GVQA: Learning to Answer Questions about Graphs with Visualizations via Knowledge Base

Sicheng Song
School of Computer Science and Technology,
Shanghai Institute of AI for Education
East China Normal University
Shanghai, China
scsong@stu.ecnu.edu.cn

Juntong Chen
School of Computer Science and Technology
East China Normal University
Shanghai, China
jtchen@stu.ecnu.edu.cn

Chenhui Li
School of Computer Science and Technology
East China Normal University
Shanghai, China
chli@cs.ecnu.edu.cn

Changbo Wang
School of Computer Science and Technology
East China Normal University
Shanghai, China
cbwang@cs.ecnu.edu.cn

ABSTRACT

Graphs are common charts used to represent the topological relationship between nodes. It is a powerful tool for data analysis and information retrieval tasks involve asking questions about graphs. In formative study, we found that questions for graphs are not only about the relationship of nodes but also about the properties of graph elements. We propose a pipeline to answer natural language questions about graph visualizations and generate visual answers. We first extract the data from graphs and convert them into GML format. We design data structures to encode graph information and convert them into an knowledge base. We then extract topic entities from questions. We feed questions, entities and knowledge bases into our question-answer model to obtain the SPARQL queries for textual answers. Finally, we design a module to present the answers visually. A user study demonstrates that these visual and textual answers are useful, credible and and transparent.

CCS CONCEPTS

• **Human-centered computing** → **Visualization**; **Natural language interfaces**; • **Information systems** → **Question answering**; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Visualization, Network Graph, Natural Language Process, Reinforcement Learning, Knowledge Base, Question Answering

ACM Reference Format:
Sicheng Song, Juntong Chen, Chenhui Li, and Changbo Wang. 2023. GVQA: Learning to Answer Questions about Graphs with Visualizations via Knowledge Base. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3544548.3581067

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04…$15.00
https://doi.org/10.1145/3544548.3581067

1 INTRODUCTION

Graphs are common visualizations to analyze topological data and answer questions when users facing decision-making tasks. However, it is not an easy task for users when there are many complex analytical questions about graphs. First, users need to understand the question before observing graphs. When observing the graph, users also need to understand the legend and textual information, perceive graphical data, analyze the community, calculate the degree values of the data, and other steps according to the different questions.

There are already some natural language systems for automatic chart question answering to help users analyze charts and answer questions faster. However, most of the existing work [35, 42] focuses on simple visualization charts such as scatter charts, line charts, and bar charts. These charts are easy to be converted into tabular data, which adapt to the tabular question answering system [54]. The graph visualizations are commonly used to represent the topological relationship between nodes, such as the character relationships in novels. People usually analyze data and answer questions through graphs. In formative study, we find that users ask questions with graphs not only about the topological relationships of nodes, but also about graph properties such as node degrees, graph communities, etc. It is difficult to convert graph visualizations into tabular data as bar charts, because graph visualizations include more attributes and graph data are not two-dimensional structures. Besides, most of the existing automatic question-answering systems use textual information to answer [35], which is dull and deviates from the original intention of visualizing the data of charts vividly. The visual answers are more convincing than textual answers [42]. We design a set of rules to visualize the answers and output graphical explanations.

We present a novel pipeline **GVQA** (Graph Visualization Question Answering) to answer the questions about graph visualizations and generate visual answers automatically. We first convert the graph visualizations into standard GML [20] formats. We then extract the graph data and the visual attributes from the GML. Based

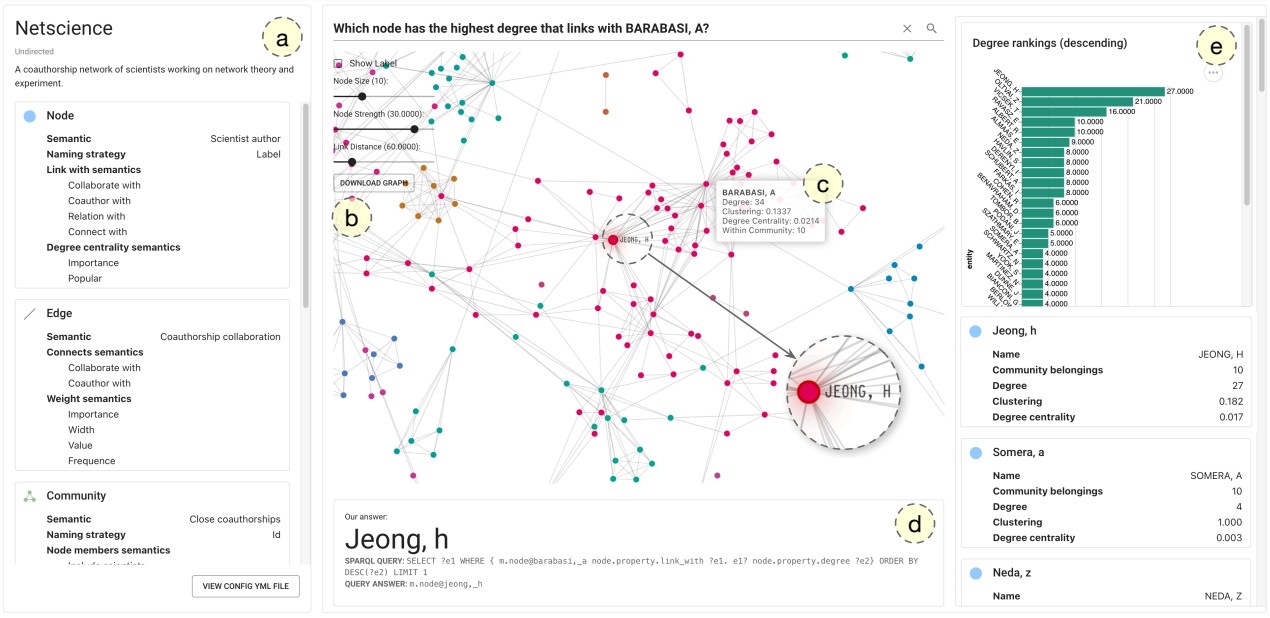


Figure 1: GVQA system interface. (a) The metadata of the graph visualization including name, semantics for essential elements and relations of the graph like nodes, edges, and community; (b) The graph visualization question answering panel. We highlight the query answer of the question in the graph visualization with a configurable and exportable force-directed layout; (c) Graph element tooltip. The tooltip will be visible when the user cursor hovered on the nodes or edges, providing users with their detailed properties; (d) The textual answer panel, providing users the readable answer and how our system queried the answers from our knowledge base; (e) The auxiliary answers panel. Providing rankings or entity details for further graph data analysis.

on the characteristics of graphs, we design the data structure of graphs and convert them into a KB (knowledge base). We also design an expansion module for the knowledge base to answer the questions with semantics and questions about graph analysis. Our question answering model is based on the BERT module [15] and a reinforcement learning model proposed by Das et al [14] in an end-to-end manner. We specifically design the aggregation operations according to the questions about graph visualizations.

We conducted evaluation experiments on a real-world corpus with 561 graph-question pairs compared with a knowledge based question answer model for web questions [36], and the results show that our design improved the ability to answer questions about graph visualizations. We also show three cases of graph visualization analysis and large network analysis. In addition, we conducted a user study to demonstrate that visual answers are more persuasive than textual answers. Our experimental results and user study show that our method has application potential in automatically question answering about graph visualizations. Our contributions include three aspects:

(1) We define a new problem of question answering about graph visualizations.
(2) We propose a pipeline to answer diverse questions about graph visualizations and generate visual and textual answers.
(3) We build a natural language question answering interface to facilitate user's analysis of graphs.

2 RELATED WORK

Our work is mainly related to three aspects: natural language interfaces for visualization, visual question answering, and knowledge base question answering.

2.1 Natural Language Interfaces for Visualization

Natural language interfaces for visualization have gained increasing popularity and attention [5, 60]. These interactive systems usually take natural language as input and visualization as output [16, 35, 47, 64]. Users can input natural language in many ways, such as characters [16] with a keyboard, audio [64] with a microphone, text selected in an article [47], or textual definition [35]. This system usually outputs results by creating new visualizations [16] or highlighting parts of existing visualizations [21, 59]. These interactive systems are useful for annotations [11, 12, 35, 42, 57, 61], telling stories [67] and generating visual natural language descriptions [43, 53]. NL2VIS is research [45] in the field of database design visual structured query language similar to database SQL (structured query language) for visual codes in recent. Huang et al. proposed FlowNL [26], an interface for asking the flow data in natural language. Badam et al. [2] proposed Elastic Documents to visualize the links between tables and charts. Interactive document readers [30, 33] focus on the links between tables and texts. Most visual analysis systems focused on charts that can be converted into tabular data, such as pie charts, bar charts, line charts, and scatter plots.

Table 1: The corpus of our visualization graphs with questions. We provide an example below the number of each question type.

Question Type	Lookup	Compositional	**Total**
Semantic	111 (19.79%)	113 (20.14%)	**224 (39.93%)**
	What scientists collaborated with Pildis, RA?	*Which scientist has the most collaborators?*	
Structural	104 (18.54%)	233 (41.53%)	**337 (60.07%)**
	What are the nodes that link with Pildis, RA?	*Which is the node with the highest degree?*	
Total	**215 (38.32%)**	**268 (61.68%)**	**561**

Only a few works focus on network graphs, Orko [65] is a visual system that pioneered the application of NLI to graph exploration, while our research focuses more on automatic question answering of graph analysis, such as some graph properties. According to our formative study in Sec. 3, the collected questions have more structural compositional questions than the semantic lookup questions that Orko focused on. For example, the question shown in Fig. 1 is a structural compositional question which usually exists in the graph analysis. Besides, our work also adopts a different approach from Orko, the knowledge base.

2.2 Visual Question Answering

Visual question answering is a semantic understanding task that aims to answer some questions based on given visual charts and legends. This is different from visual question answering in computer vision, which focuses on images that are not natural objects but visual charts. Such tasks require separate coding and mixing of visualizations and questions to generate answers [46]. Kim et al. [31] proposed a visual question answering pipeline based on a semantic parsing model named Sempre [54, 76] to automatically answer questions about charts and generate explanations for why the answer was generated. There are some works [10, 27, 28, 46, 62] using the OCR module to extract textual information in infographics for answer generation. These works mainly focus on bar charts, scatter charts, line charts, etc. They are relatively easy to convert into two-dimensional tabular data then graphs. Our work focuses on automatic question answering of topological network graphs, which are difficult to convert into tabular data because of containing many visual attributes. There are also some works focusing on scientific diagrams [29, 30, 74], such as the AI2D datasets. This type of data is actually the target of natural images, not visualizations. Questions about scientific diagrams are simple multiple-choice questions that only ask about the relationships between different targets. In contrast, our work focuses more on node relationships. In addition to the relationship between nodes, our questions also contain the community structure, node degree and the others which are unique and usual questions for the visualization graph.

2.3 Knowledge Base Question Answering

Knowledge base is an emerging application in visualization. Most research used knowledge base in visualization recommendation [17, 40]. KBQA (Knowledge Base Question Answering) aims to answer factual questions from a knowledge base. They have received much attention in recent years [22, 41, 56]. Early knowledge base question answering work mainly focused on some simple questions with a single relationship [8, 73]. In recent years, knowledge base question answering work has developed towards complex question datasets, which is also due to the fact that real problems are often complex. The complexity is mainly reflected in two aspects: The first one is a single relation with constraints [44, 72], such as "*where was the first ACM CHI Conference held?*". There is only one relationship "*location*" between the answer entity and the topic entity "*ACM CHI Conference*" in this question, but the limitation word "*the first*" has been added. Existing methods use staged queries, first identifying single-hop relational paths and then adding constraints to them to form a query graph. Second, in the question with multi-hop relationships, such as "*who is the subcommittee chair of the Visualization field of CHI2023?*", in this question, the answer entity needs to go through the two-hop relationship of "*Visualization field*" and "*subcommittee chair*" to connect with "*CHI2023*". The major challenge of this type of question is to limit the search space. Some existing work [13, 37] proposes to limit the search space by considering only the best matching relationship instead of all relationships when expanding the relationship path. There are also some studies [36] that combine both methods to handle more complex questions. Knowledge base question answering is previously applied to question answering of huge relational databases or knowledge graphs in the real world, while our work focuses more on applying it to question answering of network graph visualizations.

3 FORMATIVE STUDY

We conducted a formative study to understand what questions people would ask about graph visualizations and how they ask them. We first built a corpus containing 16 graph visualizations (13 undirected graphs, 3 directed graphs), which came from the network dataset collected by Mark Newman [52]. We then recruited 40 volunteers (μ_{age} = 28.8 years). Each volunteer has more than 2 years of experience researching graph theory or visualization. Before participants were asked questions, we spent 30 minutes explaining what each graph represented to ensure they comprehended these graphs. Each participant was then allowed to ask up to two questions per graph visualization. Since the graphs are mutually independent of each other, the order we presented these graphs to the participants is randomized.

We ruled out subjective questions like "*Can you output my favorite character in the graph?*" with the help of 3 graph visualization experts (more than 5 years of experience in graph visualization). We finally collected 561 valid question-answer pairs. We found that in terms of the answer type, the questions are primarily divided into two categories: Lookup and Compositional. We define

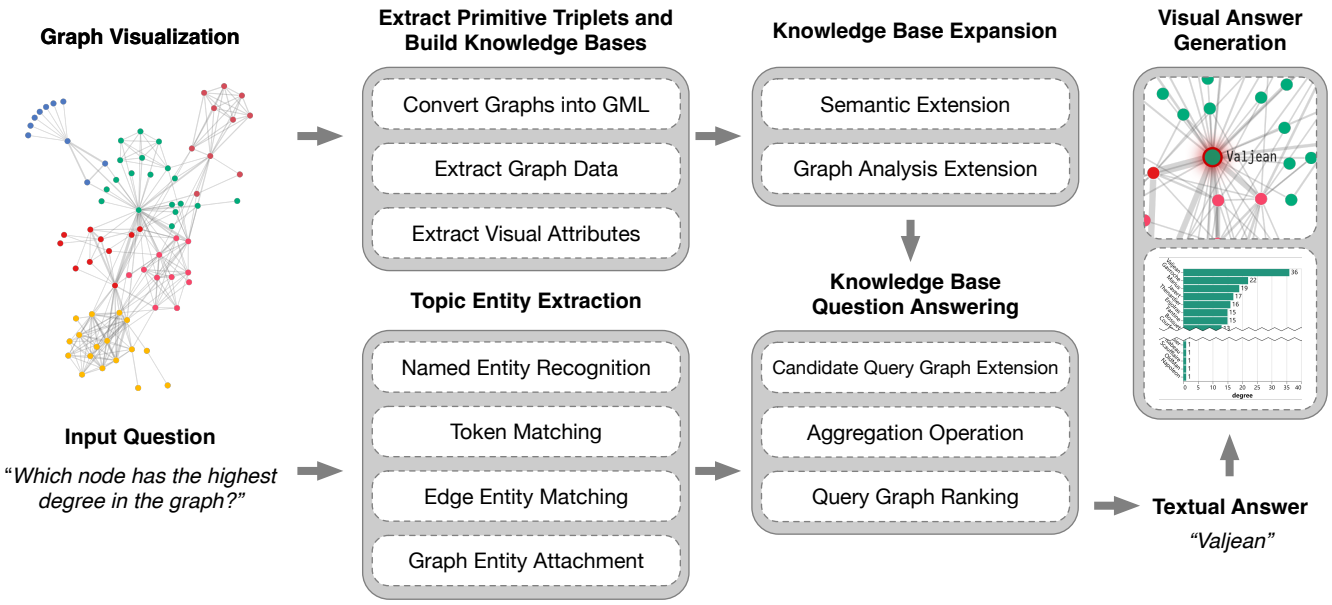


Figure 2: The pipeline of GVQA including five modules.

a question to be lookup if its answer can be found directly within the graph data. These are typically simple questions that only require information from the GML files, such as asking for a node's neighbors. Compositional questions, on the other hand, require knowledge base expansion (Sec. 4.2) or aggregation (Sec. 4.4). For example, "*What is the node with the lowest degree centrality?*" needs a superlative value aggregation. Compared with other charts, we found that participants usually ask about the properties that are not in the original graph data but are often used in graph analysis. For example, 118(21.03%) questions are related to the community, and 103(18.36%) questions are related to the properties of the node. This indicates that there are more properties involved in graph visualizations than in other charts such as bar and line charts.

Additionally, inspired by Srinivasan et al. [65] and Zhao et al. [77], we classify questions according to their utterance into two types of questions: Semantic and Structural. A semantic question is one that is related to the meaning and context of the graph, while a structural question focuses on the properties of the graph itself, without regard for the graph's specific meaning or semantics. For example, for the character relationship [32] in the novel Les Misérables, there are Semantic questions containing the background story of the graph visualization like "*How many characters coappeared with Valjin?*", and Structural questions like "*How many nodes does the graph have?*".

Our formative study reveals that participants tend to ask more Structural (60.07%) and Compositional (61.68%) questions than Semantic (39.93%) and Lookup (38.32%) questions, respectively. This indicates that people prefer to ask Structural questions that are more concrete and definitive, while we also find that there there is a wide range of utterances asking for graph properties in Semantic questions. In terms of the answer type, we find that Lookup questions were primarily asked about nodes, while Compositional questions were primarily asked about community belongings and properties of nodes with aggregations. This suggests that users are interested in not only the basic properties of the nodes in graph visualizations but also the relationships and patterns of nodes, edges, and communities within the graph. The composition of these questions and some examples are shown in Table 1. After verifying that the answers to the questions in our corpus are correct, we use these 561 graph-question pairs as our evaluation dataset. The datasets are included in the supplementary materials for further research.

4 METHODS

Our goal is to answer the questions about graph visualizations and output visual answers. The traditional method [35] is only appropriate for simple charts, such as bar charts and line charts. Graphs have more visual attributes and relation data than the above charts. It is hard to convert graph visualizations into tabular data. To solve these problems, we design a pipeline to convert graphs into knowledge bases and apply a novel reinforcement model [14] to adapt it to visualization question answering.

We propose a pipeline named GVQA, which automatically answers the question about graph visualizations and generate visual answers. Figure 2 shows our pipeline. The pipeline includes five components:(1) extract primitive triplets and builds knowledge bases, (2) knowledge base expansion, (3) topic entity extraction, (4) automatic question answering, and (5) visual answer generation.

4.1 Extract Primitive Triplets and Build Knowledge Bases

Graph visualizations generally consist of topological relationships and visual attributes [50]. The visual attributes of graphs include color, edge width, node area, node position, etc [63]. GML [20] (Graph Modelling Language) is a portable file format for graphs

with simple syntax and high flexibility. Graph data can be stored in many formats and can be easily converted to GML, and GML can also be easily converted to other formats. A KB (Knowledge Base) represents a repository for storing knowledge [69]. We typically use a set of triplets to represent a KB. They are written as $\mathcal{K} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where h represents the head entity, r represents the relation, t represents the tail entity, $\mathcal{E}$ represents the entity set, and $\mathcal{R}$ represents the relation set.

In the first step, we convert the input graph to GML format and extract topological data and visual attributes. Then, we design a data structure in the form of KB triple for graph visualizations. These data originally existed in GML, so we call the extracted triples as primitive triplets.

Convert Graphs into GML. There are many formats of graph visualizations. The formats of these graphs are generally divided into two types. The first type is in vector format, such as visualizations designed by D3 [9], G6 [70], E-Charts [39], etc. For this type of graph, we can easily parse and deconstruct their JavaScript code to automatically extract data [18]. The other type is in bitmap format, which is more commonly seen in real cases. For these graph images, we use VividGraph [63], a state-of-the-art method for automatically extracting graphs, to extract the original data of graph visualizations. We then convert the data into GML formats.

Extract Graph Data. Given a visualization graph in GML formats as shown in Figure 3, the graph category includes two main types of category, node and edge. We first convert all nodes into entities. According to different naming strategies, we assign the id or the label of the nodes to the name of entities. We then convert all edges into relations. For example, the edge in Figure 3 represents that the node "*Myriel*" links with the other node "*Napoleon*". We then add a triplet *<Myriel><Relation><Napoleon>* to the KB. In this example, the graph is an undirected graph, so we also add a reversed triplet *<Napoleon><Relation><Myriel>* to our KB.

Extract Visual Attributes. Graph visualizations have a large number of visual attributes (e.g., color, area, position, shape) of nodes and edges. We convert the attributes into entities and add a relation with the node entities. In addition to the visual attributes of nodes, edges also have visual attributes such as color, width, etc. In most of the designed KBs, edges exist only as relations in triplets. We specifically convert edges into entities into the KB as well. Then we can add the visual attributes of the edges *<Edge Entity><Attribute><Value>* to the edge entities. For example, the weight of edge connecting "*Myriel*" and "*Napoleon*" is 1, as shown in Fig. 3. The *<Edge Entity>* is "*Myriel and Napoleon*"; the *Attribute* is "*Weight*"; and the *<Value>* is 1. Since there are some visual attributes (e.g. background color) that are specific to the whole graph, we treat the whole graph as an entity and insert it into our knowledge base. We then establish connections between the graph entity and the graph members by adding the relations (*<Graph Entity><Relation><Node Entity>* and *<Graph Entity><Relation><Edge Entity>*) from graph entities to all node entities and edge entities. Since not all graph visualizations have complete visual attributes, we only convert the visual attributes it has when building the KB. If semantics are included in the GML, we will further extend our KB by the method introduced in Sec. 4.2.

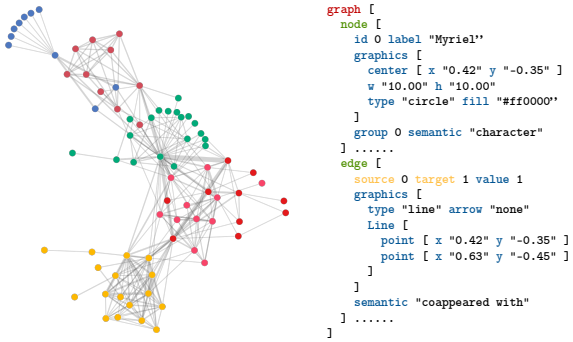


Figure 3: The samples of the graph visualization in GML formats. This format has the greatest keyword *graph* (red keys) with child objects *node* and *edge* (green keys) in no particular order. There are graph data (golden keys) and visual attributes (blue keys including x-position, y-position, width, height, shape, color, etc.) in the formats. This is an example of an undirected graph. A directed graph is stored in the same format but with an additional directed attribute specified in the *graph*.

Table 2: The data structure of the knowledge base. The entities, and primitive triplets are extracted from the GML. The derived triplets are computed and generated by primitive triplets and other derived triplets.

Entities		*<Graph Number> <Name>* *<Node Number> <Name>* *<Edge Number> <Name>*
Primitive Triplets	**Graph Data**	*<Node Entity> <Relation> <Node Entity>* *<Edge Entity> <Relation> <Node Entity>* *<Graph Entity> <Relation> <Node Entity>* *<Graph Entity> <Relation> <Edge Entity>*
	Visual Attribute	*<Node Entity> <Attribute> <Value>* *<Edge Entity> <Attribute> <Value>*
Derived Triplets	**Semantic**	*<Node Entity> <Semantic Relation> <Node Entity>* *<Edge Entity> <Semantic Relation> <Node Entity>* *<Graph Entity> <Semantic Relation> <Node Entity>* *<Graph Entity> <Semantic Relation> <Edge Entity>*
	Graph Analysis	*<Node Entity> <Property> <Value>* *<Edge Entity> <Property> <Value>* *<Community Entity> <Property> <Value>* *<Community Entity> <Relation> <Node Entity>* *<Community Entity> <Relation> <Edge Entity>* *<Graph Entity> <Property> <Value>* *<Graph Entity> <Relation> <Community Entity>*

4.2 Knowledge Base Expansion

In our formative study, we found that the questions people asked about graph visualizations are diverse and complicated. In order to answer more kinds of questions and improve the generalizability of our system, we designed a KB expansion module in our pipeline. We refer to these triplets computed and generated by primitive triplets and other derived triplets as derived triplets. There are two aspects of extension, one is the semantic extension, and the other is graph analysis extension.

Semantic Extension. Semantic extension is introduced for two purposes. The first purpose is to handle basic semantic questions caused by synonyms. For example, the question "*How many nodes*

does the graph contain?" is equivalent to "*What is the number of nodes included in the graph?*" in semantics. To address the first issue, we introduce the standard BERT model [15] in the question-answer model to obtain the context information in the question to solve. Our system also utilizes predicate aliases, which is the synonymous utterances for relationships. Specifically, we use WordNet [48] to first find all synonyms for "*include*" and "*connect*", and artificially designated multiple aliases for the relations.

The second purpose is to handle questions with domain-related utterances. Given a graph with background stories and concrete meanings, people will ask questions syntactically, but the answer is still derived from the entities and properties of the graph. For example, people will ask "*Who coappeared with Myriel?*" for a graph of co-appearance relationship. Its underlying graph analysis question is "*What nodes does Myriel link with?*". To address the second issue, our system utilizes a semantic description file containing configurable semantic information. This description file can be populated beforehand by graph providers, and can also be updated by users with an editor interface (as shown in Fig. 1a) to expand semantic information. Configurable objects including graphs, nodes, edges, and communities. All *<relation>*s can be artificially specified with any number of <semantic relation>. For example, graph providers can specify two aliases "*coapperance with*" and "*knows character*" for "*link with*" relationship, and our system will add two triplets using aliases into the KB. With this approach, we can extend the semantic understanding capabilities of our system while delivering high semantic extensibility that can be dynamically updated.

Graph Analysis Extension. Different from other types of charts, people often use terminologies (e.g., in/out-degree, community) from graph theory to analyze graph visualizations [77]. When the original graph data do not include these attributes, we will calculate them and add them to our KB as visual attributes. We use Louvain algorithm [6] to detect the best community partitions for the graph. The Louvain algorithm is an efficient unsupervised heuristic algorithm based on modularity optimization. The modularity can Q be defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta\left(c_i, c_j\right) \tag{1}$$

where A_{ij} indicates the weight of Node i and Node j; k_i represents the weight count of the edges linked with Node i; c_i denotes the community that Node i belongs to. $\delta\left(c_i, c_j\right)$ is set to 1 when Node i and Node j belong to the same community, or 0 when Node i and Node j belong to different communities. The Louvain algorithm repeats the two steps of modularity maximization and node merging until the modularity no longer changes. We convert each community partition into an entity as a child of the graph entity. We also add the relations between the community entities and the member entities (node and edge entities) to the triplets. Similarly, we also calculate the commonly queried properties [77] including degree, degree centrality, clustering, the bridges contained in the entire graph and each community, etc. The rules of all triplets are shown in Table 2.

4.3 Topic Entity Extraction

Topic entities are tokens in the questions that are grounded entities. They will be used to generate query graph in later process. For example, "*Courfeyrac*" is a topic entity of the question "*What is the number of people that coappeared with Courfeyrac?*". When receiving user input, the system will search for a query graph starting from the entity "*Courfeyrac*" to find candidate query graphs. Topic entity extraction will output the topic entity candidates $\mathcal{T} = \{< t_1, c_1 >, < t_2, c_2 >, \dots\}$ where t_i is the entity name, c_i is its confidence score. A confidence score is a decimal number between 0 and 1, serving as an indicator of how confident the model is with its prediction. A higher value indicates a more confident prediction for the result. We designed four steps for topic entity extraction to obtain $\mathcal{T}$ as shown in Figure 4.

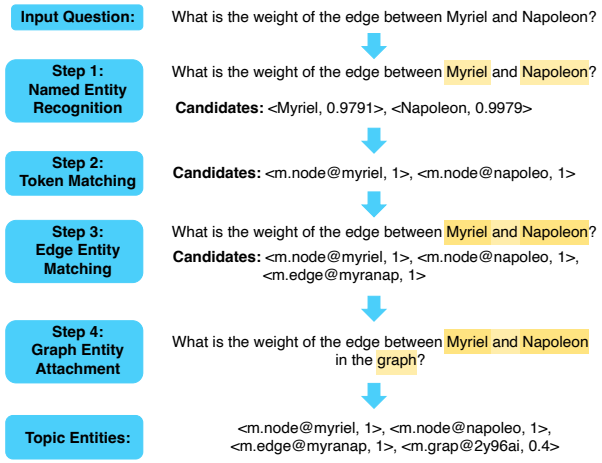


Figure 4: Topic entity extraction procedure. In Step 1, NER model outputs two named entities "*Myriel*" and "*Napoleon*" with confidence values of 0.9791 and 0.9979. In Step 2, we match the tokens in the questions with the entities in the KB. In Step 3, we use connectives to determine whether there is an edge topic entity in the question. In Step 4, we attach the graph entity to the candidates.

Step 1: Named Entity Recognition. We utilized NLP framework Flair [1] and the state-of-the-art 18-class NER model Flert [58] for named entity recognition. The NER model will output a prediction confidence score $\hat{c}$ for each token detected as a named entity. $\hat{c}$ is a probabilistic output value of the last layer of BiLSTM-CRF network used in the model [23]. We also use shallow parsing (chunking) model of Flair to extract noun phrases. We add them to token candidates to avoid missing some topic entities that are not related to named entities. We match these tokens with the entities in the KB. If entity t is matched, the tuple $< t_{\text{NER}}, \hat{c} >$ will be added to the topic entity candidates $\mathcal{T}$.

Step 2: Token Matching. To prevent the NER model from missing topic entities, we will first tokenize the question and generate n-grams. N-gram is a contiguous sequence of n tokens from a tokenized sequence. Here we generate 1 to 4 - grams. We match all the n-grams with entities in the KB. If an entity t is matched, $< t_{\text{token}}, c = 0.5 >$ will be added to $\mathcal{T}$. If the entity already exists in the set of entities generated by NER model with confidence score $\hat{c}$, we update the confidence score $c := \min(\hat{c} + 0.5, 1)$.

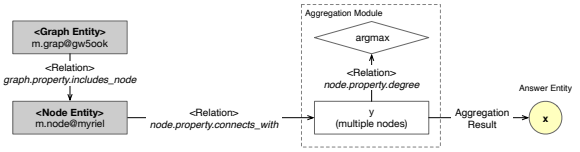


Figure 5: An example query graph for question "*Which node that links with Myriel has the highest degree?*". Shaded rectangles are n_{g}, the unshaded rectangle is n_{ug}, the diamond-shaped node is the n_{ag} and the circle represents n_{an}. The core relation path in this graph starts with the graph entity and ends at x.

Step 3: Edge Entity Matching. We have designed a set of rules for edge entity matching. Consider the question "*What is the weight of the edge between Myriel and Napoleon?*", NER model will output two seperated entities "*Myriel*" and "*Napoleon*" with their confidence score c_a and c_b. The question is more concerned with the edge connecting the two nodes than refer to each node separately. To handle this, when connectives such as "*and*", "*or*", "*to*" and "*&*" are detected in a question, we check whether the tokens before and after the connective are in the set of topic entity candidates. If so, we concatenate them to derive two edge entity names (e.g., "*Napoleon and Myriel*" and "*Myriel and Napoleon*") and check whether they exist in the KB. If a candidate exists, the edge entity $< t_{\text{edge}}, c = \max(c_a, c_b) >$ will be added to $\mathcal{T}$, where c_a and c_b represents the confidence of the topic entity before and after the connective respectively, calculated in aforementioned Step 1 and 2.

Step 4: Graph Entity Attachment. As all the questions are relevant to the graph, if the graph entity is still not present after the above steps, we will add the graph entity $< t_{\text{graph}}, c = 0.4 >$ to $\mathcal{T}$. This makes questions like "*How many people?*" and "*How many people in the graph?*" equivalent to our system. Also, if $\mathcal{T} = \emptyset$ after the above three steps, we add $< t_{\text{graph}}, c = 1 >$ to $\mathcal{T}$.

4.4 Automatic Question Answering

To query the correct answer out of the knowledge base, our Q&A model needs to generate a query graph [3] containing a core relartion path that starts from a topic entity and ends at the answer node. Since we have $\mathcal{T}$ constructed, we need to construct query graphs starting from each topic entity in $\mathcal{T}$ and pick one query graph to use its core relation path and generate answers. We define the query graph as $\mathcal{G} = \{N, E\}$. Node set $N = \{n_1, n_2, \dots\}$ where there are four types of nodes in the graph: $n_i \in \{n_{\text{g}}, n_{\text{ug}}, n_{\text{ag}}, n_{\text{an}}\}$. Grounded nodes are entities existed in the KB. Ungrounded nodes are nodes that do not represent a grounded entity. It can be used to represent multiple entities or an intermediate query result. Aggregation node can be used to perform aggregation operations and the answer node represent the query answer. Edge set $E = \{e_1, e_2, \dots\}$ where e_i is <*Relation*>s from the KB.

Our question-answer model is based on a reinforcement learning model proposed by Das et al. [14]. We trained this model on a large dataset of questions named ComplexWebQuestions [3] and a large knowledge base named Freebase [7]. The process of generating query graph is an iterative process, described as follows.

Table 3: The procedure of Aggregation operation. If a keyword is detected in the question, our model performs corresponding operations. The query language used for triplets querying is SPARQL [66]. "*<Core Relation Path>*" represents the core path of the current query graph, "*<Property>*" represents the operation key for the entities, and "*LIMIT 1*" indicates that the system retrieves the top-ranked entity.

Aggregation	Keywords	SPARQL
COUNT	['how many', 'how much', 'number of', 'count', 'quantity', 'amount', 'capacity', 'size']	SELECT COUNT (?<Entity>) WHERE { <Core Relation Path> }
ARGMIN	['min', 'minimum', 'minimal', 'least', 'lowest', 'smallest', 'littlest', 'shortest', 'poorest', 'merest', 'last', 'worst', 'undermost', 'lowermost']	SELECT ?<Entity> WHERE { <Core Relation Path> <Property> } ORDER BY (?<Property> <Value>) LIMIT 1
ARGMAX	['max', 'maximum', 'maximal', 'most', 'highest', 'biggest', 'top', 'topmost', 'supreme', 'largest', 'greatest', 'longest', 'best', 'first', 'utmost']	SELECT ?<Entity> WHERE { <Core Relation Path> <Property> } ORDER BY DESC (?<Property> <Value>) LIMIT 1

Candidate Query Graph Extension Let the candidate query graph set at iteration t to be $\mathcal{G}_t$. Initially, $\mathcal{G}_0 = \{G_1, G_2, \dots, G_{|\mathcal{T}|} | G_i = \{N = t_i, E = \emptyset\}\}$ where t_i is the i-th topic entity in $\mathcal{T}$. For each iteration, we try to extend each $G \in \mathcal{G}_t$. This can be done by extending a feasible relation to the graph or an aggregation operation (explained below). A feasible relation is all triplets whose head entity is the tail of the current g's core relation path. If there is only one topic entity in the graph, we will attach the relation and set the end of the newly attached relation to be n_{an}. If n_{an} already exists in g, we will change n_{an} to n_{ug}, execute the current query graph in KB and get all feasible relations according to the query result. These feasible relations will be attached and the end of the relation will be set to n_{an}. We used beam search [38] with beam size of 3 to narrow the search space in the iterative process and obtain $\mathcal{G}_{t+1}$, where ranking algorithm explained below.

Aggregation Operation Matching We designed an aggregation module to make it adapt to graph visualization. We predefined three keyword lists as shown in Table 3. If a keyword is detected in the question, an n_{ag} can be attached to a n_{ug} or the n_{an}. For example, for the question "*Which node that links with Myriel has the highest degree?*", one query graph that starts from the graph entity and attached with n_{ag} is shown in Figure 5. We also experimented training a BERT-based classification network [42] with WikiSQL [78] dataset to learn the aggregation intent of the question. However, it fails in terms of accuracy compared with keyword-based methods, whereas it also lacks the explainability and extensibility of keyword-based matching methods since the keyword list can be easily and purposefully expanded.

Query Graph Ranking For the candidate query graphs generated by Step 1 and Step 2, we rank these query graphs by a reinforcement learning model [14]. First, we generated a 6-dimension feature vector.

- BERT-based semantic matching: We use the standard BERT model to measure the semantic similarity between the tokenized question sequence s_q and query graph sequence s_g. s_g is generated by concatenating the ground entity names and relation names sequentially along the core relation path. For example, for the question in figure 5, s_q = [which, node, that, links, with, myriel, has, the, highest ,degree], s_g = [graph,

Table 4: The answer generation rule set of GVQA. AT stands for Answer Type, Agg. stands for Aggregation, Comm. stands for Community, Prop. stands for Property. We designed several rules for generating visual and auxiliary answers based on different types of questions and different types of source and target of the core relation path in their query graphs. The Example Question Pattern shown here is one possible pattern and does not represent any concrete question. Source Info Card can be a Node Info card, Edge Info Card or Community Info Card, depending on the source type of the query graph.

AT	Agg.	Source	Target	Example Question Pattern	Answers		
					Textual Answer	Visual Answer	Auxiliary Answers
Literal	NONE	Edge	Prop.	What is the weight of edge X?	Property Value	／ Highlight Edge ● Highlight Connected Nodes	／ Edge Info Card ● Node Info Card
	NONE	Node	Prop.	What is the degree of node X?	Property Value	● Highlight Node	● Node Info Card
	NONE	Comm.	Prop.	What is the clustering of communinty X?	Property Value	Highlight Community	Community Sub Graph Community Info Card
Entity	NONE	Node / Edge / Comm. / Graph	Edge	What bridges does the graph include?	<Edge Source> and <Edge Target>	／ Highlight Edge ● Highlight Connected Nodes Highlight Source (Except Graph)	／ Edge Info Card Source Info Card
	NONE	Node / Comm. / Graph	Node	What nodes are included in community X?	Node Label / ID	● Highlight Node Highlight Source (Except Graph)	● Node Info Card Source Info Card
	NONE	Node / Edge	Comm.	What community does X belong to?	Community ID	Highlight Community Highlight Source	Community Sub Graph Community Info Card ● Node (In Community) Info Card ／ Edge (In Community) Info Card
Statistical	ARGMAX / ARGMIN	Node / Edge / Graph	Edge	What is the edge with the highest weight?	<Edge Source> and <Edge Target> (Aggregated)	／ Highlight Edge & Connected Nodes Highlight Source (Except Graph)	Bar Chart of Prop.s among <Source> ／ Edge Info Card Source Info Card
	ARGMAX / ARGMIN	Node / Comm. / Graph	Node	What is the node with the highest degree?	Node Label (Aggregated)	● Highlight Node Highlight Source (Except Graph)	Bar Chart of Prop.s Among <Source> ● Node Info Card Source Info Card
	ARGMAX / ARGMIN	Node / Graph	Comm.	Which community has highest clustering?	Community ID (Aggregated)	Highlight Community	Bar Chart of Prop.s among <Source> Community Sub Graph Community Info Card
	COUNT	Node / Comm. / Graph	Edge	How many edges are in community X?	Number of Edges	／ Highlight Edges ● Highlight Connected Nodes Highlight Source (Except Graph)	／ Edge Info Cards ● Node Info Cards Source Info Cards
	COUNT	Node / Comm. / Graph	Node	What is the number of nodes connecting X?	Number of Nodes	● Highlight Nodes Highlight Source (Except Graph)	● Node Info Cards Source Info Cards
	COUNT	Graph	Comm.	What is the number of communities?	Number of Communities	⃠ None	Community Sub Graphs Community Info Cards

includes, node, myriel, connects, with, nodes, degree, highest]. We then feed the sequence "[CLS] s_q [SEP] s_g" to the BERT model to calculate their semantic similarity. We use the pre-trained model [71] and then fine-tune it.

- Entity Confidence: The accumulated confidence score c (derived in Sec. 4.3) of all topic entities.
- Entity Number: The number of n_g in the query graph.
- Entity Type Number: The number of entity types.
- Answer Entity Number: The number of n_{an} in the query graph.
- Aggregation Number: The number of n_{ag} in the query graph.

We feed the feature vector $\mathcal{V}$ of each candidate query graph q into a fully connected layer of the reinforcement model to obtain $p(q|Q)$. The training goal is to learn the policy function $p_\theta(q|Q)$, where θ represents the parameters in the model. We use the F1 score between the predicted answer and the ground truth as the reward.

After the aforementioned process, we obtain an optimal query graph for the question. By generating a SPARQL query from its core relation path and execute the query in the KB, we can obtain the textual answers.

4.5 Answer Generation

We then design an answer generation module to provide intuitive and insightful answers to the user. GVQA's answer is composed of three components: Textual Answers, Visual Answers, and Auxiliary Answers. Textual answers are generated in Section 4.4, while the Visual and Auxiliary Answers are generated by a set of rules determined by the answer type, the aggregation intent of the question, and the type of source and target in the query graph's core query path.

In terms of answers presentation, we design Entity Highlighting, Subgraphs and Information Card (hereinafter referred to as Info Card). Entity Highlighting is the most commonly used method, where we render red borders and shadows for the entity, scale up the node size or edge width, and display the label beside it to make it more noticeable. Highlighting a community will highlight all the nodes and edges belonging to it. Subgraphs are also designed for communities. We take the nodes in the community and the connections between them and form a separate subgraph for presentation to the user, providing an exploration of this community. Info Cards (shown in Figure 1.e) contain commonly-concerned properties, such as labels, degrees, or centralities of the entity, providing more information about the entities of the answer and the query process of

the question, since we will also provide Info Cards for the entities traversed along the core relation path.

In terms of answer type, we propose three types. The first type is Literal Answers. Literal Answers are responses to lookup questions such as "*What is the clustering of Myriel?*". The target of their core relation paths is a property value, providing a number or a string as the answer. We highlight the source entity for these attributes, and provide an info card as the auxiliary answers. The second type is Entity Answers. The target of their core relation paths represents a topic entity. This includes questions like "*What nodes does Myriel link with?*". We designed different highlight methods for them respectively and also provide info cards for the target and source of the query graph. The third type is statistical Answers. These answers are responses to questions with aggregation intent like "*How many communities does the graph have?*" or "*Which node has the highest degree?*". The former includes *COUNT*, while the latter includes *ARGMAX*. These answers are in the form of entities or properties for questions with aggregation intent. We will provide highlights and info cards, and draw a bar chart for *ARGMAX* and *ARGMIN* questions, containing the ranking of properties in the scope of the question concerned, in corresponding order.

The generated answers also varies depending on different types of aggregation intent and core relation paths. Specifically, we consider the types of entities involved in the core relation path, including nodes, edges, communities, graphs, and property values. Based on these factors. Based on these factors, different combinations of visual and auxiliary answers will be presented accordingly. The detailed rule sets are listed in Table 4.

5 APPLICATIONS

There are many scenarios where automated question-answering about graph visualizations can be applied. In this section, we introduce three cases with our pipeline: the Les Misérables coappearance character network, the network of purchased political books, and the coauthorship network of scientists. They demonstrate the effectiveness of our pipeline for undirected graphs, directed graphs, and large graphs, respectively. We deploy our system on a PC with Intel Core i7-11850H @ 2.50GhZ, NVIDIA RTX A3000 Laptop GPU 6GB, and 32 GB of memory. The reinforcement learning framework was implemented based on Pytorch [55]. We set the hidden hidden size as 768, the dropout ratio as 0.1, the initializer range as 0.02, the vocabulary number as 30522, the number of attention heads as 12, the number of layers as 6. The activation function is GELU [19]. Figure 6 shows a variety of samples generated by our pipeline.

5.1 Scientific Study of Narratives

Some literary or sociological researchers conduct structured narrative research through the relationships of characters in a novel [49]. A good story, although often fictional, is tightly coupled with the real environment through the frequent interactions of the characters and going through a series of events. Analyzing the structural relationships of the characters therein can lead to an understanding of the narrative that may lead to new insights into various social phenomena and literature. Les Misérables [24] is a famous novel published by French writer Victor Hugo in 1862, covering the Napoleonic Wars and the decades that followed. There is a network graph [32] of the coappearance of the characters in the novel. People can analyze the relationships of characters by analyzing the network of relationships in which the characters appear together.

We input the question "*Which node has the highest degree in the graph?*" to our pipeline as shown in Figure 6 (Q1). Our model will highlight the node with the highest degree in the graph, and sort all the nodes by degrees in a bar chart visualization output. The node with the highest degree is *Valjean*, who is the protagonist of the novel. We then input the question "*What is the number of characters who co-appeared with Valjean?*" and "*What characters coappeared with Valjean?*". We obtain the answer "*36*" and highlights the nodes, which link with the node Valjean. These nodes represent the characters in the novel that are related to Valjean. In the visual answer, we can sort out the characters who are directly related to the protagonist. We find a striking edge that connects the node Valjean and the other node Cosette. Cosette is the heroine of the novel and the adopted daughter of Valjean. We then input a question with semantics "*How many times did Valjean and Cosette co-appear?*". Our pipeline selects the entity on the edge between Valjean and Cosette after topic entity extraction, and then selects the core path leading to the edge weight in the GVQA module. We finally obtain the weight of edge "*31*" for the textual answer and the highlight of the edge for visual answers as shown in Figure 6 (Q2). The weight represents the number of coappearance of Valjean and Cosette in the novel, which rationally explains how important their relationship is from the frequency of appearance.

Additionally, we asked "*What bridges does the graph have?*". Our pipeline highlights all edges that are bridge edges (The removal of these edges will increase the number of connected components) and show their attributes in the auxiliary panel. We can use these bridge edges to analyze what characters played the marginal supporting roles in the relationship among the characters.

5.2 Product Sales Analysis

There is a network [34] that represents books about American politics around the 2004 presidential election sold at Amazon.com. The edge stands for book co-purchasing, specially, an edge from book A to books B indicates that customers tend to purchase book B after purchasing book A. For merchants, analyzing such networks can quickly capture market trends and adjust inventory. For politicians, they can also adjust their political strategy.

For this directed graph with sources and targets, we ask the question "*Which node has the highest out degree in the graph?*". The topic entity extraction module extracted the graph entity from the question. The GVQA searches a core path referring the out degree. Eventually, we obtain the answer *Arrogance* as shown in Figure 6 (Q5). This suggests that the bestseller drove sales for many political books. We then input the question "*What is the node with the highest in degree that links with Arrogance?*". The model output the answer *Off wit their heads*. We now know that amongst the books that drove people to buy Arrogance, Off with thiermheads is the book that's most likely to be the purchasing destination.

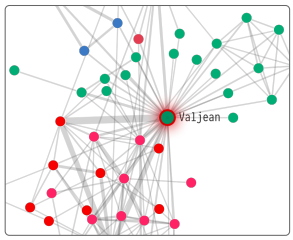
Les Miserables
Node: 77, Edge: 254, Undirected

Q1. Which node has the highest degree in the graph?

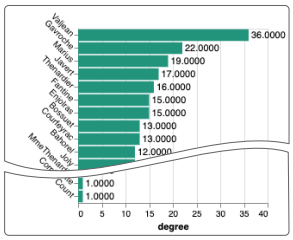
Textual Answer: **Valjean**
SPARQL:

```
SELECT ?e1 WHERE {
m.grap@hhts3e graph.property.contain_nodes ?e1 .
e1? node.property.degree ?e2} ORDER BY DESC(?e2) LIMIT 1
```

Visual Answer:



Auxiliary Answers:

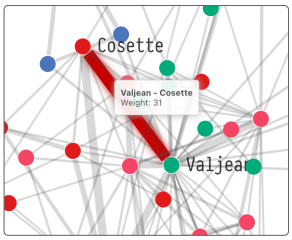


Q2. How many times did Valjean and Cosette co-appear?

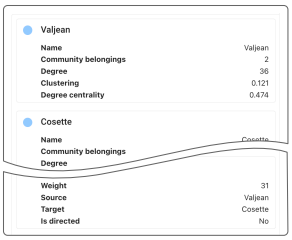
Textual Answer: **31**
SPARQL:

```
SELECT ?e1 WHERE {
m.edge@valacos edge.property.importance ?e1}
```

Visual Answer:



Auxiliary Answers:



(Les Miserables is the coappearance network of characters in the novel *Les Miserables*)

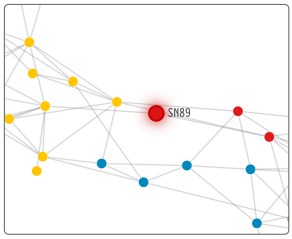
Dolphins
Node: 62, Edge: 159, Undirected

Q3. What is the node with the lowest centrality in community 0?

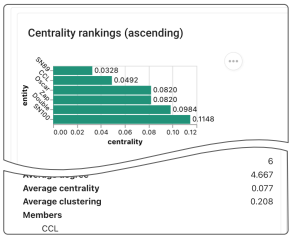
Textual Answer: **SN89**
SPARQL:

```
SELECT ?e1 WHERE {
m.comm@saivpn community.property.contain_nodes ?e1.
e1? node.property.degree_centrality ?e2} ORDER BY (?e2) LIMIT 1
```

Visual Answer:



Auxiliary Answers:

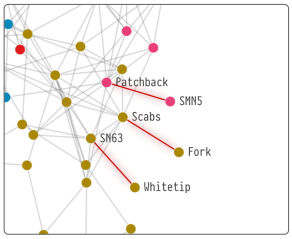


Q4. What bridges does the graph include?

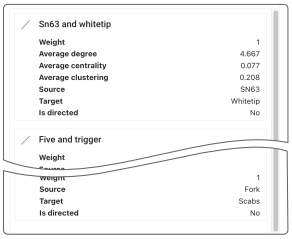
Textual Answer: **SN63 and Whitetip, Five and Trigger, ...**
SPARQL:

```
SELECT ?e1 WHERE {
m.grap@aket_i graph.property.include_bridges ?e1}
```

Visual Answer:



Auxiliary Answers:



(Dolphins is an undirected social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand)

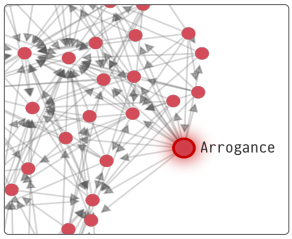
Political Books
Node: 105, Edge: 441, Directed

Q5. Which node has the highest out degree in the graph?

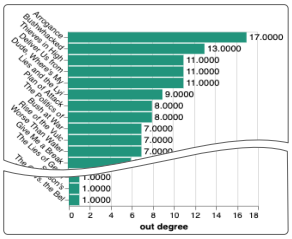
Textual Answer: **Arrogance**
SPARQL:

```
SELECT ?e1 WHERE {
m.grap@1ge2wr graph.property.contain_nodes ?e1.
e1? node.property.out_degree ?e2} ORDER BY DESC(?e2) LIMIT 1
```

Visual Answer:



Auxiliary Answers:

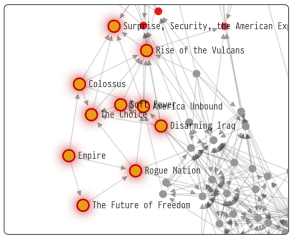


Q6. What books are included in community 2?

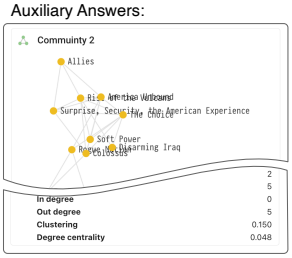
Textual Answer: **Rogue Nation, the Choice, Rise of the Vulcans...**
SPARQL:

```
SELECT ?e1 WHERE {
m.comm@6ppfnc community.property.node_members ?e1}
```

Visual Answer:



Auxiliary Answers:



(Political Books is network of books about US politics published around the time of the 2004 presidential election and sold Amazon.com)

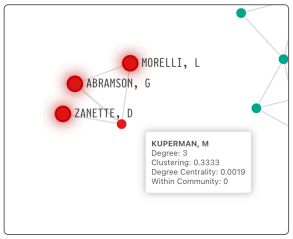
Net Science
Node: 1589, Edge: 2742, Undirected

Q7. Who have co-authored with KUPERMAN, M?

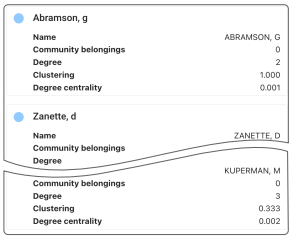
Textual Answer: **ABRAMSON, G, ZANETTE, D, MORELLI, L**
SPARQL:

```
SELECT ?e1 WHERE {
m.node@kuperman,_m node.property.coauthor_with ?e1}
```

Visual Answer:



Auxiliary Answers:

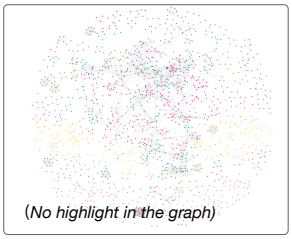


Q8. How many communities does the graph have?

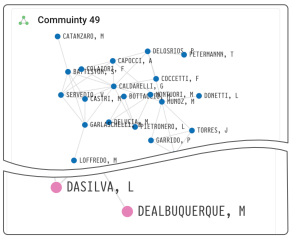
Textual Answer: **405**
SPARQL:

```
SELECT COUNT(?e1) WHERE {
m.grap@ets3ss graph.property.community_members ?e1}
```

Visual Answer:



Auxiliary Answers:



(Net Science is a large graph describing the coauthorship of scientists working on network theory and experiment)

Figure 6: The samples of visual answers generated by our system.

5.3 Analysis of scientific research hotspots

With the gradual development of open science, researchers are no longer independent individuals, and their collaboration promotes the basic understanding of scientific research topics. [68]. Analyzing the network of co-authorship can help researchers obtain research hotspots, but such networks are often large. Manually retrieving and sorting data for a large-scale graph will be a tedious task. Newman [51] has compiled a coauthorship network of scientists who worked on network science with 1589 nodes and 2742 edges. We use this graph to demonstrate our system's ability to handle large graphs.

First, we are curious about who is the most important writer in this network. The degree centrality is a direct measure of node importance. The higher the degree centrality, more important the node is in the network. We ask the question "*What is the node with the highest centrality in the graph?*" as shown in Figure 1. Our model applied the aggregation module since a *ARGMAX* keyword "*highest*" was detected in the question. The model queries all the node entities in the graph through the graph entity and sorts the nodes in descending order according to the degree centrality. We finally obtain the answer *BARABASI, A*, which represents Albert-László Barabási, an American physicist known for his research on network theory. We then want to know the most influential authors of Prof. Barabási's collaborators. We ask "*Which node has the highest degree that links with BARABASI, A?*" and obtain the answer *JEONG, H*, which represents Hawoong Jeong, a scientist who has done a large number of scale-free network research with Prof. Barabási. We also learn that the weight of their cooperation is 4.225 through the question "*What is the weight of the edge between JEONG, H and BARABASI, A?*".

By applying the Louvain algorithm for community discovery in our knowledge base expansion module, our pipeline can automatically answer community analysis questions. We can learn that this graph has 405 communities by asking "*How many communities does the graph have?*" as shown in Figure 6 (Q8). The nodes in the graph are highlighted according to their community belongings, and the visualization and visual properties of each community can be viewed in the auxiliary panel with an overview. Each community represents a group of scientists who work closely together. We continue to focus on Prof. Barabási and ask what community he belongs to. When we ask the question "*Which community does BARABASI, A belong to?*", our model will output the answer *Community 10* and highlight all the scientist nodes in this community as the visual answer. We can know the number of scientists in this group via the community attributes in the auxiliary panel or by asking "*How many scientists does Community 10 have?*". We get an answer with a relatively large value 95. By inspecting the average degree centrality of this community, we also gain the insight that the researchers working closely in this community have significant impact to other researchers.

6 USER STUDY

To validate the efficiency of our pipeline and measure the usefulness, and transparency of the generated visual answers, we conducted a user study. We propose two hypotheses:

H1: It will take users less time and effort to answer graph visualization questions using our automatic question answering system than using traditional network exploration tools [4], when both methods can find the answers.

H2: Users will find the combination of textual and visual answers are either better than or at least as good as text-only answers in terms of usefulness and transparency, when GVQA can provide correct answers. Usefulness refers to whether users accept the answer and obtain the information they want. Transparency refers to whether users feel the answers are explainable, reliable, and credible.

Different from the usefulness and transparency defined by Kim et al. [31], our hypotheses is under the condition that GVQA can give the correct answer. In fact, using both a traditional network exploration tool and GVQA may fail to answer the question. In the user study section, we want to access these two aforementioned hypotheses regarding the cost of finding answers and quality of our user interface. Therefore, we ensure that users can complete the QA task itself and provide meaningful feedback, and the answers can be found using both tools. In terms of GVQA's accuracy, we discuss it in Sec. 7.1.

Design: We first inform our participants with the statement that their answers will be only used for academic purposes. We then introduce question-answering about graph visualizations to ensure that each participant understands the motivation of our work. We then briefly introduced Gephi [4], an powerful open-source software for exploring and manipulating network graphs to our users. While Gephi has comprehensive graph visualization capabilities, we tell the participants that our main focus is the effort used to find the answer. We picked 12 correctly-answered graph-question pairs from our corpus to ensure that they can finish the answering task and provide meaningful feedback in terms of the cost of efforts and the quality of our user interface. These 12 graph-question pairs came from the three graphs we introduced in Sec. 5. These questions are divided into three sets: the first set (Q1-Q4) is about Les Misérables character relationships ($V = 77, E = 254$); the second set (Q5-Q8) is related to political books ($V = 105, E = 441$), and the third set (Q9-Q12) is about scientist coauthorship ($V = 1589, E = 2742$). Each set contains four questions including a lookup structural question, a lookup question with semantics, a compositional structural question, and a compositional question with semantics. These sets of questions will be presented in random order.

We divided our user study into two parts. In the first part, we showed participants 12 graph-question pairs without their answers and ask them to answer these questions with two different tools: Gephi and GVQA. They are free to use all Gephi's features including graph layouting, statistic tools, data laboratory and UI to find the answers. Half of the participants used Gephi to answer the questions first, while the other half used GVQA first. We recorded their time consumption and mouse click for each question using both systems separately. In the second part, we present the textual answers and visual answers of these graph-question pairs to participants via a questionnaire. Participants will evaluate the textual answers and visual answers in terms of usefulness and transparency with a 5-point Likert scale, scoring from 1 to 5. Participants were not informed of how the scores were assigned to the options. In the

Table 5: The time consumption comparisons of GVQA and Gephi. Each result is represented as mean value μ (± 95% confidence intervals). Metrics show that our NLI automatic question answering system takes less time than traditional network exploration systems.

Question	Method			
	Gephi		GVQA	
	Time/Second	Mouse Click	Time/Second	Mouse Click
Set 1 (Q1-Q4)	63.45 (57.25-69.65, p<0.001)	57.4 (51.5-63.4, p<0.01)	13.42 (12.35-14.49, p<0.001)	8.4 (8.0-8.8, p<0.01)
Set 2 (Q5-Q8)	55.69 (50.54-60.84, p<0.001)	50.9 (46.0-55.8, p<0.01)	13.36 (12.33-14.38, p<0.001)	8.6 (8.2-9.1, p<0.01)
Set 3 (Q9-Q12)	63.15 (58.14-68.16, p<0.001)	58.2 (52.7-63.6, p<0.01)	23.63 (22.62-24.63, p<0.001)	8.7 (8.3-9.1, p<0.01)

questionnaire, they were asked to indicate to what extent they agreed with the textual and visual answers.

Recruitment: We recruited 40 participants (μ_{age} = 27.9 years) who do not overlap with the participants of our formative study from the campus. 30 participants have previously engaged in data visualization or graph theory research while the remaining participants did not have prior knowledge in these fields but were able to comprehend our work through our explanation. Due to the COVID-19 pandemic, all user study was conducted online in a contactless manner.

Assessing H1: We recorded the consumed time and number of mouse clicks (including the left and right buttons) while participants answered 12 questions about graph visualizations by Gephi and GVQA. In this stage of the experiment, we pay more attention to the time the participants consumed to deliver the answer, so all answers including the incorrect ones were counted. Time metrics of our GVQA include the initial loading of the network graph and inputting questions by users. The results are shown in Table 5. We found that using our system to answer these questions significantly reduced the time consumption (Mann-Whitney $U_1 = 889, p < 0.001$; $U_2 = 1457, p < 0.001$; $U_3 = 711, p < 0.001$) and number of mouse clicks ($U_1 = 191.5, p < 0.001$; $U_2 = 469, p < 0.001$; $U_3 = 3.5, p < 0.001$). As such, we accept H1. We elucidated the time efficiency of our pipeline in more detail in Sec. 7.2.

Assessing H2: Participants took an average of 2112.5 (±95% CI: 1672.0 − 2552.9, $p < 0.001$) seconds to complete the questionnaire.

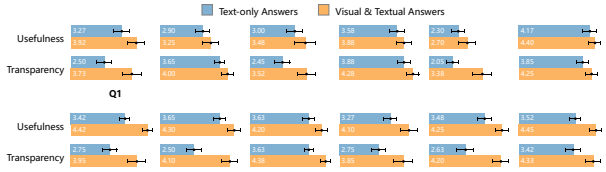


Figure 7: The results of 5-point Likert scale on usefulness and credibility. Q1-Q4 are questions and answers about undirected graphs, Q5-Q8 are questions and answers about directed graphs, and Q9-Q12 are questions and answers about large graphs. Q1, Q5, Q9 are lookup structural questions. Q2, Q6, Q10 are lookup questions with semantics. Q3, Q7, Q11 are compositional structural questions. Q4, Q8, Q12 are compositional questions with semantics. Circles depict group averages (± 95% confidence intervals).

The results of our questionnaire are shown in Figure 7. For text-only answers, we obtain an average of 3.35 (±95% CI: 3.26 − 3.44, $p < 0.001$) on usefulness, and a average of 3.00 (±95% CI: 2.90 − 3.10, $p < 0.001$) on transparency. For the combination of textual and visual answers, we obtain an average of 3.95 (±95% CI: 3.84 − 4.05, $p < 0.001$) on usefulness, and a average of 4.00 (±95% CI: 3.89 − 4.10, $p < 0.001$) on transparency. The mean values ≥ 3 indicates that our final output answers were accepted and believed by users. We also came to the conclusion that introducing visual answers improved the answers in terms of usefulness ($U = 7652, p < 0.001$) and transparency ($U = 8483.5, p < 0.001$). Therefore, we accept H2. In interviews after the questionnaire session, participants reported that they believed that the visual answers delivered more intuitive information and sometimes included additional information not found in the textual answers. In terms of concrete analysis, the main advantage of the visual answers is their transparency. This also explains the greater variation in scores assessing transparency compared to those assessing usefulness. For instance, the question Q1 is "*What is the degree of Marguerite?*", a lookup structural question with a literal answer of "*12*". In this case, our module highlights the node *Marguerite* and show its visual attributes on the auxiliary panel. Many participants felt that textual answers were sufficient as the answer to this question, so visual answers were not as useful for this type of question. However, the users can quickly perceive that the node are linked with two edges, making the answers more persuasive than solely textual answers.

We conduct a more specific analysis of the results by question. We found that Q2 and Q6 have a similar result on usefulness ($U_2 = 47, p = 0.15$; $U_6 = 45, p = 0.30$) and transparency($U_2 = 49.5, p = 0.05$; $U_6 = 51, p = 0.03$). Q2 and Q6 are both lookup questions with semantics. For this type of questions, the answers are always literal answers. Users feel that textual answers are sufficient for their needs, so the introduction of visual answers do not change much. Q5 is a lookup structural question with small changes in usefulness ($U_5 = 48, p = 0.09$) and large changes in transparency ($U_5 = 63.3, p < 0.001$). These lookup structural questions are often related to the properties of graphs. Although textual answers meet the needs of users, visual answers and auxiliary windows can make the answers more credible and understandable. We found that visual answers to Q7, Q8, Q11 and Q12 performed Well, both in terms of usefulness ($U_7 = 67, p < 0.001$; $U_8 = 56, p < 0.001$; $U_{11} = 58, p < 0.001$; $U_{12} = 63, p < 0.001$) and transparency ($U_7 = 62, p < 0.001$; $U_8 = 67.5, p < 0.001$; $U_{11} = 66, p < 0.001$; $U_{12} = 60, p < 0.001$). They are all compositional questions. This

is mainly because compositional questions are more difficult than lookup questions, and textual answers cannot meet the needs of users in most cases. Users also made suggestions for the visual answers of compositional questions in the interview, and we will enrich the form of visual answers in the future open version.

7 EVALUATION

We evaluate our pipeline from two aspects, including the accuracy of question answering and the time performance.

7.1 Question-Answering Accuracy

In formative study, we collected 561 graph-question pairs. Since the questions we collect are all objective questions, the answers are unique. We consider the output textual answer is a correct answer if it is equal to the human answer. The accuracy represents the number of questions answered correctly divided by the total number of questions. We compare our pipeline with two methods. The first method is the GVQA without the reinforcement layer. During the query graph ranking process, we only use BERT-based semantic similarity to rank the candidate query graphs instead of the reinforcement learning-based method. The second method is the KBQA model [36] designed for answering natural questions for general knowledge bases. For the fairness of the comparison, we keep our design of GML extraction module. Additionally, since the aggregation operation is keyword-based, we also considered answers from the baseline models to be correct if they obtained the correct entity without aggregation. For example, if the baseline model outputs all node entities in the graph instead of the number of nodes when faced with the question "*How many nodes in the graph?*", we would consider this to be a correct answer. Besides, all information about graphs was still presented in the form of triplets.

As shown in Figure 8, our evaluation results showed that GVQA achieved an overall accuracy of 90.73%, with high performance in semantic (88.39%), structural (91.9%), lookup(90.23%) and compositional (91.04%) questions. Additionally, GVQA performed well in both lookup and compositional questions that combined semantic and structural information. In comparison, the KBQA model only achieved an overall accuracy of 27.09%. Questions like "*What*

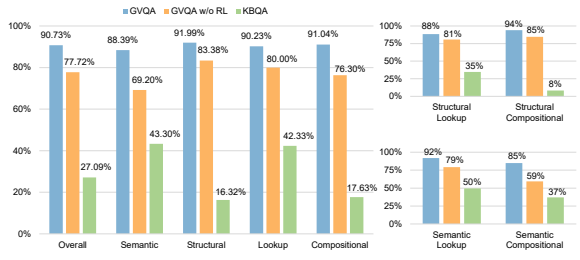


Figure 8: The results of our evaluation experiments. We compared our pipeline (blue bar) with GVQA w\o RL (without Reinforcement Layer, orange bar) and a KBQA model (green bar) on our 561 graph-question pairs. The Y-axis represents the accuracy of question answering. The X-axis represents different types of questions.

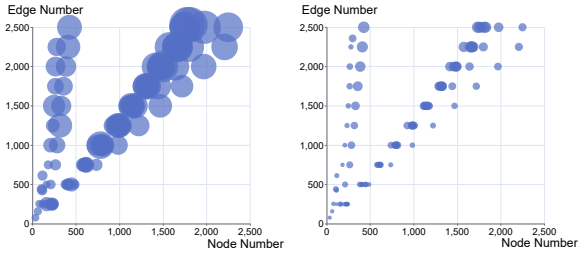
Table 6: Error analysis for GVQA. For the 52 questions that GVQA failed to answer, we counted the question type and the cause of errors. The last column and the last row are the accumulated number of each cause of error and each question type respectively.

Cause of Error \ Question Type	Structural Look up	Semantic Look up	Structural Compositional	Semantic Compositional	Accu. Percentage
Query Graph Ranking Errors	7	6	10	5	28(53.85%)
Fail to Extract Semantics	0	2	0	14	16(30.77%)
Information Absent in KB	5	2	1	0	8(15.38%)
Accu. Percentage	12(23.08%)	10(19.23%)	11(21.15%)	19(36.54%)	

community does Myriel belong to?" or "*Which node has the highest clustering value?*" are outside KBQA's capability, and KBQA will simply output Myriel's neighbors or a list of all nodes in the graph. The significant difference in accuracy can be attributed to the knowledge base expansion module we designed according to our formative study, which is inspired by the previous work [31, 65]. It provides the system with availability of more information and enables it to better cope with questions. Among the structural questions, most of them are querying the properties of graph elements, such as community partitions or the degree of nodes. The graph analysis extension module in our pipeline expanded computed properties of graph elements into the knowledge base, enabling GVQA to answer these questions that the KBQA is incapable of. Besides, our question set focuses more on the objective questions of graph analysis, which is where our task differs from Table QA [25].

When compared to GVQA without the reinforcement learning layer (GVQA w/o RL), our pipeline also showed a significant improvement in overall accuracy, as well as in most question types. When the reinforcement layer is not used, GVQA can not find the correct answer for some compositional questions like "*What is the node with the maximum centrality that links with Beak?*", and will output Beak's centrality instead. This demonstrates the contribution of the reinforcement learning layer to GVQA's performance, as it enables the system to make more accurate predictions when trying to find the optimal query graphs. Overall, our results show that GVQA is a highly promising approach for answering natural questions about graph visualizations.

We then analyzed the failed 9.73% (52) questions, the results are shown in Table 6. We classify the cause of error into three categories. The first and most common cause is Query Graph Ranking Error (53.85%), where the system uses the wrong query graph as the optimal graph during the query graph ranking phase. For example, GVQA fails to answer question "*What bridge edges does the graph contain?*" and outputs all the edges instead of bridges as the answer. The second cause is Failing to Extract Semantics (30.77%), where the system fails to understand the semantics related to the background of the graph. For example, QVQA fails to understand that the question "*Which scientist does Andrade Mad collaborate with most frequently?*" is actually asking for the node with the highest edge weight connecting Andrade Mad and just output all of the nodes connecting Andrade Mad as the answer. The third cause is Information Absent in KB, where the user is asking us for information that does not exist in the KB. For instance, users may ask "*What is the meaning of these nodes?*". GVQA will just output



(a) Time Consumption I **(b) Time Consumption II**

Figure 9: The results of evaluation experiments on time performance. The Y-axis represents the number of edges. The X-axis represents the number of nodes. The size of the scatter represents the amount of time the graph visualization spent in the pipeline.

all nodes in the graph as the answer. In terms of question types, Semantic Compositional questions have the highest failure rate (36.54%), due to their structural and semantic complexity, while the other three types of questions have failure rates of around 20% each. Based on our findings, the Q&A accuracy can be potentially improved by either improving the accuracy of query graph ranking process or increasing our KB's information capacity. This could involve implementing more sophisticated ranking algorithms capable of handling complex and multi-hop queries or collecting more relevant and diverse information in the KB.

7.2 Question-Answering Time Performance

To explore the QA time performance of our pipeline, we conducted a series of experiments in which we measured the time it took for our pipeline to provide answers. In our experiments, we used a set of experimental datasets that were derived by downsampling larger graphs at equal intervals, using the number of edges as the downsampling metric. For graphs with more than 2500 edges, we randomly sampled a number of edges at an interval of 250 and preserved the nodes connected to them as members of the subgraph. This allowed us to create a total of 96 graph visualizations for our experiments. We recorded the time consumption for two major phases: (1) Consumption I: the time it takes for the pipeline to answer a question for a graph visualization for the first time, including initializing the graph data and running all modules of the pipeline; (2) Consumption II: the average time it takes for the pipeline to answer a subsequent question for graph visualizations, including all modules of the pipeline except knowledge base construction and system initialization.

The results of our experiments are shown in Figure 9. We found that the principal time consumption of our system is the knowledge base construction at the first run and graph data initialization. The average time complexity of the entire pipeline is $O(n)$, where n mainly considerates the scale of the graph. When dealing with large-scale graphs with approximately 2500 edges, the average time cost of Consumption I is 17.15 seconds. We also discovered that subsequent question answering tasks took substantially less time, since the knowledge base data has been loaded into the memory. The time complexity for visual answer generation is $O(1)$, and the time complexity of aggregation is $O(n)$. Since the time consumption of query graph ranking using reinforcement learning is stably within one second for any graph of this scale, and the process of visual answer generation and aggregation operation using keyword-based method has low time complexity, the time cost of Consumption II only has slight fluctuations. The average time for answering subsequent questions (not the first run) about a graph visualization at the scale of 2500 edges is 2.75 seconds.

Overall, our experiments show that our pipeline has an average time complexity of $O(n)$ where n is most significantly influenced by the scale of the graph associated with the graph visualizations. Our system is able to provide answers within an acceptable amount of time, even for large graphs with up to 2500 edges.

8 DISCUSSION

8.1 Graph-based QA Versus Relational QA

GVQA is based on a graph-based database querying, while Table QA is based on a relational database. There are several differences between graph and relational querying. One of the main differences is that graph databases store the relationships between data as data, whereas relational databases infer relationships between columns of data tables. This means that in a graph database , it is easier to query and manipulate data that is related in complex ways, because the relationships between data points are explicitly stored in the database. Another difference is that graph databases are often more flexible and scalable than relational databases. Graph databases do not rely on fixed table schemas, so they can handle data that is structured in complex or evolving ways. In contrast, relational databases are typically more structured and rigid. They are based on the relational model, which requires that data be organized into fixed schemas. This makes it easier to query and manipulate data that follows a predictable structure, but can make it difficult to handle data that is complex or unstructured.

Our pipeline needs to convert a graph visualization to a database. For the question "*who is the subcommittee chair of the Visualization field of CHI2023?*", a relational query should be used by joining together different tables in the database that contain information about the CHI2023 conference, the different subcommittees within the conference, and the chairs of those subcommittees. However, we only need one graph for a graph-based database. In addition, using a graph-based database is more suitable for real-time updated graph visualization data because of scalability of knowledge bases. Most of the graph visualization questions focus on the complex relationship between nodes, which makes graph-based databases more suitable for graph visualization, while relational databases are better suited to applications that involve more structured data, such as bar charts and line charts.

8.2 Limitation and Future Work

The current pipeline of GVQA system retains some limitations.

The Type of Questions. Although the corpora we collected through our formative study covered lots of scenarios and aspects of graph analysis, we are aware that we haven't addressed all possible questioning perspectives in real cases. Besides, current version of GVQA cannot answer true or false questions, which is also a

limitation of many current question answering pipelines [35, 42] designed for other types of charts. Additionally, there are some questions with high-level semantics that cannot be answered in our corpus. For example, due to the limitation of our rule definition and knowledge base capability, our pipeline cannot answer the question "*Who is the most lonely?*" on the dataset Karate [75], which is a social network of friendships in a karate club. We plan to make our system publicly available and collect more graph-question pairs from the visualization community and further extend our system's Q&A ability.

The Type of Visual Answers. Currently, visual answers are intuitive enough but lacks interaction. Most of them are highlights or annotations. Visual answers to complex questions are static subgraph structures and bar charts. Our user study also shows that there is room for improvement on usefulness, especially for literal answers. We plan to enrich the form of our visual answers and enhance their interactivity, such as fisheye focus, more graph layouts, question-answering based on the previous visual answers, etc., in our future research.

The Type of Visualizations. Our pipeline currently only supports network graph visualizations. However, the ideology of using knowledge bases to accomplish Q&A tasks can be applied to other kinds of visualization charts such as geovisualization as well. We will explore the generalizability of our pipeline in the future.

9 CONCLUSION

We proposed a pipeline to answer questions about graph visualizations and generated visual answers. We built a natural language interface to help users analyze graphs. We further conducted a formative study to learn what questions people would ask and how they ask them. Our evaluation of the corpus collected by formative study demonstrates that GVQA can correctly and efficiently answer diverse questions about graphs with different semantics and scales. Our user study shows the combination of visual answers and textual answers are more informative than textual answer in usefulness and credibility. We also discuss the possible improvements of our work in the future.

ACKNOWLEDGMENTS

This work was supported by the NSFC under Grant 62072183, the Shanghai Committee of Science and Technology, China (Grant No. 22511104600), and NSSFC under Grant 22ZD05. Chenhui Li and Changbo Wang are the corresponding authors.

REFERENCES

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*. 1638–1649.

[2] Sriram Karthik Badam, Zhicheng Liu, and Niklas Elmqvist. 2018. Elastic documents: Coupling text and tables through contextual visualizations for enhanced document reading. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 661–671.

[3] Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2503–2514.

[4] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: an open source software for exploring and manipulating networks. In *Proceedings of the international AAAI conference on web and social media*, Vol. 3. 361–362.

[5] Leilani Battle and Carlos Scheidegger. 2020. A structured review of data management technology for interactive visualization and analysis. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 1128–1138.

[6] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[7] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[8] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075* (2015).

[9] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 data-driven documents. *IEEE TVCG* 17, 12 (2011), 2301–2309.

[10] Ritwick Chaudhry, Sumit Shekhar, Utkarsh Gupta, Pranav Maneriker, Prann Bansal, and Ajay Joshi. 2020. Leaf-qa: Locate, encode & attend for figure question answering. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 3512–3521.

[11] Yang Chen, Scott Barlowe, and Jing Yang. 2010. Click2annotate: Automated insight externalization with rich semantics. In *2010 IEEE symposium on visual analytics science and technology*. IEEE, 155–162.

[12] Yang Chen, Jing Yang, Scott Barlowe, and Dong H Jeong. 2010. Touch2Annotate: Generating better annotations with less human effort on multi-touch interfaces. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, 3703–3708.

[13] Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. UHop: An Unrestricted-Hop Relation Extraction Framework for Knowledge-Based Question Answering. In *Proceedings of NAACL-HLT*. 345–356.

[14] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning. In *International Conference on Learning Representations*.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[16] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G Karahalios. 2015. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th annual acm symposium on user interface software & technology*. 489–500.

[17] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[18] Jonathan Harper and Maneesh Agrawala. 2017. Converting basic D3 charts into reusable style templates. *IEEE transactions on visualization and computer graphics* 24, 3 (2017), 1274–1286.

[19] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[20] Michael Himsolt. 1997. *GML: A portable graph file format.* Technical Report. Technical report, Universitat Passau.

[21] Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. 2017. Applying pragmatics principles for interaction with visual analytics. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 309–318.

[22] Sen Hu, Lei Zou, and Xinbo Zhang. 2018. A state-transition framework to answer complex questions over knowledge base. In *Proceedings of the 2018 conference on empirical methods in natural language processing*. 2098–2108.

[23] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *ArXiv* abs/1508.01991 (2015).

[24] Victor Hugo. 1863. *Les misérables...* C. Lassalle.

[25] Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022. OmniTab: Pretraining with Natural and Synthetic Data for Few-shot Table-based Question Answering. *arXiv preprint arXiv:2207.03637* (2022).

[26] Huang Jieying, Xi Yang, Hu Junnan, and Tao Jun. 2023. FlowNL: Asking the Flow Data in Natural Languages. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–11.

[27] Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5648–5656.

[28] Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, and et al. 2018. FigureQA: An Annotated Figure Dataset for Visual Reasoning. In *Proc. ICLR'18*. arXiv:1710.07300 http://arxiv.org/abs/1710.07300

[29] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A diagram is worth a dozen images. In *European conference on computer vision*. Springer, 235–251.

[30] Daesik Kim, YoungJoon Yoo, Jee-Soo Kim, Sangkuk Lee, and Nojun Kwak. 2018. Dynamic graph generation network: Generating relational knowledge from diagrams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern*

Recognition. 4167–4175.
[31] Dae Hyun Kim, Enamul Hoque, and Maneesh Agrawala. 2020. Answering questions about charts and generating visual explanations. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–13.
[32] Donald Ervin Knuth. 1993. *The Stanford GraphBase: a platform for combinatorial computing*. Vol. 1. AcM Press New York.
[33] Nicholas Kong, Marti A Hearst, and Maneesh Agrawala. 2014. Extracting references between text and charts via crowdsourcing. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 31–40.
[34] Valdis Krebs. 2004. Books about us politics. *unpublished, http://www. orgnet. com* (2004).
[35] Chufan Lai, Zhixian Lin, Ruike Jiang, Yun Han, Can Liu, and Xiaoru Yuan. 2020. Automatic annotation synchronizing with textual description for visualization. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.
[36] Yunshi Lan and Jing Jiang. 2020. Query Graph Generation for Answering Multi-hop Complex Questions from Knowledge Bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 969–974. https://doi.org/10.18653/v1/2020.acl-main.91
[37] Yunshi Lan, Shuohang Wang, and Jing Jiang. 2019. Knowledge Base Question Answering with Topic Units. In *IJCAI*. 5046–5052. https://doi.org/10.24963/ijcai.2019/701
[38] Yunshi Lan, Shuohang Wang, and Jing Jiang. 2019. Multi-hop knowledge base question answering with an iterative sequence matching model. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 359–368.
[39] Deqing Li, Honghui Mei, Yi Shen, Shuang Su, Wenli Zhang, Junting Wang, Ming Zu, and Wei Chen. 2018. ECharts: a declarative framework for rapid construction of web-based visualization. *Visual Informatics* 2, 2 (2018), 136–146.
[40] Haotian Li, Yong Wang, Songheng Zhang, Yangqiu Song, and Huamin Qu. 2021. KG4Vis: A knowledge graph-based approach for visualization recommendation. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2021), 195–205.
[41] Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*. Association for Computational Linguistics (ACL), 23–33.
[42] Can Liu, Yun Han, Ruike Jiang, and Xiaoru Yuan. 2021. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*. IEEE, 11–20.
[43] Can Liu, Liwenhan Xie, Yun Han, Datong Wei, and Xiaoru Yuan. 2020. AutoCaption: An approach to generate natural language description from visualization automatically. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, 191–195.
[44] Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2185–2194.
[45] Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. 2021. Synthesizing natural language to visualization (NL2VIS) benchmarks from NL2SQL benchmarks. In *Proceedings of the 2021 International Conference on Management of Data*. 1235–1247.
[46] Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. 2022. InfographicVQA. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 1697–1706.
[47] Ronald Metoyer, Qiyu Zhi, Bart Janczuk, and Walter Scheirer. 2018. Coupling story to visualization: Using textual analysis as a bridge between data and interpretation. In *23rd International Conference on Intelligent User Interfaces*. 503–507.
[48] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.
[49] Semi Min and Juyong Park. 2016. Narrative as a Complex Network: A Study of Victor Hugo's Les Misérables. In *Proceedings of HCI Korea*. 100–107.
[50] Tamara Munzner. 2014. *Visualization analysis and design*. CRC press.
[51] Mark EJ Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* 74, 3 (2006), 036104.
[52] Mark EJ Newman. 2013. Network data. *http://www-personal. umich. edu/˜ mejn/netdata/* (2013).
[53] Jason Obeid and Enamul Hoque. 2020. Chart-to-text: Generating natural language descriptions for charts by adapting the transformer model. *arXiv preprint arXiv:2010.09142* (2020).
[54] Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 1470–1480.
[55] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).
[56] Michael Petrochuk and Luke Zettlemoyer. 2018. Simplequestions nearly solved: A new upperbound and baseline approach. *arXiv preprint arXiv:1804.08798* (2018).
[57] Donghao Ren, Matthew Brehmer, Bongshin Lee, Tobias Höllerer, and Eun Kyoung Choe. 2017. Chartaccent: Annotation for data-driven storytelling. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*. Ieee, 230–239.
[58] Stefan Schweter and Alan Akbik. 2020. Flert: Document-level features for named entity recognition. *arXiv preprint arXiv:2011.06993* (2020).
[59] Vidya Setlur, Sarah E Battersby, Melanie Tory, Rich Gossweiler, and Angel X Chang. 2016. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th annual symposium on user interface software and technology*. 365–377.
[60] Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2022. Towards Natural Language Interfaces for Data Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics* (2022), 1–1. https://doi.org/10.1109/TVCG.2022.3148007
[61] Danqing Shi, Xinyue Xu, Fuling Sun, Yang Shi, and Nan Cao. 2020. Calliope: Automatic visual data story generation from a spreadsheet. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 453–463.
[62] Hrituraj Singh and Sumit Shekhar. 2020. Stl-cqa: Structure-based transformers with localization and encoding for chart question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 3275–3284.
[63] Sicheng Song, Chenhui Li, Yujing Sun, and Changbo Wang. 2022. VividGraph: Learning to Extract and Redesign Network Graphs from Visualization Images. *IEEE Transactions on Visualization and Computer Graphics* (2022).
[64] Arjun Srinivasan, Bongshin Lee, Nathalie Henry Riche, Steven M Drucker, and Ken Hinckley. 2020. InChorus: Designing consistent multimodal interactions for data visualization on tablet devices. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–13.
[65] Arjun Srinivasan and John Stasko. 2017. Orko: Facilitating multimodal interaction for visual exploration and analysis of networks. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 511–521.
[66] Harris Steve, Seaborne Andy, and Prud'hommeaux Eric. 2013. SPARQL 1.1 Query Language. https://www.w3.org/TR/sparql11-query/.
[67] Chao Tong, Richard Roberts, Rita Borgo, Sean Walton, Robert S Laramee, Kodzo Wegba, Aidong Lu, Yun Wang, Huamin Qu, Qiong Luo, et al. 2018. Storytelling and visualization: An extended survey. *Information* 9, 3 (2018), 65.
[68] Mati Ullah, Abdul Shahid, Muhammad Roman, Muhammad Assam, Muhammad Fayaz, Yazeed Ghadi, Hanan Aljuaid, et al. 2022. Analyzing Interdisciplinary Research Using Co-Authorship Networks. *Complexity* 2022 (2022).
[69] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.
[70] Yanyan Wang, Zhanning Bai, Zhifeng Lin, Xiaoqing Dong, Yingchaojie Feng, Jiacheng Pan, and Wei Chen. 2021. G6: A web-based library for graph visualization. *Visual Informatics* 5, 4 (2021), 49–55.
[71] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6
[72] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*.
[73] Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 643–648.
[74] Zhaoquan Yuan, Xiao Peng, Xiao Wu, and Changsheng Xu. 2021. Hierarchical Multi-Task Learning for Diagram Question Answering with Multi-Modal Transformer. In *Proceedings of the 29th ACM International Conference on Multimedia*. 1313–1321.
[75] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.
[76] Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. *arXiv preprint arXiv:1707.07806* (2017).
[77] Ying Zhao, Jingcheng Shi, Jiawei Liu, Jian Zhao, Fangfang Zhou, Wenzhi Zhang, Kangyi Chen, Xin Zhao, Chunyao Zhu, and Wei Chen. 2021. Evaluating effects of background stories on graph perception. *IEEE Transactions on Visualization and Computer Graphics* (2021).
[78] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR* abs/1709.00103 (2017).