SH-CoDE: Scholarly Hybrid Complex Question Decomposition and Execution

1st Tilahun Abedissa Taffa
Semantic Systems (SEMS)
University of Hamburg (UHH)
Hamburg, Germany
tilahun.taffa@uni-hamburg.de
ORCID: 0000-0002-2476-8335

2nd Ricardo Usbeck
Artificial Intelligence and Explainability (AIX)
Leuphana University Lüneburg (LEU)
Lüneburg, Germany
ricardo.usbeck@leuphana.de
ORCID: 0000-0002-0191-7211

***Abstract*—Our research addresses the challenge of answering complex scholarly hybrid questions, often demanding multi-faceted reasoning and iterative answer retrieval over scholarly knowledge graphs (KGs) and text. The question complexity is simplified by decomposing it into simple questions and utilizing symbolic representation. However, existing scholarly hybrid Question Answering (QA) models lack question decomposition and symbolic representation. In response, we propose SH-CoDE (Scholarly Hybrid Complex Question Decomposition and Execution). This approach breaks down questions into simple queries and employs symbolic representations, resulting in a natural and interpretable format - HQ (Hybrid Question) representation. SH-CoDE also includes an HQ-Executor, transforming the HQ representation into a tree structure and executing operations within its nodes. During execution, if the executor encounters symbolic representations such as KGQA or TextQA, it retrieves answers from KG and text data sources, respectively. The KGQA module automatically generates and runs SPARQL queries against the KG SPARQL endpoints. Similarly, the TextQA component employs semantic searching and an FLAN-T5-based reader to answer over text. Our model demonstrates competitive results on the test dataset, showcasing its effectiveness in answering complex scholarly questions.**

***Index Terms*—Question Answering, Question Decomposition, Scholarly Hybrid Question Answering, Scholarly Hybrid Question Representation**

I. Introduction

A natural language question - a common way to define our information needs in day-to-day activities - is classified as simple or complex based on the phases necessary to gather relevant facts from the underlying data sources and the operations required to answer a question [8], [25]. For example, an answer to a simple question like 'Who is the author of the Learning SPARQL book?' (Answer: Bob DuCharme) needs only the book author's information from a certain underlying data source. In contrast, complex questions like 'Among the publications of the Learning SPARQL book's author, which one has more citations?' need identifying '*the author of the book*', *the author's publications with their citation*, *comparing the citations and returning the one that has more citations*.

Beyond the reasoning steps and the complexity of the operations, complex scholarly hybrid questions[1] shown in Figure 1

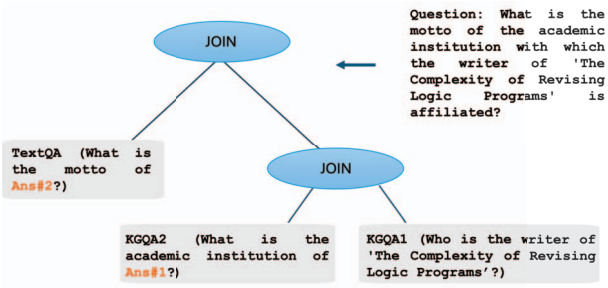

Fig. 1: An example of a scholarly hybrid question with its Hybrid Question (HQ) representation.

need searching facts over scholarly Knowledge Graphs (KGs) and text.

Generally, questions with such complexity indulge question decomposition - representing the complex question as a sequence (combination) of simple questions [10], [15]. The introduction of the decomposition component improves Question Answering (QA)[2] model's capability of answering complex questions [16], [17]. However, the QA systems assume that the answer search for the simple questions derived from the complex questions is on a single data source - KG or text, which does not fit the scholarly hybrid QA where the underlying data have both sources. On the other hand, generic hybrid QA models first unify the underlying data into KG or text and then utilize question decomposition [11]. Unlike that, Xu et al. [24], while decomposing the question, do not formulate a simple question but rather use relational mappings to both sources simultaneously. Apart from that, existing scholarly hybrid QA models [7], [9], [19] lack question decomposition and do not leverage symbolic representations[3].

Therefore, we propose SH-CoDE (**S**cholarly **H**ybrid **C**omplex Questi**o**n **D**ecomposition & **E**xecution) a symbolic representation-driven scholarly hybrid question decomposition and execution method that offers a natural and more inter-

[1]Questions requiring facts from a scholarly KG and text.

[2]QA is the task of answering questions over a specific data source.

[3]Symbolic representations refers to using symbols — such as logical expressions, structured knowledge, or formal rules—to represent the meaning of questions and potential answers.

2472-9671/25/$31.00 ©2025 IEEE
DOI 10.1109/ICSC64641.2025.00024

pretable representation. The decompose stage generates an intermediate representation - HQ-representation (see Figure 1) - by identifying embedded questions within the complex question and forming a series of sub-questions along with symbols like JOIN operations to replace the result obtained from the right part of the expression onto the placeholder in the left. Besides, symbols like KGQA and TextQA trigger an answer retrieval of the sub-question over KG and text data sources, respectively. The first challenge for complex scholarly question decomposition is correctly identifying the words' spans that make a valid sub-question. Subsequently, a valid sub-question formulation must paraphrase the identified span of the text. When the candidate's span of text does not contain the 'Wh' term, it is necessary to add an appropriate interrogation to have a valid, simple question—failure to identify sub-questions results in an incorrect answer to the complex question [23]. Hence, we leverage few-shot prompting to address the span identification and correct simple question formulation. We provide examples with their respective symbolic representation-based decomposition and prompt a Large Language Model (LLM).

On top of that, the scholarly complex questions have different forms, such as bridging, comparison, and bride-comparison questions. Bridging-type questions require a bridging entity that answers the embedded question(s) and is part of the succeeding sub-question. For instance, to answer the bridging type question given in Figure 1, the sub-questions should be answered recursively: (i) the author of the article, (ii) the author's institution, and (iii) the moto of the institution. In the meantime, the HQ-expression executor replaces the result from KGQA with the placeholder 'Ans#1' & 'Ans#2' in the left node before looking for the final answer due to the JOIN operator. Bridging questions are recursive and expressed as a sequence of sub-questions; except for the first innermost sub-question, all other sub-questions depend on their preceding sub-question. In contrast, comparison questions compare two or more entities based on their properties. For example, "Whose hIndex is higher, Tim Berners-Lee or Christopher Manning?", hIndex is used to compare the two scholars. In comparing questions, the answer is not the final comparative result; instead, it is the value of the parent of the node that satisfies the comparison condition. Hence, the model needs to backtrack and identify the respective node value. Bridge-comparison questions have both structures, i.e., identify the bridge entities with their requested properties, then compare and return the result.

In addition to the symbolic-driven question representation and execution method, we generate SPARQL automatically for the KGQA and develop our own TextQA model that fetches text from Wikipedia, chunks the text, retrieves semantically related chunks, and then uses a FLAN-T5-based [4] reader[4] to extract the final answer. Hence, automatic query generation and TextQA distinguish our scholarly hybrid QA method from the existing scholarly and generic hybrid QA models. Overall, we summarise our contributions as follows:

- New Method: we propose a new hybrid scholarly QA method that uses symbolic representation-based question decomposition & execution, KG SPARQL generation, and textual QA. The resulting HQ representations are parsable to binary tree structures, which makes them easily executable. Our method achieves a competitive result of 70.51 exact match scores and 71.83 F-Score on an existing Scholarly Hybrid QA dataset [20].
- Explainable: Breaking down complex questions into simple questions and generating an intermediary representation allows us to trace the information flow and decisions made at each stage. For example, by identifying operators like JOIN or COMPARE in decomposed questions, we can clarify which intermediate data pieces the model retrieved and how they contributed to the final answer. This hierarchical approach enables us to observe the final and intermediate results and the reasoning paths to understand why the model chooses specific answers. Additionally, this process allows the model to highlight potential sources of uncertainty or error at each stage, thus supporting a more explainable Scholarly QA.

The source code is available at https://github.com/semantic-systems/sh-code.

II. Related Work

A. Question Decomposition

The divide-and-conquer approach is a common method existing QAs follow to answer complex questions [3], [15]. The work in [22] introduced a QA model that answers bridging and comparison questions by decomposing the questions into single fact-seeking search queries suitable for surfing the web through a search engine. However, the search queries are inefficient in answering questions over KG and unstructured textual sources.

DECOMPRC [15] is a QA model that decomposes questions using Bidirectional Encoder Representations from Transformers (BERT) [5] based module via supervised training of human-annotated questions. Even though the training set is minimal, obtaining examples that enable the model to generalize from the given example is difficult. Unlike that, the ONUS (One-to-N Unsupervised Sequence transduction) [17] model proposed a decomposition component that learns from synthetic examples formulated from single-hop questions in an unsupervised way. However, it is not easy to control the quality of the auto-generated training sets, and training starts by assuming the existence of a pseudo-decomposition set.

Recent advancements in QA explore the interplay between neural models and symbolic reasoning to address complex questions with enhanced interpretability. Liu et al. [14] parses questions into hierarchical expressions by training the T5 [18] model. Moreover, it merges symbolic rules and neural readers for combined precision and explainability in textual QA. Additionally, recent techniques involving LLMs enhance reasoning

[4]A reader is a model that processes a context passage to extract the correct answer text span to a given question.

capabilities by building query trees and using a probabilistic tree of thought reasoning to leverage parametric and external knowledge [2]. Our proposed method, SH-CoDE, also emphasizes the decomposition of complex queries, similar to the strategies noted by Liu et al. [14]. SH-CoDE uniquely employs symbolic representations to create a natural and interpretable scholarly hybrid question representation. Unlike others, SH-CoDE specifically integrates a KGQA module for automatic SPARQL query generation and execution, along with a TextQA component using semantic search and a FLAN-T5 reader. These components adeptly retrieve answers from both KG and text data sources, setting our work apart in its ability to integrate multiple answer retrieval methods seamlessly. Moreover, our model uses few-shot prompting to decompose the questions and does not require annotated decomposition training data.

B. Few-shot Prompting

Recent developments in complex QA focus on enhancing interpretability by using symbolic question decomposition methods [14]. LLMs, providing limited examples to guide their responses, perform various tasks through few-shot prompting [1]. Few-shot prompting-driven decomposition gains traction because it can simplify complex queries into sequential sub-questions (answerable in a single reasoning step), allowing models to generalize with minimal supervision. Studies such as [6] and [12] have utilized few-shot methods for breaking down questions and identifying entities.

Our approach differs by leveraging few-shot prompting for question decomposition, incorporating symbolic representations into the query. The decomposition of questions focuses on creating and executing these questions through symbolic representations, providing a more comprehensive framework for handling complex queries. This dual focus on decomposition and execution with symbolic representation sets SH-CoDE apart from those that rely primarily on few-shot prompting for decomposition tasks.

C. Scholarly Question Answering

Among recent studies in scholarly QA, [21] describes a scholarly KGQA model that employs a few-shot learning strategy to generate SPARQL queries. This model selects few-shot examples from the training questions similar to the input query using a BERT-based encoder. Similarly, JarvisQA [13] addresses questions related to tabular data in scholarly literature, leveraging BERT to retrieve direct answers from various tabular formats. Unlike methods focusing solely on KGs [21] or unstructured text [13], SH-CoDE integrates both KGQA and TextQA into its framework for complex questions. In scholarly hybrid QA, [19] extracts relevant context from the underlying KGs and Wikipedia by utilizing entities mentioned in the question and prompts Llama3.1[5] to retrieve answers. The work in [7] presents a pipeline incorporating context retrieval and evidence matching to generate coherent responses to bibliographic questions, achieved through a fine-tuned model. Additionally, [9] combines SPARQL queries with divide-and-conquer algorithms and BERT predictions to improve the accuracy of their model. Besides, the scholarly hybrid QA [20] identifies scholarly entities expressed through sub-question phrases within the main question. Then, it retrieves relevant text from Wikipedia and entity-related triples from the underlying KGs via templated SPARQL queries. Finally, it utilizes a Retrieval-Augmented Generation (RAG) model that processes the retrieved text chunks alongside KG triple labels to produce an answer. In contrast, SH-CoDE distinguishes itself with interpretable, hierarchical symbolic representations and a tree-based execution framework, effectively querying KG and text-based QA and setting it apart from other scholarly QA approaches.

[5]https://ai.meta.com/blog/meta-llama-3-1/

III. Method

As shown in Figure 2, an input question comes into the SH-CoDE HQ-parser, transforming into an intermediate expression - Hybrid Question (HQ) representation. Then, an HQ executor forms a binary tree from the HQ representation and executes the operations in each node. While sending the sub-question to the KGQA module, it generates a SPARQL and runs the query on one of the SPARQL endpoints. If the TextQA module receives a sub-question, it hits Wikipedia, retrieves the entity text, chunks it, and passes the top three chunks highly similar to the respective sub-question. The reader part of the TextQA extracts a span of text as an answer and returns the one with the highest score among the top three candidate answers to the executor. Finally, upon the completion of all the operations' execution, an answer is returned.

A. Hybrid Question (HQ) Parsing

The parser employs the few-shot prompts for the HQ parsing task presented in III-A (Prompt 1) to generate an intermediate representation of scholarly questions, referred to as HQ representation, using ChatGPT-3.5. The prompts guide the transformation of complex scholarly hybrid questions into a structured HQ-representation format. This process involves decomposing questions into simpler, sequential sub-questions using operators such as JOIN and comparison operators like COMP_>, COMP_<, and COMP_=, which specify how results from one sub-question feed into another. For example, as shown in Table I, JOIN is employed for bridging questions, while JOIN with COMP is used for handling comparative and bridge-comparison questions. The examples included in the prompt demonstrate the practical usage of these operators. The prompts ensure that questions are precisely structured to facilitate accurate execution against KGs and TextQA systems via the HQ representation.

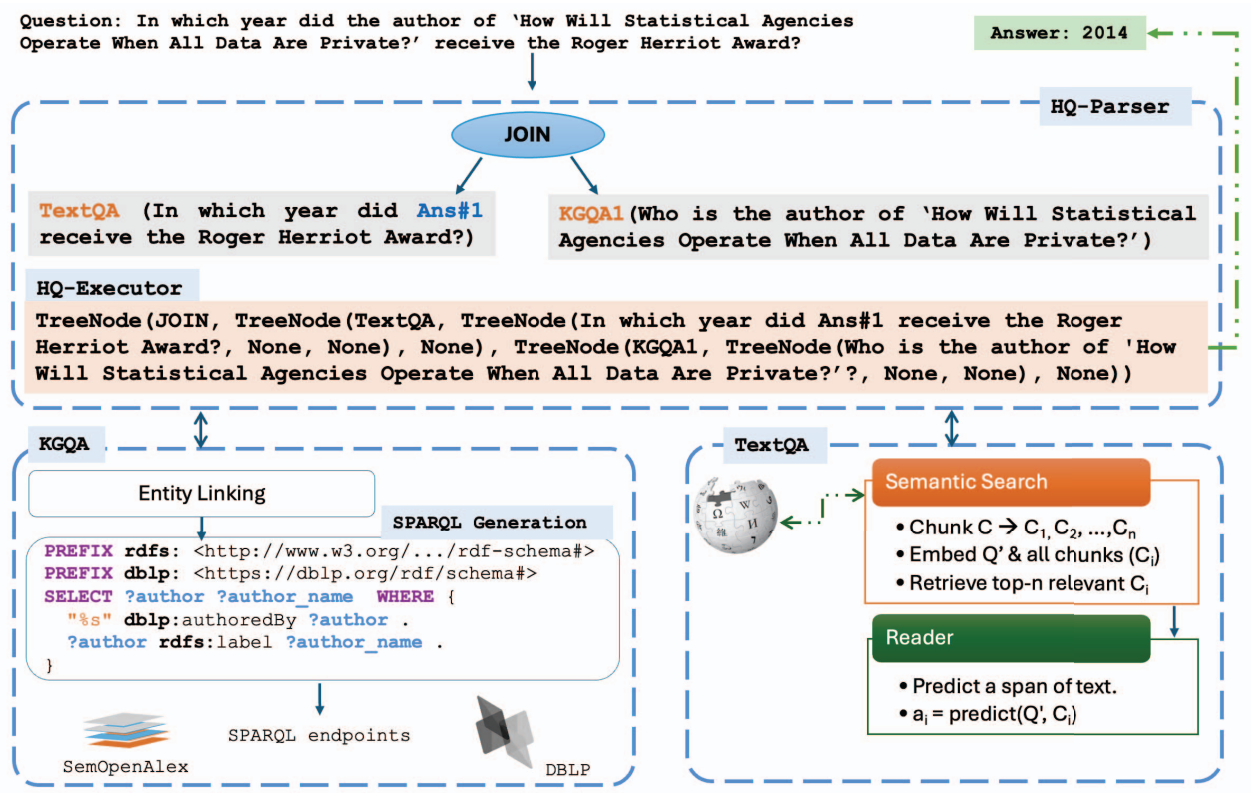

Fig. 2: The SH-CoDE model and its core components — HQ Parser, HQ Executor, KGQA, and TextQA—are illustrated through a practical example highlighting their interactions and functionalities.

Prompt 1: HQ-representation Generation Prompt

```
Convert question to HQ-representation:
the question is complex. The HQ
representation contains operators
and elements. Each element is a
natural question. The operator is
from [JOIN, AND, UNION, COMP_>,
COMP_<, COMP_=], and each is a binary
operator containing two elements.
Placeholder (e.g., Ans#1, Ans#2) is to
represent the answer to the predecessor
sub-question.
Generate an HQ representation. Please
follow the demo examples.
Example: {example}
Given question: {question}
HQ-representation:
```

B. HQ-Execution

The HQ-Executor converts the HQ representation into a binary tree through a structured approach. The HQ-representation is a symbolic expression made up of nested functions and operators that denote complex queries or operations, such as `JOIN`, `COMP_>`, `COMP_<`, and specific query types like `KGQA` and `TextQA`. Each HQ representation expresses complex queries or operations as nested function calls, where each function typically has one or two arguments.

The binary tree has the following characteristics: (i) Each node symbolizes a function/operator, (ii) The left and right

TABLE I: Example questions with their corresponding HQ Representation.

Example Question	HQ representation
Q: What is the citation of the author who collaborated with Piero Fraternali on 'The Story of the IDEA Methodology'?	JOIN(KGQA2(What is the citation of Ans#1?), KGQA1(Who is the author who collaborated with Piero Fraternali on 'The Story of the IDEA Methodology'?))
Q: Which institution, Joshua Smith or Duncan Watts affiliation, has fewer publications?	COMP_>(JOIN(KGQA2(What is the publication of Ans#2),KGQA2(What is Joshua Smith affiliation?)), JOIN(KGQA2(What is the publication of Ans#1),KGQA1(What is the affiliation of Duncan Watts?)))
Q: What is the main research focus of the author of 'Your System Is Secure? Prove It!'?	JOIN(TextQA(What is the main research focus of Ans#1?), KGQA1(Who is the author who collaborated with Piero Fraternali on 'The Story of the IDEA Methodology'?))
Q: What is the motto of the academic institution with which the writer of 'The Complexity of Revising Logic Programs' is affiliated?	JOIN(TextQA(What is the motto of Ans#2?), JOIN(KGQA2(What is the academic institution Ans#1 affiliated?), KGQA1(Who is The writer of 'The Complexity of Revising Logic Programs'?)))

children represent the arguments of this function/operator, and (iii) Leaf nodes represent specific query terms or values.

The parser recursively constructs TreeNode objects based on function names and operators in the expression. The step-by-step process to convert the HQ representation into a tree structure:

1. **Input Processing:** Begin by taking the HQ representation as input. Evaluate if it conforms to recognizable patterns such as `JOIN`, `AND`, `UNION`, comparison operators (e.g., `COMP_>`), or other query types (e.g., `TextQA`, `KGQA1`, `KGQA2`,).
2. **Extract functions with their argument:** Utilize regular expressions to identify function names and extract their corresponding arguments.
3. **Single Query Node Creation:** If the expression corresponds to a specific query function (e.g., `KGQA1`, `KGQA2`, `TextQA`), construct a node with the function's name as the primary value and the argument as the left child.
4. **Binary Operator Handling:** When the expression begins with a binary operator (e.g., `JOIN`, `AND`, `UNION`, or a comparison operator), separate the arguments. Handle nested structures by counting parentheses to ensure splitting occurs only at the top level, thereby enabling correct segmentation of complex expressions.
5. **Recursive Parsing:** Develop the left and right subtrees recursively by parsing each argument in turn.

Subsequently, the executor begins processing the parsed HQ representation with the rightmost leaf node. Each leaf node contains a sub-question and a symbolic representation that indicates the type of QA system, such as TextQA or KGQA. When the executor encounters a TextQA symbol, it sends the question to the TextQA module. For symbols like KGQA1 or KGQA2, the executor forwards the sub-question to the KGQA module. These symbols specify SPARQL types corresponding to DBLP and SemOpenAlex executables. The HQ executor recursively traverses the tree structure to execute an HQ query. It evaluates each node based on its type (e.g., `JOIN`, `COMP_>`) to retrieve relevant answers and entities. For specific nodes, such as `JOIN`, the executor applies a right-to-left execution order to propagate responses and entity references across nested expressions. Additionally, the executor incorporates entity tracking during comparisons to maintain correct associations throughout the evaluation process.

1. **Evaluate Comparison Nodes (`COMP_>`, `COMP_<`, `COMP_=`):**
   - For comparison nodes, retrieve and track values and entities from the left and right child nodes.
   - Compare these values according to the operator type (>, <, or =).
   - Return the entity corresponding to the greater, lesser, or equal value, as applicable.
2. **Evaluate Join Nodes (`JOIN`):**
   - For a `JOIN` node, first evaluate the right child to obtain the answer and associated entity URI (Uniform Resource Identifier).
   - Replace any placeholders (e.g., `Ans#`) in the left child's query with this answer.
   - Then, evaluate the left child with the updated query, using the entity from the right child as the current URI context.
3. **Evaluate Logical Operators (`AND`, `UNION`):**
   - For `AND`, recursively evaluate both children and return `"true"` only if both results are `"true"`.
   - For `UNION`, evaluate each child and merge their outcomes, ensuring no duplicates in the final result.
4. **Evaluate Query Nodes (`TextQA`, `KGQA1`, `KGQA2`):**
   - Execute each query type by invoking the corresponding function, which processes the question and provides an answer and entity (URI or name).
   - These outcomes rely on external systems, `TextQA` or KG functions like `KGQA1` and `KGQA1`.
5. **Return Leaf Node Values Directly:**
   - If the node is a leaf with no further operations required, return its value directly.

C. QA Modules

1) KGQA: The KGQA module has two sub-modules, namely KGQA1 for addressing questions related to DBLP and KGQA2 for answering questions answerable by SemOpenAlex.

Prompt 2: SPARQL Generation Prompt

```
Convert question -> SPARQL: The
SPARQL contains commands and
variables. URI is the identifier
of the entity under question.
Generate a SPARQL for the question.
Please follow the examples.
For example: {examples}
Given question: {question}
URI: {uri}
SPARQL:
```

KGQA1: The KGQA1 module focuses on answering questions related to authors by using their DBLP (Digital Bibliography & Library Project) URIs. First, it employs an entity linker to identify and connect the entity mentioned in the question. It uses ChatGPT-3.5 to determine the entity name within the question. Using the resolved entity URI, it formulates a SPARQL query by prompting ChatGPT-3.5 through Prompt 2 (III-C1. The module then runs the SPARQL against the DBLP-dump SPARQL endpoint[6] to retrieve details such as the author's name, ORCID[7] (Open Researcher and Contributor ID), and Wikipedia page URI ensuring the Wikipedia link is in English. The ORCID links an author across both DBLP and SemOpenAlex.

[6]https://dblp-april24.skynet.coypu.org/sparql
[7]https://orcid.org/

KGQA2: The KGQA2 module transforms a given question into a SPARQL query using a specific URI and executes that query against the SemOpenAlex KG endpoint[8]. If the query returns an answer, the module provides it as the response. While generating the SPARQL query, the module prompts ChatGPT-3.5 using the prompt in III-C1 (Prompt 2). The prompt includes the input question, the target URI, and examples that guide the LLM in selecting appropriate relation names for the query.

2) TextQA: The TextQA module extracts text from the corresponding Wikipedia page via the entity Wikipedia URI. Once the text is available, TextQA integrates a semantic search and reader models to process a long document and pinpoint a precise answer. This two-stage approach handles significant texts by narrowing down the relevant text chunk before extracting the answer. The model splits the text into chunks of 200 words and identifies the top three possible answers. Finally, the module returns the best answer and a list of top-3 answers.

Semantic Searching: The Semantic searching component employs a pre-trained sentence transformer model to compute embeddings for both the question and the text chunks. The searching process involves splitting the input text into chunks of 200 words, generating an embedding vector for the question and each chunk using all-MiniLM-L6-v2[9], and selecting the top three relevant chunks for further processing based on similarity scores from the sentence transformer, with the highest similarity scores indicating relevance.

Reader: The reader component leverages the FLAN-T5-based[10] model [4] to process each top-ranked chunk. The reader creates an input dictionary containing the question for each chunk, formatted with a classification token (*cls*), and the chunk as the context. Then generates an answer and its associated score using the reader pipeline[11]. Once all chunks are processed, determine the final answer by selecting the result with the highest score using the *max* function. The final answer undergoes a cleaning process to remove any leading or trailing whitespace and unwanted characters such as parentheses, periods, and trailing commas.

IV. Experiment

A. Dataset

For the evaluation of our proposed method, we use an existing dataset released for the Scholarly Hybrid QALD Challenge[12]. The questions in the dataset require reasoning over scholarly text and KG facts. Each question is aligned with DBLP and SemOpenAlex KG sub-graphs and Wikipedia text. The released dataset contains 10.5K question-answer pairs, which includes a test set of 700 human-verified questions [20].

B. Evaluation

To assess the overall performance of our proposed method in answering questions correctly, we use the exact match (EM) and F-Score metrics. EM calculates the ratio of answers that precisely match the gold standard answers. The F-Score evaluates how well the predicted answers balance precision and recall. It measures the model's accuracy by determining how well the predicted answers match the correct answers regarding overlapping tokens or words, typically analyzed at the word level for unstructured text.

The F-score is defined as follows:

$$\text{F-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1}$$

Where:

- Precision is the proportion of words in the predicted answer that also appear in the correct answer:

$$\text{Precision} = \frac{\text{Number of overlapping words}}{\text{Total words in the predicted answer}} \tag{2}$$

- Recall is the proportion of words in the correct answer that also appear in the predicted answer:

$$\text{Recall} = \frac{\text{Number of overlapping words}}{\text{Total words in the correct answer}} \tag{3}$$

The F-Score represents the harmonic mean of precision and recall, rewarding the model for accurately and comprehensively identifying correct answer tokens. A higher F-Score indicates that the predicted answers overlap effectively with the correct answers in both directions, achieving a balance between precision and recall.

TABLE II: Exact Match and F-Score of different models and our model.

Models	Exact Match (EM)	F-Score
Fondi and Jiomekong [9]	43.59	45.05
Contri(e)ve [19]	32.05	40.70
SH-CoDE (Ours model)	**70.51**	**71.83**

TABLE III: SH-CoDE performance of on different question types

Question Types	EM	F1
Bridge (KG-KG)	58.93	58.93
Comparison and Bridge-Comparison (KG-KG)	37.14	38.37
Bridge (KG-TEXT)	**79.74**	**81.55**

As shown in Table II, the model in [9] achieves an EM score of 43.59 and 45.05 F-Score. While Contri(e)ve [19] reports an EM of 32.05 and an F-Score of 40.70, lower than the other models listed. In contrast, our proposed model significantly outperforms the others, achieving an EM score of 70.51 and an F-Score of 71.83. This higher performance highlights the effectiveness of our approach, which integrates

[8]https://semoa.skynet.coypu.org/sparql

[9]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[10]FLAN-T5 is a fine-tuned version of T5 (Text-to-Text Transfer Transformer) that incorporates instruction tuning to improve performance on a wide range of text-to-text tasks.

[11]We instantiate the reader model pipeline using a Hugging Face transformer pipeline specifically configured for QA, setting the model and tokenizer to "sjrhuschlee/flan-t5-base-squad2".

[12]https://codalab.lisn.upsaclay.fr/competitions/19747

symbolic question parsing supported by SPARQL generation and semantic searching with the reader.

Additionally, as Table III illustrates our evaluation of the different question types, the SH-CoDE highest EM and F1 scores on the bridge (KG-Text) question types indicate the model ability in decomposing the questions and answering their respective sub-questions correctly. In contrast, on comparison and bridge comparison (KG-KG) question types, SH-CoDE performs the lowest due to the complexity of the questions. Besides, as the error analysis shows, the cumulative effect of the KGQA and HQ representation and execution errors impacts its performance. On the other hand, SH-CoDE shows a moderate performance in the bridge (KG-KG) type. Overall, SH-CoDE's higher performance on bridging-type questions highlights the model's efficient capability of handling questions that need a chain of evidence.

V. Error Analysis

Figure 3 presents our error analysis of the SH-CoDE model, focusing on 100 randomly selected incorrect results. The model's performance suffers from three primary errors: incorrect HQ representation generation & execution, erroneous KGQA responses, and TextQA-related mistakes. The most frequent issue, counting for 46% of errors, involves incorrect HQ representation & execution. This category includes three main problems. First, 36% of errors occur due to missing sub-question spans. For example, as shown in Table IV (row 1), the model should order the sub-questions as 'the writer of…', 'the academic institution of Ans#1…', and 'How many publications…'. Instead, the model combines the first two sub-questions, producing 'the academic institution of the writer of…'. Second, misplaced placeholders account for 6% of errors. In the HQ representation, placeholders must appear from right to left, with Ans#1 nested in the innermost sub-question. If a placeholder precedes its predecessor, it causes errors. Finally, 4% of errors result from incorrect operator usage in HQ representations.

The second error category involves erroneous responses from the KGQA component, constituting 23% of errors, which includes 11% due to inaccurate SPARQL query generation, such as introducing a non-existent relation name (for example, *bio:pdate*), shown in Table IV (row 2). Improper entity linking causes 5% of errors, where the model fails to identify entity names with their URI accurately. Additionally, 7% of errors arise from incorrect answers due to inaccurate entity URI usage.

The final category pertains to TextQA errors, which make up 31% of the total. In this category, the model struggles with extracting relevant text from Wikipedia, causing 5% of errors, and an incorrect semantic search results in 7% of errors when the retriever fails to extract relevant text chunks. The predominant error contributing 19% to this category involves inaccurate identification of the correct answer span. For instance, as shown in Table IV (row 3), the predicted answers are either overly detailed or insufficient. Our analysis highlights areas needing improvement, particularly in representing hybrid questions and optimizing interactions between KGQA and TextQA components.

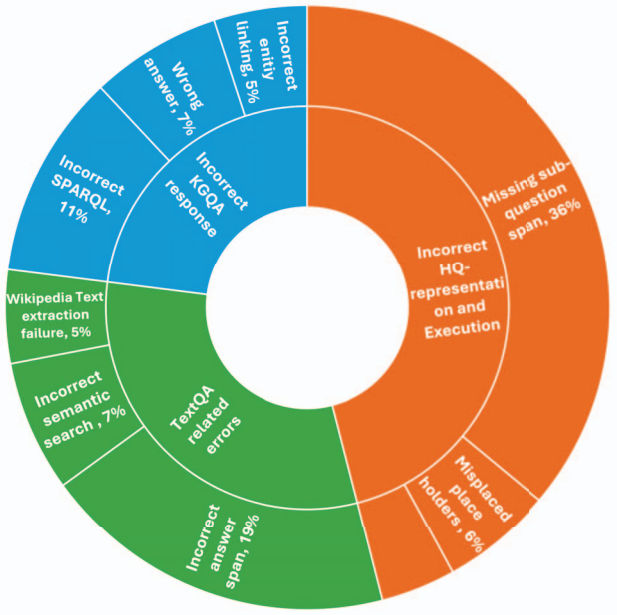

Fig. 3: An error analysis on the SH-CoDE model using randomly selected 100 questions.

TABLE IV: Sample questions with wrong HQ representation, KGQA, and TextQA results.

Question	Error Examples
How many publications are attributed to the academic institution of the writer of 'Advancing Reproducibility in Parallel and Distributed Systems Research'?	JOIN(KGQA2(How many publications are attributed to Ans#1),KGQA1(the academic institution of the creator of 'Advancing Reproducibility in Parallel and Distributed Systems Research'))
What is the i10Index of the scholar associated with Microsoft Research and involved in the publication of Information voyeurism: Social Impact of Physically Large Displays on Information Privacy in 2003?	SELECT ?answer WHERE { author_uri bio:pdate ?answer . author_uri org:memberOf ?uri .}
For what work did the author of Efficient haplotype matching and storage using the positional Burrows-Wheeler transform (PBWT) receive the Gabor Medal in 2017?	Gold Answer: computational biology Predicted Answer: contributions to computational biology

VI. Conclusion

In conclusion, this paper presents SH-CoDE, a new approach to scholarly hybrid QA that seamlessly integrates question decomposition with symbolic representation and execution. SH-CoDE effectively breaks down complex questions into simpler queries and utilizes a hierarchical symbolic representation (HQ representation) to establish a natural and interpretable framework for query processing. The HQ-Executor

enhances this framework by transforming the HQ representation into a tree structure, facilitating efficient operations across nodes. Additionally, our method incorporates automated SPARQL generation for KGQA and employs semantic search alongside a FLAN-T5-based reader for text-based queries, enabling precise answer retrieval from heterogeneous data sources.

SH-CoDE demonstrates superior performance compared to existing models and excels in handling bridging questions and indicating areas for improvement in handling comparison questions. Future work will focus on expanding SH-CoDE's capabilities to support a broader range of question types and improve its performance.

VII. Acknowledgment

We are grateful for the travel support the Department of Informatics, MIN-Faculty, University of Hamburg offers. Also, we would like to thank Dr.-Ing. Mathias Fischer, for the assistance.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[2] S. Cao, J. Zhang, J. Shi, X. Lv, Z. Yao, Q. Tian, L. Hou, and J. Li. Probabilistic Tree-of-thought Reasoning for Answering Knowledge-intensive Complex Questions. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12541–12560, 2023.

[3] W. Chen, H. Zha, Z. Chen, W. Xiong, H. Wang, and W. Y. Wang. HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data. In T. Cohn, Y. He, and Y. Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online, Nov. 2020. Association for Computational Linguistics.

[4] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling Instruction-Finetuned Language Models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[6] D. Dua, S. Gupta, S. Singh, and M. Gardner. Successive Prompting for Decomposing Complex Questions. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1251–1265, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[7] S. Efeoglu, N. Rauscher, E. Rubinov, Y. Xue, and S. Sonja. Large Language Models for Scholarly Question Answering Using Hybrid Data Sources, 2024.

[8] R. Etezadi and M. Shamsfard. The state of the art in open domain complex question answering: a survey. *Applied Intelligence*, 53(4):4124–4144, 2023.

[9] F. B. Fondi and A. J. Fidel. Integrating SPARQL and LLMs for Question Answering over Scholarly Data Sources. *arXiv preprint arXiv:2409.18969*, 2024.

[10] R. Fu, H. Wang, X. Zhang, J. Zhou, and Y. Yan. Decomposing Complex Questions Makes Multi-Hop QA Easier and More Interpretable. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 169–180, Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[11] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann, and A.-C. Ngonga Ngomo. Survey on challenges of question answering in the semantic web. *Semantic Web*, 8(6):895–920, 2017.

[12] Y. Hua, Y.-F. Li, R. Haffari, G. Qi, and T. Wu. Few-shot complex knowledge base question answering via meta reinforcement learning. In *Empirical Methods in Natural Language Processing 2020*, pages 5827–5837. Association for Computational Linguistics (ACL), 2020.

[13] M. Y. Jaradeh, M. Stocker, and S. Auer. Question Answering on Scholarly Knowledge Graphs. In *International Conference on Theory and Practice of Digital Libraries*, pages 19–32. Springer, 2020.

[14] Y. Liu, S. Yavuz, R. Meng, D. Radev, C. Xiong, S. Joty, and Y. Zhou. HPE: Answering Complex Questions over Text by Hybrid Question Parsing and Execution. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4437–4451, 2023.

[15] S. Min, V. Zhong, L. Zettlemoyer, and H. Hajishirzi. Multi-hop Reading Comprehension through Question Decomposition and Rescoring. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy, July 2019. Association for Computational Linguistics.

[16] P. Patel, S. Mishra, M. Parmar, and C. Baral. Is a question decomposition unit all we need? In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4553–4569, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[17] E. Perez, P. Lewis, W.-t. Yih, K. Cho, and D. Kiela. Unsupervised Question Decomposition for Question Answering. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8864–8880, Online, Nov. 2020. Association for Computational Linguistics.

[18] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[19] K. Shivashankar and N. Steinmetz. Contri(e)ve: Context + Retrieve for Scholarly Question Answering, 2024.

[20] T. A. Taffa, D. Banerjee, Y. Assabie, and R. Usbeck. Hybrid-SQuAD: Hybrid Scholarly Question Answering Dataset. *arXiv preprint arXiv:2412.02788*, 2024.

[21] T. A. Taffa and R. Usbeck. Leveraging LLMs in Scholarly Knowledge Graph Question Answering. In *The First Scholarly QALD Challenge at The 22nd International Semantic Web Conference (ISWC 2023).*, Athens, Greece, November 6 – 10 2023. CEUR-WS.org.

[22] A. Talmor and J. Berant. The Web as a Knowledge-Base for Answering Complex Questions. In M. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[23] Y. Tang, H. T. Ng, and A. Tung. Do Multi-Hop Question Answering Systems Know How to Answer the Single-Hop Sub-Questions? In P. Merlo, J. Tiedemann, and R. Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3244–3249, Online, Apr. 2021. Association for Computational Linguistics.

[24] K. Xu, Y. Feng, S. Huang, and D. Zhao. Hybrid Question Answering over Knowledge Base and Free Text. In Y. Matsumoto and R. Prasad, editors, *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2397–2407, Osaka, Japan, Dec. 2016. The COLING 2016 Organizing Committee.

[25] Q. Zhang, S. Chen, D. Xu, Q. Cao, X. Chen, T. Cohn, and M. Fang. A Survey for Efficient Open Domain Question Answering. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14447–14465, Toronto, Canada, July 2023. Association for Computational Linguistics.