# PASemiQA: Plan-Assisted Agent for Question Answering on Semi-Structured Data with Text and Relational Information

Hansi Yang hyangbw@cse.ust.hk CSE, HKUST Hong Kong, China Qi Zhang Ant Group Beijing, China

Wei Jiang Ant Group Beijing, China

Jianguo Li Ant Group Beijing, China

# **Abstract**

Large language models (LLMs) have shown impressive abilities in answering questions across various domains, but they often encounter hallucination issues on questions that require professional and up-to-date knowledge. To address this limitation, retrievalaugmented generation (RAG) techniques have been proposed, which retrieve relevant information from external sources to inform their responses. However, existing RAG methods typically focus on a single type of external data, such as vectorized text database or knowledge graphs, and cannot well handle real-world questions on semi-structured data containing both text and relational information. To bridge this gap, we introduce PASemiQA, a novel approach that jointly leverages text and relational information in semi-structured data to answer questions. PASemiQA first generates a plan to identify relevant text and relational information to answer the question in semi-structured data, and then uses an LLM agent to traverse the semi-structured data and extract necessary information. Our empirical results demonstrate the effectiveness of PASemiQA across different semi-structured datasets from various domains, showcasing its potential to improve the accuracy and reliability of question answering systems on semi-structured data.

#### **CCS** Concepts

• **Information systems** → *Retrieval models and ranking*; **Question answering**; • **Computing methodologies** → Knowledge representation and reasoning.

# **Keywords**

Large Language Model, Question Answering, Graph-structured Data

#### 1 Introduction

Recent years have witnessed the emergence of powerful large language models (LLMs) obtained from large-scale pre-training on

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

https://doi.org/10.1145/nnnnnn.nnnnnn

vast amount of corpus [1, 3, 4, 6, 18, 19, 26]. Despite their strong language understanding and instruction following ability [22], directly using LLMs for downstream applications may encounter the hallucination issue [11], where LLM randomly generates an answer that satisfies semantic rules but does not match the reality. Such issue makes it not accountable to directly use LLM on accuracy-critical applications, and motivates the development of retrieval-augmented generation (RAG) techniques [31] that try to utilize accurate and up-to-date external information to alleviate the hallucination issue. A common RAG pipeline is to first retrieve relevant contents from a given database based on the input, and let the LLM model generate its output from the original input and retrieved content together. RAG techniques have found application in diverse domains, including molecular generation for drug discovery [21], biomedical literature understanding [8], and code completion within a repository [29].

Based on the type of database that stores relevant contents for retrieval, existing RAG methods can be categorized into several types. The most common type is based on vectorized document database [15, 27, 29], where we retrieve text chunks based on their relevance to the input. Such approach requires all information be stored in text format, and cannot perform exact step-by-step reasoning since the text chunks are separately retrieved based on the input. Another common approach is based on knowledge graphs (KG [10]), also referred as KGQA methods [14, 16, 25]. An example of KG is shown in Figure 1(b), where nodes correspond to entities in the real world, and edges between nodes indicate relations between pairs of entities. While KG allows easy implementation of stepby-step reasoning by traversing along a path connecting different entities in KG, its format may also restrict the expressive power, as an edge connecting two entities cannot express complex relations that can cover multiple entities.

To overcome the above limitations of each data modality, a straight-forward idea is to combine these two data modalities together, which leads to so-called semi-structured data [24]. It combines knowledge graph with node-level documents to allow more flexible information storing. Due to its flexibility, semi-structured data naturally find existence in diverse domains. An example in biomedical domain is shown in Figure 1(c), where we use nodes to represent different biomedical objects, edges describe interactions between them, and each node can have additional documents

as the description for the corresponding biomedical object. Another example is from e-commerce platforms, where products correspond to nodes with rich text information such as description and reviews, and connections between products can indicate the co-purchasing behaviors recorded by the platform. Despite such wide application, limited works consider how to answer questions with semi-structured data. Existing baseline methods [24] simply borrow the baseline methods for RAG, which can neglect complex connections between nodes in semi-structured data, and perform badly when the question present includes complex reasoning on different relations across nodes in the semi-structured data.

Motivated by the limitations of existing approaches, we propose PASemiQA (Plan-Assisted Question Answering with Semistructured data), a novel method that leverages both text and relational information to answer questions. Unlike standard RAG or KGQA methods, our approach tackles the unique challenge of identifying both text and relational information from the question. To address this, we introduce a novel planning module that selects relevant nodes and edges to inform the question-answering process. This plan is then used to guide an agent framework, implemented by an LLM, to traverse the semi-structured data and extract relevant information. Our method's effectiveness is demonstrated through empirical results on different semi-structured datasets from various domains.

Our contribution can be listed as follows:

- We propose a novel planning module that can generate informative plans for answering questions with both relational and text information.
- We propose an agent framework that can traverse the semistructured data based on the generated plan to extract related information for this question.
- We demonstrate the effectiveness of our proposed method through empirical results on different semi-structured datasets from various domains.

#### 2 Related Works

# 2.1 Retrieval-Augmented Generation (RAG)

Generally, retrieval-augmented generation (RAG) methods [31] consist of two parts: retrieval, which finds relevant information from external data sources for the given input, and generation, which generates the output based on retrieved information and the original input. A simple yet effective way of retrieval is to use text embedding similarity, which selects text chunks whose text embeddings have the largest cosine similarity to the text embedding of the input. Some later works propose to improve the retrieval process to be more effective than simply comparing text embeddings. For example, ReAct [27] uses multiple round of retrieval enabled by Chain of Thought (CoT [5, 23, 30]) techniques to retrieve more relevant information for the input. REPLUG [15] proposes to treat the generation process as a black box, and fine-tune a retrieval model that can lead to better final results of generation.

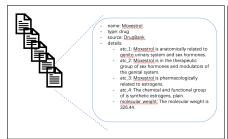
The most simple way to generate the output is to directly fed the retrieved contents into a LLM model and use it to generate the output based on the original input and retrieved content. RETRO [2] further proposes to fine-tune the LLM on the desired output based on the retrieved data to improve the quality of generated content. Another work [13] uses CoT to break down complex problems into multiple sub-problems based on the retrieved content before generating the answer.

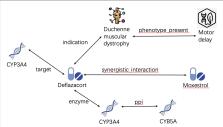
Despite the above works that focus on vectorized text databases, some recent works have considered retrieving contents from a graph database. For example, G-retriever [9] utilizes RAG techniques on a text-attributed graph, where nodes and edges are described by a short piece of text, and selects relevant node and edge in this graph to answer the given question. Nevertheless, the considered text-attributed graph only has small amount of text information, and G-retriever cannot be applied to semi-structured data where each node can have large amount of text information. GraphRAG [7] proposes a RAG pipeline on a set of text chunks that are connected based on their semantic similarities. Based on such connections, GraphRAG can answer complex questions that may involve summarizing across different text chunks. We note that this approach is also distinct from our method as we need to utilize external relational information that is not present in GraphRAG.

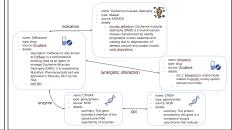
# 2.2 Knowledge Graph Question Answering (KGQA)

Mathematically, a knowledge graph (KG [10, 20]) can be defined as a set of triplets (s, r, o), where s and o are two entities, and r is the relation between these two entities. Various methods have been proposed to utilize the language understanding ability of LLM to utilize relational information in a knowledge graph to answer questions. Based on the underlying mechanism, these methods can be classified into two categories: Semantic Parsing (SP) methods [12] and Retrieval Augmented (RA) methods [14, 16]. SP methods first transform the question expressed by natural language to a structural query using LLMs. Such queries can then be directly executed on a knowledge graph to directly derive the answers. An example SP method is KB-BINDER [12], which first generates logical forms from the given question, and then binds the logical forms to executable structural queries with entity and relation binders. However, the effectiveness of these methods relies heavily on the quality of the generated queries and the completeness of KGs. RA methods share more similarities to RAG methods, which also retrieve related information from the KG to generate accurate answers for the given querstion [16, 25]. ToG [16] treats the LLM as an agent to interactively explore relation paths step-by-step on KGs and perform reasoning based on the retrieved paths. RoG [14] first generates relation paths as faithful plans, and then use them to retrieve valid reasoning paths from the KGs for LLMs to reason. GoG [25] improves upon ToG and allows more flexible traversing of KG instead of separate paths, and utilize LLM knowledge to answer questions if the LLM detects that the KG is incomplete.

While semi-structured can be regarded as a KG supplemented by text description on different entities, no existing KGQA methods consider how to utilize the additional text information, which can limits the answer quality if they are directly generalized to semi-structured data. Moreover, most entities in KG have a clear and unique name that can be used for reference, while such names do not always exist for semi-structured data and text information on each node can be important to distinguish different nodes.







(a) (Vectorized) text database. Example content is shown (b) Knowledge graph (KG). Edge types (relations) are shown (c) Semi-structured data. Node text descriptions are shown in the blue box. corresponding blue boxes. Edge types (relations) are shown near each edge.

Figure 1: Comparison of different data structures to supplement question answering of LLM.

Example question	Text information	Relational information
What climbing guide do most people purchase with Black Diamond White Gold Loose Chalk?	Black Diamond White Gold Loose Chalk	also_buy
What are some fashionable reversible bucket hats that provide excellent sun protection against UVA and UVB rays?	fashionable reversible bucket hats	text
Search publications by Hao-Sheng Zeng on non-Markovian dynamics.	non-Markovian dynamics	author_writes_paper
What are some nanofluid heat transfer research papers published by scholars from Philadelphia University?	nanofluid heat transfer	author_writes_paper author_affiliated_with_institution
What is the name of the condition characterized by a complete interruption of the inferior vena cava, falling under congenital vena cava anomalies?	a complete interruption of the inferior vena cava	parent-child
Which gallbladder illness serves as a contraindication to medications prescribed for small cell neuroendocrine carcinoma of the urinary bladder?	gallbladder illness, small cell neuroendocrine carcinoma of the urinary bladder	contraindication indication
What are the pathways with both a 'parent-child' hierarchy related to the TCR signaling pathway and interactions with the CD101 gene or protein?	TCR signaling pathway CD101	parent-child interacts with
What are the common gene targets for both Hydrocortisone butyrate and 2-Methoxyestradiol?	Hydrocortisone butyrate 2-Methoxyestradiol	target

Table 1: Example questions on semi-structured data. Relevant parts on relational and text aspects are highlighted.

# 3 Problem Definition

We first give the formal definition of semi-structured data, which can be expressed by a text-attributed heterogeneous graph as follows:

DEFINITION 1 (SEMI-STRUCTURED DATA [24]). Semi-structured data can be described by a text-attributed heterogeneous graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{R}, f_{\mathcal{T}}, f_{\mathcal{R}}\}$ , where  $\mathcal{V}$  is the set of nodes,  $\mathcal{E}$  is the set of edges,  $f_{\mathcal{T}}: \mathcal{V} \to \text{text}$  is a mapping from nodes to their text description, and  $f_{\mathcal{R}}: \mathcal{E} \to \mathcal{R}$  is a mapping from edges to edge types.

An example of semi-structured data in biomedical domain is shown in Figure 1(c). The nodes in V are represented by blue boxes, and all texts in each blue box correspond to the text descriptions of each node obtained from  $f_T$ . Edge types from  $f_R$  are shown near

each edge in  $\mathcal{E}.$  With the above definition for semi-structured data, we can also formally define the QA task on such data as follows:

DEFINITION 2 (QA ON SEMI-STRUCTURED DATA). Given semi-structured data G defined in Definition 1, QA on this data set aims to find the answer nodes  $a \in V$  for a question q.

Note that our work. Some example questions on different semi-structured data are shown in Table 1. Compared with either standard RAG or KGQA task, retrieving contents from semi-structured data poses special challenges, as we need to jointly utilize relational and text information. We note that existing methods fail short to tackle such complex challenges. Standard RAG algorithms only use text information to directly answer the questions, and cannot perform multi-step reasoning based on relational information in

semi-structured data. On the contrary, existing KGQA methods neglect additional text information on different entities. Such limitations call for the development of novel methods that can jointly utilize relational and text information.

#### 4 Proposed Method

Following the limitation raised in section 3, in this section, we introduce how to effectively utilize the text and relational information together to answer questions with semi-structured data. The proposed method, called PASemiQA (Plan-Assisted Semi-structured data QA), is composed by two connected parts as in Figure 2. First, we introduce how to generate reasoning plans based on the given question, which consists of finding relevant nodes as well as edge types from the given semi-structured data, as in section 4.1. Then based on the generated plan, in section 4.2 we propose to use a LLM model as an agent to traverse the whole data set and find the final answer. The complete algorithm is in section 4.3.

#### 4.1 Planning Generation

While answering a question defined in Definition 2 may involve both text and relational information in the given semi-structured data, we propose to disentangle the complete reasoning process into the following two parts that separately depend on text and relational information:

- Text information: given nodes with rich text description in the whole semi-structured data set, which nodes may be related to this question?
- Relational information: given numerous relations in the whole semi-structured data set, which relations may be related to this question?

To answer the two questions raised above, we introduce the following definition of generating a plan from the given question on semi-structured data:

DEFINITION 3 (PLANNING GENERATION ON SEMI-STRUCTURED DATA). With a semi-structured data set G defined in Definition 1 and a question q defined in Definition 2, planning generation maps this question q to a set of nodes  $V_q \subseteq V$ , and a set of edge type sequences  $\{z_i\}$  where each  $z_i = (r_{1,i}, \ldots, r_{n_i,i})$  is a sequence of edge types (also called a **relation path**) with each edge type  $r_{j,i} \in \mathcal{R} \cup \{text\}, j = 1, \ldots, n_i \text{ and } n_i \text{ denoting the length of sequence } z_i$ 

Note that the relation sequence consists of an additional "type" *text*, which indicates that the answer only depends on the text information in semi-structured data, and no relation in the given semi-structured data need to be involved to answer this question. Examples of queries from different semi-structured data sets are shown in Table 1, along with the extracted related text information from the question and edges in the semi-structured data. We then introduce how the plan generation module is designed for both text and relational information respectively.

4.1.1 Text information. For the first part on text information, a simple method utilized by most previous KGQA methods [16, 25] is keyword matching, which matches names of nodes from the given KG in this question. The success of such method in KGQA methods is due to the clear reference of nodes by their names in KG, which may not still hold in the context of semi-structured data. As

shown in the second row in Table 1, some queries does not directly point to a specific node, but gives node descriptions that cannot be well matched by exact name matching. Meanwhile, a standard baseline in various RAG methods [24, 31] is based on the similarity between question and document embeddings, which in turn can be inaccurate when we need to match the exact names, such as the drug or gene names shown in Table 1.

Motivated by the limitation of both methods, we then propose a hybrid solution to combine these two methods together. We first go through the question to extract all mentioned nodes based on their names as keyword ("Keyword matching" in Figure 2). If we do not obtain any nodes in keyword matching, we will select top-k nodes based on the cosine similarity between the text embedding for this question and the documents on different nodes ("Embedding similarity" in Figure 2). The matched nodes together form the node set  $\mathcal{V}_q$  in Defition 3.

4.1.2 Relational information. To utilize relational information in a given semi-structured data, as in Definition 3, we propose to generate a set of edge type sequences  $\{z_i\}$  that may be helpful to find the answers for a question. Specifically, given the question q and semi-structured data G, the probability of choosing node a as the answer can be expressed by conditioning on the relation path a as follows:

$$P_{\theta}(a|q,\mathcal{G}) = \sum_{z \in \mathcal{T}} P_{semi}(a|q,z,\mathcal{G}) P_{\theta}(z|q), \tag{1}$$

where  $\mathcal{Z}$  denotes the set of all possible relation paths. The first term  $P_{semi}(a|q,z,\mathcal{G})$  refers to the probability of choosing node a as the answer given the question q, relation path z and semi-structured data  $\mathcal{G}$ , and we let it to be uniform distribution on all nodes that can be reached by relation path z from nodes in  $\mathcal{V}_q$  obtained in previous section. Mathematically, we can define  $\mathcal{A}_{q,z}$  as the set of nodes a that can be reached from any nodes  $e \in \mathcal{V}_q$  as follows:

 $\mathcal{A}_{q,z} = \{a : \text{there exists a relation path } z \text{ starts from } e \in \mathcal{V}_q \text{ to } a\}$ 

Then  $P_{semi}(a|q, z, \mathcal{G})$  takes the following form:

$$P_{semi}(a|q, z, \mathcal{G}) = \begin{cases} \frac{1}{|\mathcal{A}_{q,z}|}, a \in \mathcal{A}_{q,z}, \\ 0, else, \end{cases}$$
 (2)

Now to maximize the probability of finding the correct answer in (1), we need to optimize the latter term  $P_{\theta}(z|q)$  that controls the generation of relation path z given the question q. Suppose we know the ground-truth answer  $a^*$  for a question q, then from a node  $e \in \mathcal{V}_q$  obtained in previous section, if we can find the path instances  $e \xrightarrow{r_1} \dots \xrightarrow{r_l} a$  connecting these two nodes in semi-structured data, this relation path  $z = \{r_1, \dots, r_l\}$  should receive larger probability in  $P_{\theta}(z|q)$ . As such, we introduce the following distribution  $Q(z|q, a^*, \mathcal{G})$  that  $P_{\theta}(z|q)$  need to fit as follows:

$$Q(z|q, a^*, \mathcal{G}) = \begin{cases} \frac{1}{|\mathcal{Z}^*|}, z = (r_1, \dots, r_l), \exists e \in \mathcal{V}_q, e \xrightarrow{r_1} \dots \xrightarrow{r_l} a \in \mathcal{G} \\ 0, else, \end{cases}$$

where we consider the shortest paths  $\mathcal{Z}^* \subseteq \mathcal{Z}$  between nodes  $e \in \mathcal{V}_q$  and the ground-truth answer  $a^*$ . Then we define the following

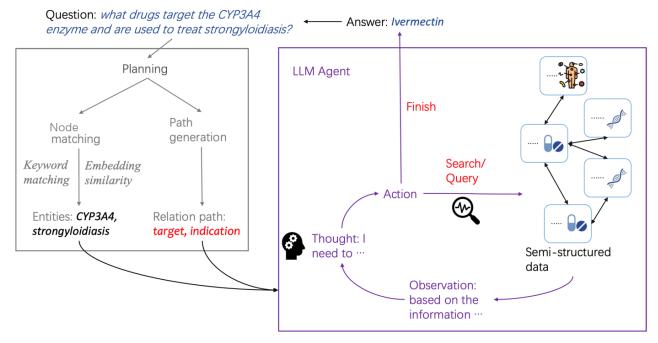


Figure 2: Complete framework of PASemiQA consisting of two parts: planning module (section 4.1) and LLM agent (section 4.2).

loss based on the KL divergence between the target distribution  $Q(z|q, a^*, \mathcal{G})$  and the path generation model  $P_{\theta}(z|q)$  as follows:

$$\mathcal{L}_{\text{plan}} = D_{\text{KL}}(Q(z|q, a^*, \mathcal{G}) || P_{\theta}(z|q))$$

$$= -\frac{1}{|\mathcal{Z}^*|} \sum_{z \in \mathcal{Z}^*} \log P_{\theta}(z|q) + C, \tag{4}$$

and the optimization of  $\mathcal{L}_{plan}$  can be achieved as

$$\arg\max_{\theta} \frac{1}{|\mathcal{Z}^*|} \sum_{z \in \mathcal{Z}^*} \log P_{\theta}(z|q) = \frac{1}{|\mathcal{Z}^*|} \sum_{z \in \mathcal{Z}^*} \log \prod_{i=1}^{|z|} P_{\theta}(r_i|r_{< i}, q), \tag{5}$$

where  $P_{\theta}(r_i|r_{< i},q)$  denotes the probability of generating next relation  $r_i$  for the relation path z given all previous relations  $r_{< i}$  and question q, and |z| denotes the total length of the relation path z. In other words, we maximize the probability of path generation model to generate faithful relation paths.

For the implementation of  $P_{\theta}(z|q)$ , we finetune a pre-trained large language model (e.g, LlaMa model family [19]) by instruction fine-tuning. Specifically, we use the following template to construct the training corpus from any given semi-structured data set:

Please generate a valid relation path that can be helpful for answering the following question:  $ext{Question}$ >. z

where  $\leq$ Question> will be replaced by the actual question q, followed by the ground-truth answer  $z \in \mathcal{Z}^*$  that is structurally formatted as a sentence like below:

$$z = \langle PATH \rangle r_1 \langle SEP \rangle r_2 \langle SEP \rangle \dots \langle SEP \rangle r_l \langle PATH \rangle$$

where <PATH>, <SEP>, </PATH> are special tokens indicating the start, separator, and end of the relation path, respectively. Then at the inference stage, we will use the same prompt to generate relation paths for the testing question q.

#### 4.2 Graph Traversing Agent

Based on the generated plan, we then need to actually retrieve relevant information from the semi-structured data. While many recent works on KGQA [16, 25] propose to use an LLM as an agent to perform step-by-step reasoning on KG, these works cannot be directly generalized to semi-structured data as they only focus on relational information, and does not utilize additional text information in different nodes. As such, we propose to design a novel agent framework to answer questions on semi-structured data. This agent, implemented by an LLM model, first thinks of which action to take and express its *Thought* in natural languages, then performs the action to retrieve relevant information from the semi-structured data to obtain the set of triplets with node text description that will be used to determine next action. Such process is repeated for multiple times. The agent can take three different actions in total: Search, Query and Finish, which focus on different aspects of semi-structured data:

(1) Search[node], which focuses on relational information from node. For simplicity, here we consider searching only one node, and it is straight-forward to search multiple nodes by repeating the same process. Given node, this action finds the most relevant top K neighbors, which uses the same strategy in ToG [16] to filter unrelated neighbor nodes: first, we retrieve all neighbors that are connected to node. Then, we utilize the LLM agent to select the top K nodes that most related to current reasoning based on their edges to node.

- (2) Query[query], which focuses on text information from query. This action finds the most relevant top *K* nodes for the given query based on the text description on different nodes. We utilize query embedding and the text embedding of each node's text description, and selects those nodes with top *K* relevance scores computed by the cosine similarity between two embeddings.
- (3) Finish[answer], which indicates that the agent finishes the task with the node answer as *a* in Definition 2.

The whole process will be repeated for T rounds (T is a hyperparameter) to let the model terminate the traversing process on semi-structured data and return the final answer.

#### 4.3 Complete Algorithm

The complete procedure of PASemiQA is shown in Algorithm 1, with the exact prompt for the agent is elaborated in Appendix A. Compared to existing RAG or KGQA methods, PASemiQA jointly utilizes the text and relational information to answer the given question.

#### Algorithm 1 PASemiQA: Plan-Assisted Semi-structured data QA.

- 1: Input: Node matching method match. Relation path generation model  $P_{\theta}(z|q)$ . LLM agent *agent*, question q;
- 2: Initialize the node set  $\mathcal{E}_0$  from the matching process:  $\mathcal{E}_0 = \{e_i\} = match(q)$  and triplet set  $O_0 = \{\}$  to empty
- 3: Generate relation paths  $\{z_j\}$  from the path generation model  $P_{\theta}(z|q)$
- 4: **for** t = 0, ..., T 1 **do**
- 5: Generate thought t from previous triplet set  $O_t$
- 6: Generate Action based on thought t.
- 7: **if** Action is Finish[answer] **then**
- 8: Break
- 9: **else if** Action is Query[query] **then**
- 10: Return k nodes with largest similarity score between their document embedding and query to form the node set  $\mathcal{E}_t$
- 11: **else if** Action is Search[query] **then**
- 12: Retrieve triplets  $(e_t, r, e_{t+1})$  from semi-structured data  $\mathcal{G}$  based on existing nodes  $e_t \in \mathcal{E}_t$  and relation paths  $\{z_j\}$
- Append new node  $e_{t+1}$  to the node set:  $\mathcal{E}_{t+1} = \mathcal{E}_t \cup \{e_{t+1}\}$
- 14: end if
- 15: Append retrieved information to triplet set:  $O_{t+1} = O_t \cup \{(e_t, r, e_{t+1})\}$
- 16: end for
- 17: Output the answer from the last Finish action

#### 5 Experiments

We conduct experiments on three semi-structured data sets from the STaRK benchmark [24] that focus on diverse fields: Amazon (product recommendation), MAG (academic network) and PrimeKG (biomedical network). The data split follows [24], and detailed information and statistics of these data sets are in Table 8 in Appendix B. For all experiments, unless otherwise specified, we finetune a LLaMa2-7B model [19] as the path generation model in section 4.1, and use the GPT-4 model as the LLM agent in section 4.2. Following

standard practice in KGQA methods [14, 16, 25], we use the Hit@1 score as the performance metric for different methods.

We compare the proposed method against baseline methods both from RAG on semi-structured data and KGQA methods. For RAG methods, we choose two baseline methods in [24]: VSS and VSS+GPT-4 reranker. We use VSS as a representative for RAG methods as it is simple, fast and easy to implement, and VSS+GPT-4 reranker achieves the best performance among all methods in [24]. For KGQA methods, we choose ToG [16], RoG [14] and GoG [25] as recent representatives, which achieves satisfying performance on existing KGQA data sets such as CWQ [17] or WebQSP [28]. For fair comparison, all KGQA methods also use the GPT-4 model for their agents.

Table 2: Hit@1 scores on different semi-structured data with different methods. Best highlighted in bold.

dataset	PrimeKG	MAG	Amazon
VSS	0.1263	0.2908	0.4044
VSS+GPT-4 reranker	0.1828	0.4090	0.4479
ToG	0.1321	0.0540	0.1667
RoG	0.2274	0.3294	0.3122
GoG	0.1892	0.4126	0.2101
PASemiQA (GPT-4 Agent)	0.2968	0.4316	0.4586

Table 3: Wall-clock time cost (mean±std in seconds) of different methods to answer a question with given semi-structured data.

	PrimeKG	MAG	Amazon
VSS	$0.54 \pm 0.01$	$2.25 \pm 0.01$	5.71±0.02
VSS+GPT-4 reranker	26.97±1.96	23.43±1.64	24.76±1.55
ToG	14.68±1.57	24.16±1.53	$32.26 \pm 1.74$
RoG	$7.89 \pm 0.98$	$11.76 \pm 1.07$	$10.42 \pm 1.02$
GoG	28.56±2.02	24.73±1.79	32.68±1.65
PASemiQA (GPT-4 Agent)	28.19±2.04	25.48±1.77	18.74±1.34

#### 5.1 Comparison on Accuracy and Latency

Table 2 compares the answering accuracy of different methods for questions on semi-structured data. Due to the space limit, results with other performance metrics (e.g., MRR or macro F1 scores) are in Appendix C. For RAG methods, VSS+GPT-4 reranker almost outperforms all the other baseline on MAG and Amazon data set. For Amazon data set where questions solely depending on text information make a large proportion, RAG baselines (VSS and VSS+GPT-4 reranker) achieves much higher performance than KGQA methods that cannot effectively utilize text information. For PrimeKG data set where some questions require complex relational information, KGQA methods significantly outperforms RAG baselines. The proposed method achieves the best overall performance, which demonstrate the necessity of joint utilization of both text information and relational information to answer questions on semi-structured data.

We further compare the time cost of different methods in Table 3. Both VSS and RoG has much smaller time cost to answer a given question than other methods, as they do not need to iteratively select relational information from semi-structured data. ToG also has much smaller time cost on PrimeKG data set, yet its performance on this data set is also much worse than other KGQA methods, which may come from ineffective utilization of the relational information as this data set is highly professional and contains biomedical terms that may not be easy to understand. We can see that the proposed method achieves comparable time cost with VSS with reranker methods, but achieve better performance. In other words, the proposed method better utilizes the language understanding abilities of LLMs as an agent to find correct answers for a given question.

Table 4: Hit@1 scores of PASemiQA with different LLM agents on semi-structured data. Best highlighted in bold.

	PrimeKG	MAG	Amazon
Owen2-72B	0.1849	0.0976	0.4163
LlaMa3-70B	0.2392	0.4113	0.3264
GPT-3.5	0.2456	0.3092	0.3781
GPT-4	0.2968	0.4316	0.4586

Table 5: Hit@1 scores on different semi-structured data with different node matching methods. Best highlighted in bold.

dataset	PrimeKG	MAG	Amazon
Exact name matching	0.2945	0.4062	0.2102
Embedding similarity	0.2598	0.4158	0.3978
PASemiQA	0.2968	0.4316	0.4586

Table 6: Hit@1 scores on different semi-structured data with different plan generation modules. Best highlighted in bold.

	PrimeKG	MAG	Amazon
None	0.2868	0.4068	0.3754
GPT-4 in-context	0.2674	0.4148	0.4162
PASemiQA	0.2968	0.4316	0.4586

# 5.2 Ablation Studies

5.2.1 Using Different LLM Models as the Agent. While our proposed method does not limit the choice of LLM model as the agent, we compare the performance of different LLM models. We consider two open-source models: LLaMa 3-70B [6] and Qwen2-72B [26], as well as OpenAI's GPT-3.5 [3] and GPT-4 [1] models as two representative closed-source LLMs. The other closed-source LLMs (e.g., Gemini [18] or Claude [4]) are not compared due to the additional API costs.

Table 4 compares the performance of these models when used as the agent. We can see that GPT-4 model generally performs the best, while GPT-3.5 model performs worse than GPT-4 on all three data

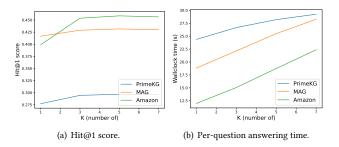


Figure 3: Comparison of PASemiQA with different values of K. T is set to 5 by default.

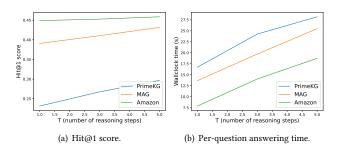


Figure 4: Comparison of PASemiQA with different values of T. K is set to 5 by default.

sets. Two open-source models achieve exceptional performance on specific semi-structured data: Qwen2 model only slightly falls short from GPT-4 on Amazon data set, while LLaMa3 model achieves almost comparable performance with GPT-4 on MAG data set. One possible reason for the performance difference across these semi-structured data is the differences in the pre-training corpus for these models, which may also motivate future works training an LLM agent specifically target on the given semi-structured data.

5.2.2 Implementation of Planning Modules. The planning module in section 4.1 involves two parts: node matching to utilize text information and plan generation to utilize relational information. In this section, we investigate how different design choices of these two parts may impact the performance of the proposed framework.

For node matching, we compare the proposed hybrid matching method in PASemiQA with solely using node matching or embedding similarity, and the performance comparison on different semi-structured data sets is shown in Table 5. We can see that these methods have different performances on different data sets. Node matching achieves satisfying performance for PrimeKG as biomedical objects have unique names and can be exactly matched. Meanwhile, for Amazon data set, the item names are often not stable and we need to better utilize the text information from each node. The performance for MAG is generally stable across different methods. Across different data sets, the proposed method achieves the best overall performance due to its flexibility on different semi-structured data.

For planning generation module, we compare against the following baseline: (i) None, which directly uses the LLM agent from the

Question q	Show me publications by Th. Friedrich on high heat flux experiments with divertor components.
Topic Node $\mathcal{V}_q$	[Th. Friedrich]
Plan $\{z_i\}$	authorwritespaper z <sub>1</sub>
Thought 1	I need to search for publications by Th. Friedrich specifically focusing on high heat flux experiments with divertor components.
Action 1	Search[Th. Friedrich]
Triplet Set 1  O <sub>1</sub>	Th. Friedrich $(e_0)$ , authorwritespaper $(r_0)$ ,  Design improvement of the target elements of Wendelstein 7-X divertor $(e_1)$ Th. Friedrich $(e_0)$ , authorwritespaper $(r_0)$ ,  Results and consequences of high heat flux testing as quality assessment of the Wendelstein 7-X divertor $(e_2)$
Thought 2	"Results and consequences of high heat flux testing as quality assessment of the Wendelstein 7-X divertor" is a publication by Th. Friedrich that focuses on high heat flux experiments with divertor components.",
Action 2	Finish[Results and consequences of high heat flux testing as quality assessment of the Wendelstein 7-X divertor]  (Finish with answer a)

Table 7: Example planning and reasoning process of PASemiQA on a question from the MAG data set.

node set  $\mathcal{V}_q$  to traverse the semi-structured data and find the answer, (ii) GPT-4 in-context, which utilizes the instruction following and few-shot learning ability of existing LLM models to generate the relation path without fine-tuning. Similar to the prompt in section 4.1, the in-context learning prompt for path generation is designed as follows:

Compared to the prompt in section 4.1, we add some additional examples from the given semi-structured data for in-context learning on the semantics of different types of edges.

Table 6 compares the performance with different path generation methods. We can see that even without the relation path generation module, the proposed method still achieves better performance than other baseline methods on PrimeKG data set, and the performance is comparable in MAG data set. However, for Amazon data set, the proposed method cannot even outperforms most simple VSS baseline without relation path generation. This may partially be attributed to large proportion of questions that only requires text information in Amazon data set. Solely utilizing relational information is not helpful for answering these queries, and we need to use a path generation module to determine whether text information is enough to answer the given question. We can also see that the influence of path generation module is more significant for Amazon and PrimeKG data set, but becomes less significant for MAG data set. While we have explained the influence on Amazon data set from the large proportion of questions that only depend on text

information, the reason on PrimeKG data set is due to its complex types of edges from the biomedical domain, which may not be so familiar for general LLM models.

# 5.3 Hyper-parameter Sensitivity

In Algorithm 1, the proposed method PASemiQA involves two hyper-parameters K and T. K controls the number of related nodes extracted in each step, while T controls the total number of steps executed to answer each question. These two hyper-parameters can impact both the answer accuracy and latency of the proposed method, which will be analyzed in this section.

Figure 3(a) compares the performance with different values of K, and Figure 3(b) compares the time cost with different values of K. Intuitively, setting K either too large or too small will cause performance downgradation: small K cannot ensure sufficient exploration in the whole semi-structured data, while large K may include too much unrelated information in our reasoning process. Moreover, setting K too large may also lead to increasing time cost, as the agent needs to process more nodes. The empirical results generally agrees with such intuition that either setting K too small or too large leads to sub-optimal performance. We also note the performance difference is not significant across different values of K, which indicates that the performance of proposed method is robust to this hyper-parameter.

Figure 4(a) compares the performance with different values of T, and Figure 4(b) compares the time cost with different values of T. Intuitively, we need to set T large enough so as to ensure the complete semi-structured data are explored. Nevertheless, setting T too large may also lead to increasing latency, as the agent needs to repeat the reasoning process for more times, even it can control the number of steps with "Finish" action. The empirical results generally agrees with such intuition that either setting T too small or too large leads to sub-optimal performance. We also note the performance difference is not significant across different values of T, which indicates that the performance of proposed method is robust to this hyper-parameter.

#### 5.4 Case Studies

Finally, we use a real example in Table 7 to demonstrate how the proposed method PASemiQA jointly uses text and relational information to answer questions. With its planning module, PASemiQA first identifies the topic node to be the author "Th. Friedrich", and the generated relation path contains only one relation "author\_\_writes\_\_paper" as this question is about the author's publication. Then starting with the topic node, PASemiQA lets the LLM agent to use the generated relation path to find more nodes related to the question. The LLM agent further uses its own language understanding ability on the paper title to select the correct answer and finish the reasoning process.

#### 6 Conclusion

In this paper, we design a novel method PASemiQA to jointly utilize text and relational information for answering questions with semi-structured data. PASemiQA first obtains a plan to determine text and relational information in semi-structured data that is useful to answer the given question. Then based on the generated plan, PASemiQA introduces an agent implemented by an LLM to traverse the semi-structured data and extract related information for this question. Empirical results across semi-structured data sets from different domains demonstrate the effectiveness of the proposed method.

#### References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023).
- [2] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving Language Models by Retrieving from Trillions of Tokens. In Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162). PMLR, 2206–2240.
- [3] Tom B Brown. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020).
- [4] Loredana Caruccio, Stefano Cirillo, Giuseppe Polese, Giandomenico Solimando, Shanmugam Sundaramurthy, and Genoveffa Tortora. 2024. Claude 2.0 large language model: Tackling a real-world classification problem with a new iterative prompt engineering approach. Intelligent Systems with Applications 21 (2024), 200336.
- [5] Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2024. Navigate through Enigmatic Labyrinth A Survey of Chain of Thought Reasoning: Advances, Frontiers and Future. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Bangkok, Thailand, 1173–1203.
- [6] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024).
- [7] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. arXiv preprint arXiv:2404.16130 (2024)
- [8] Giacomo Frisoni, Miki Mizutani, Gianluca Moro, and Lorenzo Valgimigli. 2022. Bioreader: a retrieval-enhanced text-to-text transformer for biomedical literature. In Proceedings of the 2022 conference on empirical methods in natural language processing. 5770–5793.
- [9] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. arXiv preprint arXiv:2402.07630 (2024).

- [10] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. IEEE Transactions on Neural Networks and Learning Systems 33, 2 (2022), 494–514.
- [11] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. ACM Comput. Surv. 55, 12, Article 248 (2023), 38 pages.
- [12] Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot In-context Learning on Knowledge Base Question Answering. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 6966–6980.
- [13] Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024. Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources. In The Twelfth International Conference on Learning Representations.
- [14] LINHAO LUO, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In The Twelfth International Conference on Learning Representations.
- [15] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-Augmented Black-Box Language Models. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). Association for Computational Linguistics, Mexico City, Mexico, 8371-8384.
- [16] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. In The Twelfth International Conference on Learning Representations.
- [17] Alon Talmor and Jonathan Berant. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). 641–651.
- [18] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805 (2023).
- [19] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yas-mine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023).
- [20] 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.
- [21] Zichao Wang, Weili Nie, Zhuoran Qiao, Chaowei Xiao, Richard Baraniuk, and Anima Anandkumar. 2023. Retrieval-based Controllable Molecule Generation. In The Eleventh International Conference on Learning Representations.
- [22] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned Language Models are Zero-Shot Learners. In International Conference on Learning Representations.
- [23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Advances in Neural Information Processing Systems, Vol. 35. 24824–24837.
- [24] Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. 2024. STaRK: Benchmarking LLM Retrieval on Textual and Relational Knowledge Bases. arXiv preprint arXiv:2404.13207 (2024).
- [25] Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Kang Liu, and Jun Zhao. 2024. Generate-on-Graph: Treat LLM as both Agent and KG in Incomplete Knowledge Graph Question Answering. arXiv preprint arXiv:2404.14741 (2024).
- [26] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. arXiv preprint arXiv:2407.10671 (2024).
- [27] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In The Eleventh International Conference on Learning Representations.
- [28] Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 201–206.
- [29] Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. 2023. RepoCoder: Repository-Level Code Completion Through Iterative Retrieval and Generation. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Singapore, 2471–2484.
- [30] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic Chain of Thought Prompting in Large Language Models. In The Eleventh International

Conference on Learning Representations.

[31] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-Augmented Generation for AI-Generated Content: A Survey. ArXiv abs/2402.19473 (2024).

# A Prompts

Solve a question answering task with interleaving Thought, Action, Observation steps. Thought can reason about the current situation, and Action can be three types:

- (1) Search[node1 | node2 | ...], which searches the exact nodes on the knowledge graph and returns their one-hop subgraphs. You should extract the all concrete nodes appeared in your last thought without redundant words, and you should always select nodes from topic nodes in the first search.
- (2) Query[question], which finds the most related node of the given question based on text embedding similarity.
- (3) Finish[answer1 | answer2 | ...], which returns the answer and finishes the task. The answers should be complete node name appeared in the triples. If you don't know the answer, please output Finish[unknown].

Nodes and answers should be separated by tab.

You should generate each step without redundant words.

Here are some examples

. . .

Question: ...

Topic Node: [...]

Thought 1: ...

Action 1: Search/Query[...]

Observation 1: ...

Thought 2: ...

Action 2: Search/Query[...]

Observation 2: ...

...

Thought n: ...

Action *n*: Finish[...]

#### **B** Experiment Details

In our experiments, we consider the following three semi-structured data sets from [24]: Amazon, MAG and PrimeKG. Amazon is a product recommendation data set. and incorporates single-hop relational data involving brands, categories, and products in complementary or substitute relationships. MAG is an academic data set featuring a complex network of nodes and edges centered around paper nodes, particularly focusing on citation and authorship. The queries involve single-hop or multi-hop relational queries along with textual properties sourced primarily from abstracts, such as the paper's topic and methodology. PrimeKG is a biomedical data set related to diseases, drugs, genes/proteins, etc. Detailed statistics of these semi-structured data is in Table 8

#### **C** More Experiment Results

Table 9 compares the macro F1 score of different methods. Note that F1 score is not applicable to RAG-based methods (VSS and VSS+GPT-4 reranker) as these two methods only output a ranked list for all nodes, and does not specify which nodes will be the final answers. As such, we can only compute the F1 score for KGQA-based methods. Here we use the macro F1 score as the micro F1

Table 8: Data statistics of semi-structured data used in experiments

	#node types	#edge types	avg. degree	#nodes	#edges	#queries	train/val/test
Amazon	4	4	18.2	1,035,542	9,443,802	9,100	0.65/0.17/0.18
MAG	4	4	43.5	1,872,968	39,802,116	13.323	0.60/0.20/0.20
PrimeKG	10	18	125.2	129,375	8,100,498	11,204	0.55/0.20/0.25

Table 9: F1 scores on different semi-structured data with different methods. Best highlighted in bold.

dataset	PrimeKG	MAG	Amazon
VSS	0.1263	0.2908	0.4044
VSS+GPT-4 reranker	0.1828	0.4090	0.4479
ToG	0.1265	0.0528	0.1639
RoG	0.2216	0.3267	0.3097
GoG	0.1832	0.4105	0.2079
PASemiQA (GPT-4 Agent)	0.2898	0.4294	0.4542

Table 10: MRR scores on different semi-structured data with different methods. Best highlighted in bold.

dataset	PrimeKG	MAG	Amazon
VSS	0.2141	0.3862	0.5035
VSS+GPT-4 reranker	0.2655	0.4932	0.5534
ToG	0.1584	0.0614	0.2118
RoG	0.2516	0.3867	0.3596
GoG	0.2108	0.4783	0.2628
PASemiQA (GPT-4 Agent)	0.3102	0.5024	0.5568

score should be exactly the same as the Hit@1 score, and we can see that the proposed method still outperforms other baseline when using the macro F1 score as performance metric.

Since KGQA methods only output several nodes that are likely to be the answer instead of a complete ranking on all nodes, computing Hit@k or Recall@k for arbitrary k is often not possible and may cause unfair comparison. Table 10 compares the MRR scores for different baseline methods, and our method PASemiQA generally achieves the best performance under this metric.