Generating SPARQL from Natural Language Using Chain-of-Thoughts Prompting

Hamada M. ZAHERA♣ [1] , Manzoor ALI♣ , Mohamed Ahmed SHERIF ,
Diego MOUSSALLEM , and Axel-Cyrille Ngonga NGOMO
[a] *DICE group, Department of Computer Science, Paderborn University*
ORCiD ID: Hamada M. Zahera https://orcid.org/0000-0003-0215-1278, Manzoor Ali https://orcid.org/0000-0001-8403-5160, Mohamed Ahmed Sherif https://orcid.org/0000-0002-9927-2203, Diego Moussallem https://orcid.org/0000-0003-3757-2013, Axel-Cyrille Ngonga Ngomo https://orcid.org/0000-0001-7112-3516

Abstract.

Purpose: SPARQL is a highly expressive query language for knowledge graphs; yet, formulating precise SPARQL queries can be challenging for non-expert users. A potential solution is translating natural questions into SPARQL queries, known as SPARQL generation. This paper addresses the challenges of translating natural language questions into SPARQL queries for different knowledge graphs.

Methodology: We propose CoT-SPARQL, our approach to generate SPARQL queries from input questions. Our approach employs Chain-of-thoughts prompting that guides large language models through intermediate reasoning steps and facilitates generating precise SPARQL queries. Furthermore, our approach incorporates entities and relations from the input question, and one-shot example in the prompt to provide additional context during the query generation process.

Findings: We conducted several experiments on benchmark datasets and showed that our approach outperforms the state-of-the-art methods by a large margin. Our approach achieves a significant improvement in F1 score of 4.4% and 3.0% for the QALD-10 and QALD-9 datasets, respectively.

Value: Our CoT-SPARQL approach contributes to the semantic web community by simplifying access to knowledge graphs for non-expert users. In particular, CoT-SPARQL enables non-expert end-users to query knowledge graphs in natural languages, where CoT-SPARQL converts user natural languages queries into SPARQL queries, which can be executed via the knowledge graph's SPARQL endpoint.

Keywords., SPARQL Generation, Large Language Models, Chain-of-Thoughts.

1. Introduction

Knowledge graphs (KGs) are valuable sources of structured information that can be queried using SPARQL, a standard query language for the Semantic Web. However,

♣ Equal Contribution

[1] Corresponding Author: Hamada M. Zahera; E-mail: hamada.zahera@uni-paderborn.de

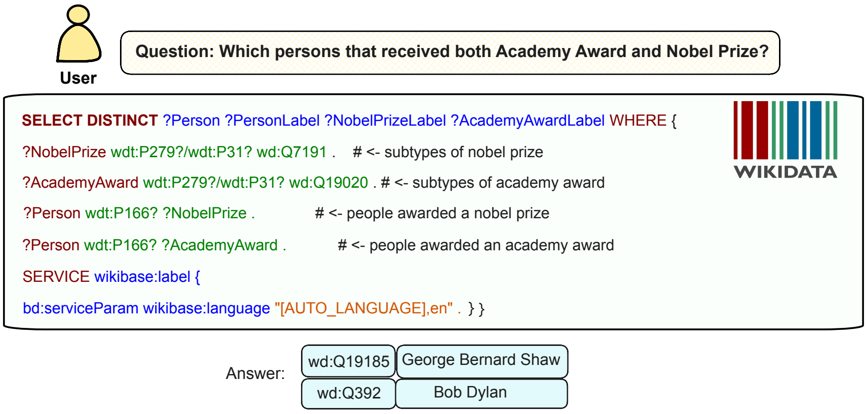

Figure 1. Example of SPARQL query used to answer an input question from Wikidata

SPARQL is an expressive language that requires users to have a deep knowledge of its syntax and semantics, as well as the specification of knowledge graph's schema [1–3]. This poses challenges for non-expert users to formulate and execute SPARQL queries, consequently limiting the accessibility and usability of knowledge graphs. To mitigate these challenges, SPARQL generation, has emerged as an active research area to bridge the gap between natural language and SPARQL [4].

SPARQL generation is the task of automatically converting natural language questions into SPARQL queries (e.g., see Figure 1), which can be executed over knowledge graphs. Current methods for SPARQL generation involve several challenges, such as mapping natural language terms to their corresponding entities and relations in knowledge graphs [5]. For example, template-based methods for SPARQL generation often require multiple steps and are tailored to specific knowledge graphs, which limit their applicability across different systems [6]. These challenges increase significantly when dealing with large and diverse knowledge graphs, such as Wikidata. Additionally, machine learning approaches for SPARQL generation aim to learn mapping and transformation rules from a large corpus of question-SPARQL pairs [7, 8]. However, these approaches require huge annotated data to train effective models. Recently, large language models (LLMs) have shown remarkable capabilities in generating database queries, such as SQL, from natural questions [9, 10]. However, generating SPARQL queries from a natural language question is more challenging, as it: 1) involves mapping natural language terms to entities and relations in knowledge graphs, and 2) requires constructing complex queries that match the semantics of questions.

To address these challenges, we leverage Chain-of-Thoughts Prompting (CoT), which has been shown to elicit the reasoning skills of LLMs for various tasks with few-shot examples [11]. We propose COT-SPARQL, our approach for SPARQL generation based on Chain-of-Thoughts reasoning. Our approach guides LLMs to think step-by-step and generate SPARQL queries similar to given few-shot examples. Unlike existing methods that rely on pre-defined templates or fixed rules, our approach can dynamically adapt to different knowledge graphs, and generate queries that capture natural language semantics. We conducted several experiments on benchmark datasets and evaluated the performance against several baselines. Our approach outperforms state-of-the-art methods in terms of accuracy, error-free SPARQL queries on several benchmark datasets. We summarize the main contributions of our paper as follows:

- We propose a new approach for SPARQL generation from natural questions using Chain-of-Thoughts prompting.
- Unlike existing methods, our approach adapts to different questions and KGs, and generates queries that capture the complex and diverse semantics of natural language.
- We show that our approach outperforms the state-of-the-art baselines on different datasets. Our implementation is open source and publicly available.[2]

2. Related Works

2.1. SPARQL Generation

Generating SPARQL queries is an essential task for accessing and analyzing Semantic Web data. Previous studies have primarily focused on two directions: *manual-* and *schema-based* SPARQL generation. In manual approaches, human experts create SPARQL queries to test ontology systems [12, 13] or to identify query features from existing datasets [14–16]. However, these approaches are not scalable to large and dynamic knowledge graphs such as *Wikidata* [17], which require a diverse set of queries to cover various aspects of the data. In contrast, schema-based approaches automatically generate SPARQL queries from pre-defined schemas or templates, which can overcome the limitations of manual approaches [18–20]. Such schemas define the structure and semantics of queries and use rules to insert data values from knowledge graphs into the queries. While these methods have shown promising results in generating complex and diverse queries [20–22], but they rely on a pre-defined set of templates, which limits the variety and scope of the queries. Moreover, creating new schemas for different question types involves manual effort, which reduces the scalability and automation of SPARQL generation process.

Another research direction has investigated the use of neural machine translation for SPARQL generation. For instance, Soru et al. [23] presented a sequence-to-sequence model that learns to generate SPARQL patterns from natural language questions. The authors used a semi-supervised approach with pre-defined templates to align questions and queries, and train their model on large-scale knowledge graphs. This approach can generate complex queries that involve multiple graph patterns, but also requires a lot of training data. Moreover, Zafar et al. [24] developed a method called SQG, which generates SPARQL queries from large-scale knowledge graphs. The proposed method has a modular design to integrate with other question answering components. Notably, this method can handle questions that are noisy or complex by finding a minimal sub-graph. However, this method encounters several challenges, such as handling out-of-vocabulary words, generalizing to unseen questions, and finding relevant query patterns.

On the other hand, Rony et al. [25] proposed the SGPT model that converts natural questions into SPARQL queries. SGPT is a comprehensive approach that does not depend on specific knowledge graphs or manual query templates. Specifically, SGPT leverages the GPT-2 language model and incorporates both linguistic and graph-specific features

[2] https://github.com/dice-group/CoT-Sparql

into its parameters. In contrast to the previous studies, our approach employs LLMs (e.g., LLaMA2-Code) to generate SPARQL queries using Chain-of-thoughts prompting without requiring predefined schemas or graph structures.

2.2. Chain-of-Thoughts Prompting

LLMs prompting has significantly improved the performance across various natural language processing tasks [26]. However, recent studies indicate that basic prompts (e.g., *"Generate a* SPARQL *code for the input question"*) may not always lead to precise results [11]. Recently, researchers have adopted Chain-of-Thoughts prompt as a means to enhance the capabilities of large language models in reasoning and generating tasks [27]. COT reflects the step-by-step learning process of humans, methodically moving through stages towards a solution, and leveraging context and supplementary information as necessary to achieve its objectives. Since our study focuses on code generation, we review related studies that apply Chain-of-Thoughts for this purpose. For example, Li et al. [28] leveraged COT approach in combination with *zero-shot* and *In-context* learning to extract specialized coding abilities from large language models. Furthermore, Jiang et al. [29] investigated the application of LLMs for code generation through a COT-based approach, including planning and implementation steps. Their structured approach demonstrates clear advantages over traditional direct generation methods using language models. Additionally, Pourreza and Rafiei [30] developed a COT-based approach for *text-to-SQL* generation, achieving a notable improvement of 10% in performance.

For SPARQL generation, Yang et al. [31] proposed an LLM-based approach to generate SPARQL queries for chinese knowledge graphs. Their method involves prompting an LLM with an input question, including entity mentions and their URIs, followed by the phrase *"the SPARQL statement corresponding to the graph is"*. However, this approach has limitations such as prompting the LLM without additional context, such as few-shot examples (question and SPARQL pairs), may not be efficient for generating complex SPARQL queries. Furthermore, the authors employed a generic pre-trained LLM, ChatGLM-6B, in contrast, we used a specialized model, LLaMa-Code, which is potentially better suited for tasks involving code and logical form generation. Similarly, Kovriguina et al. [32] introduced the SPARQGen approach, a one-shot prompt method for instructing the GPT-3 model to generate SPARQL queries. Their approach involves a basic LLM prompt, which contains a single example of a question, and its corresponding SPARQL query, instructions to explain the task of SPARQL generation for LLM and a test question. This method only considers a fixed set of questions/SPARQL pairs known as guiding examples. During the experiments, the author randomly selected a guiding example to provide a context for the LLM prompt. However, the randomly-selected example may not be relevant to the input question.

In contrast, our approach consider few-shot examples based on semantic similarity. In particular, we cluster the training set into groups of $\langle$question, SPARQL$\rangle$ pairs, then select the most semantically similar example to the input question from the appropriate cluster. Furthermore, SPARQGen approach employs a basic LLM prompt (`‘‘Given the following user question and RDF graph ... generate the corresponding SPARQL query...’’`), our Chain-of-Thoughts prompt incorporates the instruction `‘‘Let's think step by step’’` which triggers the reasoning capabilities of LLM during token generation, resulting in more precise results [33].

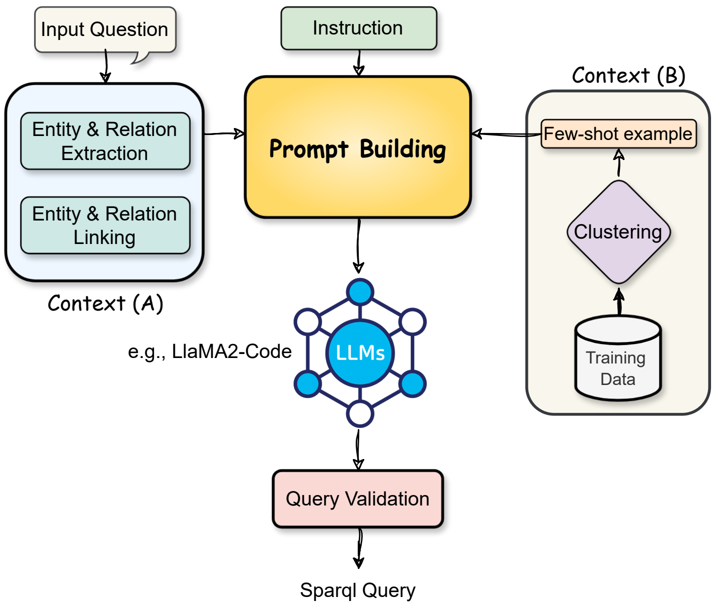

Figure 2. Overview of our approach (CoT-SPARQL) with LlaMA-Code model. Context (A) includes the entities and relations extracted from the input question and Context (B) include one-shot example.

3. Approach

In this section, we present our approach (CoT-SPARQL) for generating SPARQL queries from natural questions, including the components: *prompt building, in-context learning* and *query validation*, as shown in Figure 2. CoT-SPARQL starts by introducing the phrase ``Let's think step by step'' into the prompt, to enforce structured reasoning capabilities of the LLM is initiated. In our study, we consider LLaMA2-code model due to it's strong performance in code generation, positioning it as one of the best open-source models for this task [34]. CoT-SPARQL then define the task in the *prompt building* step (see Section 3.1 for more details), where it providing additional context from the input question, and including a few-shot example. This additional *in-context learning* helps the LLM better understand the question and generate accurate SPARQL queries (See Section 3.2 for more details). Finally, we verify the correctness of the SPARQL queries prior to their execution (See Section 3.3 for more details).

3.1. ***Prompt building.***

In this component, we use the CoT prompting [35] to enhance the reasoning capabilities of LLMs for Text-to-SPARQL generation. Specifically, we incorporate the phrase ``Let's think step by step'' into the LLM prompt to initiate a structured reasoning process and sequentially convert the given question into a SPARQL query.

Figure 3 shows an example run of the prompt building process for the user input query ``Was Gerald Gibbs the cinematographer of X the unknown?''. In particular our prompt building process includes three parts: i) `[INST]` $\cdots$ `[\INST]`, where we define the the *task* and its *description* to the LLM to be SPARQL generation for a target knowledge graph (e.g., DBpedia in Figure 3), ii) `Context (A)` provides the LLM with additional information such as entities and relations extracted from the input question, and iii) `Context (B)` presents a few-shot example for generating SPARQL query with the same syntax.

3.2. ***In-context learning.***

In this component, we add two contexts (`Context (A)` and `Context (B)` as shown in Figure 2) in our CoT prompt to provide the LLM with additional information for generating precise SPARQL queries.

3.2.1. *Context (A)*

We enrich the LLM prompt with entities and relations information from the input question. This information helps the LLM to disambiguate entities and understand the intended meaning correctly, thus reducing the chance of getting irrelevant or incorrect results (i.e, hallucination). For this purpose, we preprocess the input question using two libraries: `spaCy fishing`[3] (for entity linking in Wikidata) and `Falcon`[4] (for entity linking in DBpedia) and `REBEL`[5] (for relations extraction). The libraries also handle the issues of prefixes and IRIs in the SPARQL query generation.

3.2.2. *Context (B)*

We include a one-shot example in our prompt to show the LLM how to convert text to a SPARQL query. This one-shot example contains a (question, SPARQL) pair relevant to the input question. To achieve this, we embed all questions in the training set (i.e., Examples) as semantic vectors using `sentence-transformer`[6] library. The `sentence-transformer` captures semantic similarities between sentences, enabling effective clustering of textual data. We apply *K-means* clustering to group similar questions together into clusters. We calculate the cosine similarity to identify the most similar example as one-shot for the input question. We further refine our process by adopting the *K-means++* initialization method[7], which optimizes the selection of initial cluster centers, thereby improving convergence and reducing the likelihood of poor clustering outcomes. Finally, we append the relevant information, such as entities and relations in the SPARQL query, for the selected question as shown in example of Figure 3 in `Context B`.

3.3. ***Query validation.***

In the final step, we use the `SPARQLWrapper` library[8] to ensure that the generated SPARQL queries are syntactically correct, avoiding the execution of invalid ones. This library allows for the execution of SPARQL queries on remote endpoints connected to respective

[3] https://github.com/Lucaterre/spacyfishing
[4] https://labs.tib.eu/falcon/
[5] https://github.com/Babelscape/rebel
[6] https://github.com/UKPLab/sentence-transformers
[7] We use the Silhouette Score to determine the optimal number of clusters
[8] https://sparqlwrapper.readthedocs.io/en/latest/main.html

Instruction

[INST]
Task: Convert question to SPARQL query for {DBpedia} knowledge graph.
Description: given an input question and a list of DBpedia URIs for the mentioned entities and relations in the question. Write a correct SPARQL code to query the {DBPedia} knowledge graph. Please wrap your SPARQL code answer using ``` :
You can formulate your SPARQL query as the following examples.
[/INST]

Question

Was Gerald Gibbs the cinematographer of X the unknown?

Context (A)

Entities:[Gerald Gibbs:dbr:Gerald_Gibbs_(cinematographer), X the unknown: dbr:X_the_Unknown]

Relations: [cinematographer: dbo:cinematography]

Context (B)

Was X the Unknown edited by James needs?
ASK WHERE {
<http://dbpedia.org/resource/X_the_Unknown> <http://dbpedia.org/ontology/editing>
<http://dbpedia.org/resource/James_Needs> }
[James: dbr:LeBron_James, X the Unknown: dbr:X_the_Unknown]
[editing: dbo:editing]

CoT Prompt

Let's think step by step

Figure 3. CoT prompt of generating SPARQL queries of DBpedia. Entities are in orange and relations in blue.

knowledge graphs. For example, we use the public endpoints of Wikidata[9] and DBpedia[10] knowledge graphs. Our evaluation study (detailed in Table 3) shows that prompting LLMs without In-context and CoT instruction often results in generating invalid queries with multiple syntactic errors. We aim to build an end-to-end system that processes natural language questions, generates SPARQL queries, and validates their correctness.

4. Experiments

We conducted our experiments to answer the following research questions:

RQ$_1$. Does In-context learning enhance the performance of LLMs in generating sparql queries?
RQ$_2$. How accurate and precise are the the SPARQL queries generated by our approach?
RQ$_3$. How does the performance of our approach compare to state-of-the-art approaches of the question answering task?

[9] https://query.wikidata.org/
[10] https://dbpedia.org/sparql

Table 1. Summary of datasets used in our experiments.

Dataset	KG	Test	Valid	Train	Language
QALD-9	DBpedia	150	58	350	Multilingual
VQuAnDa	DBpedia	1000	500	3500	English
LC-QuAD 2.0	Wikidata	5969	2389	21497	English
QALD-10 2.0	Wikidata	394	-	412	Multilingual

4.1. ***Datasets.***

We used four benchmark datasets in our evaluation, namely: *LC-QuAD 2.0*, *VQuAnDa*, *QALD-9*, and *QALD-10*. Table 1 shows an overview of the datasets, including number of questions in *train, valid* and *test* splits and the language of questions. *LC-QuAD 2.0* [36] contains 30*k* question-query pairs over Wikidata and DBpedia, with 10 categories that vary in complexity and structure, where each question is annotated with its answer type and entities. *VQuAnDa* [37] has 5*k* question-query pairs over DBpedia with their verbalised answers, covering different question types, such as Boolean, list, and resource. *QALD-9* [38] has 558 total question-query pairs over DBpedia and Wikidata, with temporal, spatial, comparative, superlative, and other reasoning. This dataset supports multilingual question answering over knowledge graphs in 10 languages. *QALD-10* [39] has 412 training set question-query pairs over Wikidata, with temporal, spatial, comparative, superlative, and other reasoning. This dataset supports multilingual question answering over knowledge graphs in 9 languages. We used this dataset for our pilot study (see Section 5.3) to evaluate the correctness of generated SPARQL queries in questions answering.

4.2. ***Baselines.***

We compared our approach against different baselines, including state-of-the-art approaches and LLMs with standard prompt (*"generate a sparql query for the input question"*) as follows:

State-of-the-art approaches:

- SQG [24], which extracts sub-graph patterns from the question and ranks candidate queries by their structural similarity with the question, using a Tree-LSTM model.
- NSpM [23], which trains a Bi-LSTM model with a sequence-to-sequence technique to map natural language questions to template SPARQL queries.
- TeBaQA [40], which predicts the SPARQL query structure from template classes derived from the training dataset, and combines them with a sequence-to-sequence model to generate the SPARQL query.
- SGPT [25] uses a stack of Transformer-encoders to encode linguistic features of natural language questions and a fine-tuned GPT-2 model to decode and generate SPARQL queries.
- SPARQLGen [32], this is the state-of-the-art baseline that prompts the GPT-3 model with a fixed one-shot example to generate SPARQL queries.

LLM baselines with standard prompt[11]:

[11] By standard prompt, we mean to directly prompt the LLM for generating SPARQL queries using natural language questions without neither context learning nor example queries.

- LLaMA2-code [34], is a variant of the LLaMA2 language model specifically designed for code generation tasks. In particular, we used the `Code Llama-Instruct` variant, a 34-billion-parameters model as a baseline for generating SPARQL queries from natural language prompts.
- CodeQwen1.5 [41] is a code-specific variant of Qwen1.5 model that has been pre-trained on a large corpus of code data, enabling to handle long context understanding and generation, supporting a context length of up to 64K tokens. Additionally, CodeQwen1.5 offers extensive languages support, supporting a total of 92 coding languages, including SPARQL.
- Mistral-Code [42] is an advanced language model with 7.3B parameters. It is designed to perform on coding tasks, outperforming other models such as Llama 34B in various benchmarks.

4.3. Metrics.

We adopt the same evaluation metrics as in Rony et al. [25] (F1 and BLEU scores) to measure the performance of generating SPARQL queries for three datasets. F1 metric compares the generated SPARQL queries with the *gold-standard* ones and calculates the harmonic mean of precision and recall. BLEU metric evaluates the generated SPARQL queries based on the similarity of n-grams with one or more *gold-standard* queries. The BLEU metric is determined through precision and recall at the token level, represented by Equation (1):

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \times \log(p_n)\right) \tag{1}$$

whereas the F1 score, represented by Equation (2):

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

In addition, we used the QALD-specific Macro F1 metric (F1-QALD), designed for evaluating performance over linked data benchmarks [43]. In particular, we employed F1-QALD metric to evaluate the systems performance for the third research question (RQ_3). Moreover, F1-QALD metric considers additional semantic information in certain scenarios. If the set of golden answers is not empty and question answering system returns empty set, then precision is set to 1, while recall and F-measure are set to 0.

4.4. Setup and Hardware Requirement

We run our experiments on a server equipped with an AMD EPYC 9334 Processor (64 Threads, 32 cores), 1032GB RAM, and NVIDIA A100 80GB PCIe GPUs. Furthermore, we implemented our approach using Python 3.10 and PyTorch 2.1.1 frameworks. We obtained the pre-trained checkpoints of all models (LlaMA-Code[12], Mistral-Code[13], CodeQwen1.5[14]) from the Hugging Face repository.

[12]https://huggingface.co/TheBloke/CodeLlama-34B-Instruct-GPTQ
[13]https://huggingface.co/TheBloke/Mistral-7B-codealpaca-lora-GPTQ
[14]https://huggingface.co/Qwen/CodeQwen1.5-7B-Chat

Table 2. Performance evaluation using F1 and BLEU metrics (RQ_1). Best results are in bold, a dash (-) means no results are available from those baselines using this metric or on this dataset.

	LC-QuAD 2.0		*VQuAnDa*		*QALD-9*	
Models	**BLEU**	**F1**	**BLEU**	**F1**	**BLEU**	**F1**
Non-LLM baselines						
NSpM	34.74	66.47	37.75	59.96	18.23	45.34
SQG	-	74.00	05.09	37.70	04.44	27.85
TeBaQA	-	22.7	13.30	22.41	12.82	28.81
LLM Baselines						
SPARQLGEN	-	-	-	-	-	67.07
SGPT	73.78	**89.04**	72.58	88.87	35.68	67.82
LlaMA2-Code	06.57	18.41	05.61	33.9	08.28	27.09
Mistral-Code	0.87	10.19	0.57	08.57	04.54	16.50
CodeQwen1.5	04.90	15.01	03.20	14.9	13.01	27.10
Our approach						
CoT-SPARQL$_{(ent)}$	50.67	68.94	63.87	84.38	30.68	67.3
CoT-SPARQL$_{(ent+rel)}$	58.08	76.91	71.61	**89.36**	34.65	**70.45**

5. Results and Discussion

5.1. *Evaluating the performance of* SPARQL *generation*

In research question (**RQ_1**), we investigate the impact of *in-context learning* within our CoT-SPARQL approach in generating precise SPARQL queries. In particular, we conduct a comparative analysis against different baselines, employing F1 and BLEU as evaluation metrics. As shown in Table 2, our approach (CoT-SPARQL) outperforms the baselines on two of three datasets[15]. For example, on the VQuAnDa dataset, CoT-SPARQL achieves a BLEU score of 71.61 and an F1 score of 89.36. In contrast, the best Seq2Seq baseline (i.e., SGPT) achieves a BLEU score of 72.58 and an F1 score of 88.87. Moreover, these findings demonstrate that CoT-SPARQL, which incorporates entities and relations into the LLM prompt, achieves the highest F1 scores on two datasets (i.e., 89.36 on VQuAnDa and 70.45 on QALD-9) and the second-best performance on the LC-QuAD 2.0 dataset with an F1 score 89.04. These results indicate that CoT-SPARQL can effectively leverage the pre-trained knowledge in large language models, and robustly encode the semantic information (e.g., entities and relations) from the input question, to generate accurate SPARQL queries.

5.2. *Evaluating the correctness of generated* SPARQL *queries*

To answer **RQ_2**, we reported the number and percentage of valid queries that return correct answers without errors via DBpedia and Wikidata endpoints, and invalid queries that return syntax errors or empty answers, in Table 3. Since the queries generated by the other baselines (NSpM, SQG, TeBaQA, and SGPT) are not publicly available, we were unable to evaluate their validness and only compare our approach with the LLM

[15]The values are obtained from the respective papers: [23–25, 32, 39]

Table 3. Evaluating the correctness of generated SPARQL queries (RQ_2). Best results are in bold.

	LC-QuAD 2.0		*VQuAnDa*		*QALD-9*	
Models	**Valid**	**Invalid**	**Valid**	**Invalid**	**Valid**	**Invalid**
LlaMA2-Code	1216 (25.3%)	3651	759 (75.9%)	241	103 (68.7%)	47
Mistral-Code	121 (2.5%)	4746	46 (4.6%)	954	6 (4%)	144
CodeQwen1.5	998 (20.5%)	3869	505 (50.5%)	495	80 (53.5%)	70
CoT-SPARQL$_{(ent)}$	**3243 (75.0%)**	1642	**951 (95.0%)**	49	**139 (92.7%)**	11
CoT-SPARQL$_{(ent+rel)}$	**4640 (96.0%)**	227	**975 (95.5%)**	25	**143 (95.4%)**	7

baselines (LlaMA2-Code, Mistral-Code and CodeQwen1.5). The evaluation results show that CoT-SPARQL significantly outperforms the LLaMA2-Code model significantly on all datasets. For instance, on the LC-QuAD 2.0 dataset, (CoT-SPARQL$_{(ent+rel)}$), which incorporates both *entities* and *relations*, generates 4640 (96%) Valid queries and only 227 invalid queries. In comparison, the LLaMA2-Code model generates 1216 (25.3%) Valid queries and 3651 (74.7%) Invalid queries. These findings suggest that prompts enriched with In-context Learning and few-shot examples significantly enhance the ability of LLMs to generate valid and correct SPARQL queries than relying only on their pre-trained knowledge. Furthermore, the results show that our approach with both entities and relations (CoT-SPARQL$_{(ent+rel)}$) consistently achieves the highest performance, compared to the variant with only entities (CoT-SPARQL$_{(ent)}$).

5.3. *Executing* SPARQL *queries in question answering (Pilot Study)*

To address **RQ**$_3$, which investigates the performance of our approach in question answering task, we used the GERBIL benchmark framework [43] to execute the generated SPARQL queries. Our goal is to evaluate the effectiveness of our approach in an end-to-end setting, where a natural language question is given as an input, converted into a SPARQL query and then executed to retrieve answers. In particular, we performed a *pilot study* on the QALD-10 dataset [39], the most recent benchmark dataset for questions answering over linked data. Furthermore, we compared the performance of our approach with the state-of-the-art baselines from GERBIL framework, namely:

- Kovriguina et al. [32] employed the GPT-3 model with one-shot example to generate a SPARQL query.
- Borroto and Ricca [44] combined neural machine translation with named entity recognition to convert natural language questions into SPARQL queries.
- Guo et al. [45] developed a system that classifies questions and generates SPARQL queries using templates and a knowledge base.
- Steinmetz et al. [46] introduced a pattern-based method to transform natural language into SPARQL queries by matching patterns and fill variables with relevant information from the question.
- Baramiia et al. [47] presented a ranking method to optimize question answering over knowledge graph, focusing on ranking items to construct SPARQL queries.

As shown in Table 4, CoT-SPARQL$_{(ent+rel)}$ outperforms the state-of-the-art baseline [44] by achieving the higher macro F1-QALD score of 63.87. The full results from GERBIL

Table 4. A pilot study of question answering task over the QALD-10 dataset. For the system of Kovriguina et al. [32], we were unable to find results on the QALD-10 leaderboard, therefore we obtained results from [32] (RQ_3). Best results are in bold.

Approach	Precision	Recall	F1	F1-QALD
Kovriguina et al. [32]	-	-	-	0.29*
Borroto and Ricca [44]	0.4538	0.4574	0.4538	0.5947
Guo et al. [45]	0.5068	0.5238	0.5070	0.5776
Steinmetz et al. [46]	0.3206	0.3312	0.3215	0.4909
Baramiia et al. [47]	0.4289	0.4272	0.4277	0.4281
CoT-SPARQL$_{(ent+rel)}$	0.4944	0.5072	0.4978	**0.6387**

Home Configure Experiment QALD10 Experiment Overview About Us

GERBIL Experiment

Experiment URI: https://gerbil-qa.aksw.org/gerbil/experiment?id=202405140002
Type: QA
Matching: Me - strong entity match

Annotator	Dataset	Language		Micro F1	Micro Precision	Micro Recall	Macro F1	Macro Precision	Macro Recall	Error Count	avg millis/doc	Macro F1 QALD	Timestamp	GERBIL version
COT-SPARQL (uploaded)	QALD10 Test Multilingual	en		0.4767	0.652	0.3757	0.4978	0.4944	0.5072	0	0	0.6387	2024-05-14 11:25:00	0.2.3
COT-SPARQL (uploaded)	QALD10 Test Multilingual	en	C2KB	0.7446	0.6951	0.8017	0.7186	0.6928	0.8114	0			2024-05-14 11:25:00	0.2.3
COT-SPARQL (uploaded)	QALD10 Test Multilingual	en	P2KB	0.6981	0.6086	0.8185	0.6771	0.6321	0.7589	0			2024-05-14 11:25:00	0.2.3
COT-SPARQL (uploaded)	QALD10 Test Multilingual	en	RE2KB	0.4587	0.4495	0.4682	0.4656	0.4698	0.4735	0			2024-05-14 11:25:00	0.2.3

Figure 4. The detail results of our approach in GERBIL benchmark framework.

[16] are also shown in Figure 4. Overall, the evaluation results (shown in the top row of Figure 4) indicates that our queries are more reliable in retrieving correct answers from knowledge graphs. Furthermore, the second row of Table 4 represents GERBIL's [43] sub-experiment called *Concept to Knowledge Base* (C2KB), which identifies all resources that are relevant for the given question. In particular, GERBIL calculates the measures precision, recall and F-measure based on the comparison of the expected resource URIs and the URIs returned by the QA system. The third row of Table 4 shows the *Properties to Knowledge Base* (P2KB) sub-experiment, where GERBIL identifies all properties that are relevant for the given question. The last row of Table 4 represents *Relation to Knowledge Base* (RE2KB) sub-experiment, which focuses on the triples that have to be extracted from the question and are needed to generate the SPARQL query for retrieving correct answers. The full evaluation results can be accessed via the public KGQA leaderboard.[17]

[16] Experiment link at GERBIL framework https://gerbil-qa.aksw.org/gerbil/experiment?id=202405140002
[17] https://github.com/KGQA/leaderboard/blob/gh-pages/wikidata/qald.md#qald-10

6. Conclusion and Future Work

This paper presents a novel approach for SPARQL generation, leveraging In-context learning and Chain-of-Thoughts prompt in large language models to generate high-quality SPARQL queries from natural language. Specifically, we incorporate additional context information from the input question, including entities and relations, into the Chain-of-Thought prompt. Furthermore, we include a semantically similar one-shot example within the prompt to facilitate generating precise SPARQL queries. In contrast to existing methods relying on pre-defined templates or fixed rules, our approach is capable of adapting to generate diverse SPARQL syntax tailored to a target knowledge graph. To assess the effectiveness of our approach, we conducted experiments on various benchmark datasets. The results demonstrate that our method outperforms state-of-the-art methods in terms of both accuracy and the validity of generated SPARQL queries. In our future research, we plan to investigate fine-tuning large language models (e.g., LlaMA2-Code) on multitask learning for both SPARQL generation and question answering over knowledge graphs.

Acknowledgement

This work has been supported by the German Federal Ministry of Education and Research (BMBF) within the projects, COLIDE (grant no 01I521005D), KIAM (grant no 02L19C115), the European Union's Horizon Europe research and innovation programme (grant No 101070305), the Ministry for Economic Affairs, Innovation, Digitalisation and Energy of North Rhine-Westphalia (MWIDE NRW) within the project Climate bOWL (grant no 005-2111-0020), and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation): TRR 318/1 2021 – 438445824.

References

1. Emil Kuric, Javier D Fernández, and Olha Drozd. Knowledge graph exploration: a usability evaluation of query builders for laypeople. In *Semantic Systems. The Power of AI and Knowledge Graphs: 15th International Conference, SEMANTiCS 2019, Karlsruhe, Germany, September 9–12, 2019, Proceedings 15*, pages 326–342. Springer, 2019.
2. Patrick Hoefler. Linked data interfaces for non-expert users. In *The Semantic Web: Semantics and Big Data: 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings 10*, pages 702–706. Springer, 2013.
3. Muhammad Rizwan Saeed, Charalampos Chelmis, and Viktor K Prasanna. Asqfor: Automatic sparql query formulation for the non-expert. *AI Communications*, 31(1): 19–32, 2018.
4. Peter Ochieng. Parot: Translating natural language to sparql. *Expert Systems with Applications: X*, 5:100024, 2020.
5. K Bretonnel Cohen and Jin-Dong Kim. Evaluation of sparql query generation from natural language questions. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2013, page 3. NIH Public Access, 2013.

6. Anna Formica, Ida Mele, and Francesco Taglino. A template-based approach for question answering over knowledge bases. *Knowledge and Information Systems*, pages 1–27, 2023.
7. Xiaoyu Yin, Dagmar Gromann, and Sebastian Rudolph. Neural machine translating from natural language to sparql. *Future Generation Computer Systems*, 117:510–519, 2021.
8. Yi-Hui Chen, Eric Jui-Lin Lu, and Ting-An Ou. Intelligent sparql query generation for natural language processing systems. *IEEE Access*, 9:158638–158650, 2021.
9. Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. Act-sql: In-context learning for text-to-sql with automatically-generated chain-of-thought. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532, 2023.
10. Yuanyuan Liang, Jianing Wang, Hanlun Zhu, Lei Wang, Weining Qian, and Yunshi Lan. Prompting large language models with chain-of-thought for few-shot knowledge base question generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4329–4343, 2023.
11. Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
12. Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. Lubm: A benchmark for owl knowledge base systems. *Journal of Web Semantics*, 3(2-3):158–182, 2005.
13. Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd international conference on World Wide Web*, pages 747–758, 2014.
14. Olaf Görlitz, Matthias Thimm, and Steffen Staab. Splodge: Systematic generation of sparql benchmark queries for linked open data. In *The Semantic Web–ISWC 2012: 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I 11*, pages 116–132. Springer, 2012.
15. Shi Qiao and Z Meral Özsoyoğlu. Rbench: Application-specific rdf benchmarking. In *Proceedings of the 2015 acm sigmod international conference on management of data*, pages 1825–1838, 2015.
16. Henrik Dibowski and Klaus Kabitzsch. Ontology-based device descriptions and device repository for building automation devices. *EURASIP Journal on Embedded Systems*, 2011:1–17, 2011.
17. Denny Vrandečić. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st international conference on world wide web*, pages 1063–1064, 2012.
18. Güneş Aluç, Olaf Hartig, M Tamer Özsu, and Khuzaima Daudjee. Diversified stress testing of rdf data management systems. In *The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13*, pages 197–212. Springer, 2014.
19. Guillaume Bagan, Angela Bonifati, Radu Ciucanu, George HL Fletcher, Aurélien Lemay, and Nicky Advokaat. gmark: Schema-driven generation of graphs and queries. *IEEE Transactions on Knowledge and Data Engineering*, 29(4):856–869, 2016.

20. Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st international conference on World Wide Web*, pages 639–648, 2012.
21. Gideon Zenz, Xuan Zhou, Enrico Minack, Wolf Siberski, and Wolfgang Nejdl. From keywords to semantic queries—incremental query construction on the semantic web. *Journal of Web Semantics*, 7(3):166–176, 2009.
22. IAIS bFraunhofer. Knowledge graph question answering using graph-pattern isomorphism. In *Further with Knowledge Graphs: Proceedings of the 17th International Conference on Semantic Systems, 6-9 September 2021, Amsterdam, The Netherlands*, volume 53, page 103. IOS Press, 2021.
23. Tommaso Soru, Edgard Marx, André Valdestilhas, Diego Esteves, Diego Moussallem, and Gustavo Publio. Neural machine translation for query construction and composition. *arXiv preprint arXiv:1806.10478*, 2018.
24. Hamid Zafar, Giulio Napolitano, and Jens Lehmann. Formal query generation for question answering over knowledge bases. In *European semantic web conference*, pages 714–728. Springer, 2018.
25. Md Rashad Al Hasan Rony, Uttam Kumar, Roman Teucher, Liubov Kovriguina, and Jens Lehmann. Sgpt: a generative approach for sparql query generation from natural language questions. *IEEE Access*, 10:70712–70723, 2022.
26. Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
27. Zihan Yu, Liang He, Zhen Wu, Xinyu Dai, and Jiajun Chen. Towards better chain-of-thought prompting strategies: A survey, 2023.
28. Zongjie Li, Chaozheng Wang, Pingchuan Ma, Chaowei Liu, Shuai Wang, Daoyuan Wu, Cuiyun Gao, and Yang Liu. On extracting specialized code abilities from large language models: A feasibility study, 2023.
29. Xue Jiang, Yihong Dong, Lecheng Wang, Zheng Fang, Qiwei Shang, Ge Li, Zhi Jin, and Wenpin Jiao. Self-planning code generation with large language models, 2023.
30. Mohammadreza Pourreza and Davood Rafiei. Din-sql: Decomposed in-context learning of text-to-sql with self-correction, 2023.
31. Shuangtao Yang, Mao Teng, Xiaozheng Dong, and Fu Bo. Llm-based sparql generation with selected schema from large scale knowledge base. In *China Conference on Knowledge Graph and Semantic Computing*, pages 304–316. Springer, 2023.
32. Liubov Kovriguina, Roman Teucher, Daniil Radyush, and Dmitry Mouromtsev. Sparqlgen: One-shot prompt-based approach for sparql query generation. 2023.
33. Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.
34. Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

35. Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
36. Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II 18*, pages 69–78. Springer, 2019.
37. Endri Kacupaj, Hamid Zafar, Jens Lehmann, and Maria Maleshkova. Vquanda: Verbalization question answering dataset. In *European Semantic Web Conference*, pages 531–547. Springer, 2020.
38. Ngonga Ngomo. 9th challenge on question answering over linked data (qald-9). *language*, 7(1):58–64, 2018.
39. Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, et al. Qald-10–the 10th challenge on question answering over linked data. *Semantic Web*, (Preprint):1–15, 2023.
40. Rricha Jalota, Daniel Vollmers, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. Lauren - knowledge graph summarization for question answering. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pages 221–226, 2021. .
41. Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
42. Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.
43. Ricardo Usbeck, Michael Röder, Michael Hoffmann, Felix Conrads, Jonathan Huthmann, Axel-Cyrille Ngonga-Ngomo, Christian Demmler, and Christina Unger. Benchmarking question answering systems. *Semantic Web*, 10(2):293–304, 2019.
44. Manuel A Borroto and Francesco Ricca. Sparql-qa-v2 system for knowledge base question answering. *Expert Systems with Applications*, 229:120383, 2023.
45. Kunpeng Guo, Clement Defretiere, Dennis Diefenbach, Christophe Gravier, and Antoine Gourru. Qanswer: Towards question answering search over websites. In *Companion Proceedings of the Web Conference 2022*, pages 252–255, 2022.
46. Nadine Steinmetz, Bhavya Senthil-Kumar, and Kai-Uwe Sattler. Conversational question answering using a shift of context. In *EDBT/ICDT Workshops*, 2021.

47. Nikita Baramiia, Alina Rogulina, Sergey Petrakov, Valerii Kornilov, and Anton Razzhigaev. Ranking approach to monolingual question answering over knowledge graphs. In *Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference (ESWC 2022)*, 2022.