

Focus, Merge, Rank: Improved Question Answering Based on Semi-Structured Knowledge Bases

Derian Boer^{a,*}, Stephen Roth and Stefan Kramer^a

^aInstitute of Computer Science, Johannes Gutenberg University Mainz, Staudingerweg 9, 55131 Mainz, Germany
ORCID (Derian Boer): <https://orcid.org/0000-0002-2304-6613>, ORCID (Stefan Kramer):
<https://orcid.org/0000-0003-0136-2540>

Abstract. In many real-world settings, machine learning models and interactive systems have access to both structured knowledge, e.g., knowledge graphs or tables, and unstructured content, e.g., natural language documents. However, most rely on either. Semi-Structured Knowledge Bases (SKBs) bridge this gap by linking unstructured content to nodes within structured data, thereby enabling new strategies for knowledge access and use.

In this work, we present FocusedRetriever, a modular SKB-based framework for multi-hop question answering. It integrates components (VSS-based entity search, LLM-based generation of Cypher queries and pairwise re-ranking) in a way that enables it to outperform state-of-the-art methods across all three STaRK benchmark test sets, covering diverse domains and multiple performance metrics. The average first-hit rate exceeds that of the second-best method by 25.7%. FocusedRetriever leverages (1) the capacity of Large Language Models (LLMs) to extract relational facts and entity attributes from unstructured text, (2) node set joins to filter answer candidates based on these extracted triplets and constraints, (3) vector similarity search to retrieve and rank relevant unstructured content, and (4) the contextual capabilities of LLMs to finally rank the top-k answers.

For generality, we only incorporate base LLMs in FocusedRetriever in our evaluation. However, our analysis of intermediate results highlights several opportunities for further upgrades including finetuning. The source code is publicly available at <https://github.com/kramerlab/FocusedRetriever>.

1 Introduction

Large Language Models (LLMs) have rapidly evolved in recent years, demonstrating impressive performance on a range of complex tasks. As a result, they are increasingly integrated into everyday workflows, scientific research, and decision-making processes. However, despite their capabilities, LLMs remain prone to significant limitations. They can produce inaccurate or biased outputs, fail to recognize gaps in their own knowledge, and generate biased or fabricated (“hallucinated”) content that is difficult to detect [2, 5]. Their effectiveness in reasoning-based tasks is also the subject of ongoing scrutiny [9]. To mitigate these shortcomings, Retrieval-Augmented Generation (RAG) systems have been introduced. These systems enhance LLMs by retrieving relevant external information before generating responses, improving factual accuracy and robustness.

RAG systems, like most AI tools, typically leverage either structured data, such as knowledge graphs and relational tables, or unstructured content like natural language documents. The former provides precise, queryable information, while the latter offers broad, context-rich knowledge that is retrieved via dense embedding-based search. However, in many real-world domains, both types of information are available and valuable. This has led to the development of Semi-Structured Knowledge Bases (SKBs), which combine the advantages of both by linking unstructured documents to nodes in knowledge graphs. SKBs thus enable new methods of knowledge access and inference by bridging the gap between purely symbolic and contextual information.

Recent work by Wu *et al.* [26] exemplifies the potential of SKBs. They constructed SKBs across three real-world domains and assembled the STaRK benchmarks of corresponding multi-hop question answering (QA) datasets. These datasets reflect realistic scenarios where answering a question requires both relational reasoning over structured elements and contextual interpretation of unstructured texts.

In response, several specialized and hybrid approaches have been proposed to leveraging SKBs for complex QA tasks. However, many existing methods focus on specific techniques in isolation, overlooking opportunities to integrate complementary ideas. Our system, FocusedRetriever, outperforms state-of-the-art methods on all STaRK datasets by synthesizing multiple existing and novel strategies into a cohesive framework:

- It utilizes the inherent ability of common LLMs to interpret natural language questions and translate them into Cypher [7] queries, the standard language for querying property graphs.
- A variant of graph-based path search incorporates relational information into the retrieval process, filtering candidate answers based on their structural connections.
- To retrieve semantically relevant entities, ensuring robust retrieval even when query terms do not exactly match entity labels, we employ vector similarity search (VSS).
- FocusedRetriever is the first method to dynamically adjust the flexibility of the vector-based search of constants to always return at least k but not many more answers to ensure a consistent pool of candidates.
- By merging candidate sets from both symbolic and purely vector-based retrieval before applying the final ranking, FocusedRetriever mitigates potential errors introduced by individual retrieval steps.

* Corresponding Author. Email: boer@uni-mainz.de

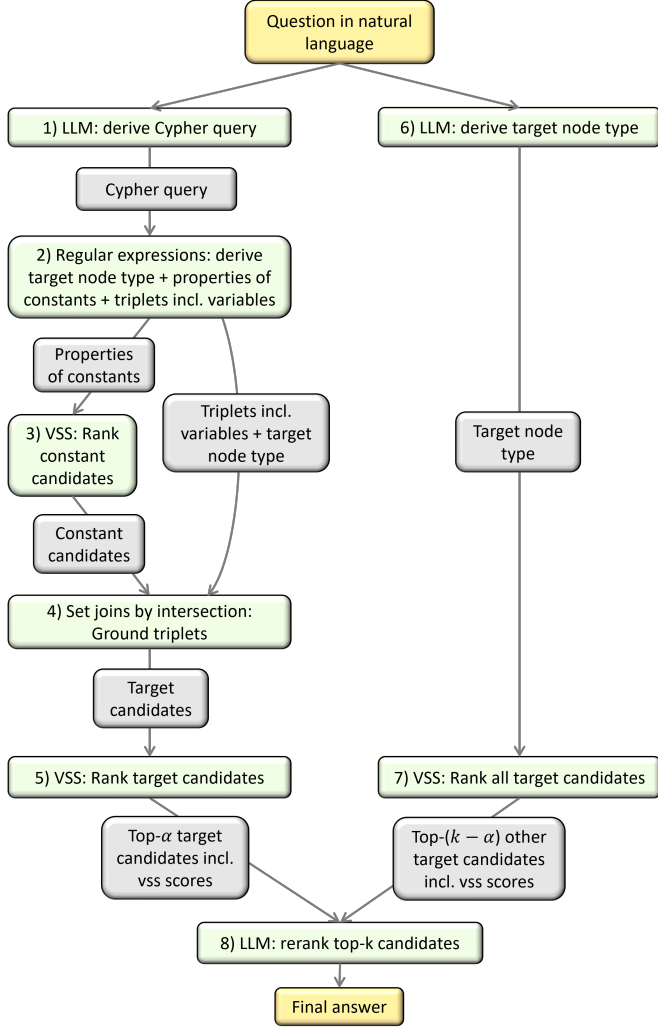


Figure 1. Overview of the processing steps of FocusedRetriever from the user question q to final answers. Gray boxes represent passed outputs of individual steps as input for subsequent steps.

- Finally, FocusedRetriever leverages LLMs to rerank the top- k candidate answers by scoring or relatively comparing their relevance and coherence with the question. In contrast to other work in this field, we make a comparison between three ranking strategies including an experimental evaluation.

The remainder of this paper is organized as follows: Section 2 surveys prior work related to question answering over Semi-Structured Knowledge Bases. Section 3 outlines the proposed methodology in detail. Section 4 reports on our experimental evaluation including additional analysis and compares our approach with current state-of-the-art systems. Finally, Section 5 concludes the paper and discusses directions for future research.

2 Related work

Recent surveys have outlined the synergy between Large Language Models (LLMs) and knowledge graphs (KGs), emphasizing both the opportunities and challenges that arise at their intersection [1, 18]. Pan *et al.* [17] further highlight the ongoing difficulties in integrating

structured reasoning with the generative capabilities of LLMs, pointing to key research directions for improving factual accuracy, context retention, and reasoning over hybrid knowledge sources.

Retrieval Across textual and relational knowledge Our work falls within the domain of retrieval-based, multi-hop question answering (QA) over semi-structured knowledge, where both structured (relational) and unstructured (textual) data must be effectively combined. Wu *et al.* [26] introduced the concept of a Semi-Structured Knowledge Base (SKB), linking knowledge graph (KG) nodes with associated free-text documents. Their STaRK benchmark suite evaluates methods designed to operate over such hybrid corpora and includes experiments with the following six baseline techniques. One of the simplest retrieval methods is *Vector Similarity Search* (VSS), which embeds a query and candidate entities, each represented by concatenated textual and relational information, into a shared space for cosine similarity computation. *Multi-Vector Similarity Search* extends this by separating and embedding text and structure separately, enabling finer-grained comparison. *Dense Passage Retriever* [10] trains query and passage encoders using QA pairs to optimize retrieval accuracy. *QA-GNN* [30] tackles the problem by constructing a joint graph consisting of QA elements and KG nodes. It performs relevance scoring using LLM-generated embeddings, followed by message-passing over the graph to facilitate joint reasoning. *VSS+[pointwise]Reranker* [4, 31] uses a language model to assign a relevance score to each of the top- k VSS results, refining the initial retrieval with semantic ranking grounded in a SKB context. Our method extends this paradigm by integrating symbolic filters, VSS for the retrieval of constants, and integration of different alternative LLM-based reranking strategies.

AVATAR [25] represents an agent-based approach, in which a comparator module generates contrastive prompts for optimizing LLM behavior during tool usage. Its performance study includes ReAct [29], which blends reasoning with action in an interleaved LLM prompt, and Reflexion [21], which employs episodic memory to refine agent decisions across tasks.

mFAR [15] addresses semi-structured retrieval by decomposing documents into multiple fields, each indexed independently. The system then adaptively weights fields based on the query. While we focus more on entity-relation-centered retrieval, mFAR contributes important insights into field-aware retrieval granularity.

HYBGRAG, a complementary line of work presented by Lee *et al.* [12], combines multiple retrievers with a critic module to refine answer generation across both structured and textual data.

4StepFocus [3] augments VSS+[pointwise]Reranker by two pre-processing steps: Triplet generation for extraction of relational data by an LLM and substitution of variables in those triplets to narrow down answer candidates. FocusedRetriever shares this aspect, but it follows a hybrid retrieval approach, features more LLM reranking strategies, and its candidate search is much more advanced.

PASemiQA [28] introduces a plan-based approach in which an LLM identifies which parts of the SKB to consult before executing a multi-stage traversal and extraction. *MoR* [13] frames retrieval in terms of three stages—planning, reasoning, and organizing—bridging graph traversal and text matching through path-based scoring. It achieved the highest average first hit rate and mean reciprocal rank so far.

Xia *et al.* [27] propose *KAR*, another query expansion method that integrates relational constraints from the KG into the retrieval pipeline.

Table 1 lists these key methods including the base LLMs they employed. We compare our results of FocusedRetriever against these

Table 1. Categorized benchmark methods developed for SKB-based multi-hop question answering and their LLMs employed. We compare the performance of all these approaches experimentally with FocusedRetriever in Section 4.

| Retrieval Category | Short Name | Method Name | Base LLM |
|-----------------------------------|---------------------|--|------------------------------|
| Textual | BM25 | Best Matching 25 [26] | - |
| | VSS (Ada-002) | Vector Similarity Search (Ada-002) [26] | - |
| | VSS (Multi-ada-002) | Multi-Vector Similarity Search (Multi-ada-002) [26] | - |
| | DPR | Dense Passage Retriever (DPR) [10] | RoBERTa |
| | VSS + Reranker | Vector Similarity Search + LMM Reranker [4, 31] | GPT4 (Claude 3 Opus) |
| Structural | QA-GNN | QA-Graph Neural Networks [30] | - |
| | ToG | Think-on-Graph [22] | - |
| Hybrid | AvaTaR | AvaTaR [25] | Claude 3 Sonnet |
| | 4StepFocus | 4StepFocus [3] | GPT4o |
| | KAR | Knowledge-Aware Retrieval [27] | GPT4o |
| | HybGRAG | Hybrid Retrieval-Augmented Generation on Textual and Relational Knowledge Bases [12] | Claude 3 Opus |
| | ReAct | ReAct [29] | Claude 3 Opus |
| | Reflexion | Reflexion [21] | Claude 3 Opus |
| | FocusedR. (ours) | FocusedRetriever (ours) | LlMa 3-3 (70b) |
| Hybrid with Finetuned Base LLM(s) | PASemiQA | Plan-Assisted Question Answering with Semi- structured data [28] | finetuned LLaMa2 (7B) + GPT4 |
| | mFAR | Multi-Field Adaptive Retrieval [14] | custom |
| | MoR | Mixture of Structural-and-Textual Retrieval [13] | fine-tuned Llama 3.2 (3B) |

systems in Section 4 to demonstrate the effectiveness of our integrated candidate generation and reranking approach.

Additional Perspectives and Supporting Work Beyond retrieval-focused QA systems, several recent efforts have tackled related challenges such as mitigating hallucinations or enhancing interpretability. Guan *et al.* [8] proposed a knowledge graph-based retrofitting method to reduce LLM hallucinations using automatically generated triplets. Wang *et al.* [24] use embedding-based techniques to enable QA across multiple documents. Vedula and Parthasarathy [23] present FACE-KEG, a KG-transformer model for fact verification with an emphasis on explainability. Kundu and Nguyen [11], as well as Shahi [19], explore approaches for identifying and structuring false claims within a KG framework for fact checking. Similarly, Shakeel *et al.* [20] take a pipeline approach combining KG embeddings and traditional NLP for factual verification. Knowledge Solver [6] builds whole subgraphs first, allowing the LLM to select entities in subsequent path-finding steps, starting from a randomly selected term in the question.

Mountantonakis and Tzitzikas [16] provide a semi-automatic web application that assists users in comparing facts generated by ChatGPT with those in KGs by using multiple RDF KGs for enriching ChatGPT responses. While it has some parallels to our work, the web interface requires manual interpretation and does not change the LLM’s response.

Querying with Cypher Property graph databases such as Neo4j and JanusGraph have gained widespread adoption in both academia and industry over the past decade. Cypher, a prominent query language for such systems, allows expressive querying of node and relationship patterns using constructs like MATCH, WHERE, and RETURN [7].

In our approach, we leverage base LLMs’ ability to generate Cypher queries from natural language without fine-tuning. Although conversion methods like text-to-Cypher and text-to-SPARQL have been studied, we show that even out-of-the-box LLMs can produce effective graph queries when paired with symbolic constraints. Nonetheless, further improvements could be achieved, e.g., by fine-tuning Cypher-specific tasks or applying reinforcement learning with downstream QA performance as feedback.

Contributions of this work Our work contributes to the evolving landscape of SKB-based question answering by introducing a synergistic triplet-, VSS- and LLM-based answer candidate generation strategy with subsequent reranking with the following distinctive key points:

- VSS-based entity search not only for answer candidates but also for other entities in a query,
- zero-shot retrieval by leveraging the capacity of base LLMs to generate Cypher queries from natural language without fine-tuning,
- a hybrid approach that not only integrates structured and unstructured knowledge but also two different retrieval strategies, and
- the introduction of listwise and pairwise reranking of the top k answers to this field as alternatives to pointwise reranking.

Our unified method improves retrieval accuracy, as presented in Section 4, while preserving the interpretability and traceability required in semi-structured QA for most answers.

3 FocusedRetriever

Figure 1 provides a high-level overview of the key components of our framework and the flow of intermediates at its interfaces. Before we explain the individual steps in more detail, we introduce the following notions:

- A Semi-Structured Knowledge Base $SKB = (V, E, D)$ [26] consists of a knowledge graph $G = (V, E)$ and associated text documents $D = \bigcup_{v \in V} D_v$, where V is a set of graph nodes and $E \subseteq V \times V$ a set of their connecting edges, which may be directed or undirected, and weighted or unweighted.
- Let $V.types = \bigcup_{v \in V} type(v)$ be the set of all existing node types in SKB, where $type(v)$ represents the node type of any node $v \in V$.
- Let $E.types = \bigcup_{e \in E} type(e)$ be the set of all existing node types in SKB, where $type(e)$ represents the edge type of any node $e \in E$.
- Let $V.attr$ be the union of node attributes names that any node $v \in V$ can have a searchable value assigned to.

Algorithm 1 FocusedRetriever**Input:**

a Semi-Structured Knowledge Base $SKB = (V, E, D)$, a query q in natural language, candidates for query answers $C \subseteq V$, number of candidates to rank and return k , an upper bound for the number of constant candidates to consider \hat{k}_{max}

Output: A list of k answers to q in descending order

```

1:  $q_{cypher} \leftarrow \text{DERIVE\_CYPHER}(q, V.types, E.types, V.attr)$ 
2:  $y, T, S_{raw} \leftarrow \text{REGEX}(q_{cypher}, V.types, E.types, V.attr)$ 
3:  $S \leftarrow \text{SYMBOL\_CANDIDATES}(S_{raw}, V, \hat{k}_{max})$ 
4:  $C_{cypher} \leftarrow \emptyset$ 
5:  $\hat{k} \leftarrow 1$ 
6: while  $|C_{cypher}| < k$  &  $\hat{k} < \hat{k}_{max}$  do
7:    $C_{cypher} \leftarrow \text{GROUND\_TRIPLETS}(T, S[\hat{k}], y, E)$ 
8:    $\hat{k} \leftarrow \hat{k}^{1.5} + 0.5$   $\triangleright$  increment  $k$  exponentially
9: end while
10:  $Y_{cypher} \leftarrow \text{VSS}(q, C_{cypher}, \emptyset, \alpha, SKB)$ 
11:  $y'.type \leftarrow \text{DERIVE\_TARGET\_TYPE}(q, V.types)$ 
12:  $Y_{vss} \leftarrow \text{VSS}(q, C(y'.type), C_{cypher}, k - |Y_{cypher}|, SKB)$ 
13:  $Y \leftarrow Y_{cypher} + Y_{vss}$   $\triangleright$  concatenate arrays
14:  $Y \leftarrow \text{RERANK}(Y, SKB)$  return  $Y$ 
```

Algorithm 2 GROUND_TRIPLETS**Input:**

a set of triplets T , a dictionary S that assigns sets of nodes (candidates) to variables representing symbols, a target variable y , a set of edges E

Output: A set of candidate nodes for y

```

1:  $S_{clone} \leftarrow S$ 
2: for all  $\tau = (h, e, t) \in T$  do
3:    $N \leftarrow \emptyset$ 
4:   for all  $h^{(j)} \in S(h)$  do
5:      $N \leftarrow N \cup \text{NEIGHBORS}(h^{(j)}, e, E, -)$ 
6:   end for
7:    $S(h) \leftarrow S(h) \cap N$ 
8:    $N \leftarrow \emptyset$ 
9:   for all  $t^{(j)} \in S(t)$  do
10:     $N \leftarrow N \cup \text{NEIGHBORS}(t^{(j)}, e, E, +)$ 
11:   end for
12:    $S(t) \leftarrow S(t) \cap N$ 
13: end for
14: if  $S_{clone} = S$  then
15:   return  $S(y)$ 
16: else
17:    $\text{GROUND\_TRIPLETS}(T, S, y, E)$ 
18: end if
```

- Let $y.type$ and $y'.type$ be a predicted node type $y.type \in V.types, y'.type \in V.types$ of the target variable y .
- Let $C \subseteq V$ be all candidate nodes for query answers and $C(y.type) \subseteq C$ all candidate nodes with node type $y.type$.
- $\text{NEIGHBORS}(v, e.type, E, +)$ returns the union of all nodes u for which an edge of type $e.type \in E.types$ exists that points from u to v .
- $\text{NEIGHBORS}(v, e.type, E, -)$ denotes the union of all nodes u for which an edge of type $e.type \in E.types$ exists that points from u to v .
- Let S be a dictionary, assigning a sorted array of nodes $S(x)$ to a symbol x (which may be a variable or a constant).
- $S[\hat{k}]$ returns a clone of S where each assigned array $S(x)$ is

Table 2. Theoretical comparison of alternative reranking approaches integrated in FocusedRetriever.

| Name | Core idea | Expected #Prompts | Expected #Tokens (relative) |
|--------------------|--|-------------------|-----------------------------|
| Pointwise Reranker | One prompt per item assigning a confidence score between 0 and 1 to each item | N | N |
| Listwise Reranker | One prompt including all entities requesting a sorted list of them | 1 | N |
| Pairwise Reranker | Binary insertion sort using one prompt for each pairwise comparison during sorting | $N * \log(N)$ | $2N * \log(N)$ |

replaced by a set of its first \hat{k} nodes.

Algorithm 1 contains all steps FocusedRetriever, calling multiple helper functions, which are explained in the following.

DERIVE_CYPHER (line 1) The first step, $\text{DERIVE_CYPHER}(q, V.types, E.types)$, uses a Large Language Model (LLM) to generate a Cypher query from the input question q . The prompt includes the available node and edge types (i.e., Cypher “labels”) and property names (e.g., ‘title’, ‘publication venue’). All prompt templates are provided in the Appendix.

REGEX (line 2) To allow full vector-based retrieval over the SKB without relying on Cypher execution, FocusedRetriever parses the generated Cypher string s into another structured format. From q_{cypher} , the target variable’s identity y and its node type $y.type \in V.types$, a set of relational triplets T , and node-level constraints S_{raw} are extracted. Each triplet $\tau \in T$ is a tuple (h, e, t) of two symbols, which can be either a constant with a given search string or a variable, and an edge type $e \in E.types$. Each condition in S_{raw} is a tuple of a symbol, a property name $p \in V.attr$, and a property value. REGEX filters out invalid components from s .

We implement two variants: a) strict: discards all triplets or attributes with nodes, edges and properties not in $V.types, E.types$, or $V.attr$; b) lenient: ignores type and attribute constraints, allowing broader matching.

SYMBOL_CANDIDATES (line 3) Subsequently, the top \hat{k}_{max} ranked candidates are retrieved for each symbol that includes at least one attribute constraint. Name or title properties are used to sort them semantically by vector cosine similarity to (see section 2), while other attributes like numeric filters (e.g., “year > 2012”) are applied as hard constraints. The following example demonstrates, why more than one candidate for a symbol are necessary in many cases: Suppose the task is to return a paper by ‘J. Smith’ published in the journal ‘Nature’. Ambiguous references, such as “J. Smith”, yield multiple plausible candidates, which is critical when only one of them satisfies all other query conditions (e.g., having published in Nature). \hat{k}_{max} is a configurable hyperparameter.

GROUND_TRIPLETS (lines 4-9) For each triplet (h, e, t) , the candidates for h are updated by their union with all incident nodes of t (lines 3-7), and vice versa (lines 8-12), taking node and edge types into account in the strict mode. This loop is repeated until S no more candidates are removed anymore in one loop (lines 1,16,17). Finally, Algorithm 2 returns the candidates for the target variable, which are potential answers to q .

Lines 4–9 aim to identify at least k candidates for the target variable y , while keeping the total number of candidates low to maintain precision. To achieve this, the number of candidates \hat{k} per symbol is gradually incremented until either k answers are found or the limit k_{max} is reached. To limit computation, we increment \hat{k} exponentially.

Grounding is done using Algorithm 2, which iteratively filters symbol candidates based on the relational triplets T and, in the strict mode, by node types, and edge types. Each triplet (h, e, t) is used to refine candidate sets by propagating constraints bidirectionally (lines 3–7 and 8–12 of Algorithm 2). This loop continues (line 17) until convergence (no further eliminations) and the final candidate set for y is returned (line 15).

VSS (lines 10-12) These candidates are ranked by their vector cosine similarity to q in line 10. The inputs of our VSS function are: the question q , the grounded candidates C_{cypher} as a filter, an node exclusion set, which is empty in this case, and a parameter $\alpha \in \{0, 1, 2, \dots, k\}$ specifying how many candidates to return.

To improve recall, $k - |Y_{cypher}| \geq k - \alpha$ additional candidates are retrieved independently of the Cypher-derived constraints, using a fallback vector-based search. Duplicates and semantically irrelevant types are excluded, the latter based on an LLM-predicted node type (line 11).

RERANKER (line 14) So far, after concatenating the candidates y_{Cypher} and y^{VSS} to y , the top- k answers have been collected. The final step is a reranking of them. It indeed cannot improve or worsen the number of hits among the top- k answers, but our evaluation shows that it substantially improves the position of hits within y . We implement and evaluate three ranking strategies, which all involve an LLM. Again, as the function `DERIVE_CYPHER`, the LLM could be finetuned for a specific dataset, but we demonstrate that a base model even ‘out-of-the-box’ can substantially improve the ranking in different domains. Table 2 compares the properties of our three reranking strategies, which are point-wise, a list-wise approach, and a pair-wise approach.

Finally, after combining Y_{cypher} and Y_{vss} from the last step into a unified set Y , the top- k answers are reranked. While reranking does not change the set of returned answers, it changes their ordering within the top- k .

We implement three LLM-based reranking strategies: point-wise, pair-wise, and list-wise reranking, as described in Table 2. As with `DERIVE_CYPHER`, the LLM can be fine-tuned for a specific domain, though we find regular base models improve ranking quality across multiple domains already.

4 Evaluation

Datasets We evaluate FocusedRetriever on the three retrieval-oriented STaRK benchmark datasets [26]: AMAZON, MAG, and PRIME, each paired with a structured knowledge base (SKB). Table 3 summarizes key statistics for these SKBs, including the number of entity types, relation types, nodes, edges, average node degree, and total token count.

Each STaRK benchmark provides a set of synthetic train/validation/test queries, along with a smaller set of human-generated questions. These benchmarks simulate realistic scenarios by combining relational and textual knowledge with natural-sounding and flexible question formats. In line with reporting of other LLM-based methods, we restrict evaluation to 10% of the

synthetic test questions. This selection of test questions is provided by the STaRK framework.

The SKBs vary substantially:

- PRIME (medical domain) contains the richest relational structure, with diverse node and edge types and a strong focus on relations.
- AMAZON (product recommendation) includes fewer node types but a large amount of unstructured textual data (e.g., reviews).
- MAG (academic search) strikes a balance, featuring one- and two-hop relational queries and textual attributes like paper abstracts.

Each SKB contains hundreds of thousands to millions of nodes and edges. In PRIME, all nodes are possible answers. In MAG and AMAZON, only specific node types (papers and products, respectively) qualify. For additional details about the SKBs and datasets, we refer readers to Wu *et al.* [26].

Experiment settings We use LLaMa 3.3 70B as the LLM backbone for FocusedRetriever and text-embedding-ada-002 for vector-based semantic search (VSS). We set the reranking output size $k = 20$, to allow a fair comparison with VSS+Reranker *et al.* [26], and the maximum number of candidates per symbol $k_{max} = 1,500,000$. Model selection (e.g., α and reranker type) is based on validation performance.

Baseline configurations for comparison methods are described in their respective publications and Table 1 of Section 2 with their eventual backbone LLMs.

We report standard retrieval metrics averaged over test queries:

- hit@ m (Hm): Whether at least one correct item appears in the top- m results.
- recall@ m (Rm): Proportion of relevant items among the top- m , or among all if fewer than m ground truths exist.
- Mean Reciprocal Rank (MRR): Reciprocal of the rank at which the first relevant answer appears, limited to k as the maximum.

Main test results Table 4 presents results for FocusedRetriever using $\alpha = 12$ and pairwise reranking. The method achieves state-of-the-art performance across the three synthetic STaRK benchmarks, outperforming all baselines in hit@1, hit@5, and MRR, and, on average, hit@20.

Due to the limited availability of published results, we report performance on the smaller human-generated question sets in the Appendix.

Component analysis and ablation study Since design choices of different steps can be seen as optimization steps, we evaluate them separately using only the validation sets to avoid biasing the final evaluation from Table 4.

Impact of node-type filtering: In contrast to MAG and AMAZON, PRIME includes answers of different node types as answers. Hence, only for PRIME filtering (line 11 of Algorithm 1) is relevant. Table 5 shows that filtering candidates by predicted answer node type improves all metrics for this dataset. For example, MRR improves from 0.384 to 0.430 compared to VSS (line 11) only.

Strict vs. lenient parsing modes: Table 7 evaluates the effect of strict vs. lenient parsing of Cypher-derived conditions (Algorithm lines 1–10). Surprisingly, lenient parsing yields better results overall, particularly on PRIME, where strict mode underperforms by 0.049 MRR. This may be due to confusion introduced by several similar entity types (e.g., ‘‘pathway’’ vs. ‘‘biological process’’). For MAG and AMAZON, results are nearly identical under both settings.

Effect of α on hybrid search: Figure 2 illustrates the effect of vary-

Table 3. Properties of the STaRK SKBs [26]

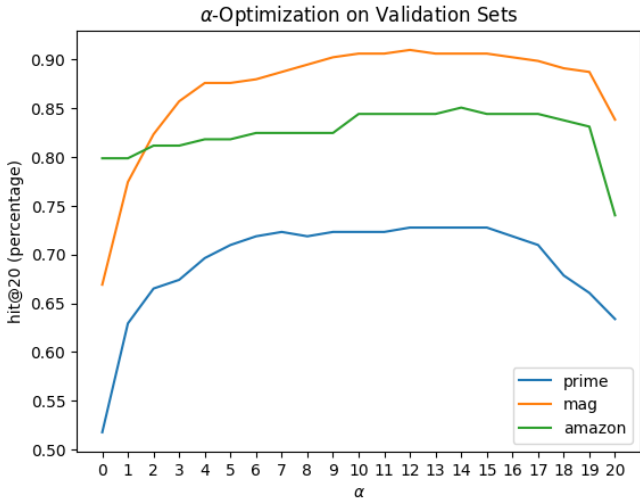
| Dataset | #entity types | #relation types | avg. degree | #entities | #relations | #tokens |
|---------|---------------|-----------------|-------------|-----------|------------|-------------|
| AMAZON | 4 | 4 | 18.2 | 1,035,542 | 9,443,802 | 592,067,882 |
| MAG | 4 | 4 | 43.5 | 1,872,968 | 39,802,116 | 212,602,571 |
| PRIME | 10 | 18 | 125.2 | 129,375 | 8,100,498 | 31,844,769 |

Table 4. Performance of benchmark methods and FocusedRetriever on the synthetically generated test sets

| Dataset Metric | AMAZON | | | | MAG | | | | PRIME | | | | Average | | | |
|------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | H1 | H5 | R20 | MRR | H1 | H5 | R20 | MRR | H1 | H5 | R20 | MRR | H1 | H5 | R20 | MRR |
| BM25 | 44.9 | 67.4 | 53.8 | 55.3 | 25.8 | 45.2 | 45.7 | 34.9 | 12.8 | 27.9 | 31.2 | 19.8 | 27.8 | 46.9 | 43.6 | 36.7 |
| VSS (Ada-002) | 39.2 | 62.7 | 53.3 | 50.4 | 29.1 | 49.6 | 48.4 | 38.6 | 12.6 | 31.5 | 36.0 | 21.4 | 27.0 | 47.9 | 45.9 | 36.8 |
| VSS (Multi-Ada) | 40.1 | 65.0 | 55.1 | 51.6 | 25.9 | 50.4 | 50.8 | 36.9 | 15.1 | 33.6 | 38.0 | 23.5 | 27.0 | 49.7 | 48.0 | 37.3 |
| DPR | 15.3 | 47.9 | 44.5 | 30.2 | 10.5 | 35.2 | 42.1 | 21.3 | 4.5 | 21.8 | 30.1 | 12.4 | 10.1 | 35.0 | 38.9 | 21.3 |
| VSS + Reranker | 44.8 | 71.2 | 55.4 | 55.7 | 40.9 | 58.2 | 48.6 | 49.0 | 18.3 | 37.3 | 34.1 | 26.6 | 34.7 | 55.6 | 46.0 | 43.8 |
| VSS + Reranker | 45.5 | 71.1 | 53.8 | 55.9 | 36.5 | 53.2 | 48.4 | 44.2 | 17.8 | 36.9 | 35.6 | 26.3 | 33.3 | 53.7 | 45.9 | 42.1 |
| GA-GNN | 26.6 | 50.0 | 52.0 | 37.8 | 12.9 | 39.0 | 47.0 | 29.1 | 8.8 | 21.4 | 29.6 | 14.7 | 16.1 | 36.8 | 42.9 | 27.2 |
| ToG | - | - | - | - | 13.2 | 16.2 | 11.3 | 14.2 | 6.1 | 15.7 | 13.1 | 10.2 | - | - | - | - |
| AvaTaR | 49.9 | 69.2 | 60.6 | 58.7 | 44.4 | 59.7 | 50.6 | 51.2 | 18.4 | 36.7 | 39.3 | 26.7 | 37.6 | 55.2 | 50.2 | 45.5 |
| 4StepFocus | 47.6 | 67.6 | 56.2 | 56.5 | 53.8 | 69.2 | 65.8 | 61.4 | 39.3 | 53.2 | 55.9 | 45.8 | 46.9 | 63.3 | 59.3 | 54.6 |
| KAR | 54.2 | 68.7 | 57.2 | 61.3 | 50.5 | 69.6 | 60.3 | 58.6 | 30.4 | 49.3 | 50.8 | 39.2 | 45.0 | 62.5 | 56.1 | 53.0 |
| HybGRAG | - | - | - | - | 65.4 | 75.3 | 65.7 | 69.8 | 28.6 | 41.4 | 43.6 | 34.5 | - | - | - | - |
| ReAct | 42.1 | 64.6 | 50.8 | 52.3 | 31.1 | 49.5 | 47.0 | 39.2 | 15.3 | 32.0 | 33.6 | 22.8 | 29.5 | 48.7 | 43.8 | 38.1 |
| Reflexion | 42.8 | 65.0 | 54.7 | 52.9 | 40.7 | 54.4 | 49.6 | 47.1 | 14.3 | 35.0 | 38.5 | 24.8 | 32.6 | 51.5 | 47.6 | 41.6 |
| PASemiQA | 45.9 | - | - | 55.7 | 43.2 | - | - | 50.2 | 29.7 | - | - | 31.0 | 39.6 | - | - | 45.6 |
| mFAR | 41.2 | 70.0 | 58.5 | 54.2 | 49.0 | 69.6 | 71.7 | 58.2 | 40.9 | 62.8 | 68.3 | 51.2 | 43.7 | 67.5 | 66.2 | 54.5 |
| MoR | 52.2 | 74.6 | 59.9 | 62.2 | 58.2 | 78.3 | 75.0 | 67.1 | 36.4 | 60.0 | 63.5 | 46.9 | 48.9 | - | - | 58.8 |
| FocusedR. (ours) | 64.0 | 76.2 | 56.0 | 69.3 | 74.1 | 84.2 | 78.8 | 78.8 | 46.4 | 63.9 | 65.5 | 53.7 | 61.5 | 74.8 | 66.7 | 67.3 |

Table 5. Impact of node-type filtering on STaRK PRIME validation set without final reranking and $\alpha = 0$

| Metric | H1 | H5 | H20 | R20 | MRR |
|----------------------------|--------------|--------------|--------------|--------------|--------------|
| With filtering | 0.129 | 0.415 | 0.518 | 0.248 | 0.430 |
| Without filtering 1 line X | 0.085 | 0.353 | 0.464 | 0.194 | 0.384 |

**Figure 2.** hit@20 rate of FocusedRetriever for different values of α

ing α , the balance between Cypher-based (lines 1–10) and vector-based retrieval (lines 11–12). When $\alpha = k$, retrieval relies fully on structured reasoning. $\alpha = 0$ uses only vector search and node type filtering. Across all datasets, hybrid retrieval (i.e., $\alpha \in [13, 15]$) achieves the highest hit@20, which is especially important since this set is passed to the final reranker.

Comparison of reranking strategies:

Table 6 compares three LLM-based reranking strategies: point-wise, pair-wise, and list-wise. All strategies substantially improve ranking quality, with the pair-wise approach consistently delivering the best results in terms of first hits and MRR.

Crucially, including relational context (up to two hops) in reranking prompts yields large gains. However, for the list-wise strategy, prompt lengths occasionally exceed the context window when including relationals, preventing evaluation in such cases. Note: For this reranker analysis, lines 1–10 of the algorithm were disabled to isolate reranking behavior.

5 Conclusion and Future Work

We introduced FocusedRetriever, a modular and traceable framework for multi-hop question answering over Structured Knowledge Bases (SKBs), which significantly outperforms existing Retrieval-Augmented Generation and other retrieval methods. Our approach leverages Large Language Models (LLMs) to interpret user queries, derive interpretable Cypher queries, and identify symbolic answer candidates from the SKB. A hybrid retrieval strategy, combining plain symbol processing and vector-based semantic search (VSS), enables robust handling of both relational and unstructured data. Finally, an LLM-based reranking step refines the top results using contextual background information.

Across diverse domains (medical, academic, and product recommendation), FocusedRetriever demonstrates strong empirical performance, achieving new state-of-the-art results on the STaRK benchmarks in hit@1, hit@5, recall@20 and MRR. Our ablation studies highlight the importance of lenient parsing, hybrid retrieval strategies, and the inclusion of relational context in reranking. Notably, the method operates in a zero-shot setting, without task-specific fine-tuning, making it adaptable across domains.

FocusedRetriever provides not only superior precision but also

Table 6. Performance of FocusedRetriever without final reranking (first row) and different reranking strategies excluding and including relational information in the prompts on the STaRK validation sets with $\alpha = 0$

| Relations included in prompts | Reranker | AMAZON | | | | MAG | | | | PRIME | | | |
|-------------------------------|------------------|--------------|--------------|-------|--------------|--------------|--------------|-------|--------------|--------------|--------------|-------|--------------|
| | | H1 | H5 | R20 | MRR | H1 | H5 | R20 | MRR | H1 | H5 | R20 | MRR |
| - | before reranking | 0.370 | 0.630 | 0.542 | 0.489 | 0.305 | 0.519 | 0.532 | 0.401 | 0.129 | 0.415 | 0.430 | 0.248 |
| no | pointwise | 0.519 | 0.727 | 0.542 | 0.615 | 0.353 | 0.553 | 0.532 | 0.445 | 0.237 | 0.446 | 0.430 | 0.326 |
| | listwise | 0.500 | 0.734 | 0.542 | 0.604 | 0.414 | 0.579 | 0.532 | 0.488 | 0.196 | 0.415 | 0.430 | 0.294 |
| | pairwise | 0.591 | 0.747 | 0.542 | 0.656 | 0.432 | 0.590 | 0.532 | 0.507 | 0.259 | 0.451 | 0.430 | 0.344 |
| yes | pointwise | 0.513 | 0.740 | 0.542 | 0.612 | 0.477 | 0.620 | 0.532 | 0.536 | 0.263 | 0.482 | 0.430 | 0.353 |
| | listwise | - | - | - | - | - | - | - | - | - | - | - | - |
| | pairwise | 0.604 | 0.753 | 0.542 | 0.670 | 0.500 | 0.620 | 0.532 | 0.549 | 0.295 | 0.482 | 0.430 | 0.375 |

Table 7. hit@20 of FocusedRetriever for strict, lenient, and intermediate modes of node type, edge type, and property name verification on the STaRK validation sets

| Mode | AMAZON | MAG | PRIME |
|--------------------|--------|--------------|--------------|
| Lenient mode | 0.844 | 0.898 | 0.679 |
| Ignore node labels | 0.844 | 0.898 | 0.714 |
| Ignore edge labels | 0.844 | 0.898 | 0.661 |
| Strict mode | 0.844 | 0.910 | 0.728 |

Table 8. Performance of benchmark methods and FocusedRetriever on the human-generated test sets in percent

| Dataset Metric | AMAZON | | | MAG | | | PRIME | | | Average | | |
|---------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | H1 | H5 | MRR | H1 | H5 | MRR | H1 | H5 | MRR | H1 | H5 | MRR |
| VSS (Ada-002) | 39.5 | 64.2 | 52.6 | 28.6 | 41.7 | 35.8 | 17.4 | 34.7 | 26.4 | 28.5 | 46.8 | 38.3 |
| VSS (Multi-ada-002) | 46.9 | 72.8 | 58.7 | 23.8 | 41.7 | 31.4 | 24.5 | 39.8 | 33.0 | 31.7 | 51.4 | 41.0 |
| DPR | 16.0 | 39.5 | 27.2 | 4.7 | 9.5 | 7.9 | 2.0 | 9.2 | 7.0 | 7.6 | 19.4 | 14.1 |
| VSS + Reranker | 54.3 | 72.8 | 62.7 | 36.9 | 45.2 | 40.3 | 28.4 | 48.6 | 36.2 | 39.9 | 55.5 | 46.4 |
| AvaTaR | 58.3 | 76.5 | 65.9 | 33.3 | 42.9 | 38.6 | 33.0 | 51.4 | 41.0 | 41.6 | 56.9 | 48.5 |
| 4StepFocus | 60.5 | 67.9 | 64.3 | 47.6 | 51.2 | 49.2 | 50.5 | 65.5 | 57.9 | 52.9 | 61.5 | 50.3 |
| KAR | 61.7 | 72.8 | 66.3 | 51.2 | 58.3 | 54.5 | 45.0 | 60.6 | 51.8 | 52.6 | 63.9 | 57.6 |
| ReAct | 45.6 | 71.7 | 58.8 | 27.3 | 40.0 | 33.9 | 21.7 | 33.3 | 28.2 | 31.6 | 48.4 | 40.3 |
| Reflexion | 49.4 | 64.2 | 59.0 | 28.6 | 39.3 | 36.5 | 16.5 | 33.0 | 24.0 | 31.5 | 45.5 | 39.8 |
| FocusedRetriever | 56.8 | 71.6 | 64.0 | 53.6 | 64.3 | 58.9 | 58.8 | 66.0 | 62.3 | 56.4 | 67.3 | 61.7 |

interpretability and modularity, two crucial features for deploying LLM-based QA systems in real-world scenarios. Future work may explore fine-tuning components such as Cypher generation and reranking for further gains and extending the framework to support more expressive logical reasoning. Overall, FocusedRetriever takes an important step toward reliable, transparent, and high-precision question answering grounded in semi-structured knowledge

Acknowledgements

This work was part of the cluster for atherothrombosis and individualized medicine (curATime), funded by the German Federal Ministry of Education and Research (03ZU1202NA).

Appendix

Appendix A: Results on human-generated test sets

Table 8 presents the experimental results on the human-generated STaRK benchmark sets. Note that some of these queries have more than 20 correct answers, recall@20 is omitted here. Although Focused retriever achieves the best results on average across all metrics, AvaTaR and KAR perform best on AMAZON.

Appendix B: Prompts

DERIVE_CYPHER Return a Cypher query derived from the given query Q. But follow my requirements precisely! Use a very basic, short Cypher syntax. Contentwise, omit everything that cannot be captured exactly with one of the given, available labels. Omit quantifications. Do not use OR or I. Format dates as YYYY-MM-DD. Only return one solution. Q: {query}

Available node labels: {nodes_to_consider}

Available properties: {properties_to_consider}

Available relation labels: {edges_to_consider}

Available keywords: [MATCH, WHERE, RETURN, AND, OR]

Example: MATCH (p:pathway)-[:interacts_with]->(g:gene/protein) WHERE g.name = 'IGF1' RETURN p.title

Point-wise reranker Examine if a {node_type} satisfies a given query and assign a score from 0.0 to 1.0. If the {node_type} does not satisfy the query, the score should be 0.0. If there exists explicit and strong evidence supporting that {node_type} satisfies the query, the score should be 1.0. If partial evidence or weak evidence exists, the score should be between 0.0 and 1.0. Here is the query: {query} Here is the information about the {node_type} Please score the {node_type} based on how well it satisfies the query. [...]

List-wise Reranker The rows of the following list consist of an ID number, a type and a corresponding descriptive text: {possible_answers} Sort this list in descending order according to how well the elements can be considered as answers to the following query: {query} Please make absolutely sure that the element which satisfies the query best is the first element in your order. Return ONLY the corresponding ID numbers separated by commas in the asked order.'

Pair-wise Reranker The following two elements consist of an ID number, a type and a corresponding descriptive text: {node1_id}, {node_type_1}, {doc_info_1}. {node2_id}, {node_type_2}, {doc_info_2}. Please find out which of the elements satisfy the following query better: {query} Return ONLY the corresponding ID number which corresponds to the element that satisfies the given query best. Nothing else.

References

- [1] G. Agrawal, T. Kumarage, Z. Alghamdi, and H. Liu. Can knowledge graphs reduce hallucinations in llms?: A survey. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3947–3960, 2024.
- [2] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proc. of FaccT 2021*, page 610–623, 2021. doi: 10.1145/3442188.3445922.
- [3] D. Boer, F. Koch, and S. Kramer. Harnessing the power of semi-structured knowledge and llms with triplet-based prefiltering for question answering. *arXiv preprint arXiv:2409.00861*, 2024.
- [4] Y. K. Chia, P. Hong, L. Bing, and S. Poria. InstructEval: Towards holistic evaluation of instruction-tuned large language models. In *Proc. of the First Workshop on the Scaling Behavior of Large Language Models (SCALE-LLM 2024)*, pages 35–64, 2024.
- [5] P. Feldman, J. R. Foulds, and S. Pan. Trapping llm hallucinations using tagged context prompts. *arXiv preprint arXiv:2306.06085*, 2023.
- [6] C. Feng, X. Zhang, and Z. Fei. Knowledge solver: Teaching llms to search for domain knowledge from knowledge graphs. *arXiv preprint arXiv:2309.03118*, 2023.
- [7] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 international conference on management of data*, pages 1433–1445, 2018.
- [8] X. Guan, Y. Liu, H. Lin, Y. Lu, B. He, X. Han, and L. Sun. Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting. In *Proc. of AAAI 2024*, pages 18126–18134, 2024.
- [9] S. Kambhampati. Can large language models reason and plan? *Annals of the New York Academy of Sciences*, 1534(1):15–18, 2024.
- [10] V. Karpukhin, B. Oguz, S. Min, P. S. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.
- [11] A. Kundu and U. T. Nguyen. Automated fact checking using a knowledge graph-based model. In *Prof. of ICAIIC 2024*, pages 709–716, 2024. doi: 10.1109/ICAIIIC60209.2024.10463196.
- [12] M.-C. Lee, Q. Zhu, C. Mavromatis, Z. Han, S. Adeshina, V. N. Ioannidis, H. Rangwala, and C. Faloutsos. Hybgrag: Hybrid retrieval-augmented generation on textual and relational knowledge bases. *arXiv preprint arXiv:2412.16311*, 2024.
- [13] Y. Lei, H. Han, R. A. Rossi, F. Derroncourt, N. Lipka, M. M. Halappanavar, J. Tang, and Y. Wang. Mixture of structural-and-textual retrieval over text-rich graph knowledge bases. *arXiv preprint arXiv:2502.20317*, 2025.
- [14] M. Li, T. Chen, B. Van Durme, and P. Xia. Multi-field adaptive retrieval. *arXiv preprint arXiv:2410.20056*, 2024.
- [15] Y. Li, R. Zhang, and J. Liu. An enhanced prompt-based llm reasoning scheme via knowledge graph-integrated collaboration. In *International Conference on Artificial Neural Networks*, pages 251–265, 2024.
- [16] M. Mountantonakis and Y. Tzitzikas. Using multiple RDF knowledge graphs for enriching ChatGPT responses. In *Prof. of the ECML PKDD 2023 Demo Track*, pages 324–329. Springer, 2023. doi: 10.1007/978-3-031-43430-3_24.
- [17] J. Z. Pan and et al. Large language models and knowledge graphs: Opportunities and challenges. *TGDK*, 1(1):2:1–2:38, 2023. doi: 10.4230/TGDK.1.1.2.
- [18] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Trans. Knowl. Data Eng.*, 36(7):3580–3599, 2024. doi: 10.1109/TKDE.2024.3352100.
- [19] G. K. Shahi. Fakekg: A knowledge graph of fake claims for improving automated fact-checking (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(13):16320–16321, Jul. 2024. doi: 10.1609/aaai.v37i13.27020.
- [20] D. Shakeel and N. Jain. Fake news detection and fact verification using knowledge graphs and machine learning. *ResearchGate preprint*, 10, 2021.
- [21] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.
- [22] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, H.-Y. Shum, and J. Guo. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph, 2023.
- [23] N. Vedula and S. Parthasarathy. FACE-KEG: fact checking explained using knowledge graphs. In *Prof. of ACM WSDM 2021*, pages 526–534. ACM, 2021. doi: 10.1145/3437963.3441828.

- [24] Y. Wang, N. Lipka, R. A. Rossi, A. Siu, R. Zhang, and T. Derr. Knowledge graph prompting for multi-document question answering. *Proc. of AAAI 2024*, (17):19206–19214, Mar. 2024. doi: 10.1609/aaai.v38i17.29889.
- [25] S. Wu, S. Zhao, Q. Huang, K. Huang, M. Yasunaga, K. Cao, V. Ioannidis, K. Subbian, J. Leskovec, and J. Y. Zou. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:25981–26010, 2024.
- [26] S. Wu, S. Zhao, M. Yasunaga, K. Huang, K. Cao, Q. Huang, V. N. Ioannidis, K. Subbian, J. Zou, and J. Leskovec. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. In *NeurIPS Datasets and Benchmarks Track*, 2024.
- [27] Y. Xia, J. Wu, S. Kim, T. Yu, R. A. Rossi, H. Wang, and J. McAuley. Knowledge-aware query expansion with large language models for textual and relational retrieval. *arXiv preprint arXiv:2410.13765*, 2024.
- [28] H. Yang, Q. Zhang, W. Jiang, and J. Li. Pasemiqua: Plan-assisted agent for question answering on semi-structured data with text and relational information. *arXiv preprint arXiv:2502.21087*, 2025.
- [29] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- [30] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proc. of the NAACL 2021*, pages 535–546, 2021.
- [31] H. Zhuang, Z. Qin, K. Hui, J. Wu, L. Yan, X. Wang, and M. Bendersky. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels. In *Proc. of the NAACL 2024*, pages 358–370, 2024.