Introduction of my research

Sitao Cheng
Websoft Lab (Advised by Yuzhong Qu)
Computer Science, Nanjing University

Contents

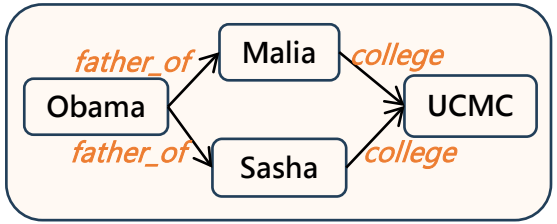
- Backgrounds
  - LMs reasoning over structured environments
  - A general pipeline
- Lines of my works
  - Question Decomposition Tree for Answering Complex Questions over Knowledge Bases **(AAAI23)**
  - MarkQA: A large scale KBQA dataset with numerical reasoning (**EMNLP23**)
  - QueryAgent: A Reliable and Efficient Reasoning Framework with Environmental Feedback based Self-Correction (submitted to **ACL24**)
  - Call me when necessary: LLMs can Efficiently and Faithfully Reason over Structured Environments (submitted to **ACL24**)

Background

- LMs reasoning over structured environments
- Multi-hop Reasoning task – Question answering
- Structured Environments: **Knowledge graph (base)**, Table, Database, etc

Question: Which college did daughters of Obama go to?

Knowledge graph instances



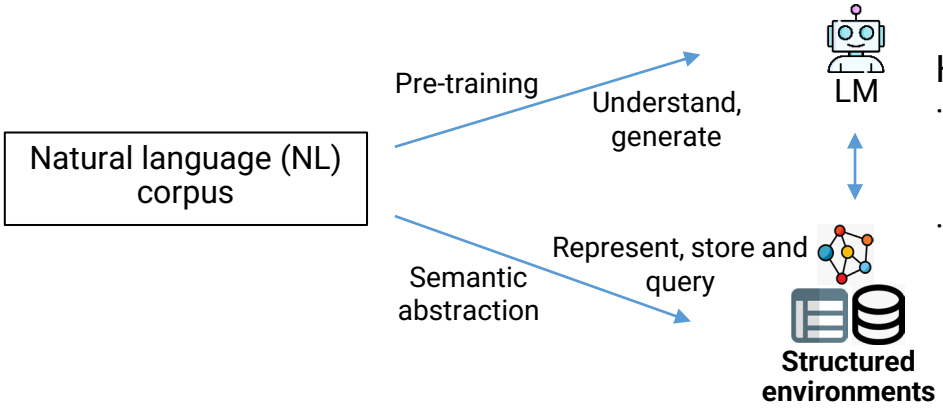
SPARQL Query

```
select ?college where {
Obama father_of ?daughter.
?daughter college ?college.
}
```

Answer: UCMC

Background - Reasoning over structured environments

- LMs – Pre-trained from natural language corpus
  - Strong NL understanding and generating ability (by embeddings)
- Structured Environments – abstraction of real-world semantics
  - Representation, store and query of semantics (by **schemas**)
- Challenges
  - **heterogeneity** between task and environment: LMs may not understand **schema representations**
  - **large-scale** environment: Cost of annotation & LMs limited context window

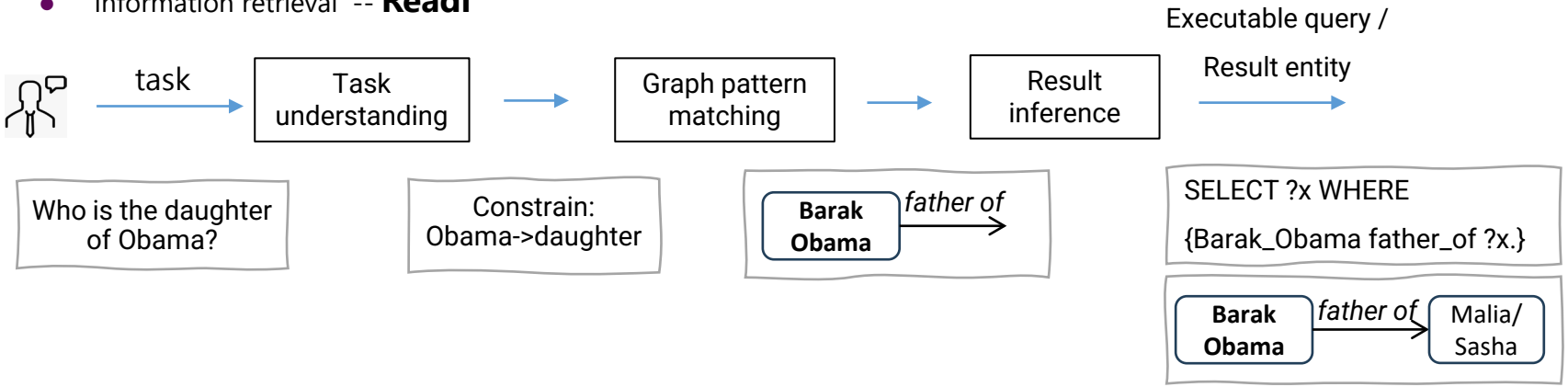


How to introduce structure information to LMs?

· For PLMs
  i. retrieve schemas with an Encoder
  ii. add candidate schemas to Seq2Seq input
· For LLMs
  i. interact with (explore) the environments

Background – a general pipeline

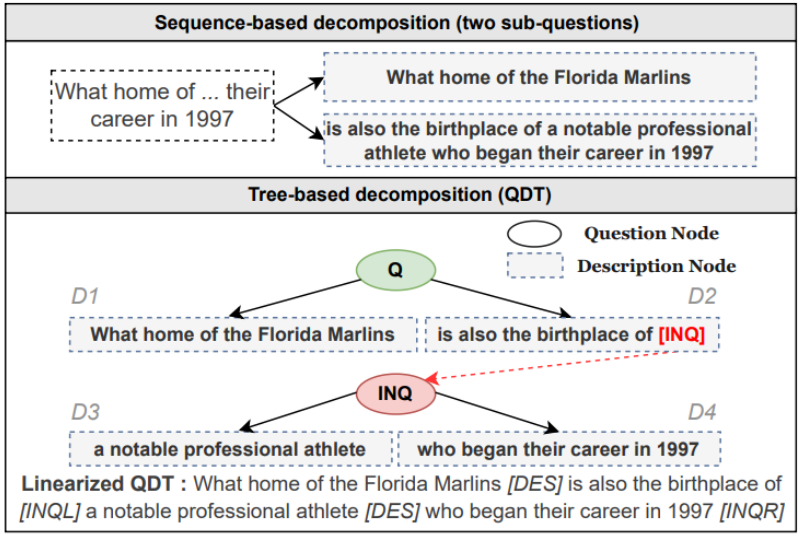
- Task understanding -- **QDT**
  - Complex question → simple constrains and their relations
- Graph pattern matching (schemas and their connections)
  - Entity / relation linking
  - Structure matching (how entities and relations are connected) -- **MarkQA**
- Result inference
  - Semantic parsing (SQL query building) -- **QueryAgent**
  - Information retrieval -- **Readi**



Question Decomposition Tree for Answering Complex Questions over Knowledge Bases

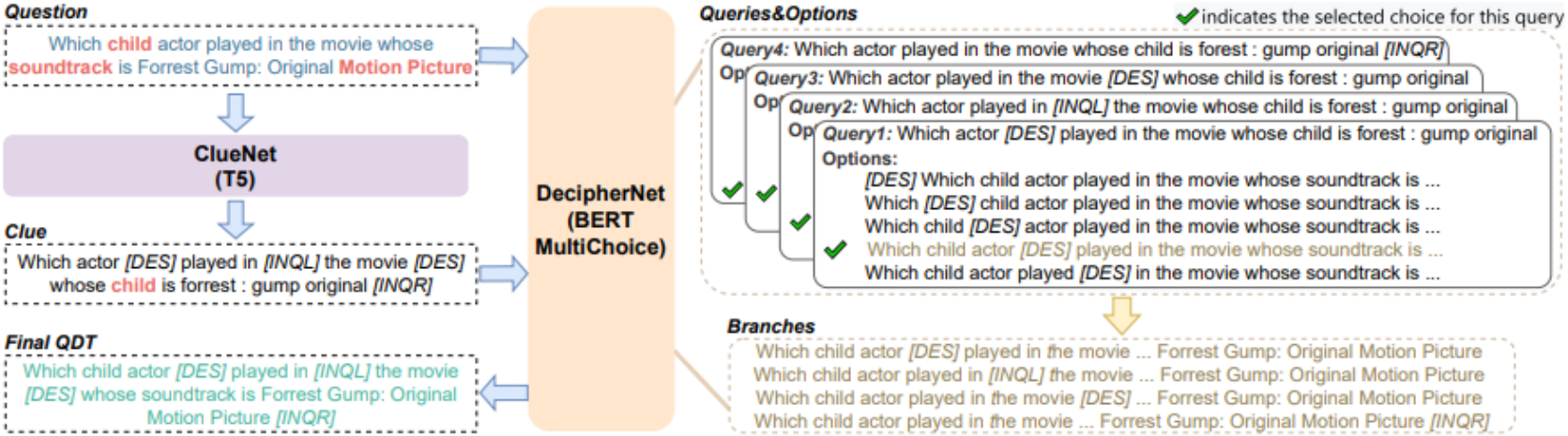
Question Decomposition Tree (QDT)

- Assumption: a Complex Questions can be decomposed into some simple questions
- Previous methods split a question into only two-parts
  - Insufficient to represent complex reasoning structure
- Tree-based decomposition
  - Recursively defined for representing complex structure
  - Can be Linearized to a sequence
    - Introducing some tags
    - can be obtained by generative LMs



Decomposition Method - Clue-decipher

- A two-staged method - Mitigation of hallucination by tag insertion
  - 1. Generate the decomposition (using the generation ability of LMs)
  - 2. Adopt a multi-choice model to determine inserting position for each tag (mitigating hallucination)



Experiments – Decomposition Results

- Annotate a dataset QDTrees from complex datasets and test on it
- Compared with tree-based and sequence-based methods
  - Different metrics
- Clue-decipher significantly outperforms others in all metrics

Method	EM	TDA	GED
EDGQA (Hu et al. 2021)	-	0.7315	0.3799
`Clue-Decipher`	**0.8332**	**0.9650**	**0.0554**
w/o DecipherNet	0.8130	**0.9650**	0.0558

Table 3: Tree-based Decomposition evaluation.

Method	EM	BLEU	ROUGE
SplitQA (Talmor and Berant 2018)	0.653	0.734	0.905
DecompRC (Min et al. 2019)	0.862	0.954	0.988
HSP (Zhang et al. 2019)	0.252	0.679	0.881
HSP + DecipherNet	0.793	0.935	0.983
`Clue-Decipher`	**0.909**	**0.970**	**0.993**
w/o DecipherNet	0.889	0.966	0.991

Table 4: Sequence-based Decomposition evaluation.

Experiments – KBQA results

- Significantly improve the results for **two** QA systems in **two** KBs
  - Our seq2seq QA system: Concat Question, QDT and linking results to a T5 model, output Logical Form
  - Tree-structured decomposition substantially boosts the result

Method	Avg. F1	Acc
(Qin et al. 2021)	0.462	-
(Huang, Kim, and Zou 2021)	0.682	-
T5-11B + Revise (Das et al. 2021)	0.582	0.556
CBR-KBQA (Das et al. 2021)	0.700	0.671
QDTQA	**0.728**	**0.679**
w/o QDT	0.715	0.666
w/o tree-based structure	0.720	0.670
w/ SplitQA	0.716	0.669
w/ DecompoRC	0.716	0.669
w/ HSP	0.717	0.669
w/ EDGQA	0.714	0.665

Table 5: QA performance of QDTQA on CWQ compared with baselines. We also report the performance with different decomposition methods.

Method	P	R	F1	ΔF1
NSQA	0.448	0.458	0.445	-
STaG-QA	**0.745***	0.548	0.536	-
(Liang et al. 2021)	0.511	0.593	0.549	-
EDGQA	0.505	0.560	0.531	0
w/ SplitQA	0.496	0.576	0.533	+0.002
w/ DecompRC	0.521	0.609	0.561	+0.030
w/ HSP	0.433	0.507	0.467	-0.064
w/ Clue-Decipher	0.548	**0.635**	**0.588**	**+0.056**

Table 6: QA performance of baseline methods and EDGQA + Clue-Decipher on LC. We also replace Clue-Decipher with other decomposition methods. * indicates that when calculating P, STaG-QA defines the empty answer to have P=1, different from others.

Limitations

- Surface form decomposition (a strong assumption)
- Contemporary LLMs already do well in question understanding
- For Seq2Seq PLMs, we augment information in input
  - for better structure matching

- What is the difficulty for LMs in graph pattern matching?
  - With golden linking results, LMs can do quite well in complex datasets (92% F1 on ComplexWebQuestions)
  - Structure can be well learned from annotations

MarkQA: A large scale KBQA dataset with **numerical reasoning**

MarkQA: A large scale KBQA dataset with numerical reasoning

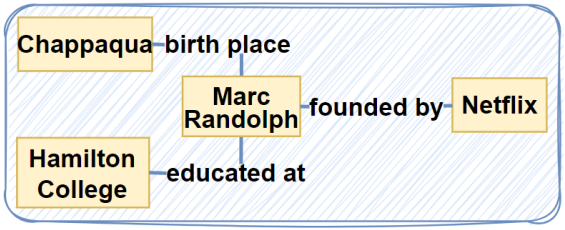
- KBQA aims to answer a question over a knowledge base (KB).
  - Need to match a **graph pattern** in the KB

- Numerical Reasoning is a critical ability in daily life
  - Require the ability of arithmetic, aggregation, comparison…

Acquire some Information → **Further Process the Information**

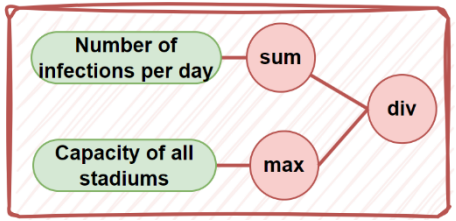
Multi-hop Reasnoing

Which company whose founder was born in Chappaqua and educated at Hamilton College?



Numerical Reasoning

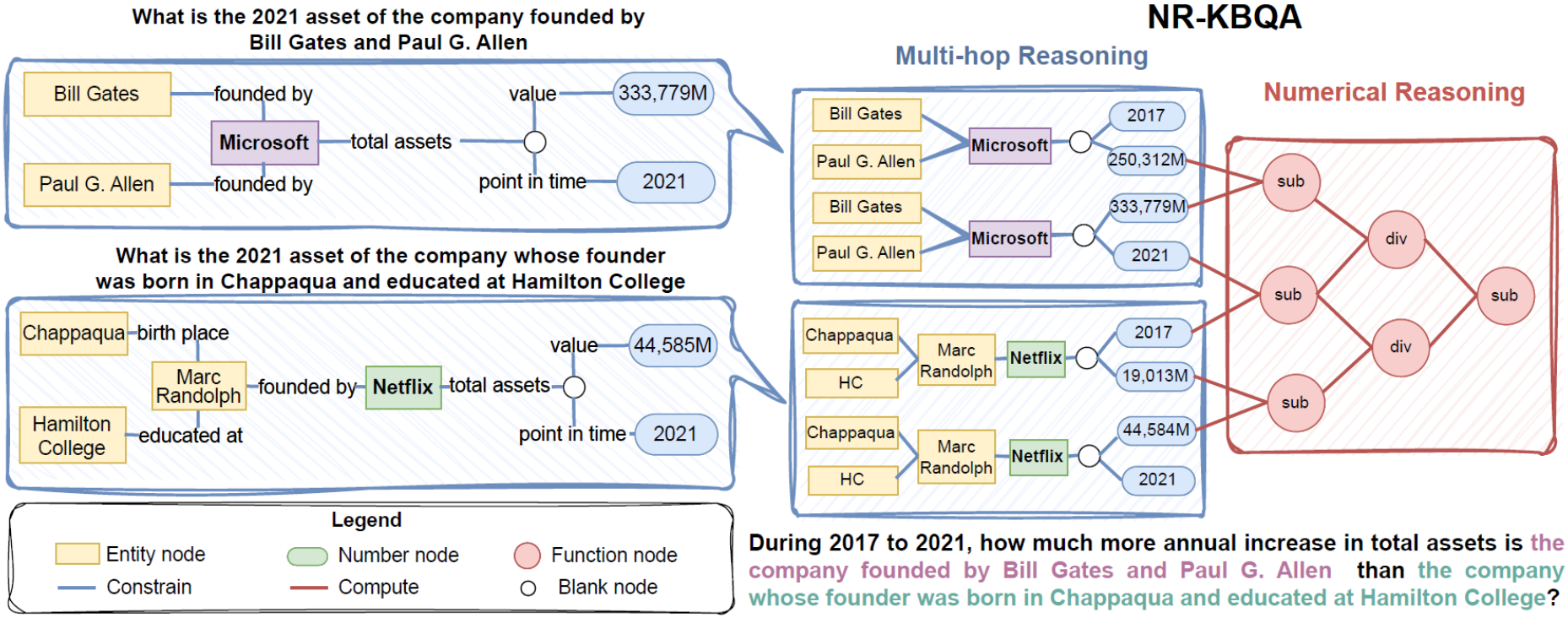
How many of Japan's largest sports stadiums could be filled with the number of new COVID-19 infections in Japan in 2021?



Numerical reasoning in previous datasets: **few** and **simple**

- Previous KBQA dataset mainly focus on Multi-hop reasoning (MR)
  - 90% of question in CWQ
  - 84.8% of question in GrailQA
- Numerical reasoning is insufficient in current KBQA datasets
  - Account for a small proportion
  - Only focus on Count, Argmax, Compare
  - Require calculation at most once
- For the first time explore and discuss Numerical Reasoning in KBQA from:
  - Task
  - Reasoning path representation
  - Dataset
  - Experiment

A New Task -- NR-KBQA



A New Representation -- PyQL

- Represented as a set of Python code
  - PyQL (Pythonic Query Language for SPARQL)
  - Each line initialize a PyQL object or calls a function
- Encapsulate various SPARQL syntax elements
  - BGP, Aggregation, Filter, Subquery, Assignment…
  - Can directly compiled to SPARQL
- Advantages:
  - User-friendly and conciseness
  - Step-by-Step reasoning path

PyQL

What is the average speed of all anti-aircraft gun with a range greater than 6,000 meters?

```
a=PyQL()
a.add_type_constrain('Q7325635','x1')
a.add_quantity('x1','P4176','x2')
a.add_filter('x2','>',6000)
a.add_quantity('x1','P2052','x3')
a.add_avg('x3','x4')
```

SPARQL

```
SELECT (AVG(?x3) AS ?x4 ) {
   ?x1 wdt:P31/wdt:P279* wd:Q7325635.
   ?x1 p:P4176 ?statement_x2.
   ?statement_x2 psv:P4176 ?value_st_x2.
   ?value_st_x2 wikibase:quantityAmount ?x2.
   FILTER(?x2 > 6000).
   ?x1 p:P2052 ?statement_x3.
   ?statement_x3 psv:P2052 ?value_st_x3.
   ?value_st_x3 wikibase:quantityAmount ?x3.
}
```

Experiment

- Overall experiments

Methods	Output	Overall	I.I.D	Compositional	Zero-shot
T5-base	SPARQL	34.24	70.05	53.71	6.32
	PyQL	40.70	78.32	63.10	10.39
GMT	SPARQL	38.68	78.32	63.58	6.07
	PyQL	43.63	82.10	68.33	11.71
QDTQA	SPARQL	37.19	76.82	57.37	7.01
	PyQL	42.57	84.59	70.89	7.01

Table 2: QA performance (%) on test set of MarkQA.

Experiment

- Different reasoning types

Type	Over.	I.I.D	Comp.	Zero.
All	40.7	78.3	63.1	10.4
NR	42.0	85.4	64.3	12.9
NR and MR	38.5	65.5	61.8	5.1
NR and MR(1)	43.3	70.1	70.2	6.5
NR and MR(2)	28.7	55.6	44.8	2.3

Table 4: Performance of different types of questions on T5 (PyQL). NR and MR mean numerical reasoning and multi-hop reasoning, respectively. MR(1) and MR(2) mean one-hop and two-hop reasoning, respectively.

- Oracle experiment

Methods	Over.	I.I.D	Comp.	Zero.
T5-base	40.7	78.3	63.1	10.4
w/ gold E	46.5	88.3	72.7	12.1
w/ gold R	47.9	79.2	65.5	23.1
w/ gold ER	57.6	89.8	76.1	31.9

Table 3: Detailed analysis of T5-base with PyQL as output. w/ gold E or R means we use golden entity or relation linking results. Over., Comp., and Zero. stands for Overall, Compositional, and Zero-shot, respectively.

Limitations

- Some human labor is involved in annotation
- PyQL query can be generalized to other environments

- How to well leverage PyQL query?
  - We prove the difficulty brought by structure, but how to handle it?
- How to incorporate LLMs?
  - How to learn the schema with **strong** understanding ability but **less** annotations?
  - Maybe by **interaction**, but how to design the interplay between input and output?

- QueryAgent: An agent framework for query building
- Readi: An interaction framework for information retrieval

Running time, query engine times, LLM token cost

QueryAgent: A Reliable and Efficient Reasoning Framework with Environmental Feedback based Self-Correction

An LLM-based agent framework
for query building

Motivations

- ICL-based generation induce massive **enumeration** and **hallucination**
  - Question decomposition for better understanding
- Agent-based methods suffer from **error propagation** and **hallucination**
  - A novel correction method for reliable generation
- PyQL query is step-wise-executable
  - ⟶ step-by-step query building (with feedback each time)
  - An interface for knowledge base (retrieval tools and functional tools)

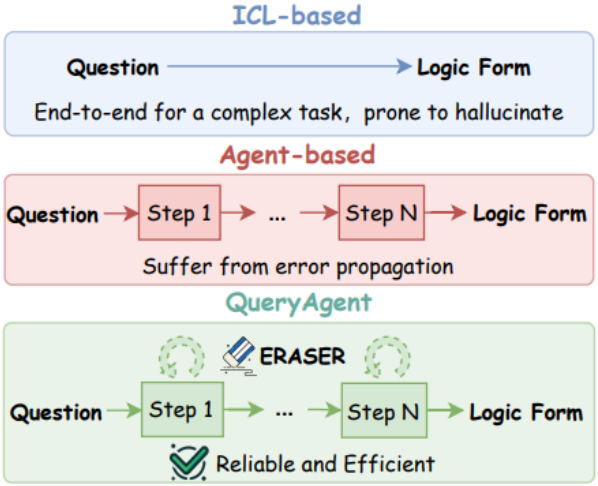


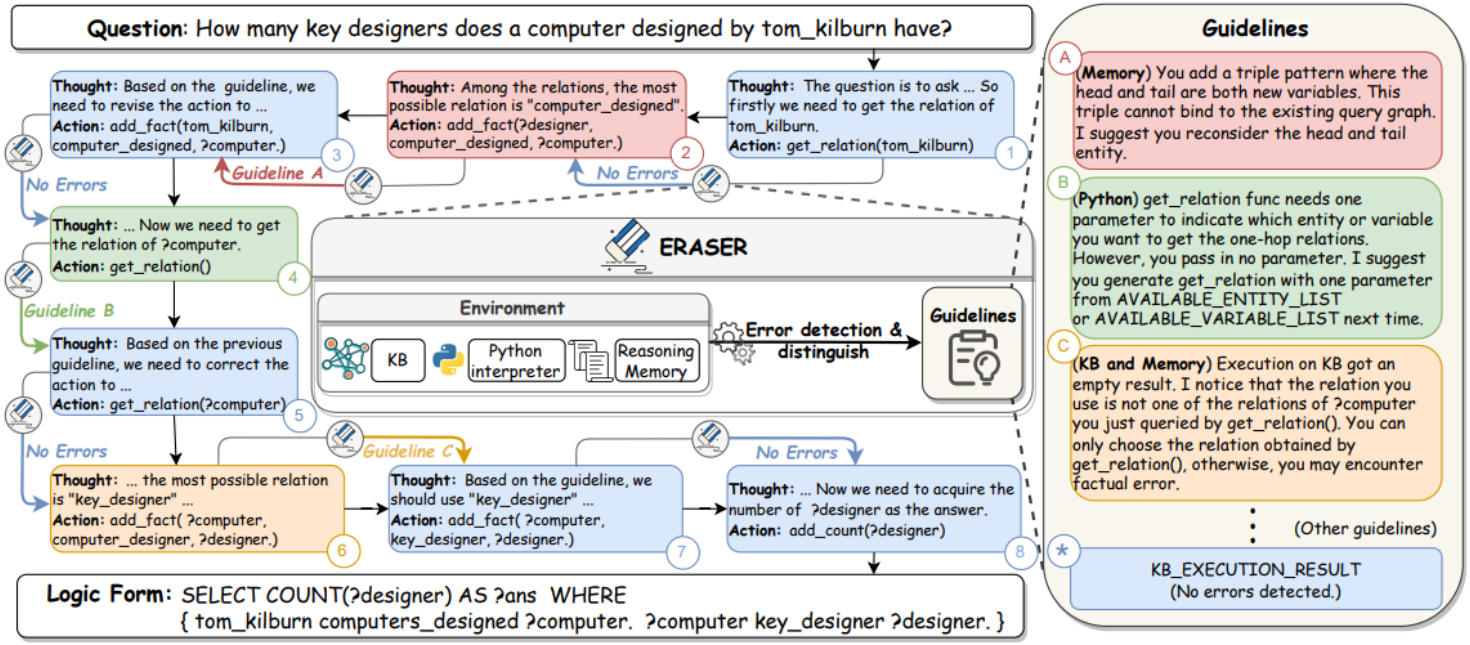
Figure 1: QueryAgent compared with two mainstream KBQA paradigms employing LLMs.

Question:
How many key designers does a computer designed by kilburn have?

PyQL:
1. get_relation(kilburn) ⟶ [‘computer_designed’, ‘sex’, ……]
2. add_fact(kilburn, computer_designed, ?computer) ⟶ {‘?computer’: [‘WK tube’]}
3. get_relation(?computer) ⟶ [‘computer_designer’, ‘time_invented’, ……]
4. add_fact(?computer, computer_designer, ?designer) ⟶ {‘?computer’: [‘WK tube’], ‘?designer’:[‘kilburn’, ……]}
5. add_count(?designer) ⟶ {‘?computer’: [‘WK tube’], ‘?designer’:[‘kilburn’, ……]}

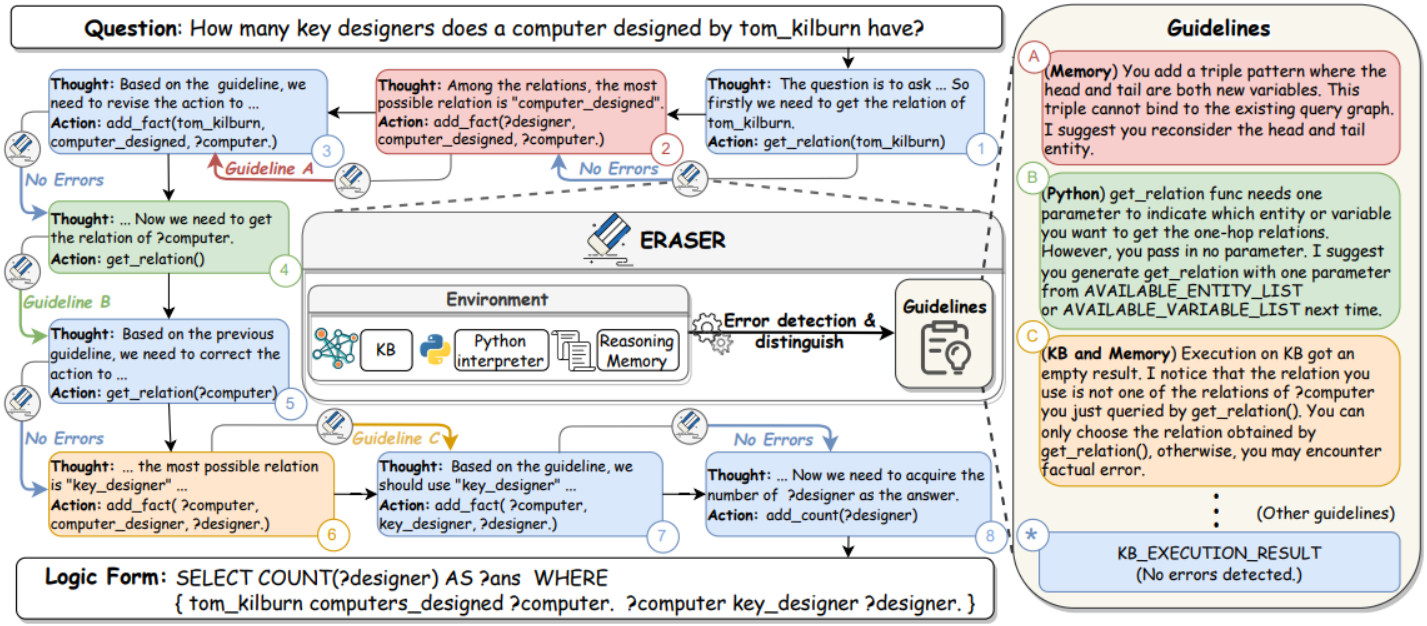
Agent framework (ReAct) with feedback-based self-correction

- LLM-Agent step-by-step query build a query with PyQL
  - based on previous step, an LLM generates **thought** and **action**. Then the action is executed.
  - A correction module, ERASER, detects and distinguishes errors, collects guidelines and adds them to **observations**.



ERASER

- Provide guidelines for detected error
  - Previous: Few-shot correction, relying on LLMs to **identify** the error and **mimic** the examples
  - Multiple feedback sources: KB engine, python interpreter and reasoning memory
  - Provide **purposeful** and **precise** guidelines as observation (zero-shot, seamless correction)



Experiments

- Significant improvement with only one-shot example

Methods	GrailQA	GraphQ	WebQSP	MetaQA-3Hop
fine-tuning				
ArcaneQA (Gu and Su, 2022)	73.7	31.8	75.6	-
TIARA (Shu et al., 2022)	78.5	-	76.7	-
DecAF (Yu et al., 2023)	81.4	-	78.8	-
Pangu(T5-3B) (Gu et al., 2023)	83.4	57.7	79.6	-
few-shot				
Pangu (Gu et al., 2023)	53.5	35.4	48.6	-
KB-BINDER (Li et al., 2023)	50.8	34.5	56.6	96.5
KB-Coder (Nie et al., 2023)	51.7	35.8	60.5	-
one-shot				
KB-BINDER (Li et al., 2023)	16.8	4.8	9.0	65.3
AgentBench (Liu et al., 2024)	30.5	25.1	26.4	-
Ours	**60.5**	**50.8**	**63.9**	**98.5**
w/ GPT4	66.8	63.0	69.0	99.9

Analysis

- Ablation study of ERASER
- Transferability of ERASER
- Efficiency in runtime, engine query time and token cost

Method	GrailQA	GraphQ
Ours	**60.5**	**50.8**
w/o ERASER	43.7	35.3
w/ zero-shot SC	38.5	30.2
w/ few-shot SC	48.0	40.1

Ablation study

Methods	GrailQA	GraphQ	WebQSP
AgentBench	30.5	25.1	26.4
w ERASER	**38.5**	**35.6**	**32.0**

Transferability

Methods	GrailQA			GraphQ			WebQSP		
	TPQ	QPQ	CPQ	TPQ	QPQ	CPQ	TPQ	QPQ	CPQ
KB-BINDER	51.2 s	3297.7	**$ 0.010**	84.0 s	2113.8	$ 0.024	138.6 s	8145.1	$ 0.017
AgentBench	40.0 s	7.4	$ 0.034	65.1 s	7.2	$ 0.035	70.4 s	7.2	$ 0.038
Ours	**16.6 s**	**5.2**	$0.019	**15.3 s**	**6.2**	**$ 0.021**	**12.6 s**	**4.7**	**$ 0.014**

Efficiency analysis

Limitations

- The correction guidelines are based on diverse feedback
  - The step-by-step manner captures various feedback sources
- The step-by-step reasoning is a fine-grained decomposition
  - Lead to lengthy prompts

- Maybe a more efficient way of interaction, like human's reasoning?
  - Human and animals tend to have a plan subconciously and ground the plan in environments
  - And we flexibly adjust our plan when grounding

How to practically invoke LLMs to interact with the environment

Call me when necessary: LLMs can Efficiently and Faithfully Reason over Structured Environments

Less LLM calls

LLM's output can be grounded on environments

Previous interaction paradigms

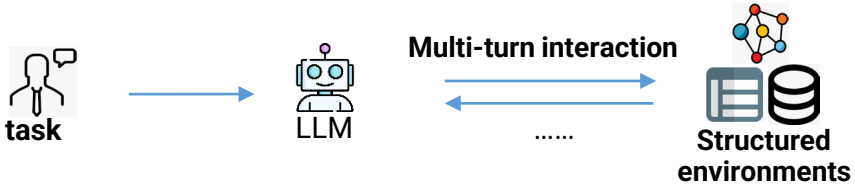
- Interactive interaction
  - Minimum effort at each step
  - pros
    - Fine grained decomposition
    - LLM discriminate at each step (better faithfulness)
  - cons
    - Iterative manner can be **inefficient**
    - Each step relies on previous steps, inducing **error propagation**
    - multiple steps and massive relations may cause **long history**

Q: Which college did daughters of Obama go to?

1. get_relation(Obama) ⟶ [gender, father_of, ...]
2. get_tail(Obama, father_of) ⟶ [Malia, Sasha]
3. get_relation(Malia) ⟶ ...

.....

Return: Obama $\xrightarrow{\text{father\_of}}$ Malia, Sasha $\xrightarrow{college}$ UCMC

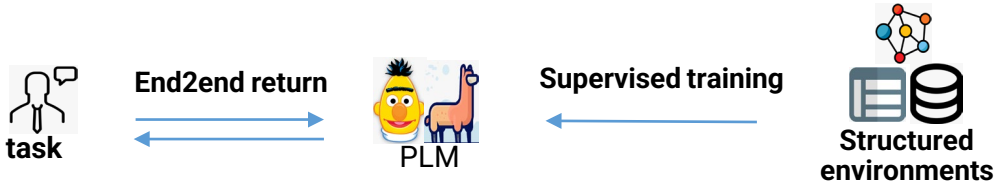


(a) Iterative interaction with LLM APIs

Previous interaction paradigms

- Training: Inject structure information to models
  Inference: End2End return
  - Direct path generation or Retrieve-and-Build
  - Pros:
    - no interaction at inference time, better efficiency
  - Cons:
    - Not ensuring **faithfulness** and relying on beam search, resulting in larger retrieved instances
    - Relying on **training data**, hard to obtain for large-scale environments

Q: Which college did daughters of Obama go to?
Return: Obama ⟶ father_of ⟶ college

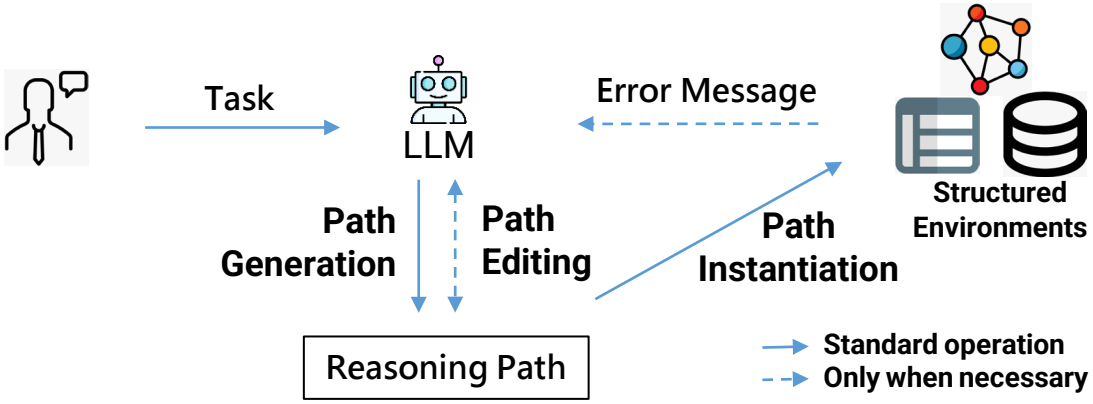


(b) fine-tuned PLM

Reasoning Path Editing (Readi)

Which college did daughters of Obama go to?
Obama ⟶ father_of ⟶ college

- How do we humans do multi-hop reasoning?
- "Reasoning path" to represent structured reasoning process
  - Utilize strong question understanding ability of LLMs
  - Can be instantiated on environments, bridging the heterogeneity gap
- Interaction framework for information retrieval
  - End2end generate an initial reasoning path (less LLM calls)
  - Instantiation on environments and edit the path when anything goes wrong (better faithfulness)



Reasoning path

- Structured representation of natural language task
  - Instantiable on structured environments (Knowledge graph)
- Relational path from topic entities
- Can represent complex constrains (conjunction)

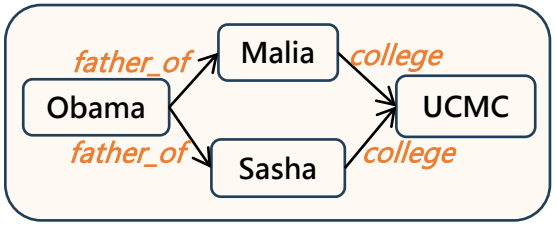
Example $Q1$

Which college did daughters of Obama go to?

Reasoning Path

[Obama] father_of→college

Path Instances



Single-constrained Reasoning Path

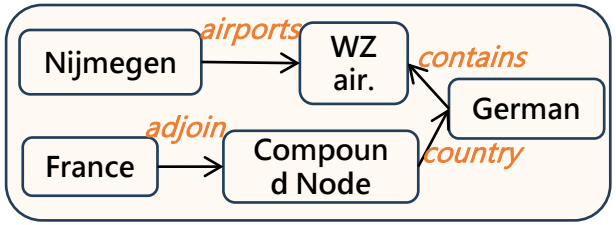
Example $Q2$

What country bordering France contains an airport that serves Nijmegen?

Reasoning Path

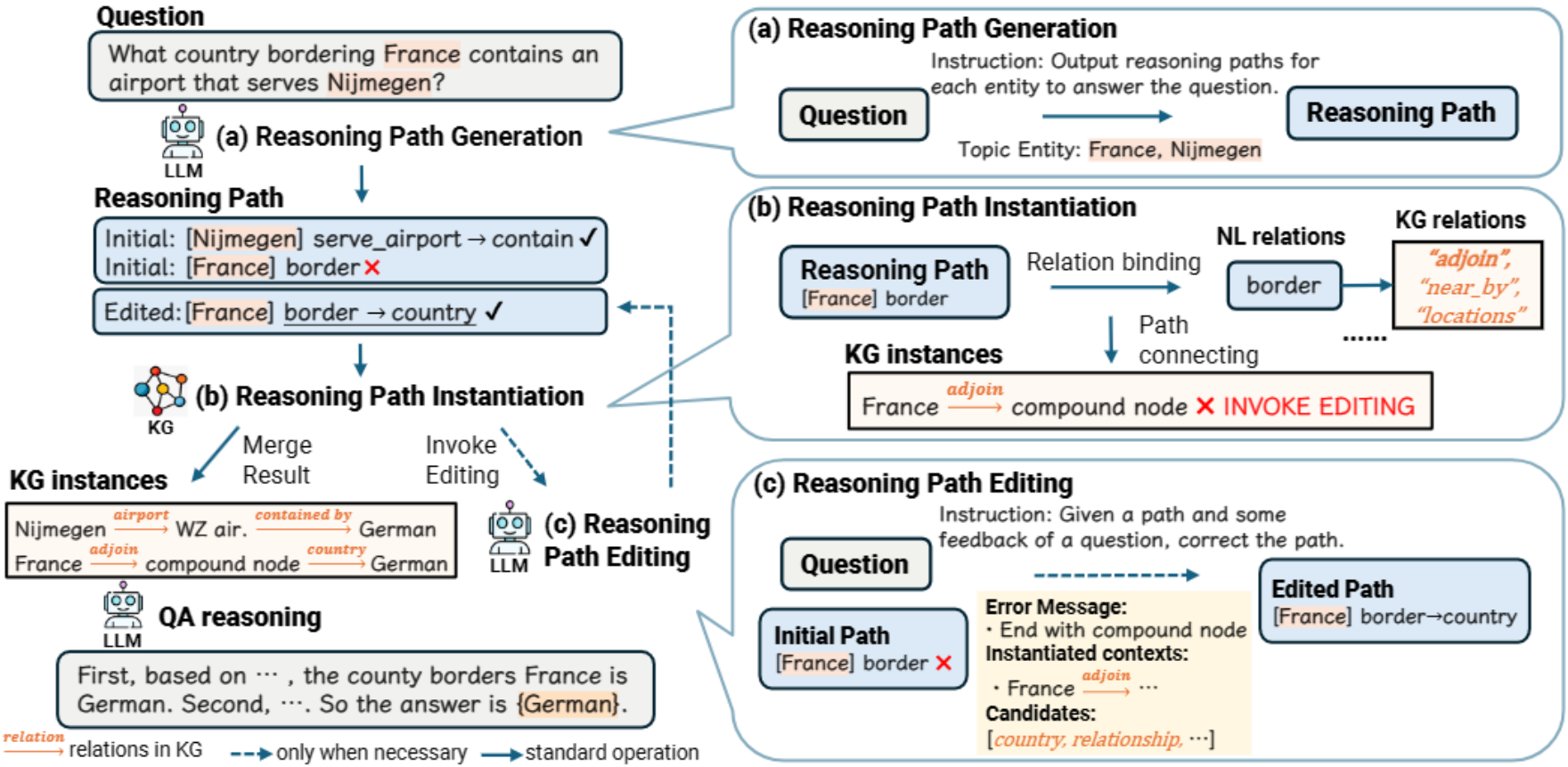
[Nijmegen] serve_airport→contain
[France] border→country

Path Instances



Multi-constrained Reasoning Path

Method – Reasoning Path Editing (Readi)



Experiments – Main Results

Methods	WebQSP	CWQ	MQA-1H	MQA-2H	MQA-3H
Training-based Method					
EmbedKGQA (Saxena et al., 2020)	66.6	-	**97.5**	98.8	94.8
NSM (He et al., 2021)	67.7	47.6	97.1	99.9	98.9
TransferNet (Shi et al., 2021)	71.4	48.6	97.5	**100***	**100***
SR+NSM+E2E (Zhang et al., 2022)	69.5	49.3	-	-	-
UniKGQA (Jiang et al., 2023c)	75.1	50.7	**97.5**	99.0	99.1
ReasoningLM (Jiang et al., 2023b)	78.5	**69.0***	96.5	98.3	92.7
RoG (Luo et al., 2023)	**85.7***	62.6	-	-	84.8
Inference-based Method					
Davinci-003 (Ouyang et al., 2022)	48.7	-	52.1	25.3	42.5
GPT3.5 (OpenAI, 2022)	65.7	44.7	61.9	31.0	43.2
GPT4 (OpenAI, 2023)	70.7	52.1	71.8	52.5	49.2
AgentBench (Liu et al., 2023b)	47.8	24.8	-	-	-
StructGPT (Jiang et al., 2023a)	69.6	-	97.1	97.3	87.0
Readi-GPT3.5	74.3	55.6	98.4	**99.9**	**99.4**
Readi-GPT4	**78.7**	**67.0**	**98.5***	**99.9**	99.2

KBQA （Hit@1）

Methods	WTQ	WikiSQL
Training-based Method		
TAPAS	48.8	83.6
UnifiedSKG (T5-3B)	49.3	86.0
TAPEX	**57.5**	**89.5**
Inference-based Method		
Davinci-003	34.8	49.1
GPT3.5	55.8	59.8
GPT4	57.0	59.9
StructGPT	52.2	65.6
Readi-GPT3.5	**61.7**	**66.2**
Readi-GPT4	61.3	66.0

TableQA （denotation accuracy）

Analysis - Ablation Study (generation and editing modules)

- Effectiveness of initial path generation (horizontal)
- Effectiveness of path editing (vertical)
  - **Robustness** : well Editing for corrupt and empty path
- With stronger model, comes better results for both generation and editing
- Plug-and-play nature for initial path generation and editing

Variance of Readi	Answer Coverage Rate (AC)				QA Performance (Hit@1)			
	Corrupt	*Empty*	GPT3.5	GPT4	*Corrupt*	*Empty*	GPT3.5	GPT4
w/o edit	-	-	56.7	62.7	-	-	51.0	57.2
w/ edit by GPT3.5	54.0	56.4	62.5	64.3	57.3	58.5	58.7	58.5
w/ edit by GPT4	55.6	63.9	68.6	65.8	58.2	59.9	58.1	59.3

Analysis – Features of Reasoning Path

- Readi's initial path is competitive (GPT3.5) or superior (GPT4)
- After Editing, Readi's path outperforms fine-tuned methods with wider beams
- Finetuned methods get exploded number of retrieved knowledge with wider beams
  - Still worse QA results than Readi
- Distribution of Editing times shows the efficiency
  - Half questions does not need editing (LLM called only once)
  - Average time: 1.55 – GPT4, 1.99 – GPT3.5

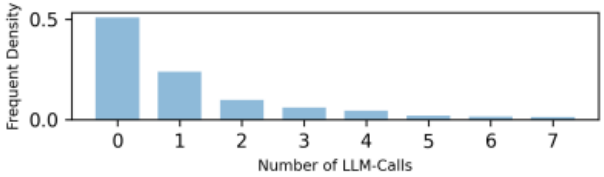


Figure 5: Distribution of number of LLM-Call for reasoning path editing of Readi-GPT4.

Methods	Graph Quality		QA Perf.
	AC	#RK	Hit@1
SR			
- beam size 1	58.4	**26.3**	50.9
- beam size 3	67.2	47.1	54.6
RoG			
- beam size 1	57.0	69.5	52.2
- beam size 3	**77.5**	170.1	57.3
Readi initial path			
- GPT3.5	56.7	134.6	51.0
- GPT4	62.7	101.4	57.2
Readi full			
- GPT3.5	62.5	93.7	58.7
- GPT4	71.8	121.5	**59.3**

Table 4: Reasoning path evaluation of Readiand compared methods. AC and #RK denotes answer coverage rate and number of retrieved knowledge, respectively.

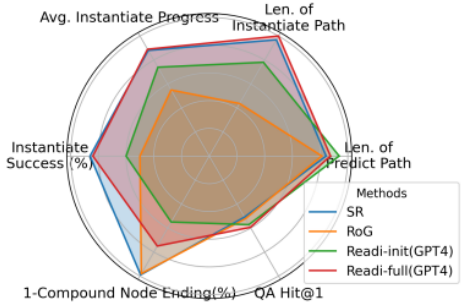


Figure 4: Extensive features of Readi's reasoning path, compared with fine-tuned methods and Golden.

Limitations

- Can try other LLMs to test generalizability
- The Instantiation is natural but a bit brute-force
- The interaction is relatively efficient and faithful, does not ensure the instances we obtain can be used to answer the question
  - The fully instantiated path does not guarantee that it's the ground truth
  - Also the limitation of IR

Thanks for listening

- Q&A