In-Context Learning for Knowledge Base Question Answering for Unmanned Systems based on Large Language Models

Yunlong Chen[1], Yaming Zhang[1], Jianfei Yu[1(✉)], Li Yang[2], and Rui Xia[1]

[1] School of Computer Science and Engineering,
Nanjing University of Science and Technology, China
[2] Wee Kim Wee School of Communication and Information,
Nanyang Technological University, Singapore
{ylchen, ymzhang, jfyu, rxia}@njust.edu.cn

Abstract. Knowledge Base Question Answering (KBQA) aims to answer factoid questions based on knowledge bases. However, generating the most appropriate knowledge base query code based on Natural Language Questions (NLQ) poses a significant challenge in KBQA. In this work, we focus on the CCKS2023 Competition of Question Answering with Knowledge Graph Inference for Unmanned Systems. Inspired by the recent success of large language models (LLMs) like ChatGPT and GPT-3 in many QA tasks, we propose a ChatGPT-based Cypher Query Language (CQL) generation framework to generate the most appropriate CQL based on the given NLQ. Our generative framework contains six parts: an auxiliary model predicting the syntax-related information of CQL based on the given NLQ, a proper noun matcher extracting proper nouns from the given NLQ, a demonstration example selector retrieving similar examples of the input sample, a prompt constructor designing the input template of ChatGPT, a ChatGPT-based generation model generating the CQL, and an ensemble model to obtain the final answers from diversified outputs. With our ChatGPT-based CQL generation framework, we achieved the second place in the CCKS 2023 Question Answering with Knowledge Graph Inference for Unmanned Systems competition, achieving an F1-score of 0.92676.

Keywords: ChatGPT · Chain-of-Thought · In-Context Learning.

1 Introduction

As an important task in Natural Language Processing (NLP), Knowledge Base Question Answering (KBQA) aims to generate accurate and complete query statements from user-provided natural language questions (NLQs), and these query statements are then used to retrieve relevant information from the knowledge base and provide accurate answers. In this work, we focus on the CCKS 2023 Question Answering with Knowledge Graph Inference for Unmanned Systems competition, which is a KBQA evaluation task where cypher query lan-
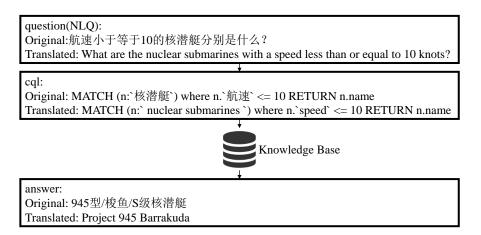
> question(NLQ):
> Original:航速小于等于10的核潜艇分别是什么？
> Translated: What are the nuclear submarines with a speed less than or equal to 10 knots?

> cql:
> Original: MATCH (n:`核潜艇`) where n.`航速` <= 10 RETURN n.name
> Translated: MATCH (n:` nuclear submarines `) where n.`speed` <= 10 RETURN n.name

Knowledge Base

> answer:
> Original: 945型/梭鱼/S级核潜艇
> Translated: Project 945 Barrakuda

Fig. 1. Illustration of an example in the evaluation task.

guage (CQL) serves as the query statements. Fig. 1 gives an example of the CCKS2023 competition.

In the literature, most existing studies on KBQA can be categorized into two types: information retrieval-based (IR-based) approaches and semantic parsing-based (SP-based) approaches. Both of them require first identifying the subject within the NLQ and linking it to an entity in the knowledge base (KB). The former line of work aims to derive answers by reasoning within a question-specific graph extracted from the KB with the assistance of those linked entities [1,2], whereas the latter line of work aims to obtain answers by executing a parsed logic form based on the linked entities [3,4]. Since the annotation in the dataset of the CCKS2023 competition contains manually annotated CQLs, we follow the latter line of approaches in this work.

However, the majority of existing SP-based approaches are built upon LSTM or pre-trained models like BERT, which are constrained by their scale or their pre-training data and may encounter challenges in effectively generating suitable knowledge base query codes based on NLQs. With the recent advancements of pre-trained language models, many Large Language Models (LLMs) have been shown to achieve surprisingly good performance on many question answering datasets under zero-shot or few-shot settings. These LLMs have also showcased an impressive capacity to deeply comprehend sentence semantics and accurately translate them into multiple languages, and even generate code when required. Therefore, we aim to explore the potential of LLMs on the Chinese KBQA task for unmanned systems.

Specifically, we propose a ChatGPT-based CQL generation framework, consisting of six parts. The first part involves an auxiliary model that takes the given NLQ as input and predicts structural information for each clause separately. The second part comprises a proper noun matcher, which identifies explicitly mentioned proper nouns existing in the KB from the given NLQ. The third part

consists of a demonstration example selector that employs a key-information-based similarity calculation criterion to retrieve demonstration samples for the given NLQ based on the aforementioned results. The fourth part encompasses a prompt constructor that constructs input text by integrating demonstration samples, NLQ, and task-specific prior knowledge. The fifth part incorporates a ChatGPT-based generation model, which inputs the constructed text into Chat-GPT to generate CQL. Subsequently, post-processing is applied to the generated CQL. Lastly, the sixth part introduces an ensemble model, in which multiple answers retrieved from the knowledge base by the post-processed CQL are combined through a voting mechanism to obtain the final result.

We conduct experiments on the dataset provided by the competition, and the results show the high efficiency of our generative framework. Therefore, we achieved the second place in the CCKS 2023 Question Answering with Knowledge Graph Inference for Unmanned Systems competition with an F1-score of 0.92676.

2   Related Work

2.1   Large Language Model

Large Language Models (LLMs) typically possess a vast number of learnable parameters and undergo extensive training on enormous text datasets, examples of which include ChatGPT[5], LLaMA[6], OPT[7], PaLM[8], CodeX[9], and so on. With the advancement of LLMs, traditional pre-trained models like BERT[10], RoBERTa[11], BART[12], T5[13], have faced great challenges. The ability of LLMs to adapt to downstream tasks without the need for retraining, but task-specific instructions, has greatly reduced the cost of solving downstream tasks.

2.2   In-Context Learning

As mentioned in Section 2.1, LLMs typically demonstrate emergent abilities [14,15] with increasing model and corpus size, i.e., the ability to learn from the given examples present in the context, known as In-Context Learning (ICL). This ability helps LLMs in better adapting to downstream tasks. While solely relying on task-specific instructions may not lead to superior performance compared to fine-tuned models in some downstream tasks, introducing ICL can often result in considerable improvements in LLMs' performance on downstream tasks.

2.3   Chain-Of-Thought

Chain-of-Thought (CoT) [16] is an extremely efficient and easy prompting strategy that endows LLMs with reasoning capabilities, enabling LLMs to decompose and comprehend complex tasks. Specifically, CoT leverages several given examples with inferred answers to assist LLMs in comprehending the reasoning process of complex tasks, thus performing reasoning on the target problem and obtaining results. In general, CoT leverages several pre-given exemplars with inferred answers to help LLMs understand the reasoning process of intricate tasks.
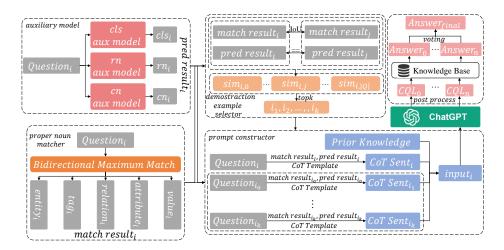
Fig. 2. The overall architecture of our ChatGPT-based KBQA framework.

3   Methodology

Recently, LLMs have showcased robust generalizability across a diverse spectrum of tasks by leveraging few-shot in-context learning. Notably, LLMs possess the capability to transform unstructured sentences into structured and executable code, rendering them valuable assets in KBQA [17].

However, the CQL solely generated by ChatGPT falls short of our expectations. Therefore, we observe CQL's overall structure and summarize empirical knowledge to design processing techniques and auxiliary tasks. These aid Chat-GPT in capturing key information and parsing CQL structures from NLQs, enabling it to adapt to downstream tasks and generate high-quality CQL.

In general, our ChatGPT-based CQL generation framework consists of six steps (excluding KB construction and answer retrieval), as illustrated in Fig. 2 and Fig. 3 provides a visualized example of using our generative framework to generate CQL from NLQ.

1. Three auxiliary tasks to predict structural information for CQL clauses.
2. Bidirectional maximum matching-based proper noun matching for NLQ.
3. Selecting demonstration examples based on the aforementioned results.
4. Combining NLQ, demonstration examples, and prior knowledge into CoT format as input text.
5. ChatGPT generates CQL based on the constructed input text, followed by post-processing of the generated CQL.
6. Voting for the answers retrieved from the given KB by CQLs.

The overall workflow is as follow:

Firstly, within the **auxiliary model**, the structural information of different clauses in CQL ($pred\ result_i$) is predicted based on NLQ ($Question_i$). Mean-

Question:
Low Rate Initial Production的英文缩写对应的中文是什么？
Auxiliary Tasks result:
cls: 0, rn: 1, cn: 1
Proper Noun Matcher result:
entity: ["Low Rate Initial Production"], tag: [], relation: ["英文缩写", "中文"], attribute: ["英文缩写", "中文"], value: []

Demonstration examples

Oregon Iron Works的英文缩写是什么
Auxiliary Tasks result:
cls: 0, rn: 1, cn: 1
Proper Noun Matcher result:
entity: ["Oregon Iron Works"], tag: [], relation: ["英文缩写"], attribute: ["英文缩写"], value: []

RS-120导弹的研发单位有哪些中文名称？
Auxiliary Tasks result:
cls: 0, rn: 1, cn: 1
Proper Noun Matcher result:
entity: ["RS-120导弹"], tag: [], relation: ["研发单位"], attribute: ["中文"], value: []

Constructed input text:
已知问题原文为："Oregon Iron Works的英文缩写是什么"，让我们一步步思考：从原文中可以抽取出：实体：Oregon Iron Works，属性：英文缩写，关系：英文缩写，条件个数：1，关系个数：1，基于抽取结果，可得相应的cql语句为： MATCH (n)-[r:`英文缩写`]->(m) where n.name="Oregon Iron Works" RETURN m.name,
......
已知问题原文为："RS-120导弹的研发单位有哪些中文名称？"，让我们一步步思考：从原文中可以抽取出：实体：RS-120导弹，属性：中文，关系：研发单位，条件个数：1，关系个数：1，基于抽取结果，可得相应cql语句为：MATCH (n)-[r:`研发单位`]->(m) where n.name="RS-120导弹" RETURN m.`中文`、

那么在已知问题原文为："Low Rate Initial Production的英文缩写对应的中文是什么？"，让我们一步步思考：从原文中可以抽取出：实体：Low Rate Initial Production，属性：英文缩写、中文，关系：英文缩写、中文，条件个数：1，关系个数：1，基于抽取结果，可得相应的cql语句为：？

ChatGPT

CQL:
MATCH (n:` Low Rate Initial Production`) RETURN n.`中文`

post-processing

CQL:
MATCH (n)-[r:`英文缩写`]->(m) where n.name="Low Rate Initial Production" RETURN m.`中文`

Fig. 3. A visualized example of generating CQL from NLQ.

while, the **proper noun matcher** identifies the proper nouns ($match\ result_i$) explicitly mentioned in $Question_i$ and existing in the KB.

Subsequently, both $pred\ result_i$ and $match\ result_i$ are fed into the **demonstration example selector** to compute the similarity between different samples, thereby selecting demonstration examples $(i_1, i_2 \ldots, i_k)$ for each sample.

Following that, within the **prompt constructor**, the $Question_i$, $pred\ result_i$, and $match\ result_i$ are firstly combined into CoT format to obtain $CoT\ Sent_i$. Next, $Prior\ Knowledge, CoT\ Sent_i$, and corresponding demonstration examples $(CoT\ Sent_{i_1}, \ldots, CoT\ Sent_{i_k})$ are combined to form $input_i$. Then, $input_i$ is fed into **ChatGPT** to generate CQL, followed by post-processing.

By repeating the aforementioned steps, multiple CQLs $(CQL_0, \ldots, CQL_n)$ are acquired. Executing these CQLs in the given KB yields multiple answers $(Answer_0, \ldots, Answer_n)$. Employing a **voting** mechanism on these answers yields the most reliable response as the final outcome $(Answer_{final})$.

3.1   Auxiliary Tasks

Upon analyzing CQLs and evaluating the task instructions, we summarize four execution intent categorized by the content of the RETURN clause: 1. Entity-Attribute retrieval; 2. Entity counting; 3. Conditional sorting; 4. Attribute-value comparison. These diverse execution purposes impact the overall CQL structure. Furthermore, the CQL structure is influenced by the number of relations and conditions, referring to the number of entity jumps in the MATCH clause and the restrictive conditions in the WHERE clause.

Due to the potential impact of the aforementioned information on the CQL structure and the difficulty in directly obtaining them from NLQs or other related sources, we design three auxiliary tasks for corresponding predictions:

- Intent classification (cls).
- Relation count classification (rn).
- Condition count classification (cn).

The aforementioned auxiliary tasks take NLQs as input and predict the intent $(0/1/2/3)$, relation number $(0/1/2)$, and condition number $(0/1/2)$ of the CQL. For example, given the CQL in Fig.1, the output of the three auxiliary tasks are 0, 0, and 1, respectively. These values signify that the intent of the CQL is to retrieve an entity's attribute, there are no relations mentioned in the CQL, and there is one condition specified.

Notably, these three auxiliary tasks share NLQs as input and yield similar outputs, all treated as simple classification results, thus allowing them to utilize a consistent model architecture. Any pre-trained model like BERT-chinese and mT5 can be used to extract global features from the text, followed by a feed-forward network for output prediction of the auxiliary tasks.

3.2   Proper Noun Matcher

In NLQs, some proper nouns are either explicitly or implicitly mentioned, which are likely to appear in the CQLs. Therefore, it is essential to extract the mentioned proper nouns from NLQs. To this end, we employ the bidirectional maximum matching based on the given proper noun vocabulary to all NLQs, obtaining the proper nouns that appear in them.

During bidirectional maximum matching, we observed that unconstrained execution could introduce noise into the matching process for relation and attribute proper nouns. Although designing a universal set of constraints is challenging, creating NLQ-specific constraints is more feasible. Therefore, our approach gives precedence to extracting entities and tags from the NLQ. Subsequently, by leveraging relevant knowledge from KB, we integrate the entities corresponding to the tags into the matched entities. Taking the entities as the starting point and considering two-hop relations as the scope, all the involved entities are referred to as entity set, and all the involved relations are referred to as *candidate* relation set. Using the entity set, we retrieve associated attributes from the given knowledge base, creating a *candidate* attribute set.

The NLQ's bidirectional maximum matching results in the *matched* attribute and relation sets. The final matching results for attribute and relation sets are obtained by intersecting the *candidate* sets with the *matched* sets.

3.3   Demonstration Example Selector

For NLQ-specific demonstration example selection, an appropriate similarity calculation criterion is vital. The conventional method, such as using BERT or

similar models for cosine similarity computation between global features (GF-Sim), may not yield accurate similarity results due to substantial noise present in NLQs. This noise considerably affects the results of GF-Sim. With approximately 35% of the text containing valid information (proper nouns) while the rest being noisy, the impact on GF-Sim is significant.

Based on the above findings, we propose a key-information-based similarity criterion (KI-Sim) for NLQs. It focuses on key components in NLQ like proper nouns and other influential details impacting CQLs, which is computed as follows:

$$Similarity(i,j) = \sum_{k}^{e,t,r,a,v} w_k * \text{IoU}(i_k, j_k) + \sum_{k}^{cls,rn,cn} w_k * (i_k == j_k) \quad (1)$$

where, i, j represent NLQs' id, e, t, r, a, v stand for entity, tag, relation, attribute, and value, cls, rn, cn indicate the predicted intent, relation number, and condition number, i_k, j_k refer to proper nouns or auxiliary task predictions from NLQs with id i, j, and w_k signifies the corresponding similarity weight.

3.4 Prompt Constructor

Prior Knowledge To bolster ChatGPT's grasp and alignment with the downstream task, we propose to incorporate task-specific prior knowledge and integrate it into text (see Appendix 1) to feed into ChatGPT. These prior knowledge are derived from observations of CQLs and hold general applicability for this downstream task, rather than being specific to any particular NLQ.

In-Context Learning To enhance ChatGPT's CQL generation, we utilize KI-Sim (Section 3.3) to select demonstration examples for ICL. This aids ChatGPT in CQL generation, thereby improving the quality of the generated CQLs.

Chain-Of-Thought Intuitively, directly generating CQL from NLQ is challenging. Yet, analyzing the composition and syntax of CQLs and NLQs reveals a high likelihood of shared proper nouns. Drawing inspiration from the CoT method, we split the CQL generation task into two sub-tasks:

1. Matching relevant proper nouns from NLQ.
2. Generating CQL based on NLQ and the matched proper nouns.

As elucidated in Section 3.2, with the completion of sub-task 1, ChatGPT now focuses on addressing sub-task 2. To achieve this, a template was devised (see Appendix 2) that integrates matched proper nouns and NLQ following the CoT approach, presenting them jointly to ChatGPT.

3.5   ChatGPT-Based Generation Model

After inputting the constructed text into ChatGPT, it generates the corresponding CQL for the given NLQ. Concerning the CQL, three common situations arise:

- Entities matched are mis-classified as tags.
- The number of relation does not match the results from auxiliary tasks.
- Matching with unprovided proper nouns.

Among these situations, the first two can result in inaccurate CQL execution and should be prevented or rectified. In contrast, the third situation is favorable. As described in Section 3.2, only explicit proper nouns can be matched, leaving implicit ones unmatched. This implies ChatGPT's successful identification of implicit proper nouns in the NLQ. In this case, post-processing methods can be used to map non-proper nouns to the provided vocabulary for correction.

Consequently, we present three post-processing methods to address these situations to get the final ChatGPT-generated CQL:

- Reclassify the mis-classified tags as entities and position them correctly.
- Use condition truncation and filling for isolated condition clause correction.
- Utilize fuzzy matching to map implicit proper nouns to vocabulary and make CQL modifications correspondingly.

3.6   Ensemble Model

Executing CQL in the KB doesn't always ensure correct answers, and sometimes no answers are found. However, generating new CQLs could enhance the retrieval success. Thus, for each NLQ, we generate multiple CQLs, retrieve corresponding answers, and apply a voting mechanism to ascertain the final answer.

4   Experiment

4.1   Dataset

We conduct experiments based on the competition's dataset, which encompasses knowledge base construction data, as well as training, validation (preliminary round), and test (final round) datasets. The training set includes annotations, while validation's annotations are released with the un-annotated test set in the final round. Annotations include answers and the CQL used for retrieval from KB. We will validate our generative framework on this dataset.

4.2   Evaluation

In the experiments, the main evaluation concern is precise answer retrieval for NLQs. The evaluation metrics include Macro Precision (Eqn. (2)), Macro Recall

(Eqn. (3)), and Averaged F1 (Eqn. (4)), which are defined below:

$$P = \frac{1}{|Q|} \sum_{i=1}^{|Q|} P_i, \quad P_i = \frac{|A_i \cap G_i|}{|A_i|} \tag{2}$$

$$R = \frac{1}{|Q|} \sum_{i=1}^{|Q|} R_i, \quad R_i = \frac{|A_i \cap G_i|}{|G_i|} \tag{3}$$

$$F1 = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{2P_i R_i}{P_i + R_i} \tag{4}$$

where $|Q|$ denotes the number of NLQs in the dataset, A_i, G_i denotes the player's and ground-truth answer sets to the question whose id is i, respectively.

4.3 Implementation

Similarity During similarity computation, entity weights are set to 5, tag weights to 3, relation weights to 3, attribute weights to 1, value weights to 0.5, cls weights to 0.5, rn weights to 0.3, and cn weights to 0.3.

ChatGPT The ChatGPT we used in this paper is gpt-3.5-turbo-0613. It should be noted that we set the temperature parameter to 1 (default) to ensure the diversity of CQLs when ChatGPT generates responses multiple times.

Auxiliary Task We use the mT5-large as the pre-trained model. For the auxiliary tasks, the global random seed is 33. The batch size is 32, trained for 100 epochs. Initial learning rates for the backbone and non-backbone part are set at 1e-6 and 1e-4, respectively. Cross-entropy loss is employed for loss calculation.

4.4 Main Results

Table 1. Main Results. Note the Prior indicate the prior knowledge, the Ensemble indicate the ensemble model, the Post indicate the post-processing in ChatGPT-based Generation Model.

+Prior	+Ensemble	+ICL+CoT	+Post	Averaged F1 (Validation)	Averaged F1 (Test)
✓	✗	✗	✗	0.72539	\
✓	✓	✗	✗	0.83865	0.86204
✓	✓	✓	✗	\	0.91561
✓	✓	✓	✓	\	**0.92676**

In Table 1, the performance of ChatGPT with different processing techniques is presented, where the last row shows the performance of our proposed ChatGPT-based CQL generation framework.

In the preliminary round, with only prior knowledge and voting mechanism, our F1 score on the validation set is 0.83865, obtaining the second place. In the final round, our ChatGPT-based CQL generation framework achieves an F1 score of 0.92676 on the test set, obtaining the second place.

4.5   Ablation Study

As shown in Table 1, all different processing techniques can improve the final performance, but their effects are different:

Ensemble Model The essence of this technique is to allow ChatGPT to generate multiple CQLs and vote on the answers. The multiple generations can help ChatGPT re-understand NLQ and increase the diversity of generated CQLs.

ICL+CoT The essence of this technique is to enable ChatGPT to capture and learn implicit relations that may exist in downstream tasks based on given demonstration examples. By using the decomposed sub-tasks, ChatGPT can achieve a deeper understanding of the downstream task and adapt to it, generating higher quality and more robust CQLs.

Post-Processing The essence of this technique is to manually correct the generation errors of ChatGPT without interfering with its process of generating CQLs. Instead, it intervenes in the results generated by ChatGPT, ensuring that the results do not contain factual errors.

4.6   Auxiliary Task Results

Table 2. The performance on three auxiliary tasks

Auxiliary Tasks	Accuracy(%)
Intent classification	99.0
Relation count classification	97.0
Condition count classification	98.2

Based on the performance of the auxiliary tasks in Table 2, we can find that the proposed model performs well on the three auxiliary tasks. Therefore, it is generally useful to incorporate the auxiliary task-related information into our generative framework.

4.7   Similarity Comparison

To verify KI-Sim's effectiveness (Section 3.3), we present the demonstration examples in Table 3. The global features are extracted from bert-base-chinese.

Table 3. Demonstration examples based on different similarity calculation criterion

NLQ		Original	Translated
		最大飞行速度 小于等于 460 的实体有几个?	How many entities have a **maximum flying speed** less than or equal to 460 ?
GF-Sim	top1	阿姆德-500M/2M沉底水雷 的 产国 是哪个?	What is the **origin country** of the **AMD-500M/2M Submarine Mine** ?
	top2	94式90毫米轻迫击炮 的 口径 是多少?	What is the **caliber** of the **Type-94 90mm Light Mortar** ?
	top3	弹径 为 1.37 的 舰地（潜地）导弹 有哪些?	Which **Ship-to-Ground (Submarine-to-Ground) Missile** has a **caliber** equal to 1.37 ?
KI-Sim	top1	最大飞行速度 大于 252 的实体有几个?	How many entities have a **maximum flying speed** greater than 252 ?
	top2	最大飞行速度 等于 850 的实体有几个?	How many entities have a **maximum flying speed** equal to 850 ?
	top3	最大飞行速度 等于 745 的实体有几个?	How many entities have a **maximum flying speed** equal to 745 ?

Key information in the NLQs is highlighted using bold and boxes, with green boxes and red boxes denoting their presence and absence in the top NLQ, respectively. It is evident that the demonstration examples selected by KI-Sim are more similar to the top NLQ. This underscores the effectiveness of KI-Sim.

5   Conclusion

In this paper, we proposed a ChatGPT-based CQL generation framework, which consists of six components: an auxiliary model that predicted structural information for CQLs based on given NLQs, a proper noun matcher that extracted explicit proper nouns, a demonstration example selector that used KI-Sim to select demonstration examples, a prompt constructor that concatenated the NLQ, demonstration examples, and prior knowledge in the form of a Chain-of-Thought, a ChatGPT-based generation model that generated CQLs using the concatenated text, and an ensemble model that produced more reliable results by voting on diversified answers. Experimental results validate the effectiveness of our generative framework, achieving a remarkable second-place rank in the CCKS 2023 Question Answering with Knowledge Graph Inference for Unmanned Systems competition.

Appendix 1

Original

给定一个中文问题，用于对知识图谱的查询。问题既包含简单问题（实体属性、关系的单一查询），也包含复杂问题（显示约束、隐式约束、比较、布尔、多跳等）。请模仿提供的示例中体现的代码风格和语法，综合分析问题和实体名称、实体属性、实体标签、关系，生成一个符合cypher语法的查询语句，用于从知识图谱中获取答案。cypher语句应该包含以下部分：MATCH：用于匹配图数据库中的节点和关系，可以指定节点和关系的类型、属性和方向；WHERE：用于过滤匹配结果，可以使用逻辑运算符、比较运算符、字符串操作等；RETURN：用于返回查询结果，可以使用聚合函数、排序函数、限制函数等；WITH：用于将查询结果传递给下一个子句，可以使用聚合函数、排序函数、限制函数等。你需要识别问题中的实体、属性、标签、关系和约束条件，cypher语句用圆括号()表示节点，用方括号[]表示关系,用冒号:表示标签，用.表示属性，用箭头->表示关系方向。例如，(n)，[r:'采用技术']，[r:'厂商制造能力']，n.'全重'，(n)-[r:'产国']->(m)-[r1:'军种涉及项目']->(l)。cypher语句用运算符如=, <, >, AND, OR, NOT来比较值和过滤结果。用函数如count(), min(), max(), avg(), sum()来对结果进行计算。请注意，若问题为布尔型问题，cypher语句返回值是True或False。例如,问题：MQ-4C是2013年首飞吗？返回值：True。若问题为比较型问题，并询问实体，cypher语句返回值是实体名称。例如，问题：M29式81毫米迫击炮的口径和M1高射炮的相比，哪个更小？对应的cypher语句：MATCH (n) where n.name="M29式81毫米迫击炮" or n.name="M1高射炮" RETURN n.name ORDER BY n.'口径' asc limit 1返回值：M29式81毫米迫击炮若问题为比较实体属性值的问题，cypher语句返回值是True 或False。例如，问题：扫描鹰无人机的交付数量是否比MQ-1"捕食者"无人机的多？对应的cypher语句：MATCH (n), (m) where n.name="扫描鹰无人机" and m.name="MQ-1"捕食者"无人机" RETURN n.'交付数量' > m.'交付数量'返回值：False请自行判断问题类型，分析问题中的逻辑关系，生成cypher语句。

Translated

Given a Chinese question for querying a knowledge graph, the question includes simple queries about entity attributes and relationships, as well as complex queries involving explicit and implicit constraints, comparisons, boolean logic, and multi-hop scenarios. Emulating the provided code style and syntax, the analysis encompasses entities, attributes, labels, and relationships to generate Cypher Query Language (CQL) for knowledge graph answers. CQL consists of several sections: MATCH, identifying nodes and relationships with specified types, properties, and directions; WHERE, filtering results with logical and comparison operators; RETURN, providing query outcomes using aggregation, sorting, and limiting functions; and WITH, forwarding results to subsequent clauses.

You need to identify entities, attributes, labels, relationships, and constraint conditions in the question. In CQL syntax, use parentheses () to represent nodes, square brackets [] to represent relationships, colons : to represent tags, periods . to represent attributes, and arrows -> to indicate relationship directions. For example: [r:Technology Used], [r:Manufacturer Capability], n.Weight, (n)-[r:Origin Country]->(m)-[r1:Military Branch Involved Project]->(l). Operators such as =, <, >, AND, OR, NOT are used to compare values and filter results. Functions such as count(), min(), max(), avg(), sum() are used to perform calculations on results. Note that if the question is boolean in nature, the CQL's return value is either True or False. For example, question: "Was MQ-4C first flown in 2013?" Return value: True. If the question involves comparing entities and inquiring about an entity, the CQL's return value is the entity name. For example, question: "In comparison to the caliber of the M1 anti-aircraft gun, is the caliber of the M29 81mm mortar smaller?" Corresponding CQL: MATCH (n) where n.name="M29 81mm Mortar" or n.name="M1 Anti-Aircraft Gun" RETURN n.name ORDER BY n.'caliber' asc limit 1 Return value: M29 81mm Mortar. If the question involves comparing entity attribute values, the CQL's return value is True or False. For example, question: "Is the delivery quantity of the ScanEagle UAV greater than that of the MQ-1 Predator UAV?" Corresponding CQL: MATCH (n), (m) where n.name="ScanEagle UAV" and m.name="MQ-1 Predator UAV" RETURN n.'delivery quantity' > m.'delivery quantity' Return value: False. Please independently determine the question type, analyze logical relationships in the question, and generate CQL accordingly.

Appendix 2

Original

已知问题原文为："question"，让我们一步步思考：从原文中可以抽取出：实体：entity，标签：tag，属性：attribute，值：value，关系：relation，条件个数：cn，关系个数：rn，基于抽取结果，可得相应的cql语句为：cql

Translated

Given the original question text: "{question}", let's think step by step: From the original text, we can extract: Entity: {entity}, Tag: {tag}, Attribute: {attribute}, Value: {value}, Relation: {relation}, Condition Count: {cn}, Relation Count: {rn}. Based on the extracted results, the corresponding CQL can be obtained as: {cql}

References

1. Yuanmeng Yan, Rumei Li, Sirui Wang, Hongzhi Zhang, Zan Daoguang, Fuzheng Zhang, Wei Wu, and Weiran Xu. Large-scale relation learning for question answering over knowledge bases with pre-trained language models. In Proceedings of the 2021 conference on empirical methods in natural language processing, pages 3653–3660, 2021.

2. Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. arXiv preprint arXiv:2202.13296, 2022.
3. Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. Knowledge base question answering: A semantic parsing perspective. arXiv preprint arXiv:2209.04994, 2022.
4. Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond iid: three levels of generalization for question answering on knowledge bases. In Proceedings of the Web Conference 2021, pages 3477–3488, 2021.
5. OpenAI. Chatgpt: Optimizing language models for dialogue, 2022.
6. Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.
7. Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. arXiv preprint arXiv:2205.01068, 2022.
8. Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311, 2022.
9. Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021.
10. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
11. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.
12. Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461, 2019.
13. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551, 2020.
14. Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. arXiv preprint arXiv:2206.07682, 2022.
15. Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.
16. 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.

17. Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. Few-shot in-context learning for knowledge base question answering. arXiv preprint arXiv:2305.01750, 2023.