Reasoning with Trees: Faithful Question Answering over Knowledge Graph

Tiesunlong Shen[1], Jin Wang[1*], Xuejie Zhang[1], Erik Cambria[2]
[1]School of Information Science and Engineering, Yunnan University, Kunming, China
[2]College of Computing and Data Science, Nanyang Technological University, Singapore
tensorshen@mail.ynu.edu.cn, {wangjin,xjzhang}@ynu.edu.cn, cambria@ntu.edu.sg

Abstract

Recent advancements in large language models (LLMs) have shown remarkable progress in reasoning capabilities, yet they still face challenges in complex, multi-step reasoning tasks. This study introduces Reasoning with Trees (RwT), a novel framework that synergistically integrates LLMs with knowledge graphs (KGs) to enhance reasoning performance and interpretability. RwT reformulates knowledge graph question answering (KGQA) as a discrete decision-making problem, leveraging Monte Carlo Tree Search (MCTS) to iteratively refine reasoning paths. This approach mirrors human-like reasoning by dynamically integrating the LLM's internal knowledge with external KG information. We propose a real-data guided iteration technique to train an evaluation model that assesses action values, improving the efficiency of MCTS. Experimental results on two benchmark KGQA datasets demonstrate that RwT significantly outperforms existing state-of-the-art methods, with an average performance improvement of 9.81%. Notably, RwT achieves these improvements without requiring complete LLM retraining, offering a more efficient and adaptable approach to enhancing LLM reasoning capabilities.

1 Introduction

Working with a large language model (LLM) is not without risks, including responses corrupted from hallucinations (Ji et al., 2023; Yeo et al., 2024), where the generated answers seem credible but are misinformation (Yao et al., 2024a). Hallucinations can be a severe problem for LLMs because they can spread misinformation, expose confidential information, and create unrealistic expectations about what they can do (Ge et al., 2024). One promising approach to address this challenge is the integration of knowledge graphs (KGs) into the LLM generation process (Xu et al., 2024; Shu et al., 2024).

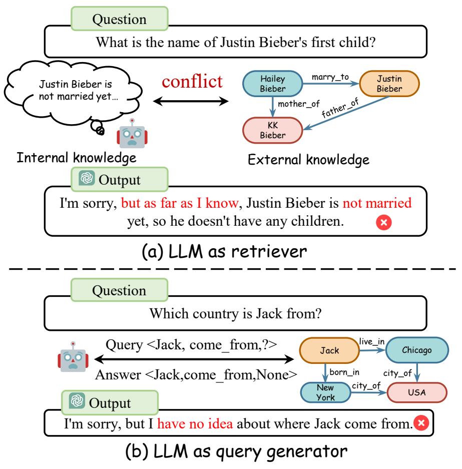
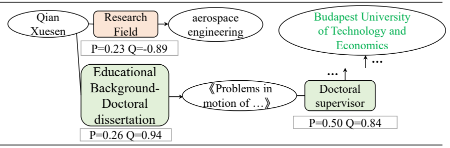

Question
What is the name of Justin Bieber's first child?
Justin Bieber is not married yet…
conflict
Hailey Bieber
marry_to
Justin Bieber
mother_of
father_of
KK Bieber
Internal knowledge
External knowledge
Output
I'm sorry, but as far as I know, Justin Bieber is not married yet, so he doesn't have any children.
(a) LLM as retriever
Question
Which country is Jack from?
Query <Jack, come_from,?>
Answer <Jack,come_from,None>
Jack
live_in
Chicago
born_in
city_of
New York
city_of
USA
Output
I'm sorry, but I have no idea about where Jack come from.
(b) LLM as query generator

Figure 1: The lack of knowledge and hallucination issues in current LLM+KG methods.

Question	Where did Dr. Qian Xuesen's doctoral advisor attend school?
Answer	Budapest University of Technology and Economics
Chat-GPT	Dr. Qian Xuesen's alma mater is Shanghai Jiao Tong University.
GPT-4o	Dr. Qian Xuesen‘s PhD advisor was Theodore von Kármán……During his time at the California Institute of Technology…
RwT	

Qian Xuesen
Research Field
P=0.23 Q=-0.89
aerospace engineering
Educational Background-Doctoral dissertation
P=0.26 Q=0.94
《Problems in motion of …》
Doctoral supervisor
P=0.50 Q=0.84
…
…
Budapest University of Technology and Economics

Figure 2: Comparison of different knowledge graph reasoning approaches

KGs are information-rich data that provide a view of entities and how they interrelate (Ji et al., 2020; Zhu et al., 2020). As structured repositories of real-world facts and relationships expressed in a machine-friendly format, KGs can augment LLMs with reliable factual knowledge and enhance their reasoning capability (Pan et al., 2024; Cao, 2024; Wei Jie et al., 2024). Three methods emerge by pairing a KG with an LLM, including taking LLM as a retriever (Kim et al., 2023; Tan et al., 2023), employing it as a query generator (Jiang et al., 2023a) or directly tuning the LLM on KGs (Luo et al., 2023). As shown in Fig. 1, LLMs often perform well on entities and relationships with high frequencies but face challenges in less popular topics.

[*]Corresponding author.

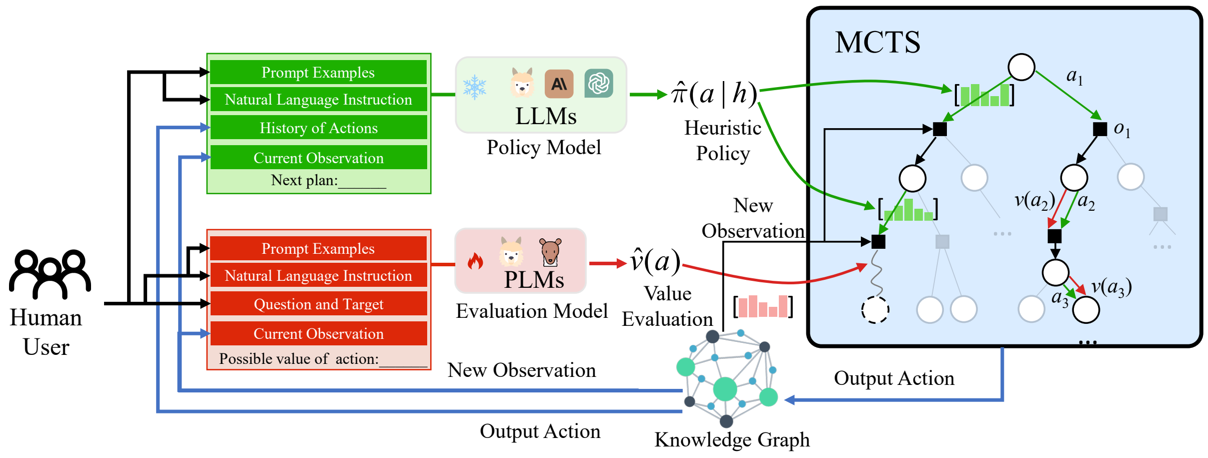


Figure 3: The overall framework of RwT. For each simulation in the MCTS, this study uses the LLM as heuristics to guide the trajectory to promising parts of the search tree and uses an evaluation model to evaluate the quality of tree nodes.

This inconsistency points to a deeper, more fundamental issue shared by all these methods: they fail to effectively integrate the internal knowledge of LLMs with external KG information in a way that aligns with human intuitive reasoning (Agrawal et al., 2024). These approaches do not follow the human intuitive process of continuously reflecting, adjusting, and integrating new information with existing knowledge. Humans iteratively reassess and modify their reasoning paths, dynamically incorporating new insights and correcting errors. This continuous refinement and integration is crucial for effective problem-solving (Wang et al., 2023c; Zhao et al., 2024).

To overcome these challenges and better mimic human-like reasoning, this study proposes a faithful question-answering method called reasoning with trees (RwT) that synergizes LLMs with KGs using Monte Carlo Tree Search (MCTS) (Browne et al., 2012; Silver et al., 2016) for reliable and interpretable reasoning. As shown in Figure 2, RwT first identifies the question entity “Qian Xuesen”. It then selects an appropriate relation to explore based on the probability P (the likelihood of this relation leading to the correct answer) predicted by the LLM and the Q-value provided by MCTS simulation. This process leads to the discovery of next hop “Problems in motion of …”. RwT differs from traditional LLM-based approaches by incorporating the LLM’s capabilities and structured KG information, enabling more accurate and interpretable reasoning. This method approaches the KG question-answering task as a discrete decision-making problem, managed through interactions between a reasoning environment and a policy model (implemented with an LLM). As shown in Figure 3, the policy model observes environmental states and suggests actions. This study employs a Monte-Carlo planning algorithm for reconcilable search to overcome the challenge of reconciling the LLM’s internal training data with conflicting external KG data. Specifically, RwT initializes candidate solutions with an action value based on LLMs’ inherent knowledge, updating it iteratively with factual knowledge as entities access their next-hop neighbors. This approach allows more informed decisions that integrate internal and external knowledge effectively. By directly reasoning over KGs, the LLM provides heuristic strategies based on observed candidate solutions, thus preventing the generation of non-executable queries. Additionally, this study develops a real-data guided iteration technique to train an evaluate model (implemented with a pre-trained language model) to assess action values. This evaluator is trained once and is adaptable across various datasets.

RwT leverages the fixed LLM’s reasoning capabilities and rich internal knowledge, while harnessing MCTS’s balanced exploration and evaluation strengths. By only requiring the training of a small PLM for evaluation, the method efficiently navigates the knowledge graph to identify the correct answer entity for the given KGQA task. Experimental results on two benchmark KG question-answering datasets show that the proposed method significantly outperforms several previous methods. The human evaluation also demonstrates that the proposed method produces valid and faithful reasoning steps compared with existing LLMs, such as GPT-4o and ChatGPT. To address the computational complexity and improve inference efficiency, we further propose a hop-level beam search (HBS) as a lightweight alternative to iterative MCTS during inference. Compared to the previous works, RwT avoids the repetitiveness and local-optimality issues of greedy decoding and mitigates the stochasticity of a single sampled generation.

2 Preliminaries

2.1 Knowledge Graph Question Answering

Multi-hop KG question-answering is a natural language processing task based on knowledge graphs. It inputs a natural language question q and retrieves the answer from a given knowledge graph G. Following the standard protocol (Saxena et al., 2020), the subject entities in question q are given and mapped to a node v_q in G via entity-linking algorithms (Yih et al., 2015). The node v_q, known as the seed node, is used to find a multi-hop adjacent node v_a as the answer to the question q. This study formulated the problem as a Markov Decision Process (MDP) (Puterman, 1990). The transition probability is set to 1 since the transition to the related entity is deterministic once the relation is selected.

2.2 Monte Carlo Tree Search

MCTS (Silver et al., 2016) is a sampling-based search algorithm for strategy optimization in decision-making problems. It iteratively builds a search tree by repeating four phases: selection, expansion, evaluation, and backpropagation. During the selection phase, it recursively selects child nodes from the root using the upper confidence bound (UCB) algorithm (Auer, 2000) (In the implementation of RwT, we use PUCT, an extension of UCB that incorporates prior probabilities) ,

$$\mathrm{UCB}(i) = w_i + C \cdot \sqrt{\frac{2 \ln N_i}{n_i}} \qquad (1)$$

where n_i and N_i are the visit counts of the node i and its parent node, respectively. C is a hyperparameter balancing exploration and exploitation and w_i is the average value of all descendant nodes of i. After selection, the tree expands according to a defined policy. In the evaluation phase, the value of the newly expanded node is estimated using either sampling or model-based methods. Finally, in the backpropagation phase, the estimated value is propagated back to all ancestor nodes of the newly expanded node.

3 Reasoning with Trees

3.1 Overview

In RwT, the MCTS algorithm explores the knowledge graph starting from an initial state s_0, which corresponds to the entity mentioned in the given question q. Each node in the MCTS represents a state s_t defined by the current entity in the knowledge graph. From each state, possible actions a_t are the relations connecting to other entities. Taking an action leads to a new state s_{t+1}, transitioning to the next entity via the chosen relation. The goal of this exploration is to find a path in the knowledge graph that leads to the correct answer entity for the given question q.

Each state-action (entity-relation) pair is evaluated using a value model, which estimates the future rewards and the current entity's relevance to the question. As shown in Figure 4, MCTS in RwT iteratively selects, expands, evaluates, and backs up the value of nodes to find the most promising paths in the knowledge graph. The detailed process is described in Algorithm 1 in appendix. Our approach leverages a fixed LLM as the policy model, focusing solely on training the evaluation model. The evaluation model, which includes a roll-out and critical models, is refined through iterative training to improve its accuracy in guiding the search process.

3.2 MCTS Planning

For each action a of the state s, this method stores a set of statistics $\{P(s,a), Q(s,a), N(s,a)\}$, where P is the prior score (i.e., the likelihood score from the policy model), Q is the action value, and N is the visit count. $Q(s,a)$ can be viewed as the posterior score, considering the future impact of a. Concretely, consider a complete reasoning path consisting of T planning steps. At a given time t, the state s_t represents the current entity in the knowledge graph, encapsulating the result of all previous reasoning steps. The subsequent reasoning step that might be taken is represented as the action a_t, which corresponds to selecting a relation connected to the current entity. The planning algorithm ends after a fixed budget of simulations. Each simulation consists of the following stages:

Selection. During the selection phase, the algorithm starts at the root node (the given question entity) and traverses the tree by selecting child nodes according to the PUCT (Policy + UCT) algorithm (Rosin, 2011). RwT employs PUCT instead

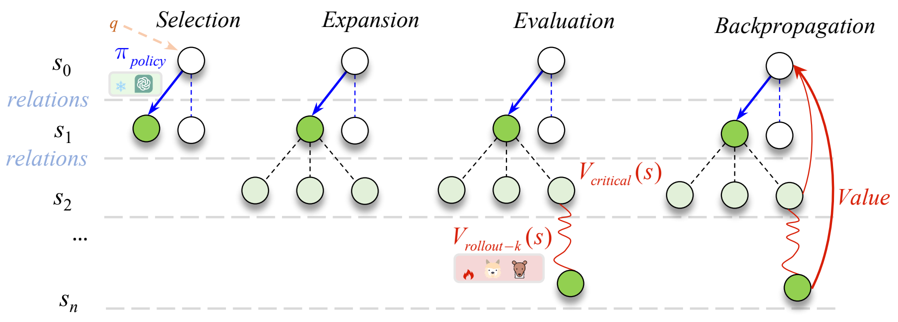


Figure 4: An overview of the four operations of RwT. A relation in KG is selected, expanded, and simulated with the policy and evaluation models until a terminal node is reached. Then, the signals from the value function are backpropagated.

of the standard UCB algorithm as it is better suited for integrating with neural network priors, which is crucial for the approach using an LLM as the policy model. PUCT incorporates prior probabilities into the selection process, allowing the method to leverage the LLM's knowledge to guide the tree search. The selection criterion for an action a in state s_t is given by:

$$a_t = \arg\max_{a \in \mathcal{T}_k} \left[\hat{Q}(s_t, a) + c_{\text{puct}} \cdot P(s_t, a) \cdot \frac{\sqrt{N_{\text{parent}}(a)}}{1 + N(s_t, a)} \right] \tag{2}$$

$$P(s_t, a) = \pi_{\text{policy}}(a \mid s_t) \tag{3}$$

where $\hat{Q}(s_t, a)$ represents the estimated state-action value, indicating the expected future reward for action a in state s_t. $\mathcal{T}_k$ represents the MCTS tree constructed in the k-th round. In addition, c_{puct} is a constant that helps balance exploration and exploitation. The probability $\pi_{policy}(a|s_t)$ denotes the likelihood of selecting action a in the state s_t as determined by the policy model. LLM is a policy model that provides the likelihood score of taking action a (choosing relation) in a state s_t (current entity). Detailed examples of LLM prompts are provided in Appendix D. Additionally, $N_{\text{parent}}(a)$ and $N(s_t, a)$ refer to the visit counts of the parent node and the state-action pair, respectively.

The selection phase continues until a leaf node is reached, ensuring that the most promising nodes are explored based on prior knowledge and accumulated statistics.

Expansion. Upon reaching a leaf node, it is expanded by generating all possible actions from that node. Each action corresponds to a transition to a new state. For the knowledge graph, the possible actions are the relations connected to the current entity, leading to new entities. Consequently, all unexplored entities connected to the current entity are added to the search tree as leaf nodes of the current node.

Evaluation. The newly expanded nodes are evaluated using an evaluation function integrating future rewards, state relevance, and actual outcomes. The value function is defined as

$$\hat{V}(s_t) = (1 - \lambda - \mu) \cdot V_{\text{roll-out}}(s_t) \\ + \lambda \cdot V_{\text{critical}}(s_t) + \mu \cdot r(\mathbf{a}^{(i)}_{t \geq t}, \mathbf{s}^{(i)}_{t' > t} \mid \mathbf{s}_t) \tag{4}$$

where $V_{\text{roll-out}}(s_t)$ is generated by the roll-out model, which is based on a pre-trained LLM and predicts the future reward (Q value) of the state through simulated future steps. $V_{\text{critical}}(s_t)$ is computed by the critical model, which uses a BERT model (Reimers and Gurevych, 2019) to evaluate whether the entity at the current node is a semantically reasonable answer to the question. The term $r\left(\mathbf{a}^{(i)}_{t' \geq t}, \mathbf{s}^{(i)}_{t' > t} \mid \mathbf{s}_t\right)$ represents the actual outcome, where $\mathbf{a}^{(i)}_{t' \geq t}$ and $\mathbf{s}^{(i)}_{t' > t}$ represent the actions and states in the i-th simulation sampled by the policy model. $r(\cdot \mid \mathbf{s}_t)$ is the reward of the outcome in one simulation from the state $\mathbf{s}_t$. The parameter μ is set as an indicator function $I(s_t)$. The reward is used if the expanded node is terminal; otherwise, the model relies on the estimated value. During inference, μ is set to 0 to ensure consistency, always relying on the value model for node evaluation, including terminal nodes.

Backpropagation. After evaluation, the value of the leaf node is propagated back through the tree, updating the value estimates and visit counts for all ancestor nodes along the path. The backpropagation process ensures that the information gained from evaluating the leaf nodes influences decision-making at higher levels of the tree. The updated rules are as follows:

$$N(\mathbf{s},\mathbf{a}) \leftarrow N(\mathbf{s},\mathbf{a}) + 1,$$

$$\hat{Q}(\mathbf{s},\mathbf{a}) \leftarrow \frac{1}{N(\mathbf{s},\mathbf{a})} \sum_{j=1}^{i} \mathbb{I}_{\mathbf{s},\mathbf{a}\rightarrow\mathbf{s}_t} \hat{V}(\mathbf{s}_t)^{(j)} \tag{5}$$

3.3 Model Training

The proposed method leverages a pre-trained LLM with extensive world knowledge and experience as the policy model. A smaller evaluation model is trained to ensure efficiency and accuracy in domain-specific knowledge. The evaluation model consists of the critical and roll-out models, working together to comprehensively evaluate the states during MCTS.

Initialization. The parameters of the evaluate model are randomly initialized. At the start, the evaluate model tends to predict values close to 0 due to its random initialization. As MCTS simulations progress, rewards from terminal nodes (±1) are back-propagated to their parent nodes. With an increasing number of simulations N, the estimated values $\hat{Q}$ of intermediate nodes converge towards the underlying true values within the range of $[-1,\ 1]$.

Roll-out Model. From the tree $\mathcal{T}_k$ constructed in the k-th round of MCTS, solution paths corresponding to terminal nodes with predicted answers are sampled, denoted as x, along with the value estimation of each node along these paths. A loss function is then applied to update the roll-out model. The loss function is defined as:

$$\arg \min_{roll-out_{(k)}} \sum_{t=1}^{T(x)} \left\| V_{\text{roll-out}_{(k)}}(s_t) + \lambda \cdot V_{\text{critical}}(s_t) - \hat{V}_{(k-1)}(s_t) \right\|^2 \tag{6}$$

In this formulation, the loss function captures the difference between the predicted value $V_{\text{roll-out}_{(k-1)}}(s_t) + \lambda \cdot V_{\text{critical}}(s_t)$ from the estimation model and the estimated value $\hat{V}(s_t)$ across all solution paths. $T(x)$ denotes the number of steps for reasoning path x. The last term $\hat{V}_{(k-1)}(s_t)$ can be calculated as follows:

$$\hat{V}_{(k-1)}(s_t) = (1-\lambda-\mu) \cdot V_{\text{roll-out}_{(k-1)}}(s_t) + \lambda \cdot V_{\text{critical}}(s_t) + \mu \cdot r(\mathbf{a}^{(i)}_{t\geq t}, \mathbf{s}^{(i)}_{t'>t} \mid \mathbf{s}_t) \tag{7}$$

where $\hat{V}_{(k-1)}(s_t)$ denotes the estimated value function from the $(k-1)$-th iteration as defined in Equation (4), which incorporates the updated evaluation model from the previous iteration.

Critical Model. The critical model checks if the entity contextually fits the question by assigning a score based on semantic suitability. When the question is first input into the policy model, it is simultaneously asked to provide three possible answers. Based on a fixed synonym-judging BERT model, the critical model calculates the semantic similarity to these three possible answers and takes the average value as V_{critical}. The integration ensures that the evaluation model's predictions are informed by both the critical model's future reward predictions and the semantic relevance.

Iterative Training Process. With the updated roll-out model $V_{roll-out_{(k+1)}}$, the next round of MCTS is initiated. This iterative training and MCTS refinement process continues, enhancing the roll-out model's accuracy and effectiveness over multiple rounds.

Hop-level Beam Search for Efficient Inference. Considering the **computational complexity** and **reasoning speed**, we propose a hop-level beam search (HBS) as an efficient inference plugin that leverages trained RwT model without iterative MCTS, achieving comparable accuracy while substantially reducing inference time. Details of HBS and its performance analysis are provided in Appendix C.

4 Experiment

4.1 Experimental Setup

Datasets. Following existing work on KGQA (Jiang et al., 2023b), this paper evaluated the reasoning performance of RwT on two benchmark KGQA datasets: WebQuestionSP (WebQSP) (Yih et al., 2016) and Complex WebQuestions (CWQ) (Talmor and Berant, 2018). These datasets include multi-hop KG reasoning problems with up to 4 hops, and both are based on the Freebase (Bollacker et al., 2008) knowledge graph. The details of the datasets are described in Appendix B.1.

Evaluation Protocol. To evaluate our reasoning approach, we frame the task as a ranking problem inspired by prior studies (Sun et al., 2018). Specifically, for each question, we rank the candidate entities by their answer scores and then determine the accuracy of the highest-ranked answer using the Hits@1 metric. Acknowledging that questions can have multiple correct answers, we also incor-

Type	Methods	WebQSP		CWQ	
		Hits@1	F1	Hits@1	F1
Embedding & Semantic Parsing	KV-Mem (Miller et al., 2016)	46.7	34.5	18.4	15.7
	EmbedKGQA (Saxena et al., 2020)	66.6	-	45.9	-
	NSM (He et al., 2021)	68.7	62.8	47.6	42.4
	TransferNet (Shi et al., 2021)	71.4	-	48.6	-
	KGT5 (Saxena et al., 2022)	56.1	-	36.5	-
	SPARQL (Sun et al., 2020)	-	-	31.6	-
	QGG (Lan and Jiang, 2020)	73.0	73.8	36.9	37.4
	ArcaneQA (Gu and Su, 2022)	-	75.3	-	-
	RnG-KBQA (Ye et al., 2022)	-	76.2	-	-
Retrieval	GraftNet (Sun et al., 2018)	66.4	60.4	36.8	32.7
	PullNet (Sun et al., 2019)	68.1	-	45.9	-
	SR+NSM (Zhang et al., 2022a)	68.9	64.1	50.2	47.1
	SR+NSM+E2E (Zhang et al., 2022a)	69.5	64.1	49.3	46.3
LLMs	Flan-T5-xl (Chung et al., 2024)	31.0	-	14.7	-
	Alpaca-7B (Taori et al., 2023)	51.8	-	27.4	-
	LLaMA2-Chat-7B (Touvron et al., 2023a)	64.4	-	34.6	-
	ChatGPT (OpenAI, 2023a)	66.8	-	39.9	-
	GPT4o (OpenAI, 2023b)	82.2	-	-	-
	ChatGPT+CoT (Wei et al., 2022)	75.6	-	48.9	-
LLMs+KGs	KD-CoT (Wang et al., 2023a)	68.6	52.5	55.7	-
	UniKGQA (Jiang et al., 2023c)	75.1	70.2	50.7	48.0
	DECAF (DPR+FiD-3B) (Yu et al., 2023)	82.1	78.8	-	-
	StructureGPT (Jiang et al., 2023a)	72.6	-	-	-
	ReasoningLM (Jiang et al., 2023b)	78.5	71.0	69.0	64.0
	RoG (Luo et al., 2023)	85.7	70.8	62.6	56.2
	RwT (Ours)	**87.0**	**79.7**	**72.4**	**66.7**

Table 1: Performance comparison with different baselines on the two KGQA datasets.

porate the F1 metric for a more comprehensive assessment.

Baselines. This study considers the following four types of baseline methods for performance comparison: (1) Traditional Embedding & Semantic Parsing methods: KV-Mem (Miller et al., 2016), EmbedKGQA (Saxena et al., 2020), NSM (He et al., 2021), TransferNet (Shi et al., 2021), KGT5 (Saxena et al., 2022), SPARQL (Sun et al., 2020), QGG (Lan and Jiang, 2020), ArcaneQA (Gu and Su, 2022), RnG-KBQA (Ye et al., 2022). (2) Retrieval methods: GraftNet (Sun et al., 2018), PullNet (Sun et al., 2019), SR+NSM (Zhang et al., 2022a), SR+NSM+E2E (Zhang et al., 2022a). (3) Direct reasoning with LLMs: Flan-T5-xl (Chung et al., 2024), Alpaca-7B (Taori et al., 2023), LLaMA2-Chat-7B (Touvron et al., 2023a), ChatGPT (OpenAI, 2023a), GPT4o (OpenAI, 2023b), ChatGPT+CoT (Wei et al., 2022). (4) LLMs + KGs methods: KD-CoT (Wang et al., 2023a), UniKGQA (Jiang et al., 2023c), DECAF (DPR+FiD-3B) (Yu et al., 2023), StructureGPT (Jiang et al., 2023a), ReasoningLM (Jiang et al., 2023b), RoG (Luo et al., 2023). The details of the baselines are described in Appendix B.2.

4.2 Implementation Details

Roll-out Model. For the roll-out model in RwT, we use LLaMA3-Base-8B (MetaAI, 2024) as the base PLM. We iteratively train this model on the training splits of WebQSP and CWQ, as described in Section 3.3. The training employs a learning rate 4e-5 and the AdamW optimizer (Loshchilov and Hutter, 2019).

Critical Model. We use a finetuned Sentence-BERT (Reimers and Gurevych, 2019) to measure the semantic similarity between the entities found in the current search and the answers predicted by the policy model.

Training Data Generation via MCTS. The training procedure for this method's value model utilizes an iterative approach with R = 4 rounds. The process continues until the improvement between rounds becomes minimal. The experiment generates 8 Monte Carlo search trees per question-answer pair during each round. The method extracts up to 5 correct and 5 incorrect reasoning paths from these trees, maintaining an approximate 1:1 ratio of positive to negative examples. More details of implementations and hyperparameter settings are described in Appendix B.3.

4.3 Main Result

We used ChatGPT as the policy model and compared RwT with other baselines on two datasets. As shown in Table 1, the results show that RwT outperforms all baselines across all datasets and metrics. Compared to the current SOTA models Reason-

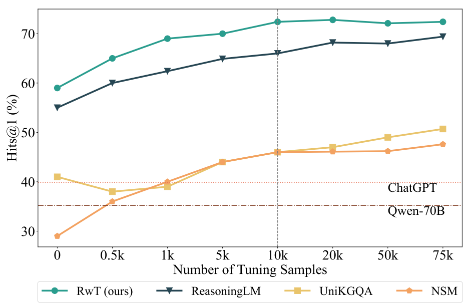


Figure 5: The Hits@1 score of our RwT compared with three strong baselines (i.e., ReasoningLM, UniKGQA, and NSM) on CWQ when tuning with various numbers of samples

Methods	WebQSP		CWQ	
	Hits@1	Recall	Hits@1	Recall
ChatGPT	66.77	49.27	39.9	35.07
+CoT	75.6	-	48.9	-
+RwT	**87**	**78.6**	**72.4**	**59.51**
Qwen-70B	68.43	54.29	35.23	30.45
+RwT	**79.23**	**68.72**	**72.36**	**57.95**
Alpaca-7B	51.78	33.65	27.44	23.62
+RwT	**84.83**	**83.56**	**68.35**	**57.34**
LLaMA2-Chat-7B	64.37	44.61	33.6	29.91
+RwT	**87.34**	**79.54**	**70.24**	**60.32**

Table 2: Effects of integrating the inference module of RwT with different LLMs for reasoning.

ingLM and RoG, RwT shows an average performance improvement of 9.81% and 5.77%, respectively. Additionally, unlike these SOTA models, RwT does not require complete retraining of the LLM. It only needs finetuning of a smaller PLM, which can then be plugged into any LLM. On the CWQ dataset, which has a higher average number of hops from question to answer, our model performs significantly better than other models. The results indicate that RwT is particularly effective in handling multi-hop deep knowledge reasoning tasks with long logical chains.

Traditional embedding and semantic parsing methods generally show limited performance. On the other hand, retrieval-based methods, which convert structured KG information into text, show significant performance improvements on more straightforward datasets like WebQSP but are limited to the more complex CWQ dataset.

Direct reasoning with LLMs has performed comparably to some supervised learning baselines (e.g., ChatGPT, LLaMA). However, these models still struggle with multi-hop reasoning tasks. By integrating RwT as a plug-and-play module with ChatGPT, we observed an 81.45% improvement in Hits@1 on the CWQ dataset.

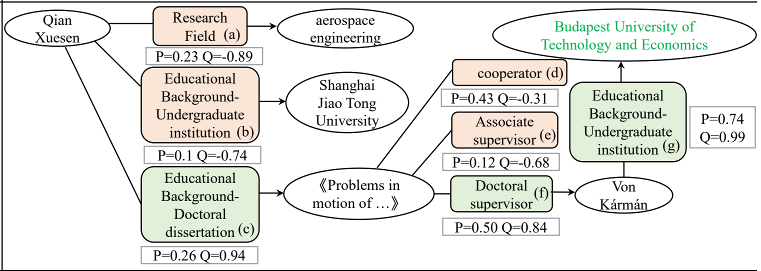
The overall performance of LLMs+KG methods surpasses other types, demonstrating the effectiveness of combining LLMs and KGs for KGQA tasks. Among these methods, PLM-based approaches (e.g., RoG and ReasoningLM) generally perform better than LLM-as-query-generator methods (e.g., StructureGPT) and LLM-as-retriever methods (e.g., KD-CoT).

4.4 Plug and Play Performance

This section evaluates the performance improvement of integrating RwT as a module into different LLMs. RwT is trained using the same settings and methods as before, with ChatGPT as the policy model on the training splits of the two datasets. After training, we replace the policy model with different LLMs, keeping other settings unchanged. As shown in Table 2, integrating RwT as a plugin with other LLMs results in an average improvement of 55.88% and 66.33% in Hits@1 for ChatGPT and Qwen, respectively. Even when combined with the most miniature model, Llama-2 (7B parameters), RwT's performance is significantly better than larger LLMs. Additionally, this combination shows a 31.57% improvement compared to ChatGPT+COT. These results indicate that finetuning a smaller LLM as the evaluation model with the RwT method effectively enhances LLM reasoning performance, especially in scenarios involving external knowledge not covered in the LLM's training data. Additionally, RwT's consistent performance boost across LLMs of varying qualities and scales indicates its ability to significantly reduce dependency on LLMs' background knowledge, mitigating the impact of domain-specific knowledge biases or deficiencies in KG+LLM reasoning methods.

4.5 Tuning Efficiency

In this section, we investigate the finetuning efficiency of different models. As shown in Figure 5, our method achieves SOTA performance for KGs+LLMs with minimal parameter tuning. As the tuning increases, the model's performance improves and eventually stabilizes. Compared to other models requiring full-scale retraining, RwT requires approximately 80% less tuning data to reach a stable state. Our method only needs to finetune the PLMs to evaluate the current state, which is a single-step reasoning task. In contrast, other methods need to train LLMs to fully master multi-hop reasoning capabilities, requiring more parameters for tuning. This result suggests that our method provides a promising approach to addressing the challenges of LLMs in multi-hop reasoning

Question	Where did Dr. Qian Xuesen's doctoral advisor attend school?
Answer	Budapest University of Technology and Economics
Chat-GPT	Dr. Qian Xuesen's alma mater is Shanghai Jiao Tong University.
GPT-4o	Dr. Qian Xuesen's PhD advisor was Theodore von Kármán. Von Kármán was a renowned aerodynamicist and aerospace engineer who played a significant role in guiding Qian Xuesen's academic development and career. During his time at the California Institute of Technology, Qian Xuesen conducted many important research projects under von Kármán's supervision.
RwT	

Qian Xuesen
Research Field (a)
P=0.23 Q=-0.89
aerospace engineering
Educational Background-Undergraduate institution (b)
P=0.1 Q=-0.74
Shanghai Jiao Tong University
Educational Background-Doctoral dissertation (c)
P=0.26 Q=0.94
《Problems in motion of …》
cooperator (d)
P=0.43 Q=-0.31
Associate supervisor (e)
P=0.12 Q=-0.68
Doctoral supervisor (f)
P=0.50 Q=0.84
Von Kármán
Educational Background-Undergraduate institution (g)
P=0.74 Q=0.99
Budapest University of Technology and Economics

Figure 6: Case study of different knowledge graph reasoning Methods.

L-PM	MCTS	ROM	CM	Hits@1
✓	×	×	×	53.1
✓	✓	×	×	59.2
×	✓	✓	×	63.9
✓	✓	✓	×	69.6
✓	✓	✓	✓	72.4

Table 3: Ablation studies on the CWQ test set of various components of RwT, including LLM as policy model (L-PM), MCTS, roll-out model (ROM), and critical model (CM).

tasks with more efficient training.

4.6 Ablation Study

We assess the effectiveness of each component in RwT and report the results on CWQ in Table 3. When the method only includes LLMs as the sole component, it degrades into a greedy strategy where the LLM selects the most likely option based on expandable relations from the current entity. This strategy achieved a Hits@1 performance of 53.1%, slightly higher than the ChatGPT+CoT strategy. This result indicates that having LLMs reason based on the graph information alone helps supplement external knowledge, providing LLMs with more information for reasoning and improving accuracy. Adding MCTS modestly improves accuracy to 59.2%, demonstrating the effectiveness of mapping KGQA as a Markov Decision Process. Without LLMs as the policy model, using a random strategy with MCTS and a trained roll-out model further enhances performance. A more significant improvement is observed when combining all three components, boosting accuracy to 69.6%. Finally, integrating the critical model with the other components yields the best performance for this task.

4.7 Case Study

In this section, we illustrate a case study in Figure 6. For clarity in presenting our method, we only illustrate a subset of nodes from the original Monte Carlo tree. As shown in Figure 6, our method found a correct relational path (c)-(f)-(g) through the MCTS search process, whereas the other solutions contained some errors. Without comprehensive background knowledge, the LLM predicted a high probability for relation (a). Subsequently, the value judgments and guidance from MCTS and the evaluation model corrected this misjudgment with a negative Q value. Similar corrections occurred in the choice between correct relation (f) and incorrect relations (d) and (e). On the other hand, we can see that ChatGPT suffers from both a lack of knowledge and misinterpretation of the question semantics, incorrectly identifying the initial entity's undergraduate alma mater as the Ph.D. advisor's alma mater. Besides, GPT-4o, through the reasoning, found the Ph.D. advisor of the initial entity and provided a detailed reasoning process, but still could not answer the question due to the lack of knowledge.

5 Conclusion

This paper introduces reasoning with trees (RwT), an innovative framework that enhances LLM reasoning by integrating internal knowledge with external knowledge graph data. RwT employs an MCTS framework to address data scarcity and multi-step reasoning complexity through iterative refinement. This process dynamically updates the LLM's understanding with reliable external information, improving accuracy and interpretability. Experimental results on benchmark datasets show that RwT significantly boosts LLM performance without requiring extra data annotations.

6 Limitation

RwT significantly outperforms leading models like GPT-4 and ChatGPT, highlighting its potential for self-improvement and more faithful reasoning. Despite the significant advancements demonstrated by the proposed reasoning with trees framework in improving the reasoning capabilities of LLMs with knowledge graph integration, several limitations remain.

- While RwT improves interpretability and accuracy by integrating external KG data, it may still struggle with questions involving high levels of ambiguity or uncertainty. The framework relies on the assumption that the knowledge graph contains accurate and relevant information, which may not always be the case. In scenarios where the knowledge graph is incomplete or contains conflicting information, RwT's performance may degrade.
- The current implementation of RwT has been primarily evaluated on specific benchmark datasets (WebQSP and CWQ) focused on multi-hop KG reasoning tasks. The generalization of RwT to other domains or types of knowledge graphs (e.g., domain-specific KGs) has not been extensively tested. Future work should explore the adaptability of RwT across various domains and investigate domain-specific optimizations.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants No. 61966038 and 62266051. The authors would like to thank the anonymous reviewers for their constructive comments.

References

Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. 2024. Can knowledge graphs reduce hallucinations in llms? : A survey. *Preprint*, arXiv:2311.07914.

Peter Auer. 2000. Using upper confidence bounds for online learning. In *Proceedings 41st annual symposium on foundations of computer science*, pages 270–279. IEEE.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Lang Cao. 2024. Graphreason: Enhancing reasoning capabilities of large language models through a graph-based verification approach. *Preprint*, arXiv:2308.09267.

Antoine Chaffin, Vincent Claveau, and Ewa Kijak. 2022. PPL-MCTS: Constrained textual generation through discriminator-guided MCTS decoding. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2953–2967, Seattle, United States. Association for Computational Linguistics.

Kaiyuan Chen, Jin Wang, and Xuejie Zhang. 2024. Learning to reason via self-iterative process feedback for small language models. *Preprint*, arXiv:2412.08393.

Kaiyuan Chen, Jin Wang, and Xuejie Zhang. 2025. Mathematical reasoning via multi-step self questioning and answering for small language models. In *Natural Language Processing and Chinese Computing*, pages 81–93, Singapore. Springer Nature Singapore.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*.

Yingqiang Ge, Wenyue Hua, Kai Mei, Juntao Tan, Shuyuan Xu, Zelong Li, Yongfeng Zhang, et al. 2024. Openagi: When llm meets domain experts. *Advances in Neural Information Processing Systems*, 36.

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.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 553–561.

Shaoxiong Ji, Shirui Pan, E. Cambria, P. Marttinen, and Philip S. Yu. 2020. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33:494–514.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating llm hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023a. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yaliang Li, and Ji-Rong Wen. 2023b. Reasoninglm: Enabling structural subgraph reasoning in pre-trained language models for question answering over knowledge graph. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3721–3735.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. *arXiv preprint arXiv:2212.00959*.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023c. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. *Preprint*, arXiv:2212.00959.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9410–9421.

Kaori Kumagai, Ichiro Kobayashi, Daichi Mochihashi, Hideki Asoh, Tomoaki Nakamura, and Takayuki Nagai. 2016. Human-like natural language generation using monte carlo tree search. In *Proceedings of the INLG 2016 Workshop on Computational Creativity in Natural Language Generation*, pages 11–18.

Kaori Kumagai, Ichiro Kobayashi, Daichi Mochihashi, Hideki Asoh, Tomoaki Nakamura, and Takayuki Nagai. 2018. Natural language generation using monte carlo tree search. *Journal of Advanced Computational Intelligence Vol*, 22(5).

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*. Association for Computational Linguistics.

Shiyang Li, Yifan Gao, Haoming Jiang, Qingyu Yin, Zheng Li, Xifeng Yan, Chao Zhang, and Bing Yin. 2023. Graph reasoning for question answering with triplet retrieval. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3366–3375.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

MetaAI. 2024. Introducing meta llama 3: The most capable openly available llm to date. Accessed: 2024-06-15.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

OpenAI. 2023a. Chatgpt: Optimizing language models for dialogue. Accessed: 2023-06-15.

OpenAI. 2023b. Gpt-4: The next generation of ai language models. Accessed: 2023-06-15.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

Martin L Puterman. 1990. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *Preprint*, arXiv:1908.10084.

Christopher D Rosin. 2011. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4498–4507.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Tiesunlong Shen, Jin Wang, and Xuejie Zhang. 2025. Knowledge distillation via adaptive meta-learning for graph neural network. *Information Sciences*, 689:121505.

Tiesunlong Shen, You Zhang, Jin Wang, and Xuejie Zhang. 2023. Graphs get personal: learning representation with contextual pretraining for collaborative filtering. *Applied Intelligence*, 53(24):30416–30430.

Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. Transfernet: An effective and transparent framework for multi-hop question answering over relation graph. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4149–4158.

Dong Shu, Tianle Chen, Mingyu Jin, Chong Zhang, Mengnan Du, and Yongfeng Zhang. 2024. Knowledge graph large language model (kg-llm) for link prediction. *Preprint*, arXiv:2403.07311.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. 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 2380–2390.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. Sparqa: skeleton-based semantic parsing for complex questions over knowledge bases. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8952–8959.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2022. Entailer: Answering questions with faithful and truthful chains of reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2078–2093.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.

Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Can chatgpt replace traditional kbqa models? an in-depth analysis of the question answering performance of the gpt llm family. In *International Semantic Web Conference*, pages 348–367. Springer.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023a. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. 2024.

Reasoning in token economies: Budget-aware evaluation of llm reasoning strategies. *arXiv preprint arXiv:2406.06461*.

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. *Preprint*, arXiv:2308.13259.

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

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023c. Self-knowledge guided retrieval augmentation for large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yeo Wei Jie, Ranjan Satapathy, Rick Goh, and Erik Cambria. 2024. How interpretable are reasoning explanations from prompting large language models? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2148–2164, Mexico City, Mexico. Association for Computational Linguistics.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Kang Liu, and Jun Zhao. 2024. Generate-on-graph: Treat llm as both agent and kg in incomplete knowledge graph question answering. *Preprint*, arXiv:2404.14741.

Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Mu-Nan Ning, Yu-Yang Liu, and Li Yuan. 2024a. Llm lies: Hallucinations are not bugs, but features as adversarial examples. *Preprint*, arXiv:2310.01469.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024b. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546.

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.

Wei Jie Yeo, Teddy Ferdinan, Przemyslaw Kazienko, Ranjan Satapathy, and Erik Cambria. 2024. Self-training large language models through knowledge detection. *Preprint*, arXiv:2406.11275.

Scott 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 Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*.

Wen-tau Yih, Matthew Richardson, Christopher 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.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. *arXiv preprint arXiv:2210.00063*.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2023. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. *Preprint*, arXiv:2210.00063.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022a. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022b. Greaselm: Graph reasoning enhanced language models for question answering. *arXiv preprint arXiv:2201.08860*.

Xinran Zhao, Hongming Zhang, Xiaoman Pan, Wenlin Yao, Dong Yu, and Jianshu Chen. 2024. Thrust: Adaptively propels large language models with external knowledge. *Advances in Neural Information Processing Systems*, 36.

Qi Zhu, Hao Wei, Bunyamin Sisman, Da Zheng, C. Faloutsos, Xin Dong, and Jiawei Han. 2020. Collective multi-type entity alignment between knowledge graphs. *Proceedings of The Web Conference 2020*.

Algorithm 1 Inference with MCTS (Where $\mathcal{T}_Q(\mathbf{s}_t, \mathbf{a})$) denotes the operation of retrieving Q-values of nodes from the tree $\mathcal{T}$)

```
Require: question $q(s_0)$, policy/roll-out/critical models $\pi_{policy}, V_{roll-out}, V_{critical}$, simulations $N$, max depth $T$.
  Build the complete tree $\mathcal{T}$ by running $\text{MCTS}_{\pi_{policy}, V_{roll-out}, V_{critical}}(s_0, N, T)$.
  $C = [s_0], t = 0$
  while $t < T$ and non-terminal path in $C$ do
    Initialize priority queue $C_{t+1}$
    for $s_t$ in $C$ do
      for $a$ in $\mathcal{T}_{children}(s_t)$ do
        $s_{t+1} = Cat[s_t, a]$
        Add $(s_{t+1}, \mathcal{T}_Q(s_t, \mathbf{a}))$ to $C_{t+1}$
      end for
    end for
    $C \leftarrow C_{t+1}$
  end while
  return Top-1 of $C$
```

A Related Work

A.1 Question Answering over Knowledge Graph

Multi-hop Knowledge Graph Question Answering (KGQA) involves deriving answers by traversing multiple edges in a Knowledge Graph (KG). The goal is to locate answer entities that may be several hops away from the subject entities in a large-scale KG. Existing approaches to KGQA can be categorized into embedding-based methods, retrieval-augmented methods, and semantic parsing methods.

Embedding-based methods such as KV-Mem (Miller et al., 2016) and EmbedKGQA (Saxena et al., 2020), utilize vector representations of entities and relations to facilitate reasoning. These methods often employ sequential models to simulate multi-hop reasoning. For example, QA-GNN (Yasunaga et al., 2021) and Greaselm (Zhang et al., 2022b) leverage graph neural networks (Shen et al., 2023, 2025) to incorporate the graph structure into the reasoning process. Despite their effectiveness, these methods require designing specialized architectures, limiting their flexibility and generalizability.

Retrieval-augmented methods aim to enhance reasoning by retrieving relevant subgraphs from KGs. Early approaches (Sun et al., 2018, 2019) utilized algorithms like PageRank and random walk to retrieve subgraphs. However, these methods often failed to capture the semantic nuances of the questions, leading to noisy retrieval results. More recent approaches by Zhang et al. (2022a) and Li et al. (2023) employ more sophisticated retrieval techniques like relation paths-based subgraph retrieval and BM25 to improve performance. Nevertheless, these methods still discard much of the KG's structural information, resulting in suboptimal outcomes.

Semantic parsing methods convert natural language questions into executable queries. Works proposed by Sun et al. (2020) and Lan and Jiang (2020) dynamically generate and execute these queries. However, these methods are heavily reliant on the quality of the generated queries, with non-executable queries yielding no answers.

Unlike previous methods, our approach focuses on harnessing the intrinsic knowledge within LLMs to iteratively enhance their understanding and utilization of KGs. By integrating question-answer pairs, our method activates the latent knowledge in LLMs, allowing them to autonomously refine their reasoning process, akin to human learning and adaptation.

A.2 Monte-Carlo Planning for Language

Monte-Carlo Planning (MCP) has been extensively used in strategic games (Silver et al., 2017; Schrittwieser et al., 2020) but its application in natural language processing (NLP) remains underexplored. Previous works have attempted to apply MCP to language generation tasks, such as decoding during text generation (Kumagai et al., 2016, 2018; Chaffin et al., 2022). These methods primarily focus on improving the quality of generated text by exploring various decoding paths.Our work is among the few that formulate question answering (QA) as a decision-making problem, leveraging MCP to optimize the QA process. Which mirrors human reasoning by iteratively refining the reasoning paths based on new insights and corrections.

A.3 Reasoning with Language Models

Recent advancements have shown that Large LLMs can effectively perform reasoning tasks by generating intermediate reasoning steps (Wang et al., 2024; Chen et al., 2025, 2024). Techniques like Chain-of-Thought prompting enable LLMs to break down complex tasks into simpler steps, thereby improving their reasoning capabilities. Tree of thoughts (Yao et al., 2024b) and graph of thoughts (Besta

Datasets	#Train	#Test	Max #hop
WebQSP	2,826	1,628	2
CWQ	27,639	3,531	4

Table 4: Statistics of datasets.

et al., 2024) further extend this concept by structuring the reasoning process as a tree or graph, exploring multiple reasoning paths simultaneously.

Despite these advances, issues like hallucinations and lack of factual knowledge still affect the faithfulness of the reasoning process (Tafjord et al., 2022; Creswell and Shanahan, 2022). Methods such as ReACT (Yao et al., 2023) and Entailer (Tafjord et al., 2022) aim to mitigate these problems by incorporating interaction with external knowledge sources and introducing verification mechanisms for the reasoning steps. Our approach enhances these methods by incorporating Monte-Carlo planning to generate and evaluate reasoning steps. By using a deterministic corpus and real-time knowledge retrieval from KGs, our method reduces the likelihood of hallucinations and improves the faithfulness of the reasoning process. This allows for a more robust and reliable reasoning capability in LLMs, closely resembling human problem-solving strategies.

B Experiment Details

B.1 Datasets

In this section, we provide more relevant details about the datasets we used in experiments. We utilized two significant benchmark datasets: WebQuestionSP (WebQSP) (Yih et al., 2016) and Complex WebQuestions (CWQ) (Talmor and Berant, 2018). These datasets are critical for evaluating the effectiveness of our proposed methods in the field of knowledge graph question answering. We provide a comprehensive description of each dataset and the specific adaptations we made for our experiments bellow, and the statistics of these datasets are provided in Table 4.

WebQuestionsSP (WebQSP) WebQuestionsSP consists of 4,737 natural language questions, each answerable by entities within a maximum of 2 hops from the topic entity in the Freebase knowledge graph. We adopted the train, validation, and test splits as defined by GraftNet [Sun et al., 2018] to ensure consistency with previous research and facilitate fair comparisons.

Complex WebQuestions 1.1 (CWQ) Complex WebQuestions 1.1 is built upon WebQSP, introducing a higher level of complexity. It includes questions that are more intricate and involve additional constraints or extended entities to restrict possible answers. The answers in CWQ are within a maximum of 4 hops from the topic entity on the Freebase knowledge graph

For our experiments, we followed the training and test splits outlined in previous works such as those by Sun et al. (2018) and Jiang et al. Jiang et al. (2022). This approach ensures that our results are directly comparable to established benchmarks. Specifically, we constructed subgraphs of the Freebase knowledge graph by extracting all triples that lie within the maximum reasoning hops of question entities in WebQSP and CWQ. This reduction in the size of the knowledge graph aligns with methodologies employed in earlier studies[He et al., 2021; Jiang et al., 2022].During the instruction-tuning phase, we utilized the training splits from both WebQSP and CWQ. To optimize planning, we generated supervisory signals by extracting the shortest paths connecting the questions and answers. These paths were then used in the retrieval-reasoning optimization phase, where they were fed into our model alongside the questions to predict the correct answers.

B.2 Baselines

For comparison, our baseline set consists of the following methods which are presented below:

(1) Traditional Embedding & Semantic Parsing Methods

- **KV-Mem**: KV-Mem (Miller et al., 2016) employs a Key-Value memory network to store triples and perform multi-hop reasoning by iteratively operating on the memory, enabling effective question answering over knowledge graphs.

- **EmbedKGQA**: EmbedKGQA (Saxena et al., 2020) addresses KG reasoning as a sequential link prediction problem by leveraging the embeddings of entities and questions, facilitating accurate answer generation.

- **NSM**: NSM (He et al., 2021) uses a sequential model to replicate the multi-hop reasoning process, thereby enhancing the ability to infer answers from knowledge graphs.

- **TransferNet**: TransferNet (Shi et al., 2021) employs a graph neural network to capture the

Hyperparameter	**Value**
c_{puct}	1.25
R	4
λ	0.15
Simulations N	40
Temperature	0.6
max depth (max steps) T	5
Batch size	8
Optimizer type	AdamW (Loshchilov and Hutter, 2017)
Learning rate	4e-5
lr scheduler type	cosine
Warmup ratio	0.03
Epochs	20
Weight decay	0.0001

Table 5: Hyperparameter settings for the RwT model.

relationship between entities and questions, thus aiding in the reasoning process.

- **KGT5**: KGT5 (Saxena et al., 2022) fine-tunes a sequence-to-sequence framework on knowledge graphs, generating answers based on the input questions.
- **SPARQL**: SPARQL (Sun et al., 2020) introduces a novel skeleton grammar to represent the high-level structure of complex questions using language modes.
- **QGG**: QGG (Lan and Jiang, 2020) generates a query graph for a question by concurrently adding constraints and extending relation paths.
- **ArcaneQA**: ArcaneQA (Gu and Su, 2022) dynamically generates the query based on results from previous steps, ensuring more accurate and contextually relevant answers.
- **RnG-KBQA**: RnG-KBQA (Ye et al., 2022) first enumerates all possible queries and then ranks them to produce the final output, optimizing the query selection process.

(2) Retrieval Methods

- **GraftNet**: GraftNet (Sun et al., 2018) retrieves relevant subgraphs from knowledge graphs using entity linking, facilitating targeted question answering.
- **PullNet**: PullNet (Sun et al., 2019) combines an LSTM with a graph neural network to retrieve a question-specific subgraph, enhancing retrieval accuracy.
- **SR+NSM**: SR+NSM (Zhang et al., 2022a) employs relation-path retrieval to obtain subgraphs for multi-hop reasoning, improving the efficiency of the reasoning process.
- **SR+NSM+E2E**: SR+NSM+E2E (Zhang et al., 2022a) adopts an end-to-end training strategy to jointly train the retrieval and reasoning modules of SR+NSM, achieving better integration and performance.

(3) Direct Reasoning with LLMs

- **Flan-T5-xl**: Flan-T5-xl (Chung et al., 2024) is an enhanced version of T5 models, instruction fine-tuned on a mixture of tasks to improve performance on a wide range of natural language processing applications.
- **Alpaca-7B**: Alpaca-7B (Taori et al., 2023) is based on LLaMA and fine-tuned on an instruction-following dataset, enhancing its ability to generate contextually appropriate responses.
- **LLaMA2-Chat-7B**: LLaMA2-Chat-7B (Touvron et al., 2023b) is a large language model optimized for dialogue purposes, facilitating more natural and coherent conversational interactions.

- **ChatGPT**: ChatGPT (OpenAI, 2023a) is a powerful closed-source language model capable of following instructions to perform complex tasks with high accuracy and fluency.
- **GPT4o**: GPT4o (OpenAI, 2023b) is OpenAI's latest and most advanced model, offering improved generation of nuanced and contextually aware text.
- **ChatGPT+CoT**: ChatGPT+CoT (Wei et al., 2022) employs the Chain-of-Thought prompting technique to enhance the reasoning capabilities of ChatGPT, resulting in more accurate and logical outputs.

(4) LLMs + KGs Methods

- **KD-CoT**: KD-CoT (Wang et al., 2023b) integrates knowledge retrieval from knowledge graphs to generate faithful reasoning plans for large language models, improving reasoning accuracy.
- **UniKGQA**: UniKGQA (Jiang et al., 2022) unifies the graph retrieval and reasoning process into a single model using LLMs, achieving state-of-the-art performance on KGQA tasks.
- **DECAF (DPR+FiD-3B)**: DECAF (Yu et al., 2022) combines semantic parsing and LLM reasoning to jointly generate answers, attaining high performance on KGQA tasks.
- **StructureGPT**: StructureGPT (Jiang et al., 2023a) integrates the structural information of knowledge graphs with LLMs, enabling more accurate and interpretable reasoning.
- **ReasoningLM**: ReasoningLM (Jiang et al., 2023b) combines the strengths of LLMs and knowledge graphs to perform detailed and accurate reasoning across complex queries.
- **RoG**: RoG (Luo et al., 2023) synergizes LLMs with knowledge graphs using a planning-retrieval-reasoning framework to generate faithful and interpretable reasoning results, achieving state-of-the-art performance on KG reasoning tasks.

B.3 Parameter Details

For the solution generation via MCTS, we set $c_{puct} = 1.25$, and set round $R = 4$, in every round, we build 8 trees for each question-answer pair and randomly sample at most 5 correct and 5 incorrect solution processes. The ratio between positive and negative examples is approximately 1:1. For supervised fine-tuning, we set the learning rate of 4e-5, batch size of 8, the weight of the λ in evaluation function(3) is setted to 0.15, and train the model for 20 epochs. We employ the AdamW optimizer (Loshchilov and Hutter, 2017) and the cosine learning rate scheduler with the warmup rate set to 0.03. More hyperparameter details can be found in Table 5.

Algorithm 2 Hop-level Beam Search

```
Require: Beam sizes $B_1$, $B_2$, question $q$ ($s_0$), policy / value models $\pi_\theta$, $V_\phi$, max hops $T$.
  $C \leftarrow [s_0] \times B_1$, $t \leftarrow 0$ {Initialize candidates}
  while $t < T$ and non-terminal path in $C$ do
    Initialize priority queue $C_{t+1}$ {Max heap}
    for $s_t$ in $C$ do
      Sample $\{a^{(b)}\}_{b=1}^{B_2} \sim \pi_\theta(a|s_t)$ {LLM generates $B_2$ samples in parallel}
      for $b = 1$ to $B_2$ do
        $s_{t+1} \leftarrow$ Cat$[s_t, a^{(b)}]$
        Add $(s_{t+1}, V_\phi(s_{t+1}))$ to $C_{t+1}$ {$V_\phi(s_{t+1})$ predicted by value model}
      end for
    end for
    $C \leftarrow$ Top-$B_1$ of $C_{t+1}$
  end while
  return Top-1 of $C$ {Return top-1 as the final solution path}
```

C Hop-level Beam Search for Efficient Inference

C.1 Hop-level Beam Search

The hop-level beam search (HBS) is an efficient inference plugin designed to leverage the RwT model after it has been fully trained using the Monte Carlo Tree Search (MCTS) process. This approach allows for rapid inference without the need for iterative MCTS during the prediction phase. HBS integrates seamlessly with the RwT framework by utilizing the policy and value models that were learned and refined during the MCTS training process.

In the HBS approach, each hop corresponds to a transition in the knowledge graph, aligning with RwT's graph-based reasoning paradigm. The algorithm employs the trained policy model π_θ to gener-

Inference Strategy	**Hits@1 (%)**	**Avg. Inference Time (s)**	**Avg. Hops**
Greedy Decoding	47.23	1.1	3.34
MCTS ($B_1 = 1$)	72.40	9.3	4.57
HBS ($B_1 = 1, B_2 = 5$)	62.32	3.0	3.41
HBS ($B_1 = 2, B_2 = 5$)	68.45	2.6	2.95
HBS ($B_1 = 3, B_2 = 5$)	71.74	2.2	2.77
HBS ($B_1 = 4, B_2 = 5$)	71.80	2.9	2.43
HBS ($B_1 = 5, B_2 = 5$)	72.38	3.8	2.56

Table 6: Performance comparison of different inference strategies on the CWQ dataset.

ate candidate actions (hops) at each step, while the value model V_ϕ evaluates the quality of each candidate path, guiding the beam search process. This integration allows HBS to maintain the robust reasoning capabilities developed through MCTS training, while significantly reducing inference time.

The key advantage of HBS lies in its ability to perform efficient inference immediately after the MCTS-based training is complete, without requiring further adjustments or additional training steps. This makes HBS a valuable tool for scenarios where rapid deployment and quick inference are crucial, while still benefiting from the deep reasoning capabilities instilled by the MCTS training process.

The HBS algorithm proceeds as follows: Initially, it generates B_1 candidates for the first hop. For each subsequent hop, it expands B_2 candidates from each of the B_1 previous candidates, resulting in $B_1 \times B_2$ new candidates. These candidates are then evaluated using the value model, and the top B_1 candidates are retained for the next hop. This process continues until a terminal node is reached or a maximum number of hops is achieved. The specific steps of the HBS algorithm are detailed in Algorithm 2.

C.2 Performance Analysis

To evaluate the effectiveness and efficiency of HBS, we conducted a comprehensive analysis with various beam sizes, comparing it with greedy decoding and MCTS. Our experiments focused on the Complex WebQuestions (CWQ) dataset, which presents challenging multi-hop reasoning tasks. Table 6 presents the performance metrics for different inference strategies. The Hits@1 score measures the accuracy of the highest-ranked answer, while the average inference time and average number of hops provide insights into the computational efficiency and reasoning depth of each method.

As evidenced by the results, HBS demonstrates a favorable balance between accuracy and efficiency. The HBS configuration with B1 = 5 and B2 = 5 achieves a Hits@1 score of 72.38%, which is nearly equivalent to MCTS (72.40%) while significantly reducing the average inference time from 9.3 seconds to 3.8 seconds. This represents a substantial improvement in efficiency without compromising accuracy.

C.3 Efficiency Analysis

The hop-level beam search offers substantial improvements in inference efficiency compared to the iterative MCTS approach. The reduced computation time is particularly noteworthy, with HBS (B1 = 5, B2 = 5) achieving nearly the same accuracy as MCTS while reducing the average inference time by approximately 59.1%.

This efficiency gain can be attributed to several factors. First, HBS allows for parallel processing in the generation of candidates at each hop, which is particularly beneficial on multi-GPU systems. Second, the beam search approach scales more effectively with the complexity of the knowledge graph, as it does not require building and expanding a full Monte Carlo tree for each inference.

Furthermore, HBS offers flexibility in balancing inference speed and reasoning quality through the adjustment of beam sizes B1 and B2. As shown in Table 6, increasing B1 generally improves accuracy but with diminishing returns and at the cost of increased inference time. Interestingly, the inference time exhibits a U-shaped curve as B1 increases, with an optimal point at B1 = 3. This pattern likely reflects the specific characteristics of the CWQ dataset, where a moderate beam size strikes the best balance between finding efficient paths and computational overhead. The average number of hops tends to decrease with larger beam sizes, from 3.41 for B1 = 1 to 2.56 for B1 = 5. This suggests that HBS can find more direct paths to the answer when considering more candidates at each step, contributing to the overall efficiency of the method. However, this benefit is partially offset by the increased computational cost of evaluating more candidates for larger B1 values.

The data reveals an optimal point at B1 = 3, where HBS achieves the best balance between accuracy (71.74%) and inference time (2.2 seconds). This configuration offers a significant speed improvement over MCTS while maintaining competitive accuracy and a relatively low average number of hops (2.77).

In conclusion, HBS serves as an efficient plugin for the RwT model, offering a practical solution for scenarios where inference speed is crucial, while maintaining reasoning quality close to that achieved by the full MCTS approach. The ability to tune beam sizes provides users with the flexibility to optimize for their specific requirements, whether prioritizing accuracy or speed. The observed performance characteristics underscore the importance of carefully selecting beam sizes to achieve the desired balance between accuracy and computational efficiency in complex reasoning tasks, particularly considering dataset-specific characteristics that may influence the optimal configuration.

D Prompt Examples

Figure 7&8 show example prompt of RwT. For a KGQA question and the provided knowledge graph, starting from the initial reasoning state, RwT first presents the question and asks the policy model to provide the three most likely answers. Then it provides the policy model with the candidate relations of the question entity and requests the LLM to evaluate the likelihood that these relations will lead to the correct answer entity. In each subsequent iteration, RwT only asks the policy model to assess the likelihood that the candidate relations of the question entity will eventually lead to the correct answer entity.

Prompt example of RwT in the initial reasoning state:
[Human]: You are a helpful assistant that when I ask you a question, you will provide the three most likely answers as responses.

Here are a few examples:
<question>Who is the student that coined the Euler-Lagrange equation?</question>
<answer>[A: Leonhard Euler, B: Joseph Louis Lagrange, C: Johann Bernoulli]</answer>

<question>What is the name of the first ruler's youngest son whom Confucius served?</question>
<answer>[A: Bo Yu, B: Kong Li, C: Zi Lu]</answer>

Remaining examples . . .

Now let's answer the question below:
<question>Where did Dr. Qian Xuesen's doctoral advisor attend school? </question>

[Policy Model]: <answer>[A: Massachusetts Institute of Technology (MIT), B: California Institute of Technology (Caltech), C: Stanford University] </answer>

[Human]: Now let's play a deduction game. In this game, you need to find the answer by reasoning based on the given questions. The inference process is divided into multiple steps. At each step, I will provide you with an entity and its related relationships. Each time, you need to fully consider the question and the current state of the entity, evaluate the possibility of ultimately inferring the correct answer based on the given relationships.

Here are a few examples:
<question>Who is the student that coined the Euler-Lagrange equation?</question>
<current entity>[Euler-Lagrange equation]</current entity>
<current relations>[A: NamedAfter, B: DerivedFrom, C: HasFormulation] </current relations>
<answer>[A: NamedAfter 0.5, B: DerivedFrom 0.2, C: HasFormulation 0.1] </answer>

Remaining examples . . .

Now let's play the game below:
<question>Where did Dr. Qian Xuesen's doctoral advisor attend school? </question>
<current entity>[Qian Xuesen] </current entity>
<current relations>[A: CollaboratedWith, B: DerivedFrom, C: HasFormulation] </current relations>

Figure 7: Prompt example of RwT in the initial reasoning state

Prompt example of RwT after the initial reasoning state:

[Human]: Now let's play a deduction game. In this game, you need to find the answer by reasoning based on the given questions. The inference process is divided into multiple steps. At each step, I will provide you with an entity and its related relationships. Each time, you need to fully consider the question and the current state of the entity, evaluate the possibility of ultimately inferring the correct answer based on the given relationships.

Here are a few examples:
<question>Who is the student that coined the Euler-Lagrange equation?</question>
<current entity>[Euler-Lagrange equation]</current entity>
<current relations>[A: NamedAfter, B: DerivedFrom, C: HasFormulation] </current relations>
<answer>[A: NamedAfter 0.5, B: DerivedFrom 0.2, C: HasFormulation 0.1] </answer>

Remaining examples . . .

Now let's play the game below:
<question>Where did Dr. Qian Xuesen's doctoral advisor attend school? </question>
<current entity>[Theodore von Kármán] </current entity>
<past reasoning paths>[Qian Xuesen $\rightarrow$ Mentored $\rightarrow$ Theodore von Kármán] </past reasoning paths>
<current relations>[A: HasAchievement, B: HasEducation, C: CollaboratedWith] </current relations>

Figure 8: Prompt example of RwT after the initial reasoning state