct manipulation: A step beyond programming languages. *Computer* 16, 08 (1983), 57–69.

[29] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

[30] Arjun Srinivasan, Steven M Drucker, Alex Endert, and John Stasko. 2018. Augmenting visualizations with interactive data facts to facilitate interpretation and communication. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 672–681.

[31] Athanasios Theocharidis, Stjin Van Dongen, Anton J Enright, and Tom C Freeman. 2009. Network visualization and analysis of gene expression data using BioLayout Express 3D. *Nature protocols* 4, 10 (2009), 1535–1550.

[32] Christian Tominski, James Abello, Frank Van Ham, and Heidrun Schumann. 2006. Fisheye tree views and lenses for graph visualization. In *Tenth International Conference on Information Visualisation (IV'06)*. IEEE, 17–24.

[33] Frank Van Ham and Jarke J Van Wijk. 2004. Interactive visualization of small world graphs. In *IEEE Symposium on Information Visualization*. IEEE, 199–206.

[34] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.

[35] Yun Wang, Zhitao Hou, Leixian Shen, Tongshuang Wu, Jiaqi Wang, He Huang, Haidong Zhang, and Dongmei Zhang. 2022. Towards natural language-based visualization authoring. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2022), 1222–1232.

[36] Zijie J Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. 2021. Putting humans in the natural language processing loop: A survey. *arXiv preprint arXiv:2103.04044* (2021).

[37] Jacob O Wobbrock, Leah Findlater, Darren Gergle, and James J Higgins. 2011. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 143–146.

[38] Bowen Yu and Cláudio T Silva. 2019. FlowSense: A natural language interface for visual data exploration within a dataflow system. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 1–11.