Understanding (Sub)graphs through LLM Commentary and Query Entity Significance Scores

Maarten Drinhuyzen*† University of Amsterdam
Jasper van der Valk*‡ University of Amsterdam
David Werkhoven*§ University of Amsterdam
Xin Yu Zhu*¶ University of Amsterdam

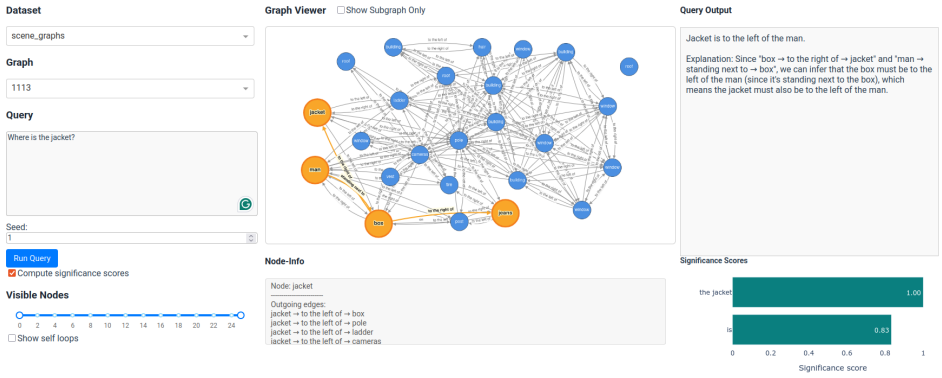


Figure 1: Visual interface front page with several interactive components, allowing the user to retrieve and understand graphs intuitively.

Abstract

Graph data has increasingly become more prevalent due to its capabilities of capturing complex structures and relationships between entities. However, querying graphs from such data remains non-trivial as it often requires knowledge of query languages. This, in turn, limits users who do not have a technical background. To address the issue, this study proposes an interactive and user-friendly visual system that allows users to retrieve graphs using natural language queries. Additionally, besides the free text graph querying component, the interface also provides users with the option to improve the interpretability of their graph queries. In particular, this is done by leveraging query term significance scores, which subsequently serve as phrase-level feedback over the input query.

The code is publicly available at: https://github.com/PirateEra/Multimedia-Analytics.git

1 Introduction

In recent years, there has been a rapid growth in network structures, such as biological, communication, social, and computer networks [19]. These networks can be represented as graphs [19], which are a powerful way to capture complex and relational data [5]. Given the growth and diversity of these data structures, an increased interest in effectively querying and data mining graphs has emerged in real-world applications [19].

However, querying graph data typically requires knowledge of query languages, which not only limits the accessibility for users without a technical background, but also restricts the users from obtaining valuable insights from the data [11]. This has raised the following question: How can we use architectures like large language models (LLMs) to allow users to query graph data without needing to learn a graph query language?

A recent study, LinkQ [11], addresses this problem by leveraging a pipeline in which an LLM analyzes the user's question and converts it into a knowledge graph query to retrieve the desired graphs.

In G-Retriever [8], the users are also able to query graphs using natural language, but use a Retrieval-Augmented Generation (RAG) approach instead for retrieval. In contrast with LinkQ [11], G-Retriever [8] is not limited to solely knowledge graphs, but also extends its applications to both scene graphs and explanation graphs (graphs for common sense reasoning).

In this work, we aim to extend upon G-Retriever [8] by delivering an interactive user-friendly interface (Figure 1) that not only allows for free text graph querying, but also provides a comprehensive explanation of why the LLM generated the final answer that it did. In addition, each noun, verb, and preposition phrase in the query also contains an indication score of how influential it was in generating the subgraph result, aiding the user to fine-tune their

*equal contribution
†e-mail: maarten.drinhuyzen@student.uva.nl
‡e-mail: jasper.van.der.valk@student.uva.nl
§e-mail: david.werkhoven@student.uva.nl
¶e-mail: xinyu.zhu2@student.uva.nl

queries if desired. Our main contributions can be summarized as follows:

- **User-friendly interface for free text graph querying:** an interactive interface that allows any user to query graph data in natural language, visualize the original graphs and their retrieved subgraphs, and answer the user's query using an LLM, accompanied by an explanation of how it got to that specific answer.
- **Improve the users' interpretability of graph queries:** phrase-level feedback over the user's query by providing significance scores of how relevant the phrase was for retrieving the final subgraph. Each noun, verb, and preposition phrase is also accompanied by a different subgraph, which is retrieved by modifying queries using a perturbation strategy based on lexical units.

In the following sections, we first outline, in Section 2, all the related work that we researched and took inspiration from in our proposed solution. Then, in Section 3, we thoroughly discuss our methodology, including the application overview, the utilized datasets, the LLM setup, and lastly, the entity-level perturbation. In Section 4, we categorize our interaction design into high-level and low-level interactions, and in Section 5, we introduce our implementation design of the created interface using Dash and Plotly. Finally, Section 6 refers to the conducted evaluation process on our MMA solution, and Section 7 finalizes the paper by summarizing the overall findings and noting the respective limitations.

2 Related Work

2.1 Multimedia Analytics

Multimedia Analytics (MMA) integrates interactive visualizations to enhance human-machine interactions. Zhalka et al. [18] developed a general multimedia model, connecting low-level methods with high-level analytical objectives through user interaction. They later built on this framework [15] to integrate with foundation models, emphasizing human-AI collaboration and enhancing transparency.

Combining MMA with Artificial Intelligence (AI) has recently gained more attention as MMA visualizations could be used to further understand, analyze, and compare various deep learning models [9]. In particular, researchers have utilized these visualizations to improve the explainability of deep learning models [9], making the models more understandable to their users. However, despite the focus on explainability in AI models, limited research has been conducted on the explainability of Graph Neural Networks (GNNs) through visualizations [9]. Therefore, Jin et al. [9] proposed GNNLens, which is a visual analytics system that consists of several visualization components, including 2D node projections and full graph visualizations. By displaying these visualizations, GNNLens not only makes the GNN and its predictions more understandable to its users, but also amplifies their cognition by reducing the information search [4].

2.2 Graphs in Retrieval-Augmented Generation

The field of RAG, first explored by Lewis et al. [10], focuses on the combined task of retrieval and question-answering, aiming to mitigate hallucination by retrieving relevant documents for response generation. Several works have researched the intersection of RAG and Graphs. *GraphToken* [13], is intended to create accurate graph presentations to improve graph-based question-answering. A variant of this is, *Graphormer* [16], a transformer-based Graph representation learner, achieving state-of-the-art performance. *G-Retriever* [8] builds on GraphToken, but also matches the query to a subgraph, using the PCST-algorithm, which is subsequently used to generate a response. The response is generated by a transformer-based LLM, which takes as input a graph token alongside a text embedding.

2.3 LLMs in Multimedia Analytics

With the rise of LLMs [3, 12], prompt engineering has gained increased interest for effectively using such models [7]. To assist users in refining their prompts, Feng et al. [7] provided PromptMagician, a visual analysis system that allows users to explore generated and retrieved results based on a given prompt, and refine their prompt based on the recommended prompt keywords from the prompt keyword recommendation model.

Similar to PromptMagician [7], LinkQ [11] also aims to refine user prompts on the visual interface, but utilizes an LLM for that process instead. In particular, LinkQ [11] uses the LLM to first interpret the user's question, followed by the conversion of the question into a well-formed knowledge graph query, then presents the user with the choice of modifying or executing the generated query, and finally summarizes the results after executing the knowledge graph query. By using this framework, any user's questions can iteratively be refined, which further facilitates both targeted and exploratory analysis of the data [11].

In G-Retriever [8], however, LLMs are not employed to refine the user's questions. Rather, they are used in the RAG approach to retrieve the relevant parts of a graph and generate a response based on the final retrieved subgraphs. Integrating LLMs into the framework enables users to query graphs in natural language and provides them with a flexible question-answering interface [8].

2.4 Graph similarity

In many applications regarding graphs, there has been a demand for a quantitative measure of graph similarities. Zager and Verghese [17] define the notion that two graphs are similar if their neighborhoods are similar. Specifically, an edge similarity score can be introduced, where an edge in one graph is similar to an edge in another graph if their respective source and destination nodes are identical [17].

3 Methodology

This section describes the method in greater detail. First, a high-level overview of the application is given. Afterward, the different parts are discussed in greater detail. The general pipeline is displayed in Figure 2.

3.1 Formalization

We define a global knowledge graph $\mathcal{G}$, from which all retrieved subgraphs are sampled.
Given a natural language prompt P, we extract a set of

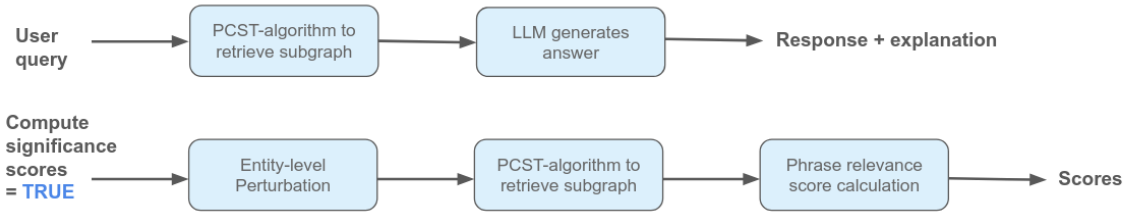


Figure 2: Pipeline of querying graphs using natural language queries and calculating query significance scores when needed. First, the PCST algorithm retrieves a subgraph based on the user's query. Then, the LLM takes the subgraph as input and generates an answer for the corresponding user query. The generated response is accompanied by an explanation of how the LLM arrived at that answer. Regarding the phrase significance values, the scores will only be calculated when the compute significance scores flag is set to TRUE. In that case, an entity-level perturbation is applied to the input query, after which the PCST-algoritm is utilized to retrieve a subgraph from each modified query version. After obtaining the subgraphs, the phrase significance scores can be calculated to present to the user.

query entities $S = E(P)$ using an arbitrary entity extraction function E, where each $s \in S$ denotes a lexical unit in P.

For subgraph retrieval, we employ the PCST algorithm [1], which takes P and $\mathcal{G}$ as input and returns a subgraph $\mathcal{G}' \subseteq \mathcal{G}$:

$$\mathcal{G}' = PCST(P, \mathcal{G})$$

We quantify the deviation between the original subgraph $\mathcal{G}'$ and the perturbed subgraph $\mathcal{G}_{-s}$ using the significance score:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Ultimately, we define the importance for entity $s \in S$ w.r.t. the subgraph $\mathcal{G}$' as the significance score:

$$\text{Significance}(s; \mathcal{G}', \mathcal{G}_{-s}) = 1 - J(\mathcal{G}', \mathcal{G}_{-s}) = 1 - \frac{|\mathcal{G}' \cap \mathcal{G}_{-s}|}{|\mathcal{G}' \cup \mathcal{G}_{-s}|}$$

The full procedure for computing these entity-level significance scores is described in our Algorithm 1.

Algorithm 1: Entity-Level Significance Scoring via Query Perturbation

Data: $P, \mathcal{G}$
Result: Entity significance scores
$\mathcal{G}' \leftarrow PCST(P, \mathcal{G})$;
$S \leftarrow E(P)$;
for $s \in S$ **do**
  $\mathcal{G}_{-s} = PCST(P_{-s}, \mathcal{G})$
  Significance$(s) \leftarrow 1 - J(\mathcal{G}', \mathcal{G}_{-s})$
end
return *Significance(s) for all* $s \in S$

3.2 Application Overview

Our work extends upon the g-retriever architecture. The flow of the application is as follows. First, the user selects a dataset and sample graph from that dataset on the top-left of the screen. This will visualize the full chosen graph in the middle-right of the screen. Subsequently, the user can write a prompt in the bottom-left of the screen, asking a question about the graph. This question is utilized by the Prize Collecting Steiner Tree Algorithm (PCST), the same one used in G-retriever [8], to create a subgraph based on the question (for reference, see step 3 in Figure 3).

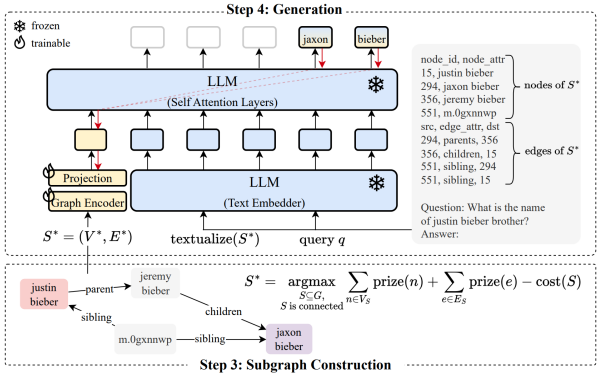


Figure 3: Step 3 and 4 of the g-retriever architecture, taken directly from its original paper [8].

This subgraph is then converted into a natural language string. The string is inserted at the start of a prompt template for context. Then this prompt is given to an LLM, which will infer the answer to the question based solely on the subgraph. The returned answer is then shown in a text box next to the subgraph.

Furthermore, when the subgraph is returned by the PCST algorithm, the visualized graph is zoomed in to highlight the retrieved subgraph. The subgraph then gets a color highlight, allowing the user to find it again when they move the camera away from it. Each noun phrase, verb phrase, and preposition phrase in the original question will also be highlighted with its contribution to the returned subgraph. This is done using significance score. There are also multiple interactions that can be utilized in the visualization. These are described in Section 4.

3.3 Datasets

The data that was utilized was retrieved from the GraphQA benchmark in G-retriever [8], which consists of the following three graph datasets: ExplaGraphs, SceneGraphs, and WebQSP. ExplaGraphs is a generative commonsense reasoning dataset, containing a total of 2,766 graphs. SceneGraphs is a visual question-answering dataset with 100,000 graph samples. Lastly, WebQSP is a knowledge graph question-answering dataset consisting of 4,737 graphs.

For triviality, we employed a subset of the original GraphQA benchmark for the interface. Specifically, we used one graph from the ExplaGraphs dataset and five graphs from each of the SceneGraphs and WebQSP datasets.

3.4 LLM Setup

The LLM is implemented using the Ollama Python package. This package allows for the utilization of quantized LLMs for personal computers. The LLM model used is Llama3.2:3b (https://ollama.com/library/llama3.2) The prompt structure can be seen in Figure 4. First the subgraph is added. Then the user's question is pasted in. Finally some miscellaneous instructions are given to make outputs more robust and consistent with each other.

```
<subgraph>
    ...
</subgraph/>

Based on the above graph only, and only use the above graph info.
Answer the following question: <userprompt> ... </userprompt/>\n
Keep in mind UnknownEntity_* represents unknown labels of nodes in
the graph, you may infer relationships through them.\n
if you think the graph does not provide the right or enough info
to answer, then mention that instead.\n
Start with your definite answer and after a very short brief
explanation on how you got the answer.\n
```

Figure 4: LLM prompt template given to the model in order to generate the output.

3.5 Entity-level Perturbation

Inspired by the Leave-one-token-out (LOTO) method [14], we employ a similar perturbation strategy based on lexical units. Specifically, for each provided user query, we remove one entity in the query, either a noun phrase, verb phrase, or preposition phrase, and recompute the corresponding subgraph using the PCST-algorithm. We employ the Python package *nltk* [2] for parsing the lexical units in the prompt (see Figure 5).

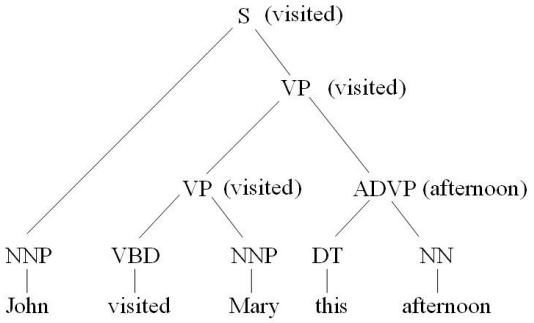


Figure 5: Example taken directly from [14]. Following our algorithm 1, the entities would be: {Noun phrase: "John", Verb phrase: "visited Mary", Preposition phrase: None}. Hence the perturbed prompts would respectively be: ["visited Mary this afternoon", "John this afternoon"].

As described in Algorithm 1, the significance scores are used to estimate the influence of each entity in the query on the retrieval of the final subgraph. This measurement was determined by considering the edges of graphs as sets, which can then be used to compute the similarity between them. [6]. The significance scores are finally calculated using the edges set of the original retrieved graph and the retrieved graph using the modified query with the one entity removed.

4 Interaction Design

In order to achieve this transparency and interpretability, we ensure our visualizations are interactive. We classify these interactions into high-level and low-level interactions.

- **High-level interactions:** We want to let users explore and browse the subgraph to ensure maximum transparency. This is part of the *"Browse"* high-level user goal.
  Another high-level user goal is *"Assess"*. This will allow users to assess the correctness of the output query based on the visualized sub-graph with highlights and the input query. Users will also be able to compare the effectiveness of different input prompts by using the significance score. This last part belongs to the *"Compare"* high-level goal.

- **Low-level interactions:** We add zooming and panning to the subgraph. This will ensure that graphs of arbitrary size can be viewed and understood by the user. The nodes can also be moved by the user. These interactions are part of the *"Explore"* class. A slider is included to show/hide a certain number of nodes. If the graph contains many nodes, showing all of them to the user could overwhelm the user or make the graph unreadable. This interaction is part of the *"Abstract/Elaborate"* high-level interaction class and the *"Filter"* low-level task. Also added is a toggle to show/hide the self-loops of the graph, as this clutters the visualization of the sub-graph but might be useful on special occasions. Finally, there is also a button to show only the subgraph, allowing for easier

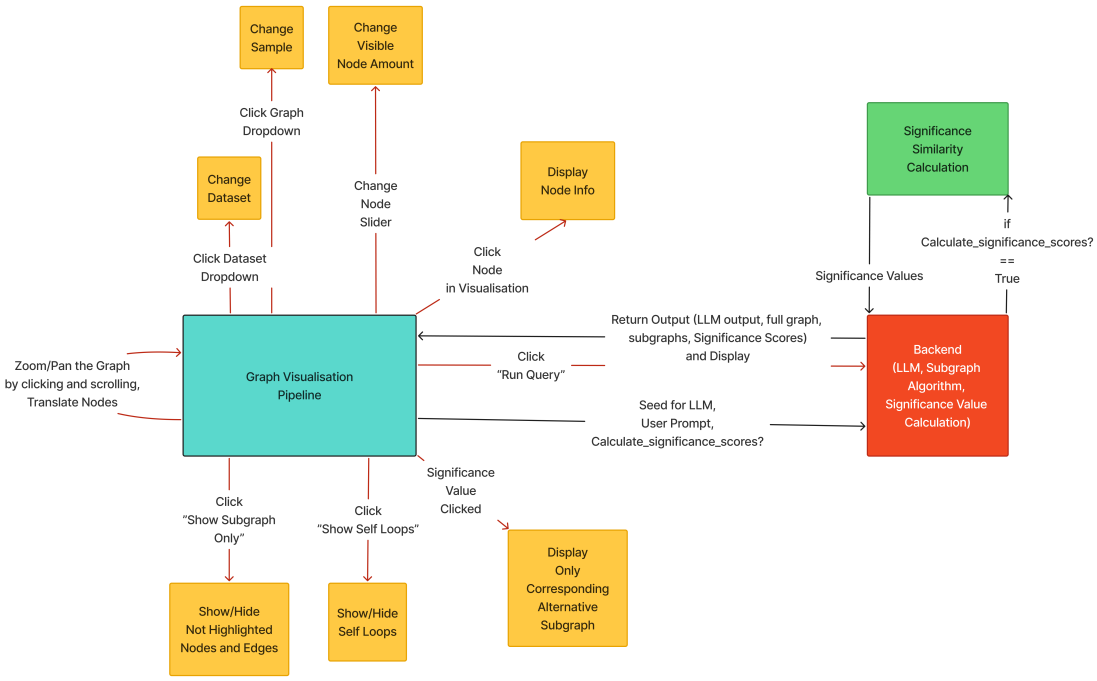


Figure 6: Interaction Pipeline, showing possible interactions within our visualization.

viewing. This is also part of the *"Filter"* low-level task.

Furthermore, when a node is clicked, info about the node and connected nodes will be shown. This will allow users to more accurately navigate through the graph in order to base their prompts on it. This is part of the *"Retrieve Value"* low-level task.

As such, the following interactions are added to the visualization and can be seen in Figure 6. The figure can be read as follows. The blocks in red are part of the backend. The blocks in green are simple functions to calculate values. The blocks in yellow can be seen as code blocks that will update or change the visualization in a specific way. The blue block in the middle is the visualization to be displayed. Finally, red arrows are manually activated by the user, while black arrows are automatic.

5 Implementation Design

To visualize and interact with the graph-based results, we choose to build our interface using Dash and Plotly. Dash is used for interactive web applications, which is perfect for both data visualization and dynamic user interaction. At the center of our architecture, we make use of Dash Cytoscape, which is a component for rendering and interacting with graph structures. We use this to visualize the subgraph returned by G-Retriever's PCST-algorithm [8]. Nodes and edges in the graph are styled based on their attributes and connection to the retrieved subgraph. This is done using visual properties such as color, text, and node size. These style differences between nodes and edges will, in turn, help the user interpret which graph elements were most relevant to the query.

The interface is organized horizontally across a single page, with three main sections. The left-hand side include dropdowns for choosing a dataset and graph, a text input form for submitting a natural language query and desired seed, a button that initiates the subgraph retrieval and its visualization process, a threshold bar for selecting the number of nodes, a toggle widget for calculating the significance scores, and finally, a toggle widget for displaying the self-loops on or off.

The center part of the interface visualizes the retrieved subgraph with the use of a Cytoscape component. Dash callbacks dynamically update the graph whenever the user submits a new query, changes the visible nodes threshold, or selects one of the left-out verb, noun, or preposition phrases (only visible if the toggle widget for calculating the significance scores is on).

On the right-hand side, we include a query output text box that displays the generated answer to the given query and provides an additional, extensive explanation of how the LLM arrived at that particular answer or conclusion. If the user decides to calculate the significance scores, the query phrase significance scores will also be displayed, but below the LLM output in a bar plot figure.

Ultimately, all components are connected through Dash's callback system, which ensures a seamless, interactive singular page. This architectural design should support the intuitive use of the interface, assisting users in re-

trieving graphs effectively and aiding them in refining their queries when desired.

6 Evaluation

We aim to evaluate the increase in the users' fluency in interpreting and interacting with graph-based language. Therefore we conduct experiments on users to assess the visualization's contribution to human learning. We measure the zero-shot satisfaction rate: the cases in which the system's initial response is satisfactory to the user (binary judgment). If the initial output is unsatisfactory, our interface provides visual feedback that enables the user to make a single, informed modification. In our experiments, we refer to the latter as one-shot satisfaction rate. The one-shot satisfaction rate includes the zero-shot cases and reflects whether the user was satisfied after at most one interaction with the system. The efficacy of our visualization is substantiated by the increase in first-shot satisfaction rates.

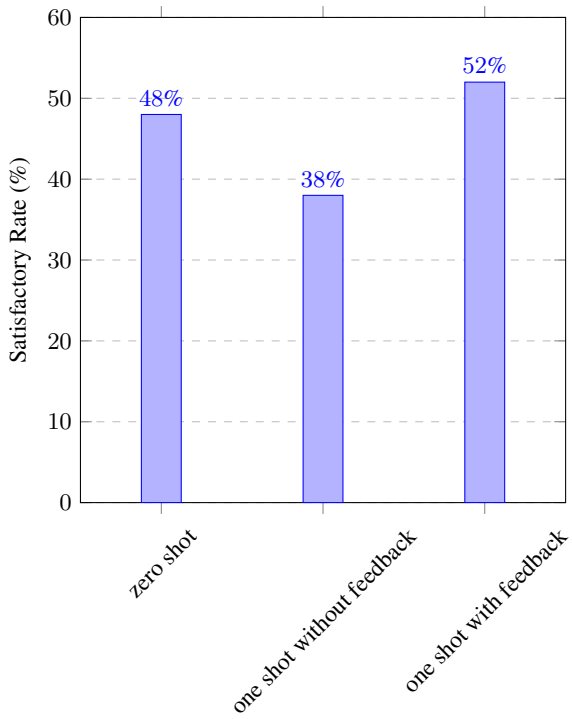
The experiments were conducted with 4 users, each completing 25 trials, resulting in a total of 100 trials. We split the population in two equal groups: the group **without feedback** and the group **with visual feedback** (see Table 1).

Table 1: Participants and trials per group, including study-wide totals.

Group	Participants	Total Trials
No Feedback	2	50
Visual Feedback	2	50
Total	**4**	**100**

In both cases, users were shown a graph per trial and asked to formulate a natural language question. Users can prompt the system twice. If the response is initially satisfactory that contributes to the zero-shot satisfaction rate, if a second prompt was needed and the response was then satisfactory, it contributed to the one-shot satisfaction rate. The group without feedback is presented with a static image of the full graph. The group with visual feedback, on the contrary, used our interactive interface with visual and phrase-level feedback.

The zero- and one-shot satisfaction rates were measured over all trials (see Figure 6). All participants were contributors to this research.



Distribution of user satisfaction rates for zero-shot (combined with and without feedback) and one-shot (seperated) interactions in our user study (n=100, zero-shot=50, one-shot-with-feedback=25, one-shot-without-feedback=25).

7 Discussion and Conclusion

This research aims at improving the interpretability and steerability of graph querying languages and enhancing human understanding. We augment the previous work on G-Retriever [8], by visualizing both the subgraph and query term significance, leveraging ideas from Entity-level perturbation strategies [14].

We implemented an interactive user interface, leveraging Dash and Plotly, enabling users to *browse*, *assess*, and *compare* graph data using natural language.

In our study, we observed a 14% increase in one-shot satisfaction rates for the group that used our visual interface, compared to the control group without visual feedback (See Figure 6). Participants using the visual interface were actually able to get a satisfactory response from the graph retriever 41 out of 50 times, suggesting that the visual feedback not only improved satisfaction rates but it also gave relevant insights.

However, to draw stronger conclusions, additional experiments with a larger sample are required, as the sample in the current experiments is small and potentially biased. Furthermore, given the novelty of graph query languages complemented by Multimedia Analytics, no relevant benchmarks exist to compare our performance. Therefore, this work serves as a baseline for future research in this direction. This places our work in quite a niche but novel direction. Unlike similar works, such as LinkQ [11] or RagEX [14], which focus on visualizing query term significance scores in graph-based retrieval set-

tings, our approach connects these significance scores to the retrieved subgraph. In summary, this work contributes a novel interactive interface for graph-based natural language querying and introduces subgraph-level phrase significance visualization specifically for graph retrieval. Our findings suggest that our interface can improve one-shot user satisfaction and simultaneously provide insights into graph querying.

Future research could focus on optimizing perturbation strategies to maximize output variance in response to query token manipulations, thereby enhancing the sensitivity and interpretability of phrase importance scores. Additionally, another promising direction is to further explain how the retrieved subgraph influences the LLM's output, accompanied by visualizations for interoperability.

Ultimately, we hope this work serves as a foundation for future research towards more explainable, steerable, and user-friendly graph querying systems, in the broader field of Multimedia Analytics.

REFERENCES

[1] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Mathematical programming*, 59(1):413–420, 1993. 3

[2] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. 01 2009. 4

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2

[4] S. K. Card, J. Mackinlay, and B. Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999. 2

[5] R. Das and M. Soylu. A key review on graph data science: The power of graphs in scientific studies. *Chemometrics and Intelligent Laboratory Systems*, 240:104896, 2023. 1

[6] A. Fender, N. Emad, S. Petiton, J. Eaton, and M. Naumov. Parallel jaccard and related graph clustering techniques. In *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, pp. 1–8, 2017. 4

[7] Y. Feng, X. Wang, K. K. Wong, S. Wang, Y. Lu, M. Zhu, B. Wang, and W. Chen. Promptmagician: Interactive prompt engineering for text-to-image creation. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):295–305, 2023. 2

[8] X. He, Y. Tian, Y. Sun, N. V. Chawla, T. Laurent, Y. LeCun, X. Bresson, and B. Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, eds., *Advances in Neural Information Processing Systems*, vol. 37, pp. 132876–132907. Curran Associates, Inc., 2024. 1, 2, 3, 4, 5, 6

[9] Z. Jin, Y. Wang, Q. Wang, Y. Ming, T. Ma, and H. Qu. Gnnlens: A visual analytics approach for prediction error diagnosis of graph neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 29(6):3024–3038, 2022. 2

[10] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020. 2

[11] H. Li, G. Appleby, and A. Suh. Linkq: An llm-assisted visual interface for knowledge graph question-answering. In *2024 IEEE Visualization and Visual Analytics (VIS)*, pp. 116–120. IEEE, 2024. 1, 2, 6

[12] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022. 2

[13] B. Perozzi, B. Fatemi, D. Zelle, A. Tsitsulin, M. Kazemi, R. Al-Rfou, and J. Halcrow. Let your graph do the talking: Encoding structured data for llms, 2024. 2

[14] V. Sudhi, S. R. Bhat, M. Rudat, and R. Teucher. Rag-ex: A generic framework for explaining retrieval augmented generation. pp. 2776–2780, 07 2024. doi: 10.1145/3626772.3657660 4, 6

[15] M. Worring, J. Zahálka, S. van den Elzen, M. T. Fischer, and D. A. Keim. A multimedia analytics model for the foundation model era, 2025. 2

[16] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. Do transformers really perform bad for graph representation?, 2021. 2

[17] L. A. Zager and G. C. Verghese. Graph similarity scoring and matching. *Applied mathematics letters*, 21(1):86–94, 2008. 2

[18] J. Zahálka and M. Worring. Towards interactive, intelligent, and integrated multimedia analytics. In *2014 IEEE conference on visual analytics science and technology (VAST)*, pp. 3–12. IEEE, 2014. 2

[19] P. Zhao and J. Han. On graph query optimization in large networks. *Proceedings of the VLDB Endowment*, 3(1-2):340–351, 2010. 1