DARA: Decomposition-Alignment-Reasoning Autonomous Language Agent for Question Answering over Knowledge Graphs

Anonymous ACL submission

Abstract

Answering Questions over Knowledge Graphs (KGQA) is key to well-functioning autonomous language agents in various real-life applications. To improve the neural-symbolic reasoning capabilities of language agents powered by Large Language Models (LLMs) in KGQA, we propose the **D**ecomposition-**A**lignment-**R**easoning **A**gent (`DARA`) framework. `DARA` effectively parses questions into formal queries through a dual mechanism: high-level iterative task decomposition and low-level grounding coupled with logical form construction. Importantly, `DARA` can be efficiently trained with a small number of high-quality reasoning trajectories. Our experimental results demonstrate fine-tuning `DARA` on small LLMs (e.g. Llama-2 7B) is not only cost-effective but also yields better performance compared to in-context learning-based agents utilizing the most powerful LLMs available to date, like Llama-2-chat (70B) and GPT-4, across different benchmarks. In addition, `DARA` attains performance comparable to state-of-the-art enumerating-and-ranking-based methods.

1 Introduction

Autonomous agents building on large language models (LLMs) such as AutoGPT [1], WebGPT (Nakano et al., 2021), and ToolFormer (Schick et al., 2023) have exhibited strong capability in planning and reasoning. They have been used to solve complex tasks in various environments involving web content, operating systems, databases, or games, among others. In a wide variety of environments, knowledge graphs (KGs), e.g., Freebase (Bollacker et al., 2008), Wikidata (Vrandecic and Krötzsch, 2014), and DBpedia (Auer et al., 2007), are core to the functionalities of the agents. Previous research, e.g., tool learning with foundation models (Qin et al., 2023) and AgentBench (Liu et al., 2023), has equipped off-the-shelf LLMs (e.g. GPT-4 (OpenAI, 2023)) with multiple functions to conduct Knowledge Graph Question Answering (KGQA) in the In-Context Learning (ICL) setup (Brown et al., 2020). However, the performance of these agents falls significantly short when compared to classical enumerating-and-ranking-based methods (Shu et al., 2022; Gu et al., 2023) in which all possible reasoning paths starting from anchor entities in the question are enumerated and ranked.

In general, ICL-based LLM agents have to heavily rely on extremely large LLMs to attain their performance (Brown et al., 2020; Chowdhery et al., 2023; Ouyang et al., 2022; Touvron et al., 2023a). However, those LLMs are not inherently pre-trained for autonomous agent use cases, which often require complex multi-step reasoning in different environments. Prior studies (Lightman et al., 2023; Dziri et al., 2023) demonstrate that LLMs are prone to produce hallucinations and logical mistakes in multi-step reasoning. Other than that, LLMs are oversensitive to prompting (Zhao et al., 2021; Chen et al., 2022; Lu et al., 2023). Crafting prompts that can guarantee LLM agents understand environments and acquire the needed abilities (e.g. planning or calling correct functions) for performing tasks is challenging (Liu et al., 2023). Beyond technical limitations, the use of commercial LLMs in ICL-based LLM agents, such as ChatGPT [2] and GPT-4 (OpenAI, 2023), raises significant practical concerns regarding privacy, costs, and flexibility of model management. For example, in terms of cost, running AgentBench with GPT-4 over a test set with only ~4,500 examples incurs an expenditure of 1,300 U.S. dollars in stark contrast to ~$30 of the proposed model (see section 5.5 for details.).

To address these challenges, this paper focuses on fine-tuning small models by proposing

---

[1] https://github.com/Significant-Gravitas/AutoGPT

[2] https://openai.com/blog/chatgpt

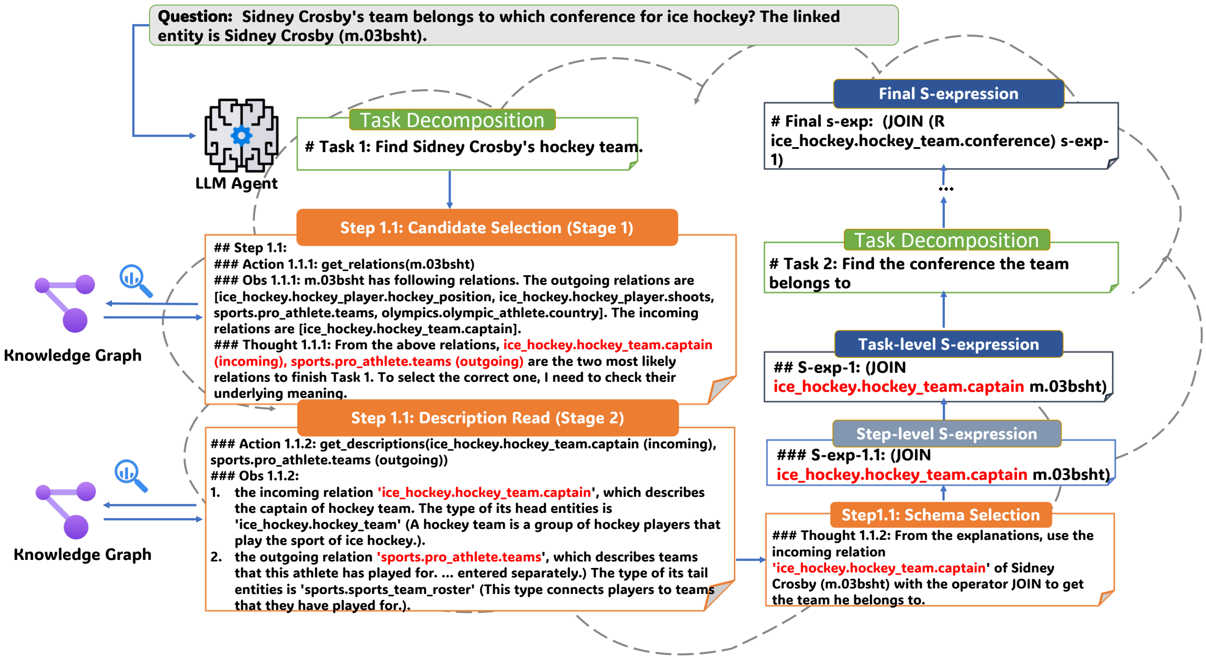
the **D**ecomposition-**A**lignment-**R**easoning **A**gent (DARA) framework (as illustrated in Figure 1). We explore solutions to the above challenges from two key components of autonomous LLM agents. (1) Question understanding: natural language questions show significant variability and diversity, and their meaning may hinge on subtle nuances. In alignment with leveraging the emerging capability of LLMs, DARA is equipped with iterative task decomposition (ITD) to dynamically decompose the question into high-level tasks. Compared to pre-decompositon (Wang et al., 2023c; Sun et al., 2023b) and implicit decomposition (Yao et al., 2023), our experiments show ITD superiority. (2) Schema alignment: KGs feature complex and extensive schemas describing entities and relations. To ground the decomposed task in KGs, the agent must align the intent with the correct relation among massive candidates which may not be seen during training. We propose a reading-to-learn method by reading descriptions of relations to select promising ones. This approach alleviates the coverage and scalability issue of KG-augmented pre-training methods (Liu et al., 2020; Yasunaga et al., 2022). Unlike AgentBench which employs ReAct (Yao et al., 2023), DARA adopts a two-level reasoning structure. At the top level, iterative task decomposition provides guidance, while at the lower level, reading-to-learn relation selection facilitates task completion. This hierarchical structure explicitly disentangles planning and downstream grounding, leading to enhanced performance.

To create high-quality reasoning trajectories for fine-tuning, we convert logical forms to natural languages using a semi-automatic method based on GPT-4 and human verification. Without human verification, our experiments show GPT-4 encounters difficulties in producing reasoning trajectories of the desired quality. To assess the efficacy of DARA, we conduct zero-shot evaluations on three popular benchmark datasets, i.e., WebQSP (Yih et al., 2016), GraphQ (Su et al., 2016), and GrailQA (Gu et al., 2021). Our findings confirm that DARA substantially outperforms ICL-based LLM agents. To conclude, our contributions are:

- We propose a novel LLM-based agent framework, **D**ecomposition-**A**lignment-**R**easoning **A**gent (DARA), for leveraging KGs to answer questions. Fine-tuned DARA proves to be both more cost-effective and powerful than ICL-based LLM agents utilizing GPT-4 and Llama-2-chat (70B). For instance, on 4,559 test examples, DARA-7B takes \$20.51 while the ICL-based agent with GPT-4 incurs \$1,208. In terms of performance, on GrailQA, DARA-7B outperforms ICL-based LLM agents with GPT-4 and Llama-2-chat (70B) by 11.82% and 42.1% (F1-score).
- DARA demonstrates notable efficiency in learning from a limited dataset using iterative task decomposition and reading-to-learn relation selection. Training with 768 reasoning trajectories, we show that fine-tuned DARA can achieve highly competitive performances comparable to enumerating-and-ranking-based models trained on larger data.
- Our experiments show that generating high-quality sequential reasoning trajectories for KGQA is still difficult for GPT-4, in contrast to previous studies that show the success of ChatGPT or GPT-4 in automatic annotation for other tasks (Gilardi et al., 2023; Xu et al., 2023). This emphasizes that high-quality reasoning trajectories are crucial for LLM agents in the KG environment, shedding light on the design of LLM agents in other digital environments such as web browsing and operating systems.

2 Related work

KG-enhanced LLM Reasoning is a popular paradigm to reduce hallucinations and unfaithful reasoning chains of LLMs. In this approach, retrieved triplets from KGs and parametric knowledge within LLMs (i.e. knowledge stored in their parameters) (Petroni et al., 2019; Roberts et al., 2020) work in tandem to derive the final answer to a given question. The Knowledge-Driven Chain-of-Thought framework by Wang et al. (2023a) refines LLM reasoning using an external QA model based on KGs. Meanwhile, Think-on-Graph (ToG) (Sun et al., 2023a) and StructGPT (Jiang et al., 2023) directly utilize off-the-shelf LLMs to traverse over the graph to find the most relevant knowledge and integrate them with parametric knowledge of LLMs to produce the final answer. Despite their potential, these methods face challenges when the parametric knowledge in LLMs is incorrect or outdated. In addition, in scenarios of conflict between parametric knowledge and external non-parametric knowledge, recent research (Qian et al., 2023) reveals that LLMs are susceptible to the distraction of external knowledge, when the latter is irrelevant, leading to



```
Question: Sidney Crosby's team belongs to which conference for ice hockey? The linked entity is Sidney Crosby (m.03bsht).
```

```
# Task 1: Find Sidney Crosby's hockey team.
```

```
## Step 1.1:
### Action 1.1.1: get_relations(m.03bsht)
### Obs 1.1.1: m.03bsht has following relations. The outgoing relations are [ice_hockey.hockey_player.hockey_position, ice_hockey.hockey_player.shoots, sports.pro_athlete.teams, olympics.olympic_athlete.country]. The incoming relations are [ice_hockey.hockey_team.captain].
### Thought 1.1.1: From the above relations, ice_hockey.hockey_team.captain (incoming), sports.pro_athlete.teams (outgoing) are the two most likely relations to finish Task 1. To select the correct one, I need to check their underlying meaning.
```

```
### Action 1.1.2: get_descriptions(ice_hockey.hockey_team.captain (incoming), sports.pro_athlete.teams (outgoing))
### Obs 1.1.2:
1. the incoming relation 'ice_hockey.hockey_team.captain', which describes the captain of hockey team. The type of its head entities is 'ice_hockey.hockey_team' (A hockey team is a group of hockey players that play the sport of ice hockey.).
2. the outgoing relation 'sports.pro_athlete.teams', which describes teams that this athlete has played for. ... entered separately.) The type of its tail entities is 'sports.sports_team_roster' (This type connects players to teams that they have played for.).
```

```
# Final s-exp: (JOIN (R ice_hockey.hockey_team.conference) s-exp-1)
```

```
# Task 2: Find the conference the team belongs to
```

```
## S-exp-1: (JOIN ice_hockey.hockey_team.captain m.03bsht)
```

```
### S-exp-1.1: (JOIN ice_hockey.hockey_team.captain m.03bsht)
```

```
### Thought 1.1.2: From the explanations, use the incoming relation 'ice_hockey.hockey_team.captain' of Sidney Crosby (m.03bsht) with the operator JOIN to get the team he belongs to.
```

Figure 1: In this illustration, (`DARA`) conducts *Task Decomposition* (green blocks), *Grounding* (orange blocks), and *Logical Form Construction* (grey and blue blocks). Starting with the task "Find Sidney Crosby's hockey team.", `DARA` uses the proposed reading-to-learn relation-selection method to select the relation *ice_hockey.hockey_team.captain*, constructing a step-level s-expression (s-exp-1.1). As it is a single-step task, the task-level s-expression is s-exp-1. This iterative process continues until the final s-expression is complete.

worse performance.

LLM-based Autonomous Agents for KGQA. LLM-based autonomous agents take LLMs as the core controllers to plan, reason, and take actions in different environments (Wang et al., 2023b). Unlike KG-enhanced LLM reasoning which focuses on refining reasoning chains using KG, LLM agents are able to conduct more complex tasks such as constructing logical forms for a given question by utilizing a set of human-like capabilities (Sumers et al., 2023) including question decomposition, logical reasoning. Tool learning with foundation models (Qin et al., 2023) and AgentBench (Liu et al., 2023) are the two most relevant works; they design different actions that LLM agents can take. Qin et al. (2023) asked the LLM agent to directly write SPARQL (Standard Protocol and RDF Query Language). In contrast, AgentBench adopts intermediate representation, namely *s-expression* (Gu et al., 2021), to represent SPARQL and construct the *s-expression* step-by-step. However, both of them achieve poor performance due to the limitations of ICL-based agents as discussed in Section 1, and imperfect framework design. `DARA` improves the reasoning abilities of LLM agents in KGQA via fine-tuning and hierarchical reasoning framework design.

Enumerating-and-Ranking-based Methods for KGQA is a prevalent paradigm (Yih et al., 2015; Lan and Jiang, 2020; Luo et al., 2018; Abujabal et al., 2017). To narrow down the large search space of KGs, it first enumerates all possible candidate logical forms from the 2-hop neighborhood of given topic entities or retrieve similar logic forms from training examples. Subsequently, a ranker is applied to select the best one. However, this brute-force approach can lead to exponential candidates and thus suffer from scalability and coverage issues. To alleviate these problems, generation-augmented methods (Ye et al., 2022; Shu et al., 2022) generate diverse logical forms based on retrieved candidates. Besides, dynamic bottom-up semantic parsing approaches (Gu et al., 2023; Gu and Su, 2022) construct the final logical form incrementally and prune the search space on the fly. Although these methods can achieve high performance, they are time-consuming and necessitate expert-crafted rules for logical form construction. In contrast, LLM agents use language as the carrier for explicit planning, grounding, and reasoning, offering enhanced explainability, efficiency, and flexibility. Moreover, LLM agents automate the entire process from task decomposition to relation selection, to logical form construction, eliminating the need for extensive enumeration based on expert-crafted rules.

3 The Approach

3.1 Overview

An overview of `DARA` is described in Algorithm 1. Formally, given a knowledge graph $\mathcal{G}$, a question $\mathcal{Q}$, and a set of actions $\mathcal{A}$, the objective is to construct a logical form $\mathcal{L}$, i.e., *s-expression* (Gu et al., 2021) that yields the final answer to the question $\mathcal{Q}$. To achieve this goal, `DARA` iteratively performs *task decomposition, grounding and logical form construction* until $\mathcal{L}$ is completed[3]. During the iteration i, for decomposed task $\mathcal{T}_i$ (*line 3*), multiple steps $S_{i1}, S_{i2}, ..S_{ij}$ are conducted until $\mathcal{T}_i$ is finished (*line 4-13*). In the step S_{ij} (*line 6-7*), `DARA` grounds the decomposed task $\mathcal{T}_i$ against $\mathcal{G}$ (Section 3.3.2) and constructs the step-level logical form $\mathcal{L}_{ij}$ (Section 3.3.3). When `DARA` determines that $\mathcal{L}_{ij}$ can finish the Task $\mathcal{T}_i$, it will assign $\mathcal{L}_{ij}$ to the task-level logical form $\mathcal{L}_i$ (*line 8-11*).

3.2 The Action Space

The action space $\mathcal{A}$ of `DARA` consists of a set of functions to interact with the KG. For Freebase, the space of $\mathcal{A}$ is as follows:

- $get\_relations(entities, topk)$: This function returns top-k 1-hop relations for given entities. Returned relations include both incoming and outgoing relations, unlike AgentBench which only considers outgoing relations.
- $get\_relevant\_relations(task, topk)$: When there is no entity mentioned in the question, this function returns top-k relevant relations based on the decomposed task.
- $get\_classes(entities, topk)$: This function provides the top-k classes associated with the given entities.
- $get\_relevant\_classes(task, topk)$: This function returns the top-k relevant classes determined by the decomposed task.
- $get\_descriptions(schema)$: For given relations or classes, this function retrieves its descriptions.

Algorithm 1: `DARA`

```
    Input: knowledge graph $\mathcal{G}$, question $\mathcal{Q}$, a
           set of actions $\mathcal{A}$
    Output: Grounded logical form $\mathcal{L}$ of the
            input question $\mathcal{Q}$
 1  while True do
 2      // upper-level guidance
 3      $\mathcal{T}_i \leftarrow$ task_decomposition($\mathcal{T}_{i-1}$);
 4      while True do
 5          // Lower-level grounding
 6          $S_{ij} \leftarrow$ grounding($S_{ij-1}$) ;
 7          $\mathcal{L}_{ij} \leftarrow$
              construct_logical_form($\mathcal{S}_{ij}$) ;
 8          if $\mathcal{T}_i$ is finished then
 9              $\mathcal{L}_i \leftarrow \mathcal{L}_{ij}$ ;
10              break ;
11          end if
12          $j \leftarrow j + 1$ ;
13      end while
14      if No further Task is needed then
15          $\mathcal{L} \leftarrow \mathcal{L}_i$ ;
16          break ;
17      end if
18      $i \leftarrow i + 1$ ;
19  end while
```

To alleviate the burden of schema selection, we adopt the top-k filtering mechanism using a bi-encoder retriever (Reimers and Gurevych, 2019), fine-tuned on schemas in training data.

3.3 The Reasoning Framework

3.3.1 Iterative Task Decomposition

Iterative task decomposition serves as the high-level planner in our framework. Decomposing questions into executable tasks is a fundamental ability in human cognition (Pelletier, 2004). By breaking down complex questions into smaller, more manageable tasks, agents can better grasp the underlying intent and devise an executable plan to find the answers.

When grounding in the KG, decomposing the question upfront into multiple tasks that align well with the KG structure can be challenging. `DARA` hence incorporates and advocates iterative task decomposition (as shown in the Task Decomposition block in Figure 1.). During the process, the agent will dynamically decide if more tasks are needed to finish the final goal. The ablation study in Section 5.2 will emphasize the consistent su-

---

[3] `DARA` will automatically terminate when it finishes the construction of the final logical form $\mathcal{L}$.

periority of iterative task decomposition over the pre-decomposition approach (Wang et al., 2023c; Sun et al., 2023b) for LLM agents in KGs.

3.3.2 Reading-to-learn Relation Selection

To ground the decomposed tasks against KGs, the primary objective is to identify the most relevant relation aligning with the task intent. To enhance the schema understanding ability of foundation models, previous efforts (Liu et al., 2020; Yasunaga et al., 2022) pre-train models on KG-augmented data. However, such methods have limitations in scalability and coverage. It can struggle to fully capture the extensive and diverse topics, entities, and relations present in large-scale KGs (e.g. Freebase has around 44 million topics and 2.4 billion facts.).

To alleviate this issue, `DARA` introduces a two-stage reading-to-learn relation selection method based on the recently emerging powerful natural language understanding ability of LLMs. `DARA` first invokes *get_relations* or $get\_relevant\_relations$ to obtain relations and selects n most likely candidates among them (*Step 1.1: Stage 1* in Figure 1). Subsequently, descriptions of these selected candidates are obtained using the $get\_descriptions$ function (*Step 1.1: Stage 2* in Figure 1). By reading these descriptions, `DARA` selects the most suitable one (*Step 1.1: Schema Selection* in Figure 1). Note that this approach eliminates the need for LLM agents to extensively learn the environment in advance, making it more adaptable and transferable across different scenarios.

3.3.3 Logical Form Construction

As depicted in Figure 1, `DARA` automatically constructs the logical form of the current step based on the selected schema item and the logical form from the previous step. `DARA` learns logical forms syntax and construction via fine-tuning while ICL-based agents have to acquire it via explanation in the prompt. This increases the learning difficulty for some flexible and complex usage of logical operations. For the syntax and complex usage of logical forms (*s-expression*), please refer to Appendix A.3.

3.4 Constructing Reasoning Trajectory Data

As we will demonstrate in Section 5.4, creating high-quality reasoning trajectory data is crucial for training. Specifically, we achieve this by initially filtering <*question, s-expression*> pairs based on GraphQ, WebQSP, and GrailQA. The selection criteria include ensuring relation diversity, removing duplicates, and including complex questions with at least two subtasks.

GrailQA	GraphQ	WebQSP	Total number
426	193	149	768

Table 1: The size of curated training data from different datasets with reasoning trajectories.

	GrailQA	GraphQ	WebQSP
data size	3,274	1,229	56

Table 2: Test data with unseen schemas of each dataset

Golden decomposed tasks are obtained by breaking down *s-expression* based on logical operations (refer to Appendix A.2). To convert these tasks into natural language, they are fed into GPT-4, coupled with the corresponding question using the prompt in Appendix C.1. For the reasoning trajectories of the reading-to-learn relation selection component, we automatically create them using the schema itmes in golden *s-expression* and functions in Section 3.2.

After inspecting and rectifying the above <*question, s-expression*> pairs, and corresponding reasoning trajectories, 768 instances are obtained (Table 1). Further details on data construction are available in Appendix A, with an exemplary reasoning trajectory of `DARA` in C.2.

4 Experiment Setup

One of the preliminary challenges of interacting with KGs is the substantial volume of unseen relations, classes, and entities during training. To simulate the real-world unseen scenario and for a fair comparison, we conduct zero-shot evaluation where schemas in the test data do not appear in training data. For GrailQA, the evaluation is performed on the development data since the golden entity linking results in the hidden test data are unavailable. The resulting statistics of the filtered test data are presented in Table 2.

4.1 Evaluation metrics

We use two evaluation metrics: exact match which evaluates if the predicted and gold logical forms are semantically equivalent (Gu et al., 2021) and F1 score based on the intersection of predicted and gold answers.

4.2 Baselines

We compare DARA with ICL-based LLM agents introduced in AgentBench, which provides seven functions to operate KG interaction and logical reasoning. Commercial GPT-4 and open-source Llama-2-chat (70B) serve as backbone LLMs. To have a comprehensive understanding of the performance of LLM-based agents in KGQA, we also include methods from the bottom-up semantic parsing paradigm. We consider state-of-art models ArcaneQA (Gu and Su, 2022) and Pangu (Gu et al., 2023) as baselines. ArcaneQA employs a constraint decoding method to generate an executable sub-program at each step. Pangu follows the enumeration-then-rank paradigm. At each step, it enumerates all possible subprograms and applies a discriminator to select top-k candidates with the highest scores.

4.3 Implementation Details

We fine-tuned DARA based on Llama 2 (Touvron et al., 2023b) with 4 80GiB H100 and use one 40GiB A100 GPU for inference. The implementation details can be found in Appendix B.

5 Experiment Results

5.1 Overall Performance

In a departure from the current trend of employing ICL-based LLM agents with advanced commercial LLMs, we have observed that fine-tuned DARA with smaller LLMs can achieve much better performance (cf. Table 3). It underscores the efficacy of fine-tuning smaller LLM agents, especially in complex environments like KGs. In particular, DARA-13B has exhibited superior performance across all datasets, surpassing GPT-4 and Llama-2-chat (70B). Notably, on GrailQA, DARA-7B and DARA-13B have outperformed GPT-4 by 11.82% and 14.46% (F1), respectively. In terms of ICL-based agents themselves, GPT-4 demonstrates a significant edge over Llama-2-chat (70B). Llama-2-chat (70B) exhibits several shortcomings, such as format errors, hallucinations, and the misunderstanding of instructions. This suggests that if constraints such as cost, data privacy, controllability, and reproducibility preclude the use of advanced commercial LLMs, fine-tuning smaller open-source LLMs is a highly viable and effective choice.

In comparison to bottom-up parsers, DARA showcases its prowess. DARA-13B outperforms both AcraneQA and Pangu (T5-large) on GraphQ and WebQSP. Notably, DARA-13B outperforms AcraneQA by 19.84% on GraphQ. For GrailQA, while DARA-13B lags behind Pangu (T5-large) by 11.41% (F1-score), it is important to note that Pangu utilized over 57 times more data (44,337 examples) for training, emphasizing the substantial efficiency of DARA in learning from a limited dataset (768 examples). On WebQSP, DARA-7B achieves the best performance, outperforming Pangu (T5-large) by 5.95%. However, an interesting observation is that the performance of all models is much lower than that of the other two datasets. Pangu (T5-large) can achieve a 78.9% F1 score on the whole test data (Gu et al., 2023) while it only achieves a 36.72% F1 score on zero-shot evaluation. Shu and Yu (2023) has a similar observation on cross-dataset evaluation. The main reason is the different data construction methods and limited expressivity of current *s-expression*. We provide detailed analysis in Appendix D.

5.2 Deatailed Analysis on DARA Components

The effectiveness of Iterative Task Decomposition (ITD). We advocate the use of ITD for various environments. As shown in Table 4, when DARA is equipped with pre-decomposition (PD), the performance drops more than 6% on GraphQ, GrailQA and 14.39% on WebQSP. After inspecting the errors made by DARA with PD, we found tasks it produced are often incorrect or redundant. For the question: *Which conference sponsor also sponsored the conference series with GridRepublic?* In the pre-decomposition approach, the decomposed tasks are as follows: *Task 1 - Find the conference series with GridRepublic, Task 2 - Find the sponsor of the conference series with GridRepublic, and Task 3 - Find the conferences sponsored by the same sponsor.* in which *Task 3* is unnecessary for answering the question. The ITD method, on the other hand, would stop the iteration at Task 2 and return the answer. These results underscore the importance of ITD for DARA, as it allows for dynamic and context-aware task decomposition, leading to improved performance across various datasets.

The role of the reading-to-learn relation selection method. The results presented in Table 4 demonstrate the significance of the reading-to-learn relation selection strategy. When this strategy is not employed, the performance experiences a decrease, especially in GraphQ (5.95% drop). To

Model	GraphQ		GrailQA		WebQSP	
	EM	**F1**	**EM**	**F1**	**EM**	**F1**
Off-the-shelf LLM Agent (in context learning)						
AgentBench (Llama-2-chat-70B)	25.63	30.33	33.20	35.72	8.93	10.18
AgentBench (GPT-4)	53.86	64.48	63.56	65.89	25.00	32.09
Fine-tuned LLM Agent (768 examples)						
`DARA-7B`	51.51	62.74	75.05	77.71	**30.36**	**42.67**
`DARA-13B`	**55.57**	**67.34**	77.03	80.35	**30.36**	41.63
Bottom-up Parser (full train data)						
AcraneQA	37.00	47.50	78.52	81.81	23.21	37.80
Pangu (T5-base)	56.06	66.70	**88.30**	**91.76**	19.64	32.64
Pangu (T5-Large)	**55.57**	67.21	$-^*$	$-^*$	23.21	36.72

Table 3: Overall zero-shot evaluation results in three different datasets. * indicates the trained T5-large on GrailQA is not provided in the GitHub repository of Pangu. Trained T5-3B models are unavailable as well. All models use the golden entity linker. The full train data size of each dataset can be found in Appendix 6.

	GraphQ	**GrailQA**	**WebQSP**
`DARA-7B`	62.74	77.71	42.67
w/ PD	56.50 (**6.24** ↓)	71.22 (**6.49** ↓)	28.28 (**14.39** ↓)
w/o RTL	56.79 (5.95 ↓)	75.12 (2.59 ↓)	40.77 (1.90 ↓)
w/o FT Retriever	61.29 (1.45 ↓)	75.76 (1.95 ↓)	36.18 (6.49 ↓)

Table 4: Ablation study of different components of `DARA`. The metric used is the F1 score. PD represents task pre-decomposition. RTL represents the reading-to-learn relation selection method. Without the FT (fine-tuned) retriever, an off-the-shelf retriever all-mpnet-base-v2 is used.

illustrate the impact, for the question: *What vocal range is Pavarotti?* In the absence of the reading-to-learn selection strategy, `DARA` selects the incorrect relation, *music.artist.track*. However, with the selection strategy in place, `DARA` chooses the correct relation, *music.opera_singer.voice_type*, whose description contains information about the vocal range. These results highlight the critical role that the reading-to-learn relation selection strategy plays in enhancing `DARA`'s performance especially when the literal meaning of relations cannot express the needed information.

The fine-tuned retriever for relation filtering. To help `DARA` better select the relations, we leverage a fine-tuned retriever to filter the top five candidate relations for the `DARA` (Section 3.2). As shown in Table 4, the fined-tuned retriever contributes to the improvement in model performance. On WebQSP, without the trained retriever, the performance drops by 6.49%.

5.3 Error Analysis

To have a concrete understanding of the utility of `DARA`, we compare its reasoning trajectories with those of ICL-based LLM agents. For ICL-based LLM agents, the majority of errors happen during grounding. Without accessing descriptions of schemas, ICL-based agents are prone to misunderstand relations or prematurely conclude the task especially when it reaches a compound value type (CVT) node with n-ary relations[4]. In contrast, `DARA` effectively identifies the CVT node through its reading-to-learn relation selection component. Apart from this, limitations in the design of ICL-based agents within AgentBench become apparent in addressing complex tasks. These agents lack functionalities for questions involving class information or incoming relations of given entities, and they struggle with questions lacking explicit entities. A thoughtful design of action space is crucial for LLM agents to solve various complex questions. The third major issue of ICL-based LLM agents is related to the instruction following. Llama-2-chat (70B) encounters challenges in following the reasoning pipeline provided in the demonstration, leading to failure function calls. While GPT-4 performs better in demonstration understanding, it struggles to halt automatically even when it reaches the trigger phrase for the final answer. This leads to continued generation beyond the necessary point,

---

[4] CVT is the mediator in Freebase so the agent needs a further step to get the final answer.

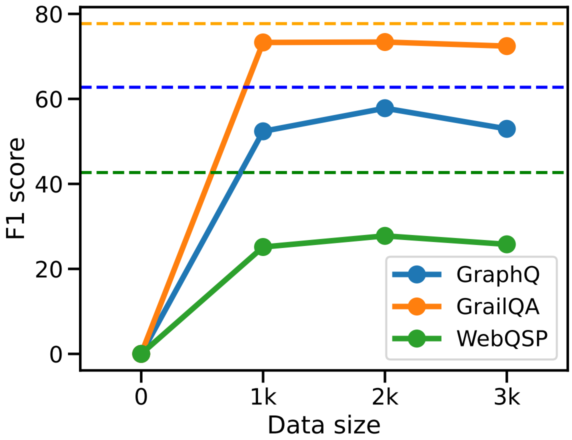


Figure 2: The performance of `DARA-7B` on three datasets trained with reasoning trajectories generated by GPT-4. - - - (dashed line) represents the performance of 768 human-rectified examples.

often resulting in repetitive or hallucinatory content. Comparison examples between the three agents are provided in Appendix E.

5.4 The Quality of Reasoning Trajectories

The utilization of the GPT series for data generation or annotation has gained popularity in recent research (Gilardi et al., 2023; Wang et al., 2023d; Xu et al., 2023). Gilardi et al. (2023) show ChatGPT outperforms crowd-workers for several classification tasks. We investigate if reasoning trajectories generated by GPT-4 could effectively serve as training data for our model without human verification. We generate trajectories from GPT-4 using the prompt in C.1 and retain 3,000 trajectories after filtering the wrong formats.

As illustrated in Figure 2, the performance of `DARA-7B` trained on GPT-4 generated data on three datasets lags behind that of 768 human-rectified examples. Upon closer inspection of the generated data, we identified the following issues. GPT-4 struggles to follow the output format, i.e., confusing 'step' and 'task'. Second, it misinterprets the intent of the given sequential logical form, generating tasks that deviate from the specified goal. Furthermore, the special schema design of Freebase (e.g. CVT) increases the difficulty of comprehension. This aligns with the findings of LIMA (Zhou et al., 2023), and Platypus (Lee et al., 2023), both of which emphasize that the quality and diversity of datasets outweigh data size when fine-tuning LLMs.

Model	Cost	
	Money	Time
AgentBench (Llama-2-chat (70B))	$1,208	**120.07 hours**
AgentBench (GPT-4)	**$1,276**	32.07 hours
`DARA-7B`	$20.51	5.01 hours
`DARA-13B`	$31.90	7.72 hours

Table 5: The total cost of different models on 4,559 examples during inference.

5.5 The Running Expense of Models

In Table 5, we calculate the cost of different models on 4,559 test examples during inference in terms of expenses and time. Overall, `DARA` is significantly cheaper and faster than ICL-based LLM agents with GPT-4 and Llama-2-Chat (70B). GPT-4 is the most expensive among them with a cost that is sixty times higher and a speed four times slower than that of `DARA-7B`. The pricing of the GPT-4 experiment is available on the Auzre OpenAI service webpage [5]. To assess the price of GPUs used for Llama-2, we refer to Replicate.com [6]. While the cost may be affected by different factors (e.g. GPU service providers, inference optimization methods), the distinct advantage of `DARA` in terms of affordability and efficiency remains evident.

6 Conclusion

In this paper, we present `DARA`, a fine-tuned LLM agent designed for question answering over knowledge graphs. `DARA` autonomously translates a given question into an executable query through iterative task decomposition and a reading-to-learn-based grounding. Experimental results highlight the superior performance of `DARA`, even when fine-tuned on a smaller LLM and a limited number of examples, compared to in-context learning LLM agents with GPT-4 and Llama-2-chat-70B. Additionally, `DARA` proves to be more cost-effective and time-efficient. However, automatically generating high-quality reasoning trajectories is still difficult. In the future, we will center on developing methods to generate high-quality reasoning trajectories for LLM agent fine-tuning in KG and other symbolic environments.

---

[5] `https://azure.microsoft.com/en-us/pricing/details/cognitive-services/openai-service/`

[6] `https://replicate.com/pricing`

7 Limitations

Although `DARA` shows superiority over ICL-based LLM agents, there are several limitations that call for further improvement. First, the process of training `DARA` relies on acquiring high-quality reasoning trajectories. This is currently achieved by prompting GPT-4 and subsequently verifying the trajectories through human validation. How to get trajectories automatically and scalably for LLM agents in various environments is a promising direction for future research.

Second, DARA lacks the error-correcting ability, which means it cannot correct itself when erroneously decomposing the question or selecting incorrect relations. Humans excel in fixing errors based on the current observation and the final goal. Exploring approaches like Reflextion (Shinn et al., 2023) could enhance error-correcting capabilities, although it is important to note that such methods currently work effectively only with very powerful LLMs, such as GPT-4.

Third, while DARA is a generic framework for different knowledge graphs, we only evaluate it on popular datasets using Freebase as the backend. For other KGs like Wikidata, there is a scarcity of high-quality data which contains both questions and golden logical forms. Therefore, how to employ weak supervision (only the final answer is available) for LLM agents reasoning in KG is an interesting direction.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1191–1200, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer.

Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM.

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. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yanda Chen, Chen Zhao, Zhou Yu, Kathleen R. McKeown, and He He. 2022. On the relation between sensitivity and accuracy in in-context learning. *CoRR*, abs/2209.07661.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaïd Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. *CoRR*, abs/2305.18654.

F. Gilardi, M. Alizadeh, and M. Kubli. 2023. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120.

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings* *of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4928–4949, Toronto, Canada. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488, New York, NY, USA. Association for Computing Machinery.

Yu Gu and Yu Su. 2022. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *CoRR*, abs/2305.09645.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974, Online. Association for Computational Linguistics.

Ariel N. Lee, Cole J. Hunter, and Nataniel Ruiz. 2023. Platypus: Quick, cheap, and powerful refinement of llms. *CoRR*, abs/2308.07317.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *CoRR*, abs/2305.20050.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: enabling language representation with knowledge graph. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2901–2908. AAAI Press.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, et al. 2023. Agentbench: Evaluating llms as agents. *CoRR*, abs/2308.03688.

Sheng Lu, Hendrik Schuff, and Iryna Gurevych. 2023. How are prompts different in terms of sensitivity? *CoRR*, abs/2311.07230.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2185–2194, Brussels, Belgium. Association for Computational Linguistics.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

R OpenAI. 2023. Gpt-4 technical report. *arXiv*, pages 2303–08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Francis Jeffry Pelletier. 2004. The Principle of Semantic Compositionality. In *Semantics: A Reader*. Oxford University Press.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Cheng Qian, Xinran Zhao, and Sherry Tongshuang Wu. 2023. "merge conflicts!" exploring the impacts of external distractors to parametric knowledge graphs. *CoRR*, abs/2309.08594.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, et al. 2023. Tool learning with foundation models. *CoRR*, abs/2304.08354.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *CoRR*, abs/2302.04761.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Yiheng Shu and Zhiwei Yu. 2023. Data distribution bottlenecks in grounding language models to knowledge bases. *CoRR*, abs/2309.08345.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: Multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8108–8121, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for QA evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572, Austin, Texas. Association for Computational Linguistics.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2023. Cognitive architectures for language agents. *CoRR*, abs/2309.02427.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023a. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *CoRR*, abs/2307.07697.

Simeng Sun, Yang Liu, Shuohang Wang, Chenguang Zhu, and Mohit Iyyer. 2023b. PEARL: prompting large language models to plan and execute actions over long documents. *CoRR*, abs/2305.14564.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023a. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *CoRR*, abs/2308.13259.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2023b. A survey on large language model based autonomous agents. *CoRR*, abs/2308.11432.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023c. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023d. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *CoRR*, abs/2304.12244.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D. Manning, Percy Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. In *NeurIPS*.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: less is more for alignment. *CoRR*, abs/2305.11206.

A Data Creation

A.1 Dataset Quality Inspection

Fine-tuning LLMs with large but very noisy data contributes little to performance improvement (Zhou et al., 2023; Lee et al., 2023). We manually inspect selected data from the original dataset. Several issues are identified: unnatural expressions, redundant logical forms, ambiguous questions, and inconsistencies between questions and logical forms. GrailQA is a large-scale dataset aiming at evaluating different level generalization abilities of KGQA models. Among 500 sampled data points, roughly 15% contained errors. The main issue is the wrong directionality of relations. In the case where the question is asking about *operating systems that include 386bsd*, the logical form is searching for operating systems belonging to 386sd *(JOIN computer.operating_system.parent_os 386BSD)*. Another common issue is improper comparative operations. For instance, the question is asking about the comparison *more than* while *ge (greater than or equal to)* is adapted in logical forms. Besides, some errors involve redundant components within the query, which is unnecessary to answer the question. GraphQ is a medium-sized dataset where questions are generated by expert annotators according to the query graph. It has multiple paraphrased questions in the same logical form. We only retain one question for each logical form. WebQSP is another medium-sized dataset comprising questions sourced from Google query logs. This dataset includes implicit entities not mentioned in the questions and complex logical operations. Furthermore, it contains many ambiguous/open questions. For example, *what did Stephen Hawking become famous for?* is an open question. The answer could be his profession, book, discovery, etc. The data statistics of these three datasets are shown in Table 6.

	GraphQ	**GrailQA**	**WebQSP**
Training	2,381	44,337	3,098
Dev	-	6,763	-
Test	2,395	13,231	1,639

Table 6: Original Dataset Statistics

A.2 Logical Operations

The common logical operations are as follows:

- Superlative (ARGMIN/ARGMAX): it is used to get entities with the maximum value of a given attribute.
- Intersection (AND): it is used to get an intersection between two sets of entities.
- Comparative (LT/LE/GE/GE): it is used to filter properties according to numerical constraints.

Logical Operation	Returns	Descriptions
(AND u1 u2)	a set of entities	AND function returns the intersection of two arguments
(COUNT u)	a singleton set of integer	COUNT function returns the cardinality of the argument
(R b)	a set of (entity entity) tuples	R function reverse each binary tuple (x, y) in the input to (y, x)
(JOIN b u)	a set of entities	Inner join based on items in u and the second element of items in b
(JOIN b1 b2)	a set of (entity, entity) tuples	Inner join based on the first element of items in b2 and the second element of items in b1
(ARGMAX/ARGMIN u b)	a set of entities	Return x in u such that (x, y) is in b and y is the largest / smallest
(LT/LE/GT/GE b n)	a set of entities	Return all x such that (x, v) in b and v </ / >/ n

Table 7: The definition and syntax of *s-expression*

- Count (COUNT): it is used to count the number of a set of entities.
- Projection (JOIN): This operation is used to get the other side entity of a triplet via the relation.

We break down the logical form into subtasks according to the above logical operations. For projection operation, it may be used due to the KG structure rather than the question. For such cases, we do not break it down.

A.3 Logical form used in `DARA`

We use s-expression (Gu et al., 2021) as the logical form to represent SPARQL. The definition and syntax of logical operations are as shown in Table 7.

B Implementation Details

Training We fine-tuned Llama-2 7B/13B using 4 80GiB H100 for 10 epochs using 2e-5 learning rate with 0.03 warm-up ratio and 8 batch size. Deepspeed is utilized to accelerate the training process.

Inference To do inference on Llama 7B/13B, we use one 40GiB A100 GPU while 2 80GiB A180 GPUs are used to test Llama-2-chat-70B. We call Azure OpenAI service API for GPT-4 inference.

C Prompt and reasoning trajectories

C.1 Prompt used to convert logical forms to natural languages

We present the prompt and demonstrations used to convert broken-down logical forms into natural languages in Table 8.

C.2 A exemplary reasoning trajectory of `DARA`

As shown in Table 9, we provide the complete reasoning trajectory of `DARA`, which includes iterative task decomposition and the two-stage reading-to-learn relation selection method.

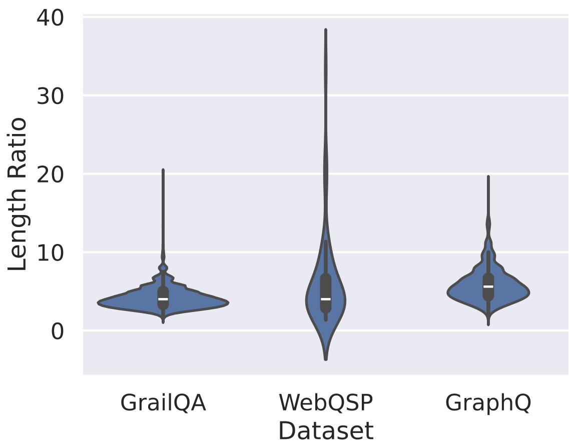


Figure 3: Ratio between tokens of SPARQL and tokens of questions in three test datasets

D Why is there a performance discrepancy between WebQSP and GrailQA, GraphQ?

One significant factor lies in the different sources of the questions. Questions of GrailQA and GraphQ are derived from logical forms, which are structured and explicitly aligned with underlying KGs. Conversely, WebQSP collects questions from real users' query logs on Google search, often containing common sense or complex logical operations. Intuitively, when the SPARQL is longer but the question is short, it often signifies a requirement for implicit knowledge or complex reasoning. To explore this hypothesis, we simply calculate the ratio between the lengths of SPARQL queries and questions, visualizing it in a violin plot as shown in Figure 3. Compared with GrailQA and GraphQ, we observe that WebQSP exhibits more extreme cases in the length ratio, suggesting that it contains more short questions with long SPARQL queries. Some of them (cf. Table 10) include complex logical constraints (e.g. time comparison) that cannot be represented by s-expression. Besides, Humans can formulate concise and abstract questions based on common sense, which poses a challenge for parsing questions in the absence of such knowledge or an ability to leverage it. For example, to answer the question *what year did Seattle Seahawks go to the Super Bowl?* The model needs to know Superbowl serves as the final match of the American National Football League. Therefore, it needs to find the season when the Seattle Seahawks were not only the champion but also the runner-up. Those real questions from humans can be more flexible, and

User: You are knowledgeable about Freebase. You need to describe what is the intent of logical forms in the given subtasks and steps. To find the answer to the question 'Which type of dish common in manado cuisine contains sugars? The linked entities are sugars (m.06x4c), and manado cuisine (m.0102k5v9).', it has the following subtasks. Note: Some tasks may have multiple steps due to the schemas of the KG.
Task 1: Step1:(JOIN food.dish.ingredients m.06x4c)
Task 2: Step1:(JOIN (R dining.cuisine.dishes) m.0102k5v9)
Task 3: Step1:(AND task1 task2)
Task 4: Step1:(JOIN food.type_of_dish.dishes task3)
To help you understand the relations used in the above steps, we provide the following descriptions of them. 1. the incoming relation 'food.dish.ingredients', which describes please enter the main ingredients usually included in this dish. The type of its head entity is 'food.dish' (A dish is a food prepared and presented in a certain way. These are foods that are ready to be served and eaten as a meal itself, or part of a larger meal). 2. the outgoing relation 'dining.cuisine.dishes', which describes the dishes of cuisine. The type of its tail entity is 'food.dish' (A dish is a food prepared and presented in a certain way. These are foods that are ready to be served and eaten as a meal itself, or part of a larger meal). 3. the incoming relation 'food.type_of_dish.dishes', which describes the dishes of type of dish. The type of its head entity is 'food.type_of_dish' (This type is used to classify dishes into broad categories, such as appetizers, soups, desserts, etc.). Please describe the intent of the above tasks and steps.
Assistant: To solve this question, we can break it down into the following tasks:
Task 1: Determine the dishes that contain sugars.
- Step 1.1: use the incoming relation 'food.dish.ingredients' of surgars with the operator JOIN to find dishes whose ingredients contain sugars.
Task 2: Identify the dishes in Manado cuisine.
- Step 2.1: we can use the outgoing relation 'dining.cuisine.dishes' of manado cuisine to find the dishes in Manado cuisine.
Task 3: Find the common dishes from Task 1 and Task 2.
- Step 3.1: we can use the operator AND to get the common dishes from Task 1 and Task 2.
Task 4: Determine the type of dish among the common dishes.
- Step 4.1: to get the type of dish, we use the the incoming relation 'food.type_of_dish.dishes' of those entities obtained from Task 3 to get the final answer.
User: To find the answer to the question 'which academic was an dashiell that was born in japan? The linked entities are Japan (m.03_3d), dashiell (m.03m3mgq).', it has following tasks.
Each task may have multiple steps due to the schemas of the KG.
Task 1: Step1:(JOIN people.person.place_of_birth m.03_3d)
...
Please describe the intent of the above tasks and steps.
Assistant: To solve this question, we can break it down into following tasks:
Task1: find people who were born in Japan.
...
User: To find the answer to the question '{question}', it has following tasks.
{subtasks}
To help you understand the relations used in above steps, we provide the following descriptions of them.
{description}
Please describe the intent of above tasks and steps.

Table 8: The prompt used to convert linearized logical forms to natural languages.

User: For a given question, your task is to parse the question into a correct logical form (s-expression) which could be executed over a KG to get the answer. To write the s-expression, you need to decompose the question into subtasks and solve them step-by-step. To get necessary schemas (i.e., relations or classes) for s-expression writing, you can use the following functions to interact with the KG.
get_relations(expression): return all relations of those entities represented by the expression.
get_classes(expression): return all class types of those entities represented by the expression.
get_relevant_relations(thought): retrieve relevant relations from the KG according to your thought.
get_relevant_classes(thought): retrieve relevant classes from the KG according to your thought.
get_descriptions(candidate): get description of candidate schemas. The question is what olympic games did ronny represent as a mascot? The linked entity is Ronny (m.04dwjbg).
Agent: The given question can be decomposed into the following subtasks: # Task 1: Find the Olympic games that Ronny represented as a mascot.
## Step 1.1:
### Action 1.1.1: get_relations(m.04dwjbg)
### Obs 1.1.1: m.04dwjbg has following relations. The outgoing relations are [olympics.olympic_mascot.olympic_games, kg.object_profile.prominent_type, common.topic.notable_for, common.topic.notable_types, type.object.name]. The incoming relations are [].
### Thought 1.1.1: From the above relations, olympics.olympic_mascot.olympic_games (outgoing), kg.object_profile.prominent_type (outgoing) are the two most likely relations to finish Task 1. To select the correct one, I need to check their underlying meaning.
### Action 1.1.2: get_descriptions(olympics.olympic_mascot.olympic_games (outgoing), kg.object_profile.prominent_type (outgoing))
### Obs 1.1.2: 1. the outgoing relation 'olympics.olympic_mascot.olympic_games', which describes the Olympic games for which this was a mascot. The type of its tail entities is 'olympics.olympic_games' (This type is for any modern Olympic Games, summer or winter.). 2. the outgoing relation 'kg.object_profile.prominent_type', which describes the prominent type of object profile. The type of its tail entities is 'prominent_type' (prominent type).
### Thought 1.1.2: From the explanations, use the outgoing relation 'olympics.olympic_mascot.olympic_games' of Ronny with the operator JOIN to find the Olympic games that Ronny represented as a mascot.
### S-exp-1.1: (JOIN (R olympics.olympic_mascot.olympic_games) m.04dwjbg)
## S-exp-1: (JOIN (R olympics.olympic_mascot.olympic_games) m.04dwjbg)
# Final s-exp: (JOIN (R olympics.olympic_mascot.olympic_games) m.04dwjbg)

Table 9: An exemplary reasoning trajectory of `DARA`.

obscurer than synthetic questions derived from logical forms. In light of these findings, it is evident that advancing the field of knowledge graph reasoning requires the design of more realistic datasets, the creation of more expressive intermediate representations, and the exploration of new model paradigms that can effectively harness commonsense knowledge. but sometimes such synthetic questions lack clarity and authenticity, which may impede the LLM agent's understanding. For example, *which industry is ayala land in which real estate and rental and leasing is also in?* is ambiguous and unnatural.

Question	what did james k polk do before he was president?

SPARQL

```
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT DISTINCT ?x
WHERE {
{
SELECT ?pFrom
WHERE {
ns:m.042f1 ns:government.politician.government_positions_held ?y .
?y ns:government.government_position_held.office_position_or_title ?x ;
ns:government.government_position_held.basic_title ns:m.060c4 ; # President
ns:government.government_position_held.from ?pFrom .
}
}
ns:m.042f1 ns:government.politician.government_positions_held ?y . # James K. Polk
?y ns:government.government_position_held.office_position_or_title ?x ;
ns:government.government_position_held.from ?from .

FILTER(xsd:dateTime(?pFrom) - xsd:dateTime(?from) >0)
}
```

Question	what super bowl did peyton manning win?

SPARQL

```
SELECT DISTINCT ?z
WHERE {
ns:m.027jv8 ns:sports.pro_athlete.teams ?y .
?y ns:sports.sports_team_roster.team ?x . #team
?x ns:sports.sports_team.championships ?z . # super bowls
?z ns:common.topic.notable_types ns:m.01xljv1 . #super bowl
?z ns:sports.sports_championship_event.champion ?c .
?z ns:time.event.start_date ?a .

# Check the time overlap
FILTER(NOT EXISTS {?y ns:sports.sports_team_roster.from ?sk0} ||
EXISTS {?y ns:sports.sports_team_roster.from ?sk1 .
FILTER(xsd:datetime(?sk1) - xsd:datetime(?a) <= 0) })

FILTER(NOT EXISTS {?y ns:sports.sports_team_roster.to ?sk2} ||
EXISTS {?y ns:sports.sports_team_roster.to ?sk3 .
FILTER(xsd:datetime(?sk3) - xsd:datetime(?a) >= 0 ) })
}
```

Question	which country in north america is divided into provinces?

SPARQL

```
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT DISTINCT ?x
WHERE {
ns:m.059g4 ns:location.location.contains ?x . # North America
?x ns:common.topic.notable_types ns:m.01mp . # Country
?x ns:location.location.contains ?y .
?y ns:common.topic.notable_types ?t .
# All the possible "province" type
FILTER ((?t = ns:m.01nm) ||
(?t = ns:m.02_1y_9) ||
(?t = ns:m.02_3ny_) ||
(?t = ns:m.02_3phk) ||
(?t = ns:m.02_3r2r) ||
(?t = ns:m.02_3rt3) ||
(?t = ns:m.02_3zf4) ||
(?t = ns:m.02_40h1) ||
(?t = ns:m.02_96lm) ||
(?t = ns:m.02yxk5c) ||
(?t = ns:m.02zd6yn) ||
(?t = ns:m.03z96kq) ||
(?t = ns:m.04g7rg9) ||
(?t = ns:m.04js0h5) ||
(?t = ns:m.065rjpr) ||
(?t = ns:m.078_8dm) ||
(?t = ns:m.0hzcb3l) ||
(?t = ns:m.0hzcb5p) ||
(?t = ns:m.0hzcb69) ||
(?t = ns:m.0hzcb7p) ||
(?t = ns:m.0hzcd76) ||
(?t = ns:m.0hzcd7v) ||
```

Question	who was president after franklin d. roosevelt?

SPARQL

```
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT DISTINCT ?x
WHERE {
# President of the United States
ns:m.060d2 ns:government.government_office_or_title.office_holders ?y1 .
# Franklin D. Roosevelt
?y1 ns:government.government_position_held.office_holder ns:m.02yy8 ;
ns:government.government_position_held.to ?to .
# President of the United States
ns:m.060d2 ns:government.government_office_or_title.office_holders ?y2 .
?y2 ns:government.government_position_held.office_holder ?x ;
ns:government.government_position_held.to ?from .

FILTER(xsd:dateTime(?from) - xsd:dateTime(?to) >0)
}
ORDER BY xsd:dateTime(?from)
LIMIT 1
```

Question	who is meredith gray married to in real life?

SPARQL

```
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT DISTINCT ?x
WHERE {
ns:m.0268x2f ns:tv.tv_character.appeared_in_tv_program ?y .
?y ns:tv.regular_tv_appearance.actor ?z .
?z ns:people.person.spouse_s ?s .
?s ns:people.marriage.spouse ?x .
?s ns:people.marriage.type_of_union ns:m.04ztj .
Filter (?x != ?z)
FILTER(NOT EXISTS {?s ns:people.marriage.from ?sk0} ||
EXISTS {?s ns:people.marriage.from ?sk1 .
FILTER(xsd:datetime(?sk1) <= "2015-08-10"^^xsd:dateTime) })
FILTER(NOT EXISTS {?s ns:people.marriage.to ?sk2} ||
EXISTS {?s ns:people.marriage.to ?sk3 .
FILTER(xsd:datetime(?sk3) >= "2015-08-10"^^xsd:dateTime) })
?s ns:people.marriage.from ?sk4 .
}
ORDER BY DESC(xsd:datetime(?sk4))
LIMIT 1
```

Question what was franklin d roosevelt ’s job before president?

SPARQL

```
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT DISTINCT ?x
WHERE {
{
SELECT ?pFrom
WHERE {
ns:m.02yy8 ns:government.politician.government_positions_held ?y . # Franklin D. Roosevelt
?y ns:government.government_position_held.office_position_or_title ?x ;
ns:government.government_position_held.basic_title ns:m.060c4 ; # President
ns:government.government_position_held.from ?pFrom .
}
}
ns:m.02yy8 ns:government.politician.government_positions_held ?y . # Franklin D. Roosevelt
?y ns:government.government_position_held.office_position_or_title ?x ;
ns:government.government_position_held.from ?from .

FILTER(xsd:dateTime(?pFrom) - xsd:dateTime(?from) >0
```

Table 10: Questions in unseen schemas test data of WebQSP with unseen schemas whose length ratio is larger than 10.

E Error Analysis

As discussed in section 5.3, we compared `DARA` with ICL-based agents with GPT4, Llama-2-chat (70B). We show three examples in Table 11 $\sim$ Table 13.

Question	guitar hero iii has what songs?
ICL Agent	**Llama-2-chat (70B)**: Thought: To find songs that can be played in Guitar Hero 3, we can use the knowledge base to find relations connected to the entity ’Guitar Hero 3’. ... Thought: From the above relations, I might use cvg.musical_game.songs to find the songs in Guitar Hero 3. Action: get_neighbors(Guitar Hero 3, cvg.musical_game.songs) Observation: variable #0, which are instances of cvg.musical_game_song_relationship Thought: I can use intersection() to find the songs that are in Guitar Hero 3. Action: intersection(#0, Guitar Hero 3) ... **GPT-4**: Thought: I need to find songs that can be played in ’guitar hero 3’. ... Thought: The relation ’cvg.musical_game.songs’ seems to be the most relevant to find the songs in ’guitar hero 3’. Action: get_neighbors(guitar hero 3, cvg.musical_game.songs) Observation : variable #0, which are instances of cvg.musical_game_song_relationship Thought: The variable #0 should be the final answer to the question as it represents the songs that can be played in Guitar Hero 3.
DARA	The given question can be decomposed into the following subtasks: # Task 1: Find the songs that can be played in Guitar Hero 3. ... ### Thought 1.1.2: From the explanations, we can use the outgoing relation ’cvg.musical_game.songs’ of Guitar Hero 3 with the operator JOIN to reach the intermediate (CVT) node first, which contains all related information of songs in the game. ... Thought 1.2.2: From the explanations, we can use the outgoing relation ’cvg.musical_game_song_relationship.song’ of the CVT node from step 1.1 with JOIN to find the songs that can be played in the game. ...
Explanation	ICL-based agents prematurely stop at the CVT node while DARA gets the correct answer.

Table 11: The incorrect reasoning trajectory is highlighted in red while the correct one is in blue.

Question	snk developed how many video game compilations?
ICL Agent	**Llama-2-chat (70B)**: Thought: To find the number of video game compilations developed by SNK, I need to first find the games developed by SNK. To achieve this, I will query the KB to find relations connected to the entity 'SNK' and see if any of them can help me find the answer. ... Thought: From the above relations, I might use cvg.cvg_developer.game_versions_developed to find the games developed by SNK. ... Thought: Now, I need to find the number of unique game compilations developed by SNK. To achieve this, I can use the intersection function to find the intersection of all the games developed by SNK. Action: intersection(#0) **GPT-4**: Thought: I need to find out the number of video game compilations developed by snk. To achieve this, I will query the KB to find relations connected to the entity 'snk' and see if any of them can help me find the answer. Thought: From the above relations, I can use cvg.cvg_developer.games_developed to find video games developed by snk. ... Thought: Now, I need to find out which of these games are compilations. To achieve this, I will query the KB to find relations connected to the variable #0. Action: get_relations(#0) ... Thought: From the above relations, I might use cvg.computer_game_compilation.games_included to find which of these games are compilations.
DARA	The given question can be decomposed into the following subtasks: # Task 1: Find the video game compilations developed by SNK. ... Thought 1.1.2: From the explanations, use the outgoing relation 'cvg.cvg_developer.games_developed' of SNK with the operator JOIN to find the video game compilations developed by SNK. ... ### Action 1.2.1: get_classes(s-exp-1.1) ... ### Thought 1.2.1: From classes in Obs 1.2.1, use the operator AND with the class 'cvg.computer_game_compilation' to filter out the entities that are not video game compilations. ### S-exp-1.2: (AND cvg.computer_game_compilation s-exp-1.1)
Explanation	ICL-based agents are incapable of getting class information due to the lack of functions in the action space while DARA calls the correct function.

Table 12: The incorrect reasoning trajectory is highlighted in red while the correct one is in blue.

Question	talk of the nation appears in what radio program episode?
ICL Agent	**Llama-2-chat (70B)**: Thought: To find the answer, I need to first find radio programs related to ’Talk of the Nation’. I will query the KB to find relations connected to the entity ’Talk of the Nation’. ... Thought: From the above relations, I might use radio.radio_program.episodes to find the radio program episodes related to ’Talk of the Nation’. ... Thought: Now I have the episodes, I need to find the specific episode that is related to the question. To achieve this, I will query the KB to find relations connected to the variable #0. ... **GPT-4**: Thought: To find the radio program episodes where ’Talk of the Nation’ appears, I need to first find the relations connected to the entity ’Talk of the Nation’ ... Action: get_neighbors(talk of the nation, radio.radio_program.episodes) Observation: variable #0, which are instances of radio.radio_program_episode Thought: The variable #0 represents the radio program episodes where ’Talk of the Nation’ appears, which is the answer to the question. Final Answer: #0 Thought: To find the radio program episodes where ’Talk of the Nation’ appears, I need to first find the relations connected to the entity... ... Thought: From the above relations, I might use cvg.computer_game_compilation.games_included to find which of these games are compilations.
DARA	The given question can be decomposed into the following subtasks: # Task 1: Find the radio program episode that contains the T̈alk of the Nations̈egment. ... Thought 1.1.2: From the explanations, use the outgoing relation ’cvg.cvg_developer.games_developed’ of SNK with the operator JOIN to find the video game compilations developed by SNK. ... ### Thought 1.2.1: From the explanations, use the outgoing relation ’radio.radio_program.episodes’ of "Talk of the Nation" with the operator JOIN to find the radio program episode that contains the "Talk of the Nation" segment. ... #Final s-exp: (JOIN (R radio.radio_program.episodes) m.017dgp)
Explanation	Llama-chat call incorrect functions for further reasoning. While GPT-4 finds the correct answer, it cannot stop at the trigger phrase (Final answer).

Table 13: The incorrect reasoning trajectory is highlighted in red while the correct one is in blue.