GVR: A Recommendation Tool for Knowledge Graph Visualizations

A Data-Driven System to Knowledge Graph Visualization through Adaptive Learning

Sjoerd Vink

A thesis presented for the Master of Science degree in Applied Data-Science

Supervisor: Dr. Michael Behrisch
Second examiner: Prof. dr. ir. Alex Telea
External partner: Prof. dr. Remco Chang

Utrecht University

Department of Information and Computing Sciences
Utrecht University

The Netherlands
July 28, 2024

Abstract

Knowledge graphs, which represent complex data through nodes and edges, offer immense potential for analysis but pose challenges in understanding desired outcomes. Visualizations serve as a crucial tool that enhances accessibility to knowledge graphs, yet their design demands expertise in data understanding and visualization design. Recommendation systems for knowledge graph visualizations aim to lower the barrier for non-expert users in the process of information discovery by autonomously recommending and constructing (graph) visualizations from data. We introduce GVR (Graph Visualization Recommender), a system that aims to bridge the gap between raw knowledge graph data and informative visual representations, facilitating efficient analysis and decision-making while paving the way for future advancements in this emerging field. The effectiveness of our approach was evaluated using generated sample data, highlighting its potential to recommend appropriate visualizations based on user interactions.

The code is available at: graphpolaris/visualization-recommendation

Keywords

Knowledge graphs, Visualization recommendation

Contents

1 Introduction **3**

2 Background **4**

3 Related Work **5**
- 3.1 Descriptive Graph Statistics 5
- 3.2 Graph Representations 6
  - 3.2.1 Node-Link Layouts 6
  - 3.2.2 Tabular Layouts 7
  - 3.2.3 Implicit Layouts 7
- 3.3 Recommendation Systems 7
  - 3.3.1 Rule-Based 8
  - 3.3.2 Machine-Learning-Based 8

4 Recommendation Model **9**
- 4.1 Decision Space 9
- 4.2 Conceptual Overview 10
- 4.3 Data Structures and Storage 11
- 4.4 Algorithmic Explanation 11
- 4.5 Updating Scheme 12

5 Evaluation **13**
- 5.1 Use-Case Scenario 13
- 5.2 Technical Evaluation 16
  - 5.2.1 Experimental Setup 16
  - 5.2.2 Policy Effectiveness 17
  - 5.2.3 Convergence 18
  - 5.2.4 Stability 19

6 Discussion and Future Work **19**

7 Conclusion **22**

A Data Generation **26**

B Implemented Code **27**
- B.1 Recommend Visualization Function 27
- B.2 Update Function 27

C Project plan **29**
- C.1 Goals 29

1 Introduction

In the era of big data, more information becomes available with increasing complexity. Among the myriad of strategies devised to manage such complexity, knowledge graphs (KGs) have emerged as a powerful paradigm. KGs transform data into semantically rich narratives that underpin modern business applications by encapsulating information in nodes and edges. They facilitate differentiating analysis and decision-making in a complex world while offering key advantages such as schema flexibility and seamless data integration [1].

Yet, the full potential of KGs remains underutilized, partly due to the steep learning curve associated with their analysis. Visualizations enhance accessibility to KGs by alleviating the cognitive load required for extracting insights [2]. They enable analysts to discern complex patterns with greater ease and serve as a tool to extract information beneficial to research, business operations, and beyond. Nonetheless, designing KG visualizations is a multifaceted endeavor that demands expertise in data understanding, objectives, and visualization techniques at hand. Even experts often find it challenging to determine the appropriate visualization for a given purpose and context.

Most visualization tools rely on users' manual specification [3], an often challenging task for non-experts. Visualization recommendation systems (VRS) represent a relatively new research area that aims to democratize data exploration by automating the visualization process. A successful VRS must proficiently handle various sub-tasks: i) understanding the semantics of the data, ii) identifying visualization objectives, and iii) producing visualization specifications that fulfill the syntax, design, task, and perceptual criteria associated with these objectives [4]. Various VRSs currently exist, with some employing rule-based approaches, while others rely on machine-learning (ML) techniques.

Despite advancements, current VRSs predominantly focus on tabular data and neglect KGs. Recommending visualizations for KGs is inherently more challenging due to various reasons. Firstly, tabular data has a fixed number of dimensions, whereas KGs can vary widely in attributes and degrees, making the underlying data structure far more complex. Secondly, tabular data typically has a fixed schema, while KGs lack a fixed schema and can evolve. Additionally, relationships in tabular data are often implicit, whereas in KGs, edges are explicitly defined. Finally, the semantics in tabular data are usually limited to column names and values, while KGs have nodes and edges rich in semantic information that needs to be conveyed. Although there are existing VRSs for KG data, they are often too rigid and potentially have too limited input dimensions to base their decisions on. To our knowledge, no attempts have been made to extend existing VRS methodologies to fully address the unique challenges posed by KGs.

The major contributions of this paper are:

- GVR (Graph Visualization Recommender), a feedback-driven visualization recommendation tool for KGs that incrementally converges through continuous learning based on user interactions.
- A library that extracts a range of descriptive graph statistics from the result set, which can be used as input for the visualization recommendation system.

To the best of our knowledge, GVR is the first dynamic knowledge graph visualization recommendation tool available. By offering these contributions, this paper not only addresses a significant gap in current VRS research but also provides a foundation for future endeavors to build upon. Ultimately, this work paves the way for more intuitive and accessible KG analysis tools.

2 Background

A KG visualization is a visual representation of the nodes and edges of a given KG. The arrangement of these nodes and edges within visualizations significantly impacts their understandability, usability, and aesthetics [5, 6]. However, selecting an effective visual representation is a non-trivial task; it involves navigating the design space at both the visualization and encoding levels [7]. At the visualization level, design choices typically involve high-level decisions, such as determining which layouts to employ. In contrast, encoding-level design choices pertain to more granular aspects, such as selecting a color encoding to represent data attributes.

We focus on a scenario where an analyst interacts with a KG database $G(V, E)$ to extract insights through visualizations and subsequently uses these insights for downstream tasks. Each node and edge in the KG can have multiple attributes, including textual, quantitative, ordinal, nominal, or set data [8]. Upon querying the database, the analyst receives a subset of the original KG as a result set. This subset, denoted as $G' = (V', E')$ (where $V' \subset V$ and $E' \subset E$), forms the basis for subsequent analysis.

The process of transforming the KG subset G' into a visual representation $V(G', D)$ involves a set of design choices D. Herein lies the core problem: not all visual representations derived from the myriad design choices D are equally valid or useful. The challenge lies in optimizing the design choices within $V(G', D)$ to ensure that the visualizations accurately and effectively convey the intended insights from the KG.

The goal of a VRS for KG data is to provide the analyst with a recommendation comprising a ranked set of N valid visualizations, denoted as

$V_1, V_2, ..., V_N$. This recommendation aims to facilitate the analysis process by presenting visualizations that effectively convey relevant insights from the KG. The effectiveness of the recommendation can be evaluated based on its ability to fulfill specific analysis tasks efficiently. By implementing a recommendation process, we aim to assist analysts in transitioning from the raw KG data $G(V, E)$ to informative visual representations $V(G', D)$ tailored to meet specific analysis objectives.

3 Related Work

GVR builds on advancements in three key areas: descriptive graph statistics, graph representations, and visualization recommendation systems. The section follows a VRS pipeline, beginning with descriptive graph analytics as the input, progressing to graph representation as the output, and finally addressing visualization recommendation systems to integrate these elements.

3.1 Descriptive Graph Statistics

Numerous measures exist to characterize KGs. Hernandez and Van Mieghem [9] broadly categorize these measures into distance metrics, connection metrics, and spectra metrics. Distance metrics, including closeness, eccentricity, diameter, and radius, help in understanding proximity and reachability within the graph. Connection metrics, such as degree, assortativity, coreness, cliques, and clustering, provide insights into the connectivity and cohesiveness of the graph. While spectra metrics are significant in mathematical analysis, they are not as crucial for our purposes due to their abstract nature and indirect relationship with visual patterns, making it challenging to translate these metrics into visual cues.

Kolaczyk and Csárdi [10] categorize metrics into node and edge characteristics, network cohesion, network partitioning, and assortative mixing. Node and edge characteristics include degree and centrality measures such as closeness and betweenness. Network cohesion involves subgraphs and censuses, density and related notions, and connectivity, cuts, and flows. Graph partitioning, also known as community detection, includes hierarchical clustering and spectral partitioning. Assortative mixing is measured using assortativity coefficients, such as degree assortativity.

The need for a statistical abstraction of the KG arises from the complexity and richness of the data represented within the result set. Such abstraction simplifies the recommendation problem by describing every possible KG in a finite set of integers ($S_{G'} = \{S^1_{G'}, S^2_{G'}, ..., S^n_{G'}\}$). These abstractions help bridge the gap between raw KG data and effective visual representations. GVR builds upon existing methods by focusing on the strategic combination of these statistics to offer a comprehensive understanding of both the data and its semantics. Future research is needed to determine which of these statistics actually influence the prediction, as this can only be identified once our solution begins to converge. This perspective resonates with [4], which underscores that the initial sub-task of a VRS involves comprehending the semantics of the data.

3.2 Graph Representations

We were intrigued by the state-of-the-art report from Nobre et al. [8] and wanted to automate it, making it generally useful for a wide audience. This section delineates the various categories and choices D in $V(G', D)$, categorizing the KG visualization space in which predictions are made. KG visualization can be categorized as explicit or implicit [11, 8]. Explicit visualizations portray nodes and edges directly, while implicit visualizations rely on the node's placement to encode edges. Explicit layouts are further divided into node-link and tabular layouts. We acknowledge that while this set of visualizations captures the majority of the KG visualization space, it does not encompass all possibilities as the field is continuously evolving. However, GVR is designed to be flexible, allowing for the easy addition of new visualization types to accommodate future advancements and emerging trends.

3.2.1 Node-Link Layouts

Node-link layouts (Figure 1) are a prevalent method to visualize KGs, with substantial research dedicated to this approach [12, 13, 14]. Schulz and Schumann [11] defines three categories in the node-link paradigm: free, styled, and fixed layouts. Generally, node layouts are unrestricted (free layouts) and can be placed by various algorithms. Conversely, fixed layouts require specific node placement, such as in map visualizations where the nodes correspond to geographical locations. Between free and fixed layouts, there are less restricted styled layouts. Nobre et al. [8] narrow this categorization down to topology-driven layouts and attribute-driven layouts.

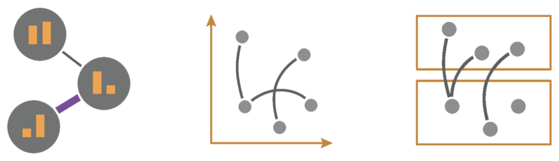


Figure 1: Node-Link layouts. From left to right: topology-driven, attribute-driven positioning, attribute-driven faceting [8].

3.2.2 Tabular Layouts

Tabular layouts (Figure 2) organize nodes into rows and columns, with edges represented as cells. This format allows for the encoding of node attributes by aligning attributes with the tabular layout and edge attributes through cell encoding. The most common tabular representation of a KG is an adjacency matrix [15]. In an adjacency matrix, each column and row represents a node, and each value in row i and column j indicates the presence of an edge from node i to node j [16]. Other examples of tabular layouts are PAOH (Parallel Aggregated Ordered Hypergraph) [17], quilts [18] and BioFabric [19].

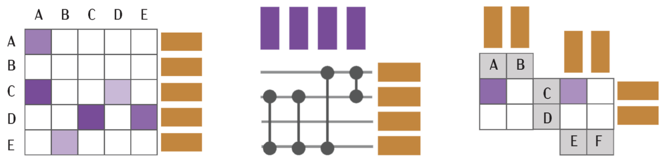


Figure 2: Tabular layouts. From left to right: adjacency matrix, BioFabric, quilts [8].

3.2.3 Implicit Layouts

Implicit layouts (Figure 3) are primarily used for trees [8]. This type of layout typically allows for the visualization of only two node attributes, using color and size. Since edges are encoded implicitly, it is not possible to show their attributes. Within this category, visualizations either display all inner nodes and leaves or only the leaves. Examples of implicit layouts are treemaps [20] and sunbursts [21].

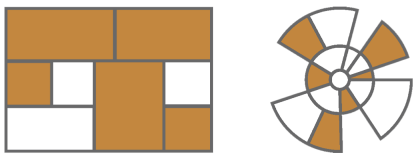


Figure 3: Implicit layouts. From left to right: treemap, sunburst [8].

3.3 Recommendation Systems

VRSs integrate input G' and output $V(G', D)$, effectively guiding the selection of design choices D to generate useful visual representations. Two predominant paradigms exist in this domain: rule-based and machine-learning-based approaches.

3.3.1 Rule-Based

Rule-based VRSs are primarily based on manually specified rules. For instance, APT [22], BOZ [23] and SAGE [24] generate ranked visualizations based on rules inspired by perceptual principles influenced by Bertin [25] and Clevelend & McGill [26]. However, these systems focus primarily on how to communicate graphically rather than what to convey. More recent systems like Tableau's Show Me[27] and Voyager [28, 29] extend these with support for column selection. There are three main drawbacks to rule-based recommendation systems [4, 7]. First, constructing rules is labor intensive. Second, visualization often requires modeling complex relationships that are difficult to capture with simple rules. Finally, as input and output dimensions grow, the combinatorial nature of the rules results in an explosion of possible recommendations [4].

Despite significant advancements in VRSs, KG visualizations lack dedicated recommendation systems. Specialized tools like TreePlus and NetLens are designed for specific topological structures within KG visualization, focusing on nodes and groups respectively [30, 31, 8]. To the best of our knowledge, there are currently no dedicated recommendation systems for KG visualization that cover all topological structures. The work of Nobre et al. [8] on multivariate KG visualization stands as the closest semblance to a recommendation system for KG visualizations. Serving as a state-of-the-art report, they focus on categorizing multivariate KG visualizations. Each visualization category is scored regarding how well they fit certain KGs by the four authors. However, Nobre et al. [8] do not account for directedness, weightedness, spatiality/temporality, self-loops, and parallel edges. GVR extends their framework by allowing the incorporation of these characteristics, along with a comprehensive range of descriptive statistics discussed in the previous section.

3.3.2 Machine-Learning-Based

Machine-learning-based approaches aim to overcome the limitations of rule-based methods by employing algorithms that learn directly from data. For example, VizML [7] utilizes the Plotly community corpus to recommend visualizations tailored to specific data sets. Similarly, DeepEye [32] recommends visualizations by interpreting query intent to recommend and rank effective visualizations. Draco [33] takes a more structured approach by enumerating the visualizations that do not violate hard constraints while optimizing for those preferred under soft constraints. Addressing transparency, KG4Vis [3] tries to overcome the black-box problem in other ML-based VRSs by making the ML-constructed rules explainable with KGs and embeddings. CompasQL [34] introduces a comprehensive framework for developing VRSs in the form of a specification language for querying over the space of visualizations. Lastly, LIDA [2] formulates visualization generation as a multi-step generation task, introducing an end-to-end system based on large language models that addresses a variety of sub-tasks, including data summarization, automated exploration, and visualization generation.

All these systems perform offline learning on large datasets. GVR, however, advances this by incorporating reinforcement learning to adapt and improve over time through user interaction. This methodology is particularly beneficial because the rules for selecting appropriate visualizations for KGs are relatively unexplored and not well understood. By employing reinforcement learning, we can iteratively develop a set of rules, refine features, and gain valuable insights into how users interact with graph visualizations.

4 Recommendation Model

We present GVR, a feedback-driven visualization recommendation system for KG data that utilizes reinforcement learning to construct recommendation policies. Our tool enhances the interpretability and usability of complex KGs through automated visualization recommendations, effectively extracting insights from G' with $V(G', D)$ for downstream tasks.

4.1 Decision Space

The decision space defines the scope of input and output dimensions that the recommendation system evaluates and operates upon. It encompasses all the factors and parameters that influence the system's decision-making process; i.e., which visualization should be recommended given the input parameters.

The input parameters to the recommendation model comprise a set of descriptive statistics ($S_{G'} = S^1_{G'}, S^2_{G'}, ..., S^n_{G'}$) derived from the result set G'. Nobre et al. [8] use a relatively small set of statistics for their recommendations, such as KG type, size, and node/edge heterogeneity. GVR extends this set by adding a more descriptive range of additional statistics derived from the result set. These include general KG measures such as degree and centrality, as well as complex topological metrics like partitioning and cohesion measures. Additionally, a domain is added as input to the recommendation function because we claim that KG visualization is heavily domain-dependent. GVR learns a separate model for each domain, allowing for the unique needs of each domain to be effectively addressed.

By incorporating a more extensive and descriptive set of statistics, GVR enhances our ability to understand and represent the underlying data. The selection of these statistics was made relatively arbitrarily, aiming to be as comprehensive as possible to capture the diverse aspects of the KG. This broad approach allows us to encompass a wide range of information about the KG. As our solution matures and begins to converge, we can refine the feature set by identifying and excluding statistics that may not significantly influence the predictions.

The goal of GVR is to provide guidance in the KG visualization design process. Hence, the output consists of the set of design choices D in $V(G', D)$, tailored to perform analysis tasks within a given domain. The output dimensions can be easily added to the model. This gives our approach an advantage over rule-based systems, where extending the output dimension is problematic. This framework can also be enhanced to include more granular encoding in the output labels, such as using multiple labels with different color encoding settings within the same visualization.

4.2 Conceptual Overview

Our data processing and prediction pipeline is split into four parts, 1) data source analysis, 2) graph feature extraction, 3) (domain) model building and update, and 4) recommendation generation. We illustrate the processing flow in Figure 4.

Descriptive graph statistics as the input to our recommender system: In a typical KG database, a subset G' of the original KG is returned upon querying. This subset represents the information for subsequent analysis. G' can be described as a set of statistics $S_{G'} = S^1_{G'}, S^2_{G'}, ..., S^n_{G'}$, capturing essential attributes of the KG subset. Together with the user domain, these statistics serve as the foundational input for the recommendation process. Leveraging both domain-specific knowledge and the extracted statistics, the system engages in the predictive task of selecting the most suitable visualization $V(G, D)$ that best represents subset G'.

Reinforcement learning to capture user/domain-feedback: Reinforcement learning is a machine-learning technique in which an agent interacts with an environment and learns from the rewards it receives based on its actions. The agent maintains a memory in the form of a Q-table, where it accumulates and refines knowledge through scores assigned to each action (visualization choice) for each state id (descriptive statistics of the graph). In GVR, the agent is the recommendation model, and the environment is the context in which the model is used. Rewards are implicitly provided by users through a scoring system, and the Q-table is updated accordingly. The ability of GVR to learn and adapt over time is central to the system's functionality. As users interact with the recommended visualizations, their implicit feedback helps refine the recommendation process. This adaptive learning process ensures that the system remains responsive to changing user needs and evolving data characteristics.

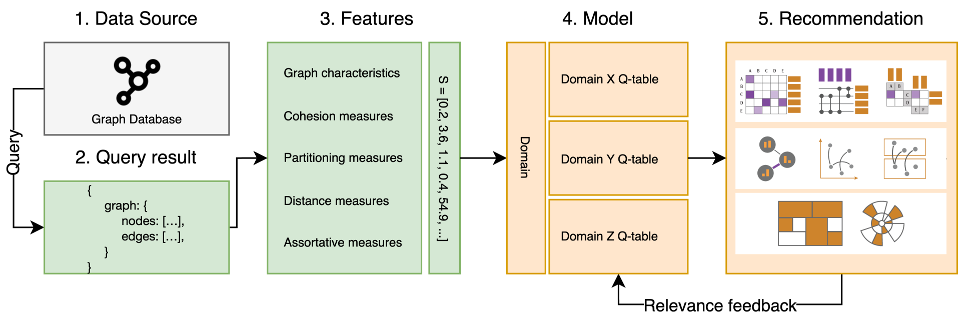


Figure 4: Using descriptive KG statistics and a domain as input for our recommender model enables GVR to incorporate domain-specific considerations.

4.3 Data Structures and Storage

The descriptive statistics that are given as input are transformed into a state id by the agent, uniquely representing the values in the list. This vector representation allows for efficient computation and comparison of similarities across the KGs. Additionally, storing the vectors as state IDs enables subsequent analysis of the learned rules. These IDs are used in a Q-table and an index in the agent's memory.

The Q-table serves as a collection of acquired knowledge which stores state IDs with their respective visualization scores. Each domain in the system maintains a Q-table that serves as a repository for mapping state representations to visualization scores within that domain. These scores are indicative of domain-specific preferences and requirements crucial for effective visualization recommendations.

The index stores the state id as vectors and is used in the algorithm to find similar previous inputs. To facilitate rapid retrieval of similar state id vectors encountered previously, our system integrates the Voyager library [35]. Voyager enables approximate nearest-neighbor searches within an n-dimensional space on an in-memory collection of vectors, significantly enhancing the system's capability to identify and utilize statistical patterns expediently. This indexing mechanism optimizes the recommendation process by swiftly pinpointing the most relevant historical data points.

4.4 Algorithmic Explanation

From a high-level perspective, the algorithm operates through a series of sequential steps. Initially, it recommends the most appropriate visualization randomly. Next, the user rates the recommended visualization with a score. Following this recommendation, the Q-table is updated to reflect user feedback. Additionally, after processing each batch, the system updates the scores of the ten nearest neighbors for each batch item based on the received feedback, ensuring generalizability and accuracy in future recommendations.

On a lower level, the recommendation process initiates with the extraction of statistics from G' using a script that relies on Graphology [36]. This script can be utilized in both front-end and back-end environments, facilitating different operational contexts. The parsed statistics, along with the user's domain, are then passed to the recommendation agent. A state id is generated, representing all statistics values as a vector. If the recommendation agent encounters unseen vectors, it searches the index for the n most similar vectors. The average scores associated with these closest matches serve as the initial values in the Q-table for subsequent decision-making processes.

A key aspect of reinforcement learning is deciding on the strategy. Given that the prediction task involves a single step, GVR uses a relatively straightforward epsilon-greedy strategy. This strategy balances exploration and exploitation based on the epsilon parameter. In the future, this can be extended to more advanced methods, such as gradient-based techniques. The default exploration rate of the agent is set to 0.1, ensuring that 10% of the time, random actions are chosen to explore alternative visualization options. There is also the possibility to set a decay rate for the epsilon, to move to a more greedy strategy over time. Finally, the statistical vectors are added to the index for future reference in similarity searches.

4.5 Updating Scheme

After the algorithm recommends a visualization, it solicits explicit relevance feedback from the user to refine and update the Q-table scores dynamically. This feedback, ranging between 0 and 5, reflects the perceived relevance or utility of the recommended visualizations. The update function requires the learning rate (α), the current score in the Q-table ($Q(s, a)$), and the relevance feedback score (R). There is also the possibility to set a decay rate for the learning rate, which reduces the weights of the updating scheme over time. Specifically, the update function follows the formula: $Q(s, a) \leftarrow Q(s, a) + (\alpha * (R - Q(s, a)))$. The scores for the visualizations are bounded within a range of -3 to 3. This capping mechanism ensures that the scores remain within a controlled interval, preventing extreme values that could unduly influence the recommendation process. To find if this update scheme can reflect user input we ran successful experiments showing the algorithm learns an effective policy in 500 iterations in section 5.2.2.

5 Evaluation

This section outlines the evaluation conducted for GVR. It begins with a use-case scenario and proceeds to a technical evaluation with a generated dataset that assesses the learning performance.

5.1 Use-Case Scenario

The use-case scenario illustrates the utility of GVR. The use-case focuses on Jane, a data analyst for Northwind Traders, a fictitious multinational company that imports and exports specialty foods [37]. Northwind Traders maintains a graph database, which is ideal for representing and analyzing complex relationships between diverse entities. This capability allows for the integration of various data sources, identification of hidden patterns, and a holistic view of the business operations, which is crucial for making informed strategic decisions. The database contains information about customers, orders, products, and suppliers.

Jane aims to gain insights into the current status of the company and provide actionable recommendations to the management team. She plans to use multivariate graph visualization techniques to gather these insights because KGs. However, Jane has limited experience with graph analytics and is unsure which visualization to use for her analysis. This is where our VRS for KG visualizations comes in.

Product Preferences: Jane wants to explore products that customers might find interesting based on their characteristics. This is a typical graph analytics question because it involves analyzing relationships and connections between different entities, such as customers and products, to uncover patterns and insights that are not immediately apparent from the raw data alone. To discover the purchasing patterns of similar customers, she queries the graph database for customers and their orders. The typical graph analytics query that Jane would write in Cypher looks like:

MATCH (customer: Customer) - [:PURCHASED] -> (product: Product) - [:BELONGS] -> (category: Category) WITH category, customer ORDER BY customer.age RETURN category.name AS Category, collect(customer) AS Customers

The database returns a complex result set, which Jane finds challenging to interpret using traditional visualization techniques due to the data complexity (thousands of clients and hundreds of products). She decides to use our recommendation system to aid in the visualization step of her data analysis pipeline. The statistics module first extracts all KG statistics, after which

Jane runs the recommendation agent. Based on the descriptive statistics, GVR returns an adjacency matrix ordered by the 'age' attribute as one of the best visualizations. This visualization reveals a clear pattern because adjacency matrices benefit cluster and group discovery. For illustration purposes, Figure 5 shows how a user can see clusters in an adjacency matrix. The conclusion would be that customers can be effectively grouped by age based on purchasing behavior.

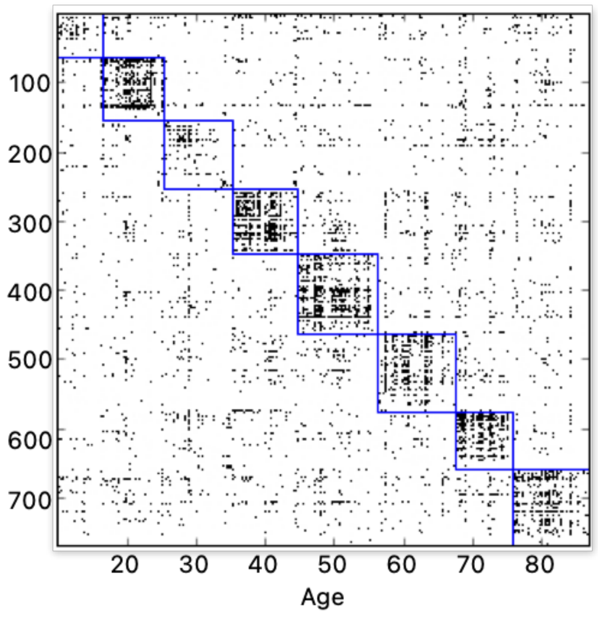


Figure 5: Adjacency matrix that shows areas with dense connectivity between nodes [38].

Common Characteristics: Among the products, Jane identifies French red wine as a potential high-revenue item. She filters her query to include only customers who have purchased this product:

MATCH (product: Product {name: 'French red wine'}) <- [:PURCHASED] - (customer: Customer) RETURN product, customer

After GVR extracts descriptive statistics, she instructs the recommendation system to suggest a visualization. The system recommends a topology-driven node-link diagram with node coloring based on the age attribute. Jane observes that a significant subset of customers aged 55 to 65 frequently buys French red wine. Additionally, she notices another non-overlapping subset within the same age range that has not purchased French red wine but might be interested based on their attributes. For illustration purposes, Figure 6 shows many nodes with a shared attribute connected to a single product.

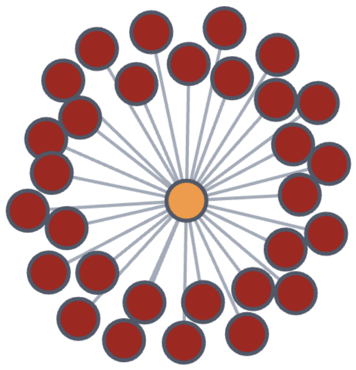


Figure 6: Node-link visualization that shows a lot of nodes with a shared attribute connected to a node in the middle [39].

High-Demand Products: Next, Jane wants to identify suppliers of French red wine and assess their capacity to meet increased demand. She queries the graph database for suppliers of French red wine and filters for customers in the greater Boston area aged 55 to 65. A typical query looks like this:

MATCH (supplier: Supplier) - [:SUPPLIES] -> (product: Product {name: 'French red wine'}), (customer: Customer) - [:PURCHASED] -> (product), (customer) - [:LIVES] -> (location: Location {name: 'Greater Boston Area'}) WHERE customer.age >= 55 AND customer.age <= 65 RETURN supplier, customer

Using the extracted statistics, the VRS recommends an attribute-driven node-link diagram as a cartographic visualization. This visualization shows two distributors in the area that supply French red wine, one of which has a large inventory and is conveniently located near many potential customers. For illustration purposes, Figure 7 shows how a node layer can be displayed on top of a geographical map to see distributors.

Future Trends: In summary, Jane discovered that a specific age group is likely interested in French red wine and identified local distributors with sufficient inventory to meet potential demand. These insights enable Jane to craft effective strategies for future revenue growth at Northwind Traders, and shows how GVR can be used in practice.

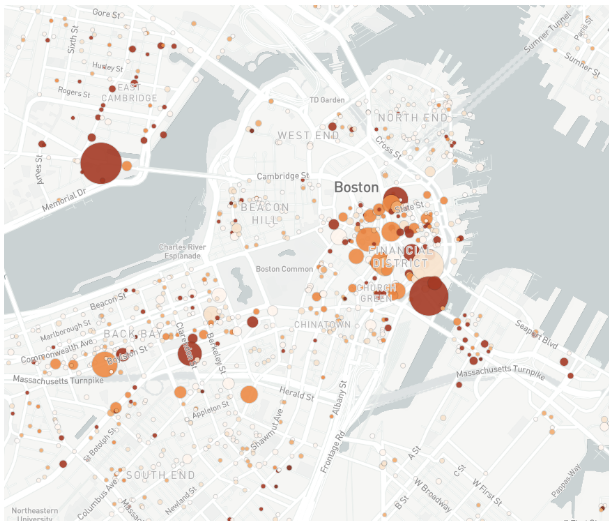


Figure 7: Map visualization that shows where the distributors are located in the Boston area [40].

5.2 Technical Evaluation

This section describes the models performance through a technical evaluation using a randomly generated collection of KGs.

5.2.1 Experimental Setup

Due to the absence of a ground truth dataset, conducting a proper technical evaluation is challenging. Instead, we simulated data generation using the categorization of Nobre et al. [8], the most current state-of-the-art report on multivariate graph visualization. We first selected random values for each of the features in their set, i.e., size, type, node/edge types, and node/edge heterogeneity. We then generated a random integer value within that selected value. This results in a set of generated descriptive statistics of KGs. Next, we used their scoring system as our reference to rank the visualizations from best to worst, resulting in a KG to visualizations mapping.

With this dataset, we can test the effectiveness of our algorithm and how quickly it learned a policy. The process involved the algorithm making a recommendation based on randomly generated KGs, which we compared to the ranked visualizations defined earlier. Based on the index in this ranking, we calculated an expected user reward, which ranged between 0 and 5. This setup allowed us to simulate a realistic evaluation environment and assess the performance of our algorithm on a set of user interactions.

5.2.2 Policy Effectiveness

The model effectively learns a policy that maximizes user reward within approximately a thousand interactions, depending on the parameter settings (taking around 20-30 seconds). Figure 8 illustrates the average rewards obtained with varying learning rates (alpha). The plot illustrates distinct learning behaviors: alpha set to 0.3 demonstrates rapid learning, achieving an average reward of 4 after approximately 300 interactions. This is logical, as the learning rate directly influences the speed at which the algorithm learns a policy. In contrast, alpha set to 0.2 shows a slower learning curve, stabilizing around 3.5 after 1000 interactions. Alpha set to 0.1 exhibits the slowest learning pace, eventually reaching a comparable average reward as alpha 0.3, albeit requiring approximately 5000 interactions to do so.

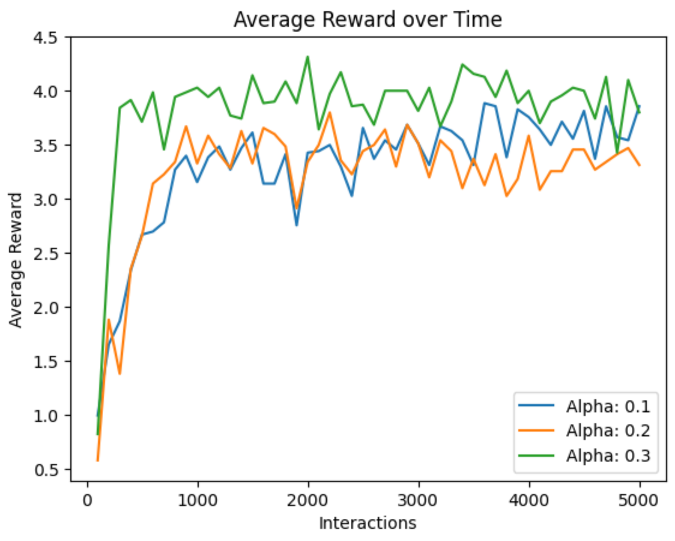


Figure 8: Comparing average rewards over interactions with different alpha values.

Figure 9 illustrates the average rewards obtained with varying epsilon values. The plot shows noticeable variance with an overall upward trend. The observed differences in variance across the lines can be attributed to a higher value of epsilon, which induces an increased number of exploratory steps. Specifically, the epsilon value of 0.3 initially shows growth but stabilizes around 500 interactions. In contrast, for epsilon values of 0.2, rewards start relatively high but display minimal growth over time. Epsilon set to 0.1 begins to stabilize after 1000 interactions and, by 5000 interactions, yields the highest average reward observed.

Generally, the model is well able to learn an effective policy based on the generated data suggesting that graph feature-centric learning of recommen-

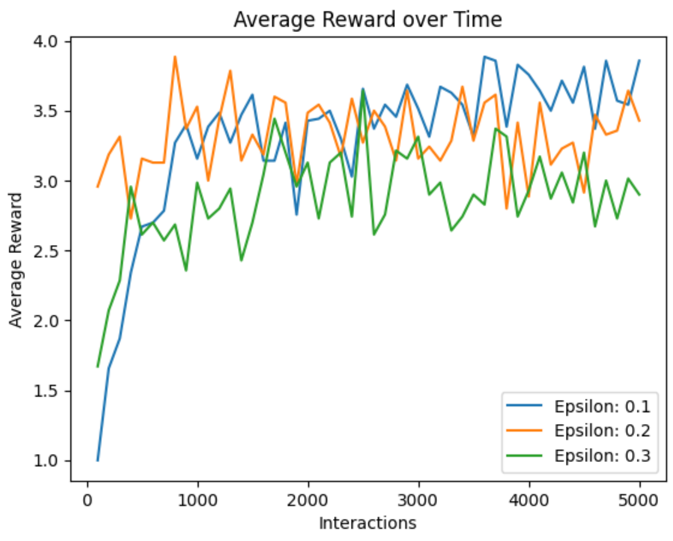


Figure 9: Comparing average rewards over interactions with different epsilon values.

dations is actually viable. A trade-off appears to exist between the number of interactions required to reach an optimum and the magnitude of the alpha and epsilon parameters.

5.2.3 Convergence

Figure 10 illustrates the rate of change in average rewards obtained with varying values of alpha. The graph shows a declining trend in the rate of change for average rewards. Notably, simulations with an alpha of 0.3 exhibit the most rapid initial decrease within the first few hundred interactions. Conversely, alphas of 0.1 and 0.2 demonstrate comparable rates of change initially. By the 1000th interaction, all simulations stabilize. This indicates that a higher alpha accelerates learning initially but converges similarly to lower alphas over time. Therefore, the choice of alpha affects early learning behavior more than the long-term outcome.

Figure 11 illustrates the rate of change in average rewards obtained with varying epsilon values. The plot shows a decreasing trend over interactions. Notably, the epsilon value of 0.1 starts with the highest average rate of change for the average reward and exhibits a gradual decline, stabilizing after approximately 100 interactions. In contrast, the epsilon values of 0.2 and 0.3 show comparable performance, displaying minimal change over time and remaining relatively stable from the initial interactions. These results demonstrate that GVR can effectively learn policies through user interaction and converge after several hundred iterations. This indicates that rules can

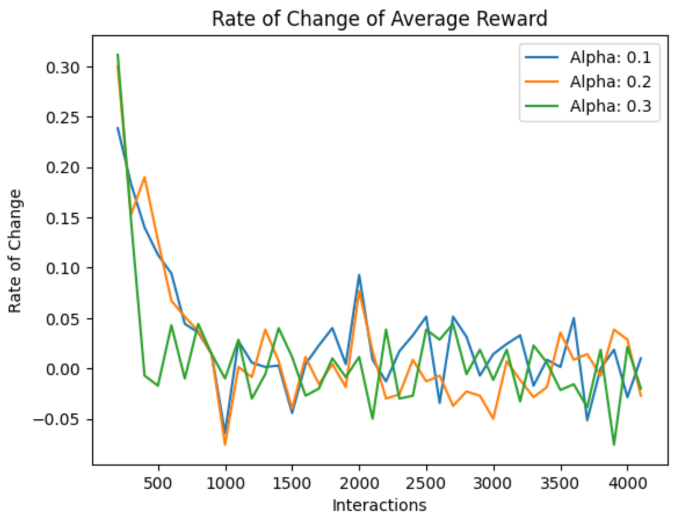


Figure 10: Compares the rate of change for average rewards over interactions with different alpha parameters.

be appropriately learned, advancing the concept of a classical rule-based learner to a machine-learning-based recommender.

5.2.4 Stability

Figure 12 illustrates the stability of the algorithm (alpha: 0.1, epsilon: 0.1) and its consistency over time. The graph shows that the algorithm initially starts far below the mean of the moving average. It then climbs quickly, reaching one standard deviation below the mean after 500 interactions. The algorithm remains within one standard deviation below and above the mean moving average, demonstrating its stability. Notably, after 3500 interactions, it continues to improve its performance gradually as it reaches the upper boundary, which is one standard deviation above the mean. The lack of large deviations demonstrates the algorithm's resilience to variations in the data or decision-making process.

6 Discussion and Future Work

We introduce GVR, a feedback-driven recommendation system for KG visualizations that constructs a rule base over time. Our tool aims to offer a data-driven approach to KG visualization, unlocking the full potential of KGs. The VRS process is designed to deliver visualizations that enable efficient analysis of complex patterns within KGs.

One of the significant strengths of GVR is its ability to generate insights over

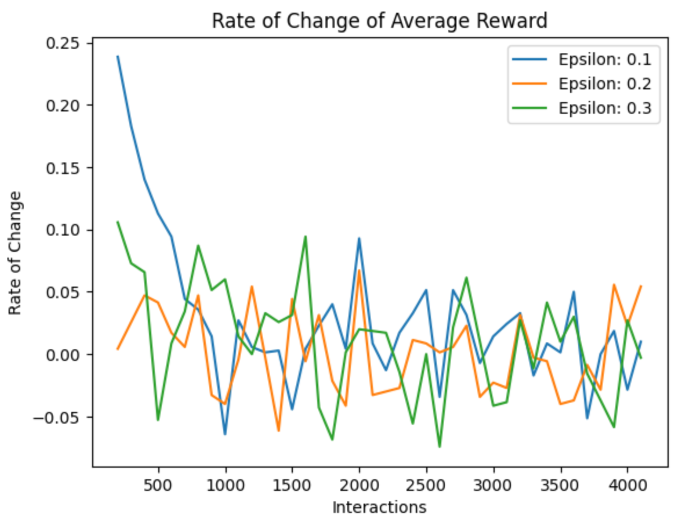


Figure 11: Compares the rate of change for average rewards over interactions with different epsilon parameters.

time regarding the optimal use of KG visualizations. The decision of which visualization to use is often ambiguous, but our system seeks to clarify this by building a rule base that evolves and improves through user interaction. By leveraging the collective wisdom of users, the system converges towards optimal solutions, thus providing increasingly accurate recommendations. Additionally, the system is highly extendable. It allows for adding new statistical dimensions, domains, and visualizations. Users can even configure the same visualization in multiple ways, making the system versatile and adaptable to various needs and contexts.

Despite its strengths, GVR faces several limitations and challenges. One challenge is that the result set may not be the sole factor influencing the choice of visualization. Other elements such as the schema and query context could also play significant roles. For example, when a query returns nodes and edges without geospatial information, but a node type with geospatial data exists in the schema, the recommended visualization might still be influenced by the presence of this geospatial information. Additionally, interpreting the characteristics of a graph solely from a result set can pose challenges. Determining its directionality, and attributes like weight, spatiality, or temporality is untrivial. While we can make assumptions about this, such as inferring weight from numerical node attribute values or spatial/temporal attributes from specific keys, these assumptions are not always reliable. Moreover, when only the result set is known, it is challenging to account for contextual factors of use without additional information. For instance, the recommended visualization may differ depending on whether the result set

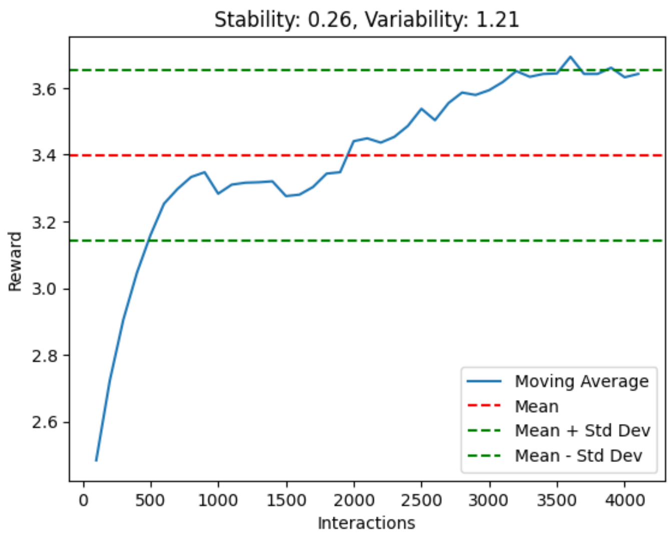


Figure 12: Stability and variability for the average reward over interactions. Parameters of the model used: alpha=0.1, epsilon=0.1.

includes all nodes present in the database or if a filtering operation is applied, returning only a specific cluster of nodes. Another significant challenge is the absence of a comprehensive repository of KG-to-visualization mappings, which hinders the initial training of models and the extraction of rules. However, we overcame this in a sub-optimal manner by generating artificial KG samples. As our system gathers more data over time, this repository will develop, improving the system's performance.

GVR has limitations in its initial stages. Initially, the system's performance is poor due to the lack of historical data and established rules, but this issue diminishes over time as more data is collected. Currently, there is no sound technical evaluation possible due to the lack of data, making it difficult to quantitatively assess the system's effectiveness at this stage. Implicit relevance feedback might introduce biases and varying interpretations. An alternative could be to use 'time-on-visualization' as a metric for relevance feedback, potentially offering a more objective measure. Additionally, the current approach does not consider view operations (juxtaposed, integrated, overloaded) or task definitions, which were excluded from the scope but could be captured by domain specifications in the future.

In the future, we will focus on enhancing the rules and improving the descriptive statistics for KGs. We plan to conduct a user study to comprehensively evaluate the algorithm's effectiveness. Additionally, we aim to incorporate visual quality metrics into the visualization recommendation process and utilize pattern extractors to identify potentially interesting elements that users might overlook. Integrating machine learning techniques could potentially enhance the model's decision-making capabilities. As more data becomes available, feature selection methods can be employed to reduce the decision space, streamlining the recommendation process and improving system efficiency. By continuing to refine and expand GVR, we aim to contribute valuable insights and tools to the field of knowledge graph visualization, ultimately facilitating more effective and informed visual data analysis. Finally, we invite researchers to explore and expand upon this work, fostering further research within the community.

7 Conclusion

To fully harness the potential of KG applications and unlock insights, there is a pressing need for automated visualization solutions that enhance its accessibility. We propose GVR, a feedback-driven visualization recommendation system that aims to address this gap by providing a versatile tool that consolidates collective knowledge into actionable rules using a reinforcement learning approach. By offering guidance for analysts, our system streamlines the visualization process and facilitates deeper insights into KGs.

Through our evaluation, we demonstrated that the algorithm can effectively learn policies based on user interactions, showing significant promise in improving visualization recommendations over time. While the topic is complex and multifaceted, and we do not claim to have a definitive solution, our proposed approach offers a viable path forward. We invite fellow researchers to engage with this issue and contribute their insights to further the discourse, ultimately advancing the field of KG visualization.

Acknowledgements

The author wishes to thank GraphPolaris, Utrecht University, and Tufts University for enabling this collaboration. Additionally, a great deal of inspiration came from the state-of-the-art report by Nobre et al., to whom I am particularly grateful.

References

[1] Harry Li et al. “Knowledge graphs in practice: Characterizing their users, challenges, and visualization opportunities”. In: *IEEE Transactions on Visualization and Computer Graphics* (2023).

[2] Victor Dibia. “LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models”. In: *arXiv preprint arXiv:2303.02927* (2023).

[3] Haotian Li et al. "KG4Vis: A knowledge graph-based approach for visualization recommendation". In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2021), pp. 195–205.

[4] Luca Podo, Bardh Prenkaj, and Paola Velardi. "Machine Learning for Visualization Recommendation Systems: Open Challenges and Future Directions". In: *arXiv preprint arXiv:2302.00569* (2023).

[5] Ales Komarek, Jakub Pavlik, and Vladimir Sobeslav. "Network visualization survey". In: *Computational Collective Intelligence: 7th International Conference, ICCCI 2015, Madrid, Spain, September 21-23, 2015, Proceedings, Part II.* Springer. 2015, pp. 275–284.

[6] Giuseppe Di Battista et al. "Algorithms for drawing graphs: an annotated bibliography". In: *Computational Geometry* 4.5 (1994), pp. 235–282.

[7] Kevin Hu et al. "Vizml: A machine learning approach to visualization recommendation". In: (2019), pp. 1–12.

[8] Carolina Nobre et al. "The state of the art in visualizing multivariate networks". In: *Computer Graphics Forum.* Vol. 38. 3. Wiley Online Library. 2019, pp. 807–832.

[9] Javier Martın Hernández and Piet Van Mieghem. "Classification of graph metrics". In: *Delft University of Technology: Mekelweg, The Netherlands* 1 (2011).

[10] Eric D Kolaczyk et al. "Descriptive analysis of network graph characteristics". In: *Statistical analysis of network data with R* (2014), pp. 43–67.

[11] Hans-Jörg Schulz et al. "A design space of visualization tasks". In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2366–2375.

[12] Giuseppe Di Battista et al. *Graph drawing: algorithms for the visualization of graphs.* Prentice Hall PTR, 1998.

[13] Linton C Freeman. "Visualizing social networks". In: *Journal of social structure* 1.1 (2000), p. 4.

[14] Ivan Herman, Guy Melançon, and M Scott Marshall. "Graph visualization and navigation in information visualization: A survey". In: *IEEE Transactions on visualization and computer graphics* 6.1 (2000), pp. 24–43.

[15] Richard A. Becker, Stephen G. Eick, and Allan R. Wilks. "Visualizing network data". In: *IEEE Transactions on visualization and computer graphics* 1.1 (1995), pp. 16–28.

[16] Chris Godsil and Gordon F Royle. *Algebraic graph theory.* Vol. 207. Springer Science & Business Media, 2001.

[17] Paolo Buono, Miguel Ceriani, and Maria Francesca Costabile. "Hypergraph Data analysis with PAOHVis." In: *SEBD.* 2021, pp. 160–167.

[18] Anastasia Bezerianos et al. "Geneaquilts: A system for exploring large genealogies". In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1073–1081.

[19] William JR Longabaugh. "Combing the hairball with BioFabric: a new approach for visualization of large networks". In: *BMC bioinformatics* 13 (2012), pp. 1–16.

[20] Brian Johnson and Ben Shneiderman. *Tree-maps: A space filling approach to the visualization of hierarchical information structures.* Tech. rep. UM Computer Science Department; CS-TR-2657, 1998.

[21] Keith Andrews and Helmut Heidegger. "Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs". In: *IEEE Information Visualization Symposium.* 1998, pp. 9–12.

[22] Jock Mackinlay. "Automating the design of graphical presentations of relational information". In: *Acm Transactions On Graphics (Tog)* 5.2 (1986), pp. 110–141.

[23] Stephen M Casner. "Task-analytic approach to the automated design of graphic presentations". In: *ACM Transactions on Graphics (ToG)* 10.2 (1991), pp. 111–151.

[24] Steven F Roth et al. "Interactive graphic design using automatic presentation knowledge". In: (1994), pp. 112–117.

[25] Jacques Bertin. *Semiology of graphics.* University of Wisconsin press, 1983.

[26] William S Cleveland and Robert McGill. "Graphical perception: Theory, experimentation, and application to the development of graphical methods". In: *Journal of the American statistical association* 79.387 (1984), pp. 531–554.

[27] Jock Mackinlay, Pat Hanrahan, and Chris Stolte. "Show me: Automatic presentation for visual analysis". In: *IEEE transactions on visualization and computer graphics* 13.6 (2007), pp. 1137–1144.

[28] Kanit Wongsuphasawat et al. "Voyager: Exploratory analysis via faceted browsing of visualization recommendations". In: *IEEE transactions on visualization and computer graphics* 22.1 (2015), pp. 649–658.

[29] Kanit Wongsuphasawat et al. "Voyager 2: Augmenting visual analysis with partial view specifications". In: (2017), pp. 2648–2659.

[30] Bongshin Lee et al. "Treeplus: Interactive exploration of networks with enhanced tree layouts". In: *IEEE Transactions on Visualization and Computer Graphics* 12.6 (2006), pp. 1414–1426.

[31] Hyunmo Kang et al. "NetLens: iterative exploration of content-actor network data". In: *Information Visualization* 6.1 (2007), pp. 18–31.

[32] Yuyu Luo et al. "Deepeye: Towards automatic data visualization". In: *2018 IEEE 34th international conference on data engineering (ICDE).* IEEE. 2018, pp. 101–112.

[33] Dominik Moritz et al. "Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco". In: *IEEE*

transactions on visualization and computer graphics 25.1 (2018), pp. 438–448.

[34] Kanit Wongsuphasawat et al. "Towards a general-purpose query language for visualization recommendation". In: *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. 2016, pp. 1–6.

[35] Peter Sobot et al. *Voyager*. `https://github.com/spotify/voyager`. 2024.

[36] Guillaume Plique. "Graphology, a robust and multipurpose Graph object for JavaScript". In: *Zenodo. https://doi. org/10.5281/zenodo* 5681257 (2022).

[37] Microsoft. *Northwind and pubs sample databases for Microsoft SQL Server*. `https://github.com/Microsoft/sql-server-samples/tree/master/samples/databases/northwind-pubs`. 2023.

[38] *How to reorder/cluster adjacency matrix to maximize the interaction along the super diagonal?* `https://scicomp.stackexchange.com/questions/31061/how-to-reorder-cluster-adjacency-matrix-to-maximize-the-interaction-along-the-su`. Accessed: 2024-06-26.

[39] *GraphPolaris Explorer Interface*. `https://graphpolaris.com/`. Accessed: 2024-06-26.

[40] *Seven data visualization techniques for location*. `https://blog.mapbox.com/7-data-visualizations-techniques-for-location-544c558cc960?gi=bf6e005da355`. Accessed: 2024-06-26.

A Data Generation

The generation process involves producing a specified number of samples, each characterized by a set of predefined attributes. The function takes two parameters: the number of samples to generate, and an optional random seed for reproducibility. If a random seed is provided, the random number generators are seeded to ensure reproducibility.

An empty list is initialized to store the generated samples. For each sample, six categorical attributes are randomly chosen. These categories include the size of the graph (small, medium, large), the type of graph (sparse, dense, tree, k-partite), the homogeneity or heterogeneity of the graph nodes, the number of node attributes (few or several), the homogeneity or heterogeneity of the graph edges, and the number of edge attributes (few or several). Based on the selected categories, the actual values for the graph attributes are determined:

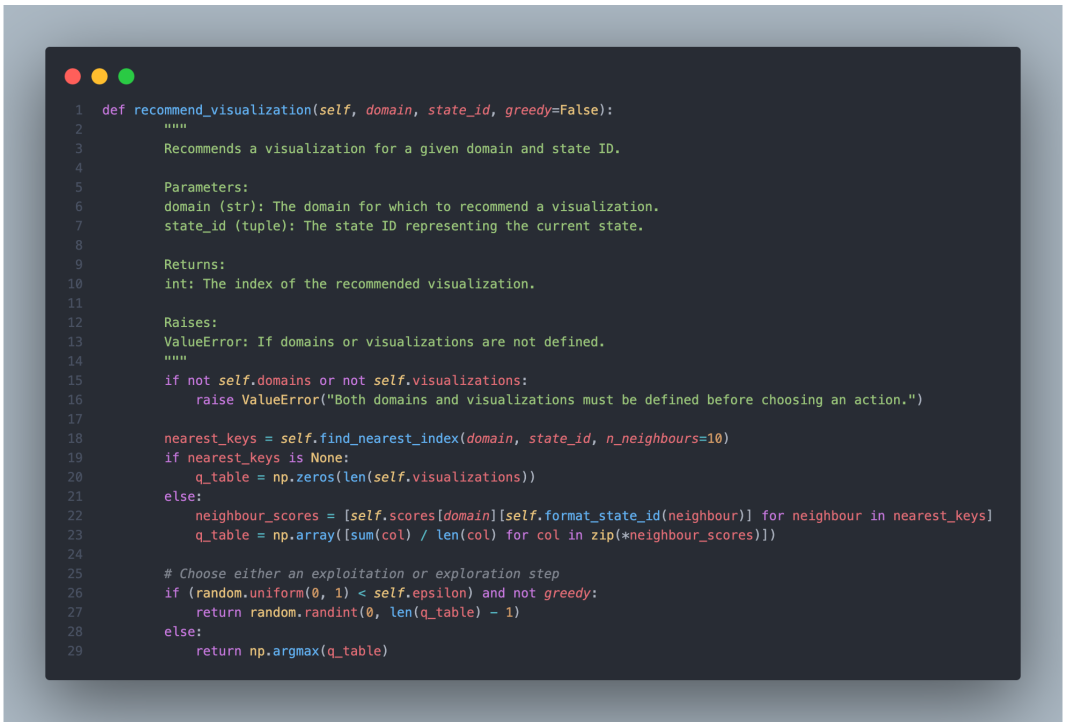
- The number of nodes in the graph is determined by the size category.
- The number of different types of nodes is determined by the node type category.
- The number of attributes associated with nodes is determined by the node attributes category.
- The number of different types of edges is determined by the edge types category.
- The number of attributes associated with edges is determined by the edge attributes category.
- The number of edges in the graph is determined by the type category.
- The density of the graph is calculated as the ratio of the number of edges to the maximum possible number of edges.

A dictionary is initialized to store the scores for different visualization types. Scores are computed based on predefined values associated with each attribute category. The visualization with the highest score is chosen as the best visualization. Visualizations are sorted by their scores to determine their rank and generate user feedback. User feedback is calculated based on the rank of the best visualization. Each sample is constructed as a dictionary containing the domain of the graph, a dictionary of the graph's statistical attributes, the calculated user feedback score, and a list of visualizations sorted by their scores. Finally, the function returns the list of generated samples. This process ensures that each generated sample is characterized by a diverse set of attributes, making it suitable for a wide range of experimental scenarios.

B Implemented Code

B.1 Recommend Visualization Function

The recommendation function (Figure 13) takes three parameters: the domain for which the visualization is needed, the state ID representing the current state, and an optional flag for greedy selection. First, the function checks if both the domains and visualizations are defined. Then, it identifies the nearest neighbors to the current state within the given domain. If no neighbors are found, a default score table is used. Otherwise, it calculates an average score table based on the scores of the nearest neighbors. The function then decides whether to choose a visualization based on exploration (random selection) or exploitation (selecting the highest-scored visualization). If exploration is chosen, a random visualization is recommended. If exploitation is chosen or the greedy flag is set, the visualization with the highest score is recommended. This process ensures that the recommendation balances between exploring new options and leveraging known best choices.

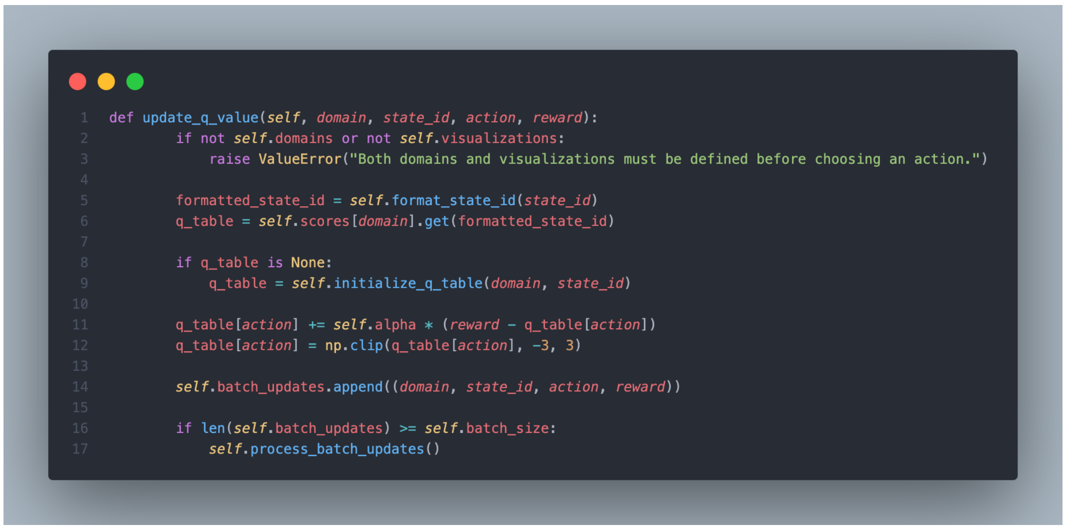


```python
def recommend_visualization(self, domain, state_id, greedy=False):
        """
        Recommends a visualization for a given domain and state ID.

        Parameters:
        domain (str): The domain for which to recommend a visualization.
        state_id (tuple): The state ID representing the current state.

        Returns:
        int: The index of the recommended visualization.

        Raises:
        ValueError: If domains or visualizations are not defined.
        """
        if not self.domains or not self.visualizations:
            raise ValueError("Both domains and visualizations must be defined before choosing an action.")

        nearest_keys = self.find_nearest_index(domain, state_id, n_neighbours=10)
        if nearest_keys is None:
            q_table = np.zeros(len(self.visualizations))
        else:
            neighbour_scores = [self.scores[domain][self.format_state_id(neighbour)] for neighbour in nearest_keys]
            q_table = np.array([sum(col) / len(col) for col in zip(*neighbour_scores)])

        # Choose either an exploitation or exploration step
        if (random.uniform(0, 1) < self.epsilon) and not greedy:
            return random.randint(0, len(q_table) - 1)
        else:
            return np.argmax(q_table)
```

Figure 13

B.2 Update Function

The update function (Figure 14) takes four parameters: the domain, the state ID representing the current state, the action taken, and the reward received. First, the function checks if both the domains and visualizations are defined. The state ID is then formatted to match the required format. The function retrieves the Q-value table for the given state in the specified domain. If no Q-value table exists for the state, it initializes a new one. The Q-value for the specified action is updated using the learning rate and the received reward. The updated Q-value is then clipped to ensure it remains within a predefined range. The update is added to a batch of updates. If the number of updates in the batch reaches a specified threshold, the function processes the batch updates to apply them. This process ensures that the Q-value table is continuously improved based on received rewards, facilitating more accurate visualization recommendations over time.



```python
def update_q_value(self, domain, state_id, action, reward):
        if not self.domains or not self.visualizations:
            raise ValueError("Both domains and visualizations must be defined before choosing an action.")

        formatted_state_id = self.format_state_id(state_id)
        q_table = self.scores[domain].get(formatted_state_id)

        if q_table is None:
            q_table = self.initialize_q_table(domain, state_id)

        q_table[action] += self.alpha * (reward - q_table[action])
        q_table[action] = np.clip(q_table[action], -3, 3)

        self.batch_updates.append((domain, state_id, action, reward))

        if len(self.batch_updates) >= self.batch_size:
            self.process_batch_updates()
```

Figure 14

C Project plan

C.1 Goals

- Must have
  - Big goal: which visualization to autoselect
  - Define the design space of visualizations
  - Develop a structure for APT-like, rule-based recommendations for multivariate graphs (methodology)
  - Rule-based system that recommends one of the three graph visualizations depending on result set topology (implementation in frontend)
  - Rule-based logic should be on user-steerable heuristics (e.g., thresholds accessible in the UI)
- Should have
  - Big goal: which visualization-layout algorithm to auto-select
  - Adaptable system for rules (new Visualizations and plugins thereof)
  - working against an interface for future replacement of rule engine with ML
  - Devise a user-evaluation strategy for an online experiment within GP
- Could have
  - Big goal: which visualization-encoding to auto-select
  - **User input**: Learning from user-feedback to adapt rules or ML
  - **ML input** ML-based system for recommendation
  - Recommend layout algorithm on top of vis (Matrix + reordering approach or NL + layout approach)
  - Recommend encodings on top of vis (NL + node size, paohvis + highlighting of certain rows)
  - Run user evaluation in an online experiment within GP
- Won't have
  - dynamic graphs in implementation
  - no uncertainty in graphs