IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 March 2026, accepted 26 March 2026, date of publication 6 April 2026, date of current version 21 April 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3681404

RESEARCH ARTICLE

STaRQ-Agent-Investigating Generalization in Knowledge Graph Question Answering via a Multi-Agent Collaborative Framework

LONGQUAN JIANG[1,2], **DEBAYAN BANERJEE**[2], **AND RICARDO USBECK**[2]
[1]University of Hamburg, 20148 Hamburg, Germany
[2]Leuphana University of Lüneburg, 21335 Lüneburg, Germany

Corresponding author: Ricardo Usbeck (ricardo.usbeck@leuphana.de)

This publication was funded by the Open Access Publication Fund of Leuphana University Lüneburg. This publication was also funded by The Ministry of Research and Education under the project 'RESCUE-MATE: Dynamische Lageerstellung und Unterstützung für Rettungskräfte in komplexen Krisensituationen mittels Datenfusion und intelligenten Drohnenschwärmen' (FKZ 13N16844). This publication was also funded by the German Research Foundation (DFG) project NFDI4DS under Grant No.: 460234259.

ABSTRACT Generalization to different Knowledge Graphs (KGs) is a core challenge of Knowledge Graph Question Answering (KGQA). Current models struggle to adapt to unseen KGs due to their underlying heterogeneity. Most approaches depend on data-hungry methods, such as supervised fine-tuning and KG-specific few-shot demonstrations. These systems are limited in cross-KG generalization, and may exhibit hallucination problems, especially in low-resource or domain-specific scenarios. This paper presents **STaRQ-Agent**, an LLM-based multi-agent collaborative framework for the KGQA generalization problem. It comprises a core agent for SPARQL query generation, accompanied by three supportive agents for schema selection, template generation, and query refinement. This collaborative framework helps mitigate hallucination problems inherent in LLMs. STaRQ-Agent was evaluated on LC-QuAD 1.0, QALD-9, MetaQA, and MatKGQA datasets in few-shot and zero-shot settings. Experimental results demonstrate strong overall generalization across diverse KGs, without requiring high-quality annotated data, re-training, or fine-tuning. Specially, on LC-QuAD 1.0, QALD-9, and MetaQA, few-shot STaRQ-Agent outperforms the strongest baseline by 5.1%, 4.3%, and 0.7%, while zero-shot STaRQ-Agent shows only modest F1 drops of 8.7%, 12.3%, and 10.8%. Moreover, the state-of-the-art results on MatKGQA show its robust out-of-distribution generalization to novel KGs in low-resource specific domains.

INDEX TERMS Knowledge graphs, question answering, KGQA, generalizability.

I. INTRODUCTION

Knowledge Graph Question Answering (KGQA) aims to access structured information by translating natural language questions into formal queries over knowledge graphs (KGs), such as DBpedia [1], Freebase [2], YAGO [3], and Wikidata [4]. KGQA's wide application across various fields has made it a long-standing research focus in both academia and industry. An important direction in KGQA is generalization across different KGs. Most existing methods assume the availability of training data for a given KG. However, in practice, obtaining high-quality annotated natural language question-SPARQL pairs is difficult and costly, especially for low-resource specialized domains [5], [6], [7], [8], [9], [10]. For instance, in scientific domains such as healthcare [6], materials science [8] and manufacturing [9], training data is often entirely unavailable due to prohibitive collection costs and stringent privacy regulations.

Recent LLM-based KGQA approaches [11], [12], [13], [14], [15], [16] primarily rely on few-shot in-context learning or supervised fine-tuning with target KG data. ChatKBQA [14] first generates logical forms by fine-tuning LLMs with question and logical form pairs in the KBQA dataset, and then retrieves and replaces the entities and

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

relations in an unsupervised method. RoG [16] first generates KG-grounded relation paths as faithful reasoning plans via the planning module, and employs these reasoning plans to extract valid reasoning paths from the KG. However, these methods remains training-intensive, i.e., requiring large scale annotated QA pairs. Moreover, they generalize poorly to unseen KGs due to the inefficacy of fine-tuning in low-resource settings and the limited coverage of fixed in-context examples.

While some methods invoke external functions to retrieve KG elements such as entities, relations, and neighbours, they lack a deeper understanding of the KG's structural semantics [17], [18]. Modular, multi-stage approaches [19] have also been explored, but insufficient interaction between modules often leads to pipeline errors that impair generalization. A lack of interaction with the real environment inevitably leads to hallucinated knowledge [15].

To address these challenges, we proposed **STaRQ-Agent**, an LLM-based multi-agent collaborative framework that focuses on **S**chema, **T**emplate, Question **a**nd SPA**RQ**L for generalizable KGQA systems. STaRQ-Agent comprises four distinct agents, each specializing in addressing a specific subtask for the KGQA pipeline. To be specific, the **Template Designer (TD)** generates an abstract intermediate representation (i.e., SPARQL template) that reflects the structural form of the question over the specific KG. The **Question Planner (QP)** breaks down a complex question into simple sub-questions and solves them with the guidance of its SPARQL template. When necessary, the **Schema Analyst (SA)** decomposes the whole KG schema into smaller sub-schemas to mitigate the influence of irrelevant information. Finally, the **Query Inspector (QI)** updates the query based on execution feedback to correct errors. To improve generalization, the **TD** employs a question-type-driven approach to avoid hallucinating query templates inconsistent with the target KG structure. Moreover, the **QI** uses an external tool[1] to execute the SPARQL query and refine it from such interaction to the target KG.

The contributions of this work can be summarized as follows:

- STaRQ-Agent, a multi-agent collaborative framework for generalizable KGQA systems, was proposed.
- The experimental results on several KGQA datasets demonstrated the strong generalization ability across different KGs, without annotating data, re-training, or fine-tuning.
- STaRQ-Agent's performance on MetaQA and MatKGQA datasets indicates its great potential to adapt to unseen graphs in domain-specific settings.
- By execution-based interaction with the real-world environment, STaRQ-Agent can mitigate knowledge hallucination problems inherent in LLMs.

The remainder of this paper is organized as follows. Section II reviews prior work on KGQA and its generalization and LLM-based multi-agent frameworks. Next, Section III describes the methodology to tackle the generalization challenge in KGQA. Then, Section IV describes the experimental setup and implementation details, and Section V presents the results and analysis. Section VI and Section VII discuss the ablation study and error analysis. Finally, Section VIII concludes the paper.

[1] https://pypi.org/project/SPARQLWrapper/

II. LITERATURE REVIEW

This section reviews prior studies on KGQA and its generalization and LLM-based multi-agent collaboration frameworks.

A. KGQA GENERALIZATION

Over the past years, several benchmarks [20], [21] have been proposed to evaluate the generalization of modern KGQA systems. Gu et al. [20] introduced a three-level taxonomy of built-in generalization for KGQA – i.i.d., compositionality and zero-shot – and proposed the GrailQA dataset and evaluation protocols that systematically test each level. These levels reveal how systems perform when encountering new structures, new compositions, and new schemas.

Dutt et al. [21] observed that the zero-shot split of GrailQA is biased toward simpler reasoning patterns, failing to sufficiently challenge models' true generalization abilities. To solve this issue, Dutt et al. introduced GrailQA++, a new zero-shot test set tailored to a more balanced distribution of reasoning path complexities. This design ensures the test set is substantially more challenging for contemporary KBQA models.

Jiang et al. [22] argued that many current KGQA approaches exhibit weak generalizability largely because datasets are constructed under a standard i.i.d. assumption (independent and identically distributed training and test splits), which does not reflect real-world variation and challenge scenarios. Instead of creating new datasets, Trivedi et al. proposed a cost-free mitigation strategy: resplitting existing datasets, such as LC-QuAD 1.0 [23] and WebQuestionsSP [24], to support evaluation at compositional and zero-shot levels.

To address cross-KB generalization, a challenge that limits development and scalability of KGQA systems, Ravishankar et al. [19] proposed a two-stage architecture that explicitly separates KG-agnostic semantic understanding and KG-specific execution. Omar et al. [25] presented KGQAn, a universal framework that maps questions to SPARQL queries via abstract representations and just-in-time linking. However, these approaches often rely on annotated data or lack collaborative modeling, limiting their applicability in low-resource scenarios.

B. KGQA

Current studies on KGQA can be broadly categorized into two main types: Information Retrieval (IR)-based methods and Semantic Parsing (SP)-based methods. Recently, with the remarkable reasoning capability of Large Language Models

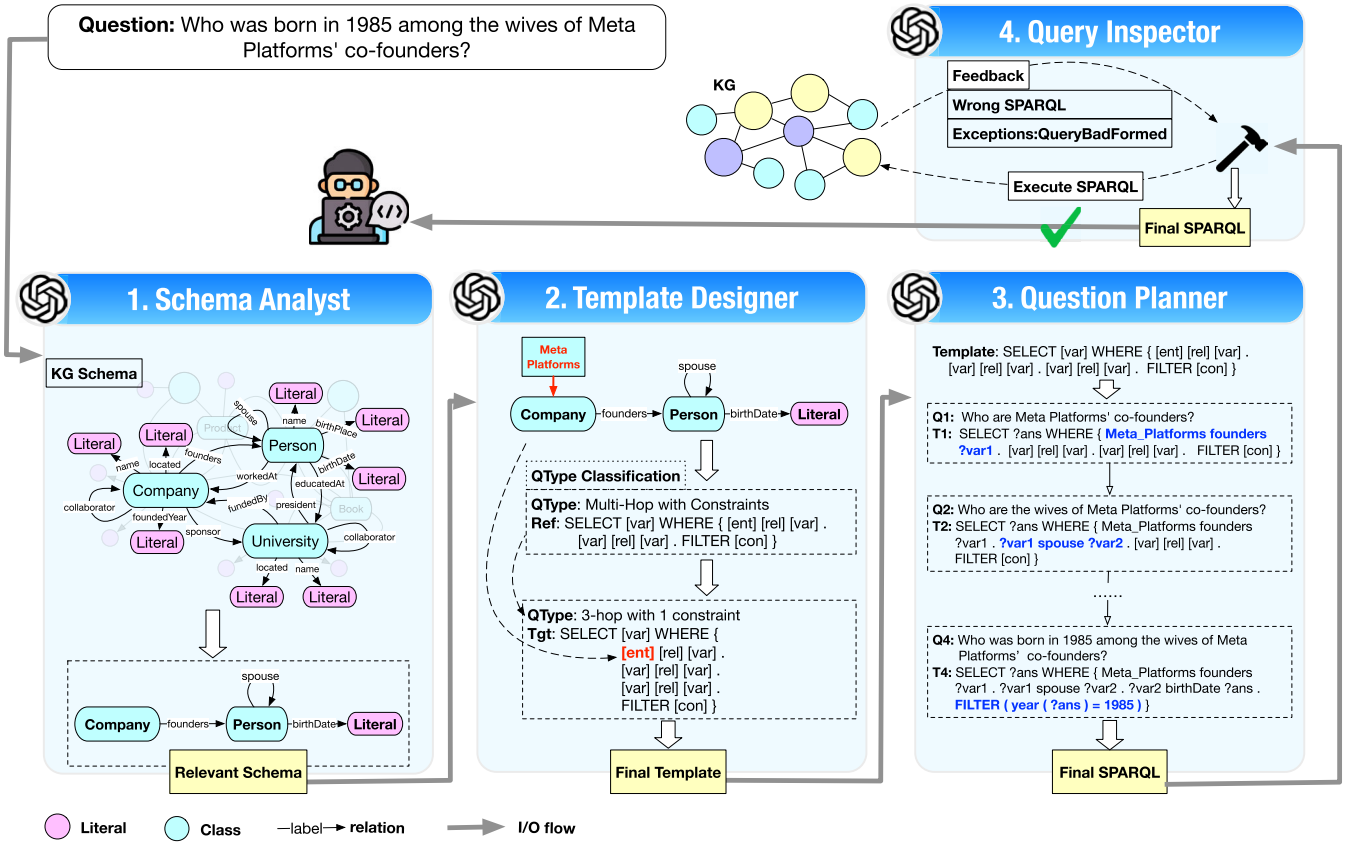

FIGURE 1. Overview of the STaRQ-Agent framework comprising four agents.

(LLMs), many works integrate LLMs with KGs for KGQA task. In this work, we primarily focus on SP-based methods and LLM-based methods.

SP-based methods initially convert natural language questions into structured logical forms (LFs) and then retrieve the correct answers by executing the logical forms on KGs. Early works [26], [27] conduct semantic parsing by generating query graphs, which are an intermediate structured representation that mirrors subgraphs of the underlying KGs and can be directly mapped to logical forms. To effectively narrow the search space and speed up the generation of candidate query graphs, Lan and Jiang [28] incorporate constraints into query graphs. Chen et al. [29] reduce noisy graphs by using predicted query structures to guide candidate graph generation. To address the problem of generating non-executable logical forms, Yu et al. [30] propose integrating the LLM's direct answer into the final answer. Pre-trained language models are increasingly utilized to produce executable logical forms, thereby enabling efficient querying over knowledge graphs [7], [31], [32]. In this context, essential components—such as entities, relations, subgraphs, or textual descriptions extracted from the graph—serve as auxiliary inputs to support logical form generation [33], [34].

LLMs-based methods unify the strong capability of reasoning and actual and comprehensive factual information that KGs provide to solve the KGQA task. Several approaches [14], [15] use LLMs as semantic parsers that produce an initial logical form for a given question, subsequent to which entities and relations are retrieved from the KG and substituted into the logical form. Recent works, such as StructGPT [35], ToG [36], and KnowAgent [38] treat LLMs as agents that conduct multi-turn interactions between KGs and LLMs to arrive at the final answer. Another line of LLMs-based approaches aims to integrate the knowledge of KG into LLMs. RoG [16] samples reasoning paths from the graph, integrates relational information from the knowledge graph into the LLM, and produces reasoning trajectories that guide the retrieval of relevant KG facts for inference. GNN-RAG [37] employs a lightweight GNN to efficiently extract KG information that supports LLM-based reasoning.

C. LLM REASONING PROMPT

Many prior studies have demonstrated that carefully designed prompting strategies can substantially enhance the performance of LLMs on a wide range of complex reasoning tasks. [39], for example, show the effectiveness of prompts constructed in an input-label pairing format, while [40] investigate the impact of varying the number of in-context examples and propose retrieving examples relevant to a given test input for prompt construction. Nevertheless,

due to the lack of sufficient contextual information, basic prompts (e.g., ''Generate a SPARQL query for the following question'') often lead to inaccurate inferences. To address these limitations, several studies have focused on strengthening the reasoning capabilities of LLMs. Chain-of-Thought (CoT) prompting [64], a gradient-free approach, encourages models to produce intermediate step-by-step explanations before outputting final answers, thereby improving reasoning accuracy. Several follow-up studies proposed least-to-most prompting [41], zero-shot CoT [42], and reasoning with self-consistency [43] for solving complicated tasks. FAME [44] employs Monte Carlo planning to construct reliable reasoning trajectories, and methods such as RR [45] and KD-CoT [46] further incorporate knowledge graph retrieval to derive more faithful reasoning plans. Plan-and-Solve [47] guides LLMs to explicitly generate a high-level plan and then perform reasoning conditioned on that plan. ReACT [48] treats LLMs as agents, which interact with the environment to get the latest knowledge for reasoning. In KGQA task, [49] first retrieves relevant facts from a knowledge graph based on entity mentions in a question, then incorporates these facts into the prompt to promote more faithful and factually correct answers. DecomP [51] prompts models to decompose a complex reasoning problem into a sequence of simpler sub-tasks and solve them iteratively. Several recent KGQA approaches instantiate these ideas at the level of logical form generation. SymKGQA [52] uses Chain-of-Symbol prompting to generate KoPL logical forms. Interactive-KBQA [53] produces logical forms through explicit interaction with the KG — decomposing the question into sub-query triples. KB-Coder [13] generates code-style S-expression representations, and KB-Binder [15] prompts LLMs to generate a preliminary logical form as a draft using exemplary demonstrations and then ground the entities and relations on the KB.

D. LLM-BASED MULTI-AGENT FRAMEWORKS

Recent studies have investigated enhancing LLM performance through collaboration among multiple agents. For example, Zhuge et al. [54] introduced NLSOMs, a collaborative framework where agents with different roles engage in iterative interactions, known as ''Mindstorms,'' to address complex tasks. In [55], a closed-loop framework - LATM - was introduced to integrate large and small models with external tools to optimize efficiency and cost. Meanwhile, Park et al. [56] simulated a community of 25 agents to explore emergent language-driven behaviors. In addition, multi-agent collaboration is often applied to translating natural language queries into structured query languages, such as SQL [57], [58], [59], GraphQL [60], and SPARQL [61]. Talaei et al. [57] proposed CHESS, a multi-agent framework for efficient and scalable SQL synthesis. Liang et al. [60] proposed NAT-NL2GQL, a multi-agent framework for translating natural language into graph query languages. Zong et al. [61] proposed Triad, a unified framework that employed multiple LLM-based agents specializing in various KGQA subtasks. Although the aforementioned Multi-Agent Collaboration frameworks resemble ours and are capable of generating SPARQL queries, they fail to consider the shared structures and divergences among multiple KGs in cross-KG scenarios, thereby constraining its generalization capability across heterogeneous KGs, especially in low-resource domain-specific scenarios.

```
[Instruction]
As a professional knowledge engineer, your task is to
extract schema items from [Knowledge graph schema]
relevant to answer [Question]. Please adhere to the
following guidelines:

{guidelines}

Here are some examples:

[Knowledge graph schema]
# Concepts:
{concept list}
# Relations:
{relation list}
[Question]
{query}
[Answer]
{answer}

Here is a new example, please start answering:
```

FIGURE 2. Prompt template for schema analyst agent.

III. METHODOLOGY

To tackle the generalization challenge in KGQA, we proposed STaRQ-Agent, a multi-agent collaborative framework where specialized agents work together to generate SPARQL queries, see Figure 1. STaRQ-Agent centers on a core reasoning agent (Question Planner (QP)) and three supportive agents, each focused on improving a specific aspect of the query generation process. To be specific, Schema Analyst extracts a question-relevant subset of the KG schema; Template Designer constructs a SPARQL query template aligned with the question structure; Question Planner decomposes the question into stepwise sub-questions guided by the template; and Query Inspector, which refines SPARQL queries using execution-based feedback.

A. SCHEMA ANALYST

The KG schema serves as a declarative specification of the knowledge graph's structure and semantics, thereby constraining the space of well-formed and semantically meaningful SPARQL queries. In KGQA, each question corresponds to a specific semantic structure, represented as a SPARQL query graph pattern. Distinct questions may share identical underlying query structures despite variations in lexical and syntactic forms. Extracting question-relevant

schema information from the KG schema is crucial for identifying the semantic structure of a given question [62], [63].

The purpose of this agent is to analyze the KG schema S and select a minimal subset of schema items S' that are semantically relevant to the question Q, from a KG G consisting of a set of classes C and relations R (and entities E optionally). Our motivation stems from the observation that including excessive irrelevant schema items in the prompt increases the risk of selecting incorrect schemas or generating hallucinated content. Assuming a KG G with its own schema S, this process can be described as follows:

$$S' = f_{analyst}(Q, S|M) \tag{1}$$

where $f_s(\cdot|M)$ denotes the function of the analyst prompting the LLM M. The prompt template is shown in Figure 2. For the example question in Figure 1, the output of this agent is a list of KG schema items that are semantically relevant to that question – 'concepts: {"Company," "Person"}, relations: {"birthDate:" {"domain:" "Person," "range:" "Literal"}, "spouse:" {"domain:" "Person," "range:" "Person"}, "founders:" {"domain:" "Company," "range:" "Person"}}'.

B. TEMPLATE DESIGNER

Given the data from the schema analyst (SA) agent, the template designer (TD) agent generates a SPARQL template T that represents the structural mapping of the question Q against the target KG G. This process can be described as follows:

$$T = f_{designer}(Q, S'|M) \tag{2}$$

where, $f_{designer}$ is the function of template designer prompting the LLM M. The prompt template is shown in Figure 3. For the example question in Figure 1, the output of this agent is the SPARQL query template – "`SELECT ?ans WHERE { [ent] [rel] [var]. [var] [rel] [var]. [var] [rel] ?ans. FILTER [con] }`".

In practical KGQA settings, user questions often involve complex structures that demand multi-hop reasoning, attribute filtering, and aggregation in the resulting SPARQL queries. To ensure better structural generalization across different KGs, we adopted a question-type-driven in-context learning strategy. Specifically, in the prompt, we included a generic SPARQL query template definition from Jiang et al. [7], representative examples for each question type (see Table 1), and detailed instructions for generating a query template aligned with both the input question and the KG schema.

The basic generation process is as follows. First, the model determines the question type and refers to its corresponding example template. Then, it refines the example template based on the understanding of the question and the schema. Repeat this step until the final template matches the predicted question type and the schema. In the generation process,

```
[Instruction]
As a professional knowledge engineer, your task is to
generate a SPARQL template for the [Question] according to
the [Generic SPARQL Template] and [Knowledge graph
schema]. You need to first identify the question type and
refers to the basic example template of that type, and refines
it based on the question and knowledge graph schema.
Please adhere to the following guidelines:

{guidelines}

[Generic SPARQL Template]
{template definition}

[Question Types]
{type description and examples}

Here are some examples:
{examples}

Here is a new example, please think step by step:

[Knowledge base schema]
{desc_str}
[Question]
{query}
[Template]
```

FIGURE 3. Prompt template for template designer agent.

the agent should adhere to the following guidelines for alignments: (1) **Relation Alignment**. The subject and object of the relation placeholder [rel] in each triple pattern must conform to the domain and range specifications defined in the KG schema. For example, for the question "who founded Apple Inc.?", if "founded" is defined with the founder as the subject and the organization as the object—the triple pattern should be "`[var] [rel] [ent]`" instead of "`[ent] [rel] [var]`". (2) **Triple Pattern Alignment**. The number of required triple patterns should correspond to the reasoning path implied by the question and the KG schema. For multi-hop questions, the minimum number of triple patterns is 2. if a new question requires three-hop reasoning, the model is instructed to add an additional triple pattern of the form `[var] [rel] [var]`. If no linked entities are available, the model is instructed to add an additional triple pattern in the form of `[var] rdfs:label [val]` for entity label matching, or include a constraint such as `FILTER [con]` for string matching or regular expression filtering. (3) **Constraint Alignment**. The required number of constraints is determined by the model's interpretation of explicit and implicit conditions in the question and the KG schema. If linked entities are not provided, the model is instructed to include a `FILTER [con]` clause for string or regex-based matching. For example, considering the question in Figure 1 with a single constraint. Both the template and final SPARQL query must include exactly one condition - "`FILTER ( year ( ?ans ) = 1985 )`".

Considering the example question shown in Figure 1, given relevant concepts and relations with their own domain

and range from the output of **Schema Analyst** Agent, it is recognized as "multiple hops with constraints" type with 3 hops and 1 constraint. The model refers to the example template of the question type —"`SELECT [var] WHERE { [ent] [rel] [var]. [var] [rel] [var]. FILTER [con] }`", and generate its own query template following a series of aforementioned guidelines. First, the model starts from the placeholder of the linked entity mentioned in the question - "Meta Platforms" and following the Relation Alignment guideline, generates the correct triple template - "`[ent] [rel] [var]`". Next, following the Triple Pattern Alignment guideline, it produces the remaining two triple patterns that align with the extracted schema structure. Finally, following the Constraint Alignment guideline, it outputs 1 filter constraint. Together, these steps yield the final query template (see the output of Template Designer in Figure 1). However, if the entity "Meta Platform" is not identified, following both Triple Pattern and Constraint Alignments, the template has to be modified to "`SELECT [var] WHERE { [var] [rel] [var]. [var] rdfs:label [var]. [var] [rel] [var]. [var] [rel] [var]. FILTER [con] FILTER [con] }`".

C. QUESTION PLANNER

Question Planner (QP) agent leverages the reasoning capability of LLMs to decompose complex questions into sub-problems, solving each independently. It receives relevant schema and the SPARQL query template for the target KG from the Template Designer (TD), then performs decomposition and sub-problem template filling iteratively until all placeholders are resolved. The process can be formally described as follows:

$$P_M(Y|Q, S', T) = \prod_{j=1}^{L} P_M(Y^j|Y^{<j}; Q^j, S', T) \quad (3)$$

where, Q^j and Y^j are the j-th sub-question and its SPARQL query generated by the LLM M given the previous SPARQL query $Y^{<j}$, the selected KG schema S', L is the number of sub-questions. The prompt template is shown in Figure 4.

Question decomposition strategy follows a structured progression: (1) single-hop questions containing the core entities from the question, (2) multi-hop complex questions building on prior results, (3) constrain-based complex questions (filtering or aggregation). The template determines the required decomposition and reasoning steps. Results are merged by filling the corresponding triple patterns or conditions into the output from the previous step in the template. For the example question in Figure 1, its decomposition follows this strategy and its the final SPARQL query – "`SELECT ?ans WHERE { Meta_Platforms founders ?var1. ?var2 spouse ?var2. ?var2 birthDate ?ans. FILTER ( year ( ?ans ) = 1985 ) }`".

```
[Instruction]
As a professional knowledge engineer, your task is to
generate a SPARQL query for the [Question] by filling up the
[Template] with the actual values. You need to decompose
the [Question] into sub-questions, and fill in the
corresponding values in [Template] of each sub-questions.
Please adhere to the following guidelines:

{guidelines}

Here are some examples:
{examples}

Here is a new example, please think step by step:

[Knowledge graph schema]
{desc_str}
[Question]
{query}
[Template]
```

FIGURE 4. Prompt tempalte for question planner agent.

Here, we employ Chain-of-Thought reasoning [64] to decompose the question Q into sub-questions and populate their corresponding components within the SPARQL template. The agent begins by identifying an entity mentioned in the question and selects relevant schema elements from the KG to populate [ent], [rel], [var] and [cls] fields. It then connects and completes the next related triples until all placeholders are filled and the answer node is reached. If the question includes constraints, it assigns values or expressions to [con] based on its interpretation of the question. Consider the example in Figure 1, at step 1, the agent starts from the entity *Meta Platforms* and the first sub-question, and begins by filling the template segment [ent] [rel] [var] with *Meta_Platforms founders ?var1*. Next, the agent proceeds from this point to identify the next connected triple pattern [var] [rel] [var], and then fills it with *?var1 spouse ?var2*. This process continues until all placeholders in the template have been fully instantiated.

D. QUERY INSPECTOR

LLMs generate hallucinations that undermine both grammaticality and faithfulness, leading to possible failures during execution [65]. To mitigate this issue, a verifier is needed to evaluate the syntactic and semantic correctness of the generated queries and revise them based on the detected errors. The objective of this agent is to correct and produce a valid SPARQL query that accurately answers Q. This process can be formally described as follows:

$$Y = f_{inspector}(E, Y', Q, S'|M) \quad (4)$$

where $f_{inspector}(\cdot|M)$ denotes the function of Query Inspector prompting the LLM M. The prompt template is shown in Figure 5. For the example question in Figure 1, the output is the new SPARQL query without any errors – "`SELECT ?ans WHERE { Meta_Platforms`

[Instruction]
As a professional knowledge engineer, your task is to inspect and refine the [Old SPARQL] for the [Question] based on [Knowledge graph schema]. If [SPARQL Error] and [Exception Class] occur, please fix it up and output the final [Correct SPARQL].

[Knowledge base schema]
{desc_str}
[Question]
{query}
[Old SPARQL]
{old_sparql}
[SPARQL error]
{sparql_error}
[Exception class]
{exception_class}

Now, please start the process.
[Correct SPARQL]

FIGURE 5. Prompt template for query inspector agent.

`founders ?var1. ?var2 spouse ?var2. ?var2 birthDate ?ans. FILTER ( year ( ?ans ) = 1985 ) }`".

As shown in Figure 1, the Query Inspector iteratively validates SPARQL queries by checking syntax, execution feasibility, and non-empty results from the target KG. This process continues until reaching the maximum attempt limit or obtaining a syntactically correct query with valid results. Its goal is self-checking and self-correction, significantly enhancing framework robustness and accuracy while reducing syntax errors, schema linking issues, and other basic failures.

IV. EXPERIMENTAL SETUP

A. KGS AND DATASETS

For the KGQA generalization evaluation, we used DBpedia [1], MatKG [8], MoviesKG [66] as target KGs, and evaluate our method with MetaQA [67], MatKGQA [68], LC-QuAD 1.0 [23] and QALD-9 [69].

B. METRICS AND MODELS

We used Mixtral-8x7b-Instruct [70] and GPT-4.5 [71] as LLM backbone. Due to cost constraints, we evaluated STaRQ-Agent on MetaQA using only Mixtral-8x7b-Instruct. We used F1-score as evaluation metric.

C. BASELINE METHODS

We compared STaRQ-Agent with the following baselines and their variants in the full-shot, few-shot and zero-shot settings: 1) **BYOKG** [68], an unsupervised QA system with a symbolic agent randomly exploring the target KG. 2) **NSM** [72], a KGQA method trained on teacher-student networks. 3) **Pangu** [33], a KGQA method that uses language models to discriminate plans generated by a symbolic agent. 4) **Triad** [61], a multi-agent framework with a few-shot prompt learning. 5) **gAnswer** [26], a subgraph matching based QA method over RDF KGs. 6) **EDGQA** [73], an Entity Description Graph (EDG) based question decomposition for complex KGQA. 7) **KGQAn** [25], a universal QA framework that maps questions to SPARQL queries via abstract representations and just-in-time linking. 8) **GPT-4** [71], a large-scale, multimodal model developed by OpenAI team.

D. IMPLEMENTATION DETAILS

We conducted all experiments on NVIDIA RTX A6000 GPUs. We used the DBpedia 2016-10 release[2] for LC-QUAD and QALD-9, and MatKG and MoviesKG dumps from Agarwal et al. [68][3] for MatKGQA and MetaQA, respectively. Each KG is deployed with a local SPARQL query endpoint via Virtuoso to support answer retrieval. The temperature of GPT-4.5 is set to 0.1. For entity linking, we used spacy dbpedia spotlight[4] for LC-QuAD 1.0 and QALD-9, and a simple string pattern matching approach for MetaQA. For MatKG, we used gold entities. To reduce redundant and irrelevant schema items, we employ a pre-processed simplified ontology for LC-QuAD and QALD-9. Using Sentence-BERT, we calculate semantic similarity between questions and schema items, retaining only those above a predefined threshold.

V. EVALUATION AND RESULTS

In this section, we conducted a series of experiments to evaluate the effectiveness of STaRQ-Agent in improving generalization of KGQA systems. To this end, we compared STaRQ-Agent to baselines under three settings: (1) **Full-shot**, where the model is trained on a large annotated dataset from the target KG; (2) **Few-shot**, where the model is trained on a limited amount of labelled data or provided with only a few in-context examples; and (3) **Zero-shot**, where the model receives no labelled data or in-context examples from the target KG during inference. Note that, in the few-shot setting, a small number of examples from the target KG are included, while in the zero-shot setting only a demonstration KG schema along with a single corresponding example is included. The full-shot setting is to indicate the upper bound of performance.

A. GENERALIZATION PERFORMANCE COMPARISON

We observe that from the primary experimental results shown in Tables 2, 3 and 4:

STaRQ-Agent exhibits strong overall generalization across various KGs: On LC-QuAD 1.0 (DBpedia), QALD-9 (DBpedia), MetaQA (MoviesKG), and MatKGQA (MatKG), we find that STaRQ-Agent, in both few-shot and zero-shot settings, performs the best, showing strong generalization

[2]https://downloads.dbpedia.org/2016-10/
[3]https://drive.google.com/drive/folders/1YfGmENowy3H1meysBi4AG7oyY2fHITDz
[4]https://spacy.io/universe/project/spacy-dbpedia-spotlight

TABLE 1. Categories for questions of varying SPARQL query complexity.

	QType	Example	Template
I	Single-hop	Who founded Meta Platforms?	SELECT [var] WHERE { [ent] [rel] [var] . }
II	Single-hop with aggregator(s)	How many co-founders does Meta Platforms have?	SELECT (COUNT([var]) AS [var]) WHERE { [ent] [rel] [var] . }
III	Multi-hop	List all wives of Meta Platforms' co-founders?	SELECT [var] WHERE { [ent] [rel] [var] . [var] [rel] [var] . }
IV	Multi-hop with constraint(s)	Who was born in 1984 among Meta Platforms' co-founders?	SELECT [var] WHERE { [ent] [rel] [var] . [var] [rel] [var] . FILTER [con] }
V	Multi-hop with aggregator(s)	What is the largest among all the birth dates of Meta Platforms' co-founders?	SELECT (MIN ([var]) as [var]) WHERE { [ent] [rel] [var] . [var] [rel] [var] . }
VI	Multi-hop with aggregator(s) and constraint(s)	What is the average age of Meta Platforms' co-founders who were born after 1980?	SELECT (COUNT ([var]) as [var]) WHERE { [ent] [rel] [var] . [var] [rel] [var] . FILTER [con] }

TABLE 2. Results of F1 scores on LC-QuAD 1.0 and QALD-9 with a general KG – DBpedia, using GPT-4.5.

Shot	Method	LC-QuAD 1.0	QALD-9
full-shot	gAnswer	-	29.8
	EDGQA	53.1	32
	KGQAn	51.6	44.1
few-shot	GPT-4	34.0	24.9
	Triad	56.4	41.6
	STaRQ-Agent	**59.3**	**43.4**
zero-shot	STaRQ-Agent	54.1	38.03

TABLE 3. Results of F1 scores on MetaQA with a domain-specific KG – MoviesKG, using Mixtral-8 × 7b-Instruct.

Shot	Method	Overall	1-hop	2-hop	3-hop
full-shot	NSM-FT (SOTA)	**98.82**	**97.1**	**99.99**	**98.9**
few-shot	Pangu-ICL	92.38	98.80	94.21	86.01
	BYOKG	97.31	98.27	90.76	76.08
	STaRQ-Agent	**98.01**	**98.92**	**96.32**	**90.22**
zero-shot	Pangu-ICL + X	64.87	63.40	66.74	63.96
	BYOKG + X	83.01	95.25	81.85	75.69
	STaRQ-Agent	**87.40**	**98.12**	**90.71**	**80.34**

across various KGs, without the need for re-training or fine-tuning. Specifically, in the few-shot setting, STaRG-Agent results in 2.9 and 1.8 F1 improvements on LC-QuAD 1.0 and QALD-9, respectively.

Few-shot STaRQ-Agent is comparable or superior to full-shot SOTA models: On LC-QuAD 1.0, few-shot STaRQ-Agent achieves an F1 score of 59.3, outperforming the full-shot methods, i.e., EDGQA (53.1) and KGQAn (51.6). On QALD-9 and MetaQA, few-shot STaRG-Agent achieves near-SOTA results, close to the best-performing full-shot method (KGQAn and NSM-FT, respectively).

TABLE 4. Results of F1 scores on MatKGQA with a domain-specific KG – MatKG, using GPT-4.5.

Method	Overall
BYOKG	62.25
STaRQ-Agent	**64.70**

Explicit KG-specific information matters: On LC-QuAD 1.0, QALD-9, and MetaQA, few-shot STaRQ-Agent performs relatively better than the zero-shot variant – (59.3/54.1, 43.4/38.03, 98.01/87.40), with 5.2, 5.37, and 10.69 F1 gains, respectively, indicating that the integration of KG-specific information improves generalization performance on that target KG.

Zero-shot STaRQ-Agent shows strong out of distribution generalization: Compared to the few-shot STaRQ-Agent, the zero-shot variant yields F1 score decreases of *5.2*, *5.97*, and *10.61* points on LC-QuAD 1.0, QALD-9, and MetaQA, respectively. It also achieves the SOTA result – *64.70* F1 on MatKGQA (MatKG) in Table 4. These findings demonstrate the zero-shot STaRQ-Agent's robust out-of-distribution (OOD) generalization capabilities and underscore the value of the proposed schema-guided, adaptive SPARQL query template design, which supports reliable generalization to unseen query structures. As shown in Table 5, this generalization manifests distinctly across question types. For single-hop questions, the pattern Q1→Q2 hinges on whether the identified entity's class belongs to the domain or range of the relevant relation, thereby determining its role as subject or object. This capability extends to complex questions in Table 1 as well: the Q3→Q5 pattern is governed by the number of identified entities, whereas Q3→Q6 depends on the number of relevant relations. Generalizing to unseen query structures relies on both the complexity of the question and the relevant schema items.

B. ROBUSTNESS ANALYSIS

Robustness is closely tied to data distribution. Distributional shifts from general-purpose knowledge graphs to domain-specific ones significantly impact the robustness in KGQA systems. We assess the robustness of our proposed method when generalizing to previously unseen domain-specific knowledge graphs. In particular, we analyze the ability of Template Designer to generalize to novel SPARQL query structures. Table 6 presents the accuracy of Template Designer in predicting SPARQL query structures on MatKGQA across different distributional settings. IID refers to test questions whose templates mach those in-context demonstrations (as shown in Table 1). Notably, for single-hop questions, "`SELECT [var] WHERE { [ent] [rel] [var]. }`" differs from "`SELECT [var] WHERE {`

TABLE 5. Question examples where in-distribution (IID) templates generalize to out-of-distribution (OOD) templates, under the guidance of schema items. Entity mentions and their corresponding placeholders in the templates are underlined, and relation mentions and their corresponding placeholders are italicized. The "Schema Items" column lists the relations mentioned in the question along with their domain and range, as well as the entities and their corresponding classes.

No.	Type	QType:Single-hop		Schema Items
Q1	IID	Who *founded* Meta Platforms?	select [var] where { [ent] *[rel]* [var] . }	founded_by: (Company, Person) Meta Platforms: Company
Q2	OOD	what films did Robert Strauss *act* in?	select [var] where { [var] *[rel]* [ent] . }	movie.starred_actors: (Movie, Person) Robert Strauss: Person
No.	**Type**	**QType:Multi-hop**		**Schema Items**
Q3	IID	List all *wives* of Meta Platforms' *co-founders*?	select [var] where { [ent] *[rel]* [var] . [var] *[rel]* [var] . }	founded_by: (Company, Person) spouse: (Person, Person) Meta Platforms: Company
Q4	OOD	which person *directed* the movies *starred* by John Krasinski?	select [var] where { [var] [rel] [ent] . [var] [rel] [var] . }	movie.starred_actors: (Movie, Person) movie.directed_by: (Movie, Person) John Krasinski: Person
Q5	OOD	What place did Edwin Adams *die* at, which *gave birth to* William A Purtell?	select [var] where { [ent] *[rel]* [var] . [ent] *[rel]* [var] . }	placeOfDeath: (Person, Place) birthPlace: (Person, Place) Edwin Adams: Person William A. Purtell: Person
Q6	OOD	what are the *genres* of the films whose *screenwriters* also *wrote* A Walk in the Clouds?	select [var] where { [ent] *[rel]* [var] . [var] *[rel]* [var] . [var] *[rel]* [var] . }	movie.written_by: (Movie, Person) movie.has_genre: (Movie, Genre) A Walk in the Clouds: Movie

TABLE 6. Accuracy of template designer on MatKGQA in predicting SPARQL query structures across different distributional settings.

QType	Acc. (%)		
	I.I.D.	O.O.D.	Overall
Single-hop	100%	80%	86%
Single-hop with agg	100%	100%	100%
Multi-hop	84%	85%	85%
Multi-hop with agg	80%	61%	66%
Overall	84%	81%	80%

TABLE 7. F1 scores of STaRQ-Agent ablation study on LC-QuAD 1.0 test set.

Method	F1 score	Perf Gain
STaRQ-Agent	59.30	-
w/o SA	40.68	↓ 31.4%
w/o TD	42.29	↓ 28.7%
w/o QP	45.52	↓ 23.2%
w/o QI	38.26	↓ 35.5%

`[var] [rel] [ent]. }`", and similarly for multi-hop questions, "`SELECT [var] WHERE { [ent] [rel] [var]. [var] [rel] [var]. }`" differs from "`SELECT [var] WHERE { [var] [rel] [var]. [ent] [rel] [var]. }`", as these represent distinct underlying graph structures.

As shown in Table 6, STaRQ-Agent achieves an overall accuracy of 81% in predicting OOD SPARQL query structures, demonstrating robust generalization capabilities to unseen query structures. Furthermore, when examined from the perspective of question complexity, both IID and OOD accuracy exhibit a consistent decline as question complexity increases.

TABLE 8. The percent of generated SPARQL queries containing hallucinated entity names and relations on MatKGQA.

Method	Percent
STaRQ-Agent w/o QI	34%
STaRQ-Agent with QI	22% (↓)

VI. ABLATION STUDY

Table 7 shows the zero-shot ablation results for STaRQ-Agent on the LC-QuAD 1.0 set, comparing different variants with or without specific components. The findings show that removing the Query Inspector and Schema Analyst from the STaRQ-Agent model leads to a significant performance decline, highlighting their importance in reducing noise and improving generation accuracy for large-scale, ontology-rich general KGs. The ablation study indicates that each component of STaRQ-Agent model plays a crucial role in achieving high accuracy, as its removal resulted in decreased performance.

LLMs are prone to hallucinate non-existent entity names and relations when applied to KGQA tasks. We further analyze the effectiveness of Query Inspector in alleviating hallucinations. We compute, in both the presence and absence of Query Inspector, the percent of generated SPARQL queries containing hallucinated entity names and relations on MatKGQA. As reported in Table 8, incorporating Query Inspector leads to a 35% reduction in hallucinated SPARQL queries compared with the setting without it. This finding indicates that execution-based interaction with the real-world environment can enhance the faithful reasoning of LLMs.

VII. ERROR ANALYSIS

To assess the limitations of STaRQ-Agent, we analyzed error types using these KGQA datasets, as shown in Figure 6. Typical examples of different error types are shown in Table 9.

TABLE 9. Different types of error cases.

No.	Question	Ground Truth	Pred SPARQL	Error Type
Q1	what genres are the movies starred by Trog actors	select distinct ?ans where { <m.41451> <movie.starred_actors> ?var1 . ?var2 <movie.starred_actors> ?var1 . ?var2 <movie.has_genre> ?ans . }	select distinct ?ans where { ?var1 <movie.starred_actors> <m.41451> . ?var1 <movie.has_genre> ?ans . }	Structure Error
Q2	Which frequent flyer program has the most airlines?	select ?uri where { ?airline rdf:type dbo:Airline . ?airline dbp:frequentFlyer ?uri. } group by ?uri order by desc(count(distinct ?airline)) offset 0 limit 1	select distinct ?ans where { ?var1 dbp:frequentFlyer ?ans . ?var1 rdf:type dbo:Airline . }	Complex Syntax
Q3	what are the descriptors of mesenchymal stem	select distinct ?ans where { ?ans <descriptor.symmetry_phase> <m.49329> . }	select distinct ?ans where { ?ans <descriptor.material> <m.49329> . }	Schema Missing
Q4	What is Batman's real name?	select distinct ?label where { dbr:Batman foaf:name ?label . }	select ?ans where { dbr:Batman dbo:name ?ans . }	Schema Missing
Q5	what is the application associated with pump that also uses zirconium oxide	select distinct ?ans where { ?ans <application.application> <m.16566> . ?ans <application.material> <m.2075> . }	select distinct ?ans where { <m.16566> <material.application> ?ans . ?ans <application.material> <m.20753> . }	Schema Linking
Q6	How many gold medals did Michael Phelps win at the 2008 Olympics?	select count(?sub) as ?c where { ?sub dbo:goldMedalist dbr:Michael_Phelps . filter (contains (str(?sub), "2008") && contains (str(?sub), "Olympics")) }	select (count(?var) as ?ans) where { ?var dbo:goldMedalist dbr:Michael_Phelps . filter (str(?var) = "2008 Olympics") }	Expression
Q7	What place did Edwin Adams die at, which gave birth to William A Purtell?	select distinct ?uri where { dbr:Edwin_Adams_(politician) dbp:placeOfDeath ?uri . dbr:William_A._Purtell dbo:birthPlace ?uri . }	select distinct ?ans where { dbr:William_A._Purtell dbo:birthPlace ?ans . dbr:Edwin_Adams dbp:placeOfDeath ?ans . }	Other

Schema linking errors are the most frequent (28%) and arise when schema items are incorrectly recognized, either due to limitations of external linking components or misunderstanding of the question. This highlights the need to improve both linking accuracy and the use of richer contextual signals. In the case of question Q5 in Table 9, for example, interpreting "pump" as a material, our model incorrectly associates the relation triggered by "pump" with "<material.application>", even though it correctly predicts the underlying template and identifies "<application.application>" as one of relevant schema items.

Schema missing errors (22%) occur when the analyst agent (SA) fails to detect all the required schema elements, leading to incomplete SPARQL queries. For questions Q3 and Q4 in Table 9, our model does not identify the relation "<descriptor.symmetry_phase>" because "mesenchymal stem" is misinterpreted as a material in Q3, and it also overlooks the relation "foaf:name" from the external FOAF ontology in Q4.

Structure errors (17%) refers to failures in selecting an appropriate SPARQL query template that reflects the structural requirements of the question. For question Q1 in Table 9, our model predicts its SPARQL query with only two triples, whereas the gold query consists of three triples. Together with schema missing errors, these cases suggest that a more thorough and precise understanding of both the question and the KG schema is required.

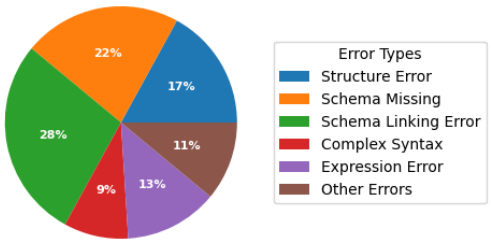

FIGURE 6. Error distribution across all KGQA datasets.

Complex syntax errors (9%) are associated with questions that demand advanced SPARQL constructs (e.g., UNION, OPTIONAL) not covered by the predefined template set, illustrating the difficulty of handling long-tail query patterns. For question Q2 in Table 9, our model fails to produce the final SPARQL query involving grouping and ordering of

the result set, which exceeds the expressive capacity of our current structural templates.

Expression errors (13%) concern cases with complex constraints, where the model often struggles to generate accurate expressions. In question Q6 in Table 9, for instance, our model does not correctly produce the filter expression that combines two string-matching conditions with a logical AND, namely "FILTER (contains (str(?sub), "2008") && contains (str(?sub), "Olympics"))". Complex syntax and expression errors together underscore the importance of developing a richer collection of reusable templates and expression patterns to better cover diverse query requirements.

Other errors (9%) pertain to content-level issues, where the model hallucinates facts that are not supported by the KG. For question Q7 in Table 9, this manifests as the generation of a non-existent entity. This highlights the need for reliable solutions to mitigate hallucinations.

VIII. CONCLUSION AND FUTURE WORK

In this work, we proposed an LLM-based multi-agent collaborative framework, **STaRQ-Agent**, with an emphasis on Schema, Template, Question, and SPARQL, for KGQA generalization. Its generalization ability across diverse KGs was evaluated. Therefore, general and domain-specific KGs of varying sizes can be used — without requiring annotation, retraining, or fine-tuning. STaRQ-Agent performs competitively in both zero-shot and few-shot settings, demonstrating that our proposed method can achieve strong generalization across diverse KGs, and explicit incorporation of KG-specific information aids adaptability. Ablation studies confirm the effectiveness of the multi-agent collaborative workflow in boosting accuracy and generalization. Notably, STaRQ-Agent generalizes particularly well to domain-specific, low-resource KGs.

Limitations. 1) The prompts used by the agents may be suboptimal and could benefit from further refinement. 2) While our template design captures most structural patterns shared by different knowledge graphs, it falls short on handling more complex queries, such as those with UNION or nested SELECT constructs. 3) In the few-shot setting, generalization could be further improved by retrieving multiple similar questions as in-context examples. 4) Hallucinations still occur at times, despite mitigation efforts. Future directions may use a symbolic verifier to explore the KG.

REFERENCES

[1] P. N. Mendes, M. Jakob, and C. Bizer, "DBpedia: A multilingual cross-domain knowledge base. European language resources association (ELRA)," in *Proc. 8th Int. Conf. Lang. Resour. Eval. (LREC)*. Istanbul, Turkey: European Language Resources Association (ELRA), 2012, pp. 1813–1817.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. data*, Jun. 2008, pp. 1247–1250.

[3] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," in *Proc. 16th Int. Conf. World Wide Web*, May 2007, pp. 697–706.

[4] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher, "From freebase to wikidata: The great migration," in *Proc. 25th Int. Conf. World Wide Web*, Apr. 2016, pp. 1419–1428.

[5] Y. Feng, L. Zhou, C. Ma, Y. Zheng, R. He, and Y. Li, "Knowledge graph–based thought: A knowledge graph–enhanced LLM framework for pan-cancer question answering," *GigaScience*, vol. 14, Jan. 2025. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/39775838/

[6] Z. Jiang, C. Chi, and Y. Zhan, "Research on medical question answering system based on knowledge graph," *IEEE Access*, vol. 9, pp. 21094–21101, 2021.

[7] L. Jiang, J. Huang, C. Möller, and R. Usbeck, "Ontology-guided, hybrid prompt learning for generalization in knowledge graph question answering," in *Proc. 19th Int. Conf. Semantic Comput. (ICSC)*, Feb. 2025, pp. 28–35.

[8] V. Venugopal, S. Pai, and E. Olivetti, "MatKG: The largest knowledge graph in materials science - entities, relations, and link prediction through graph representation learning," 2022, *arXiv:2210.17340*.

[9] G. Buchgeher, D. Gabauer, J. Martinez-Gil, and L. Ehrlinger, "Knowledge graphs in manufacturing and production: A systematic literature review," *IEEE Access*, vol. 9, pp. 55537–55554, 2021.

[10] Y. Fettach, M. Ghogho, and B. Benatallah, "Knowledge graphs in education and employability: A survey on applications and techniques," *IEEE Access*, vol. 10, pp. 80174–80183, 2022.

[11] C. Tan, Y. Chen, W. Shao, and W. Chen, "Make a choice! Knowledge base question answering with in-context learning," 2023, *arXiv:2305.13972*.

[12] Z. Li, S. Fan, Y. Gu, X. Li, Z. Duan, B. Dong, N. Liu, and J. Wang, "Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 17, pp. 18608–18616.

[13] Z. Nie, R. Zhang, Z. Wang, and X. Li, "Code-style in-context learning for knowledge-based question answering," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 17, pp. 18833–18841.

[14] H. Luo, E. Haihong, Z. Tang, S. Peng, Y. Guo, W. Zhang, C. Ma, G. Dong, M. Song, W. Lin, Y. Zhu, and A. T. Luu, "ChatKBQA: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models," in *Proc. Findings Assoc. Comput. Linguistics ACL*, 2024, pp. 2039–2056.

[15] T. Li, X. Ma, A. Zhuang, Y. Gu, Y. Su, and W. Chen, "Few-shot in-context learning on knowledge base question answering," in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 6966–6980.

[16] L. Luo, Y. F. Li, G. Haffari, and S. Pan, "Reasoning on graphs: Faithful and interpretable large language model reasoning," in *Proc. 12th Int. Conf. Learn. Represent.*, 2024. [Online]. Available: https://openreview.net/forum?id=ZGNWW7xZ6Q

[17] Y. Gu, Y. Shu, H. Yu, X. Liu, Y. Dong, J. Tang, J. Srinivasa, H. Latapie, and Y. Su, "Middleware for LLMs: Tools are instrumental for language agents in complex environments," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2024, pp. 7646–7663.

[18] Z. Zhang, L. Wen, and W. Zhao, "Rule-KBQA: Rule-guided reasoning for complex knowledge base question answering with large language models," in *Proc. 31st Int. Conf. Comput. Linguistics*, 2025, pp. 8399–8417. [Online]. Available: https://aclanthology.org/2025.coling-main.562/

[19] S. Ravishankar, D. Thai, I. Abdelaziz, N. Mihindukulasooriya, T. Naseem, P. Kapanipathi, G. Rossiello, and A. Fokoue, "A two-stage approach towards generalization in knowledge base question answering," in *Proc. Findings Assoc. Comput. Linguistics, EMNLP*, 2022, pp. 5571–5580.

[20] Y. Gu, S. Kase, M. Vanni, B. Sadler, P. Liang, X. Yan, and Y. Su, "Beyond I.I.D.: Three levels of generalization for question answering on knowledge bases," in *Proc. Web Conf.*, Apr. 2021, pp. 3477–3488.

[21] R. Dutt, S. Khosla, V. B. Kumar, and R. Gangadharaiah, "Designing harder benchmarks for evaluating zero-shot generalizability in question answering over knowledge bases," in *Proc. 1st Workshop Natural Lang. Reasoning Structured Explanations (NLRSE @ ACL 2023)*, Stroudsburg, PA, USA. Toronto, ON, Canada: Association for Computational Linguistics, Jul. 2023. [Online]. Available: https://virtual2023.aclweb.org/paper_ACL_90.html

[22] L. Jiang and R. Usbeck, "Knowledge graph question answering datasets and their generalizability: Are they enough for future research?" in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2022, pp. 3209–3218.

[23] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann, "LC-QUAD: A corpus for complex question answering over knowledge graphs," in *Proc. Int. Semantic Web Conf.*, 2017, pp. 210–218.

[24] A. Talmor and J. Berant, "The web as a knowledge-base for answering complex questions," 2018, *arXiv:1803.06643*.

[25] R. Omar, I. Dhall, P. Kalnis, and E. Mansour, "A universal question-answering platform for knowledge graphs," *Proc. ACM Manage. Data*, vol. 1, no. 1, pp. 1–25, May 2023.

[26] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao, "Answering natural language questions by subgraph matching over knowledge graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 824–837, May 2018.

[27] W.-T. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, 2015, pp. 1321–1331.

[28] Y. Lan and J. Jiang, "Query graph generation for answering multi-hop complex questions from knowledge bases," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 969–974.

[29] Y. Chen, H. Li, Y. Hua, and G. Qi, "Formal query building with query structure prediction for complex question answering over knowledge base," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3751–3758.

[30] D. Yu, S. Zhang, P. Ng, H. Zhu, A. H. Li, J. Wang, Y. Hu, W. Wang, Z. Wang, and B. Xiang, "DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases," in *Proc. 11th Int. Conf. Learn. Represent.*, 2023. [Online]. Available: https://openreview.net/forum?id=XHc5zRPxqV9

[31] M. R. A. H. Rony, U. Kumar, R. Teucher, L. Kovriguina, and J. Lehmann, "SGPT: A generative approach for SPARQL query generation from natural language questions," *IEEE Access*, vol. 10, pp. 70712–70723, 2022.

[32] D. Banerjee, P. A. Nair, J. N. Kaur, R. Usbeck, and C. Biemann, "Modern baselines for SPARQL semantic parsing," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2022, pp. 2260–2265.

[33] Y. Gu, X. Deng, and Y. Su, "Don't generate, discriminate: A proposal for grounding language models to real-world environments," in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 4928–4949.

[34] X. Ye, S. Yavuz, K. Hashimoto, Y. Zhou, and C. Xiong, "RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 6032–6043.

[35] J. Jiang, K. Zhou, Z. Dong, K. Ye, X. Zhao, and J.-R. Wen, "StructGPT: A general framework for large language model to reason over structured data," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2023, pp. 9237–9251.

[36] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. M. Ni, H. Y. Shum, and J. Guo, "Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph," in *Proc. 12th Int. Conf. Learn. Represent.*, 2024. [Online]. Available: https://openreview.net/forum?id=nnVO1PvbTv

[37] C. Mavromatis and G. Karypis, "GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs," in *Proc. Findings Assoc. Comput. Linguistics, ACL*, 2025, pp. 16682–16699.

[38] Y. Zhu, S. Qiao, Y. Ou, S. Deng, S. Lyu, Y. Shen, L. Liang, J. Gu, H. Chen, and N. Zhang, "Knowagent: Knowledge-augmented planning for llm-based agents," in *Proc. Findings Assoc. Comput. Linguistics*, 2025, pp. 3709–3732.

[39] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, "Rethinking the role of demonstrations: What makes in-context learning work?" in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2022, pp. 11048–11064.

[40] J. Liu, D. Shen, Y. Zhang, W. B. Dolan, L. Carin, and W. Chen, "What Makes Good In-Context Examples for GPT-3?" in *Proc. Deep Learn. Inside Out (DeeLIO) 3rd Workshop Knowl. Extraction Integr. Deep Learn. Archit.*, Dublin, Ireland, 2022, pp. 100–114.

[41] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, and E. Chi, "Least-to-most prompting enables complex reasoning in large language models," 2022, *arXiv:2205.10625*.

[42] S. S. Gu, Y. Iwasawa, T. Kojima, Y. Matsuo, and M. Reid, "Large language models are zero-shot reasoners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 22199–22213.

[43] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," 2022, *arXiv:2203.11171*.

[44] R. Hong, H. Zhang, H. Zhao, D. Yu, and C. Zhang, "Faithful question answering with monte-carlo planning," in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 3944–3965.

[45] H. He, H. Zhang, and D. Roth, "Rethinking with retrieval: Faithful large language model inference," 2023, *arXiv:2301.00303*.

[46] K. Wang, F. Duan, S. Wang, P. Li, Y. Xian, C. Yin, W. Rong, and Z. Xiong, "Knowledge-driven CoT: Exploring faithful reasoning in LLMs for knowledge-intensive question answering," 2023, *arXiv:2308.13259*.

[47] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R. K.-W. Lee, and E.-P. Lim, "Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models," in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 2609–2634.

[48] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *Proc. 11th Int. Conf. Learn. Represent.*, 2022. [Online]. Available: https://openreview.net/forum?id=WE_vluYUL-X

[49] J. Baek, A. F. Aji, and A. Saffari, "Knowledge-augmented language model prompting for zero-shot knowledge graph question answering," in *Proc. 1st Workshop Natural Lang. Reasoning Structured Explanations (NLRSE)*, 2023, pp. 78–106.

[50] T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal, "Decomposed prompting: A modular approach for solving complex tasks," in *Proc. 11th Int. Conf. Learn. Represent. (ICLR)*, Kigali, Rwanda, May 2023. [Online]. Available: https://openreview.net/forum?id=_nGgzQjzaRy

[51] T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal, "Decomposed prompting: A modular approach for solving complex tasks," in *Proc. 11th Int. Conf. Learn. Represent.*, 2023. [Online]. Available: https://openreview.net/forum?id=_nGgzQjzaRy

[52] P. Agarwal, N. Kumar, and S. Bedathur, "SymKGQA: Few-shot knowledge graph question answering via symbolic program generation and execution," in *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics*, 2024, pp. 10119–10140.

[53] G. Xiong, J. Bao, and W. Zhao, "Interactive-KBQA: multi-turn interactions for knowledge base question answering with large language models," in *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics*, 2024, pp. 10561–10582.

[54] M. Zhuge et al., "Mindstorms in natural language-based societies of mind," *Comput. Vis. Media*, vol. 11, no. 1, pp. 29–81, Feb. 2025.

[55] T. Cai, X. Wang, T. Ma, X. Chen, and D. Zhou, "Large language models as tool makers," in *Proc. 12th Int. Conf. Learn. Represent.*, 2024. [Online]. Available: https://openreview.net/forum?id=qV83K9d5WB

[56] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proc. 36th Annu. ACM Symp. User Interface Softw. Technol.*, Oct. 2023, pp. 1–22.

[57] S. Talaei, M. Pourreza, Y.-C. Chang, A. Mirhoseini, and A. Saberi, "CHESS: Contextual harnessing for efficient SQL synthesis," 2024, *arXiv:2405.16755*.

[58] X. Xie, G. Xu, L. Zhao, and R. Guo, "OpenSearch-SQL: Enhancing text-to-SQL with dynamic few-shot and consistency alignment," *Proc. ACM Manage. Data*, vol. 3, no. 3, pp. 1–24, Jun. 2025.

[59] S. Chaturvedi, A. Chadha, and L. Bindschaedler, "SQL-of-thought: Multi-agentic text-to-SQL with guided error correction," 2025, *arXiv:2509.00581*.

[60] Y. Liang, T. Xie, G. Peng, Z. Huang, Y. Lan, and W. Qian, "NAT-NL2GQL: A novel multi-agent framework for translating natural language to graph query language," 2024, *arXiv:2412.10434*.

[61] C. Zong, Y. Yan, W. Lu, J. Shao, Y. Huang, H. Chang, and Y. Zhuang, "Triad: A framework leveraging a multi-role LLM-based agent to solve knowledge base question answering," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2024, pp. 1698–1710.

[62] S. Gao, J. H. Lau, and J. Qi, "Beyond seen data: Improving KBQA generalization through schema-guided logical form generation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Suzhou, China, 2025, pp. 8764–8783.

[63] Z. Wen, Q. Si, Z. Lin, H. Liu, P. Fu, and W. Wang, "Schema item matters in knowledge base question answering," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2023, pp. 1–8.

[64] M. Bosma, E. Chi, B. Ichter, Q. V. Le, D. Schuurmans, X. Wang, J. Wei, F. Xia, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. Adv. Neural Inf. Process. Syst. 35*, 2022, pp. 24824–24837.
[65] Y. Tian, W. Yan, Q. Yang, X. Zhao, Q. Chen, W. Wang, Z. Luo, L. Ma, and D. Song, "Codehalu: Investigating code hallucinations in llms via execution-based verification," in *Proc. AAAI Conf. Artif. Intell.*, 2025, vol. 39, no. 24, pp. 25300–25308.
[66] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1400–1409.
[67] Y. Zhang, H. Dai, Z. Kozareva, A. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/3504035.3504780
[68] D. Agarwal, R. Das, S. Khosla, and R. Gangadharaiah, "Bring your own KG: Self-supervised program synthesis for zero-shot KGQA," in *Proc. Findings Assoc. Comput. Linguistics: NAACL*, 2024, pp. 896–919.
[69] N. Ngomo, "9th challenge on question answering over linked data (QALD-9)," *Language*, vol. 7, no. 1, pp. 58–64, 2018.
[70] A. Q. Jiang et al., "Mixtral of experts," 2024, *arXiv:2401.04088*.
[71] J. Achiam et al., "GPT-4 technical report," 2023, *arXiv:2303.08774*.
[72] G. He, Y. Lan, J. Jiang, W. X. Zhao, and J.-R. Wen, "Improving multi-hop knowledge base question answering by learning intermediate supervision signals," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, Mar. 2021, pp. 553–561.
[73] X. Hu, Y. Shu, X. Huang, and Y. Qu, "Edg-based question decomposition for complex question answering over knowledge bases," in *Proc. Int. Semantic Web Conf.*, 2021, pp. 128–145.

LONGQUAN JIANG received the B.Sc. degree in computer science from Yancheng Normal University, China, in 2013, and the M.Sc. degree in computer science from Shanghai Normal University, China, in 2018. He is currently pursuing the Ph.D. degree in computer science with the University of Hamburg, Germany. His research interests include question answering systems, deep learning, federated learning, reinforcement learning, and knowledge graphs.

DEBAYAN BANERJEE received the B.Sc. degree in information technology from the National Institute of Technology Durgapur, Durgapur, India, in 2009, and the M.Sc. degree in computer science from the University of Bonn, Germany, in 2020, and the Ph.D. degree in computer science from the University of Hamburg, Germany, in 2024. He is currently a Postdoctoral Researcher with the Leuphana University of Lüneburg, Germany. His research interests include LLMs and natural language processing (NLP), and more specifically, on the topic of knowledge graph question answering (KGQA).

RICARDO USBECK received the B.Sc. and M.Sc. degrees in computer science from the Martin Luther University Halle-Wittenberg, Germany, in 2010 and 2012, respectively, and the Ph.D. degree in computer science from Leipzig University, Germany, in 2017. Until 2019, he was a Postdoctoral Researcher with the Data Science Group, University of Paderborn. Then, he became a Team Leader with the Fraunhofer Institute for Intelligent Analysis and Information Systems. In 2021, he took up the junior professorship for semantic systems with the University of Hamburg. In 2023, he was a Professor of business informatics with the Leuphana University of Lüneburg, with a focus on artificial intelligence and explainability. He works in and leads several national and international research projects on knowledge graphs, large-scale language models, chatbots, and AI for social purposes. He is interested in transformation-related areas, such as sustainability and creative spaces for technology. He strives to build a bridge between research and application.