Tree-of-Traversals: A Zero-Shot Reasoning Algorithm for Augmenting Black-box Language Models with Knowledge Graphs

Elan Markowitz[*,1,2], Anil Ramakrishna[1], Jwala Dhamala[1], Ninareh Mehrabi[1],
Charith Peris[1], Rahul Gupta[1], Kai-Wei Chang[1], and Aram Galstyan[1]

[1]Amazon AGI
[2]University of Southern California

Abstract

Knowledge graphs (KGs) complement Large Language Models (LLMs) by providing reliable, structured, domain-specific, and up-to-date external knowledge. However, KGs and LLMs are often developed separately and must be integrated after training. We introduce Tree-of-Traversals, a novel zero-shot reasoning algorithm that enables augmentation of black-box LLMs with one or more KGs. The algorithm equips a LLM with actions for interfacing a KG and enables the LLM to perform tree search over possible thoughts and actions to find high confidence reasoning paths. We evaluate on two popular benchmark datasets. Our results show that Tree-of-Traversals significantly improves performance on question answering and KG question answering tasks. Code is available at `https://github.com/amazon-science/tree-of-traversals`

1 Introduction

Large language models (LLMs) are used for a range of knowledge-intensive tasks such as information retrieval (Zhu et al., 2023b), summarization (Zhang et al., 2023), and question answering (Tan et al., 2023). Trained on large amounts of textual data, these models learn a wide breadth of information. However, LLMs suffer from several limitations – they produce hallucinated information (Ji et al., 2022; Bang et al., 2023), lack deep domain-specific knowledge (Pan et al., 2023b), and have a static knowledge cutoff when training ends.

Knowledge graphs (KGs) naturally complement LLM weaknesses. KGs contain up-to-date information covering general (Vrandečić and Krötzsch, 2014; Lehmann et al., 2015) and/or domain-specific topics (Abu-Salih, 2020; Liu et al., 2019; Zhu et al., 2017; Choi and Lee, 2019; Farazi et al., 2020) in highly structured and interpretable format. Augmenting an LLM's ability to reason and respond in natural language with an external KG's up-to-date knowledge presents a path toward a reliable and factual LLM.

The rise of powerful LLMs with new capabilities has renewed interest in combining LLMs with KGS. Numerous survey and position papers have recently emphasized their combined potential (Pan et al., 2023a; Zhu et al., 2023a; Yang et al., 2023; Pan et al., 2023b). Existing works augmented LLMs with KGs in multiple ways, such as integration into pretraining (Yasunaga et al., 2022), fine-tuning (Zhang et al., 2022), or later adaptation with subsequently trained components (Lin et al., 2019; Hu et al., 2022). All of these carry some limitations. In particular, training or fine-tuning a large scale LLMs is computationally expensive. In some cases, model weights are unavailable publicly. Finally, the largest KGs require their own servers and cannot be integrated in-memory with LLMs. Additionally, previous works do not consider the case of augmenting with multiple KGs.

An algorithm that allows the augmentation of a powerful black-box LLM with any number of internal or external KGs without training from scratch or fine-tuning the model is valuable. Such a zero-shot algorithm would enable several innovative use cases such as (1) customers using a black-box LLM API in conjunction with an internal domain-specific KG, (2) integration of personalized KGs into an LLM without the risks associated with training a model with such personal data, (3) integration of deep domain knowledge via an array of API accessible KGs (e.g., IMDb[1], MusicBrainz[2]).

We introduce **Tree-of-Traversals**, a novel algorithm that addresses the above issues by allowing the augmentation of any powerful LLM with arbitrary number of KGs in a zero-shot fashion. It re-
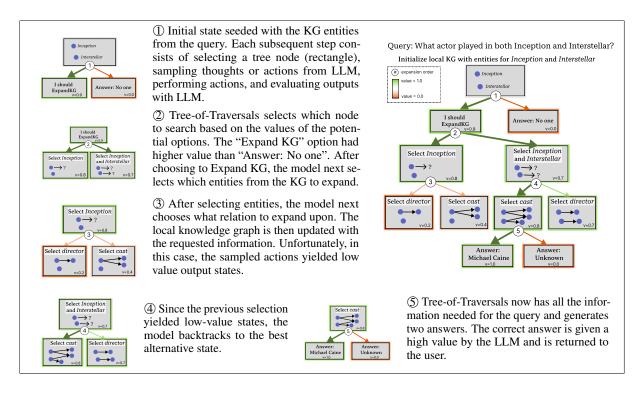
---

[*]Work done while interning at Amazon
Corresponding author: `esmarkow@usc.edu`

[1]IMDb API
[2]MusicBrainz API

① Initial state seeded with the KG entities from the query. Each subsequent step consists of selecting a tree node (rectangle), sampling thoughts or actions from LLM, performing actions, and evaluating outputs with LLM.

② Tree-of-Traversals selects which node to search based on the values of the potential options. The "Expand KG" option had higher value than "Answer: No one". After choosing to Expand KG, the model next selects which entities from the KG to expand.

③ After selecting entities, the model next chooses what relation to expand upon. The local knowledge graph is then updated with the requested information. Unfortunately, in this case, the sampled actions yielded low value output states.

④ Since the previous selection yielded low-value states, the model backtracks to the best alternative state.

⑤ Tree-of-Traversals now has all the information needed for the query and generates two answers. The correct answer is given a high value by the LLM and is returned to the user.

Figure 1: An example of how Tree-of-Traversals uses a KG interface for the query, "What actor played in both Inception and Interstellar?".

quires no training, functions with black-box access to the LLM, and works with any API accessible KG. Our contributions are:

1. Tree of traversals: A novel zero-shot algorithm for augmenting any powerful LLM with arbitrary number of KGs and enabling advanced KG reasoning using tree search.

2. Evaluation of Tree-of-Traversals on two question answering tasks: 2WikiMultiHop and QALD-10 and comparison with baselines.

3. Development of a new dataset to test combined reasoning over a general and a domain-specific KG, and evaluation of Tree-of-Traversals on this dataset.

We conduct detailed experiments on three models of varied sizes hosted on Amazon Bedrock and present detailed ablation studies.

2 Tree-of-Traversals

The Tree-of-Traversals algorithm maintains a local KG subgraph that is repeatedly expanded until it contains all the information required by an LLM to answer the given query. At the start, a local KG subgraph is initialized to contain the entities present in the original query. It is then expanded

using a tree search algorithm to choose actions and thoughts generated by an LLM in order to obtain relevant knowledge from the KG using a KG interface. The algorithm halts when the LLM is able to answer the original query using the local KG subgraph as context. Tree-of-Traversals consists of three major components. (1) **A knowledge graph interface** implemented to interact with one or more required KGs. (2) **An action state machine (ASM)** which is a finite state machine that defines the feasible space of actions, states, and prompt templates when the LLM interacts with a KG to expand the local KG subgraph. (3) **A tree search algorithm** which defines the overall LLM search trajectory such as best first search, backtrack upon making a mistake, and termination condition when an answer is found.

2.1 Knowledge Graph Interface

The knowledge graph interface allows Tree-of-Traversals to interact with one or multiple KGs. Let $\mathcal{K} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a single KG. $\mathcal{E}$ is the set of entities in which each entity consists of an identifier, a label, and an optional description (e.g., Q35332, '*Christopher Nolan*', '*British-American filmmaker*'). $\mathcal{R}$ is the set of relation types, each consisting of an identifier, a label, and an optional inverse label (P57, '*director*', '*is director of*'). $\mathcal{T}$
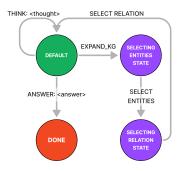
Figure 2: Action State Machine

is the set of edges or facts in the KG in which each edge is of the form (s, r, o) where $s, o \in \mathcal{E}$ and $r \in \mathcal{R}$, e.g., ('*Inception*', '*director*', '*Christopher Nolan*'). Tree-of-Traversals can be used for $\mathcal{K}$ as long as the following interfaces are implemented.

1. `initialize`$(q) \rightarrow \mathcal{E}_0$: It takes as input a query q, extracts entities from q and returns the linked entities $\mathcal{E}_0 \subset \mathcal{E}$ where $\mathcal{E}_0$ are the entities from $\mathcal{K}$ that are referenced in q.

2. `get_relations`$(\mathcal{E}_{selected}) \rightarrow \mathcal{R}_{options}$: It takes a set of entities, $\mathcal{E}_{selected}$, and returns the relation types, $\mathcal{R}_{options} \subset \mathcal{R}$, that $\mathcal{E}_{selected}$ have in $\mathcal{K}$: $\{r | (s, r, o) \in \mathcal{T}, s \in \mathcal{E}_{selected}\}$

3. `get_edges`$(\mathcal{E}_{selected}, r) \rightarrow \mathcal{T}_{added}, \mathcal{E}_{added}$: It takes a set of entities and a selected relation type, and returns all edges with relation type r for source entities in $\mathcal{E}_{selected}$: $\{(s, r, o) \in \mathcal{T} | s \in \mathcal{E}_{selected}, r = r, o \in \mathcal{E}\}$. It also returns the new entities, $\mathcal{E}_{added}$, that are entities reached with $\mathcal{T}_{added}$.

This interface is implemented with SPARQL queries when available; otherwise, we use the graph API that is available for the KG. For multiple KGs, each interface is implemented separately.

2.2 Action State Machine (ASM)

One of the challenges with developing a zero-shot LLM algorithm that works with arbitrary KGs is that the LLM does not know what relations are available in the graph or what relations are valid for a given entity. Few-shot or in-context learning approaches can only cover a handful of the possible relation types (e.g., Wikidata has over 11,000 relation types) (Brown et al., 2020).

To overcome these issues we break the task of expanding a local KG subgraph into multiple subtasks. We use a finite state machine with the following actions: `Think`,

`Answer`, `ExpandKG`, `Select_Entities`, and `Select_Relation`, and states: *default*, *selecting-entities*, *selecting-relation*, and *done* as shown in Figure 2. This is named as Action State Machine (ASM) throughout the paper.

From the *default* state, Tree-of-Traversals can either `Think`, `Answer`, or choose to `ExpandKG`. After Tree-of-Traversals chooses to `ExpandKG`, it first is prompted to `Select_Entities` about which it needs more information (e.g., '*Inception*' in Figure 1). It is then prompted to `Select_Relation` from a list of candidate relations provided by the KG interface's `get_relations` method. After selecting a relation (e.g., '*cast member*' in Figure 1), all edges containing one of the selected entities as the source and the selected relation as the relation are added to the local KG subgraph. Tree-of-Traversals is then able to `Answer`, `Think`, or `ExpandKG` again.

Prompt Templates. Each state in the ASM other than the '*done*' state is associated with a unique prompt template. Prompt templates are filled with information from the local KG subgraph and the KG interface before presenting them to the LLM. A customized prompt for each state allows us to present precise and relevant information along with specific instructions for each state to the LLM simplifying the LLM's task. For instance, the prompt for '*selecting-entities*' presents the available options of entities to choose from that are in the local KG subgraph (see Figure 3). Prompt templates for all the states are in the Appendix F.1-F.3.

Local KG subgraph is represented using a a token efficient YAML format that minimizes repetition for multiple edges on the same entity.

Invoking KG Interfaces. Outside of initialization, there are two times in which the ASM needs to invoke the KG interface: (1) When constructing the *selecting-relation* prompt, the algorithm calls `get_relations`$(\mathcal{E}_{selected})$ to gather available choices of relations. (2) When executing the transition from *selecting-relation* to *default* after having selected a relation type r, the algorithm calls `get_edges`$(\mathcal{E}_{selected}, r)$ so that it can add the new edges and entities to the local subgraph.

2.3 Tree Search Algorithm

Our approach draws inspiration from the Tree-of-Thoughts approach (Yao et al., 2023) in which an LLM is given increased reasoning power by allowing generation of multiple thoughts at a time and

```
Original Query:
    Who is Bob Dylan's maternal grandmother?
Knowledge Graph Entities:
    Q392:  Bob Dylan - American singer-songwriter
    Q62519478:  Beatrice Stone
Knowledge Graph Edges:
    Bob Dylan:
        mother:
            Beatrice Stone
Previous Actions:
...
Current task: Select entities to expand from [Q392,
    Q62519478]
Selection:
```

Figure 3: Prompt for the '*selecting-entities*' state of the ASM. The original query and the local KG subgraph are part of the prompt for every action state while the 'Current task' differs. The entity options show what entities can be selected based on the current local KG subgraph. If the model selected Q62519478: Beatrice Stone here, then in the next action state (*selecting-relation*), the dynamic prompt would list the relation types that Beatrice Stone has edges for.

building a search tree over the resulting reasoning chains. We extend this approach to augment LLMs with KGs through allowing the generation of actions in addition to thoughts, and building a search tree over them. Challenges arise because Tree-of-Thoughts was not designed to incorporate actions and was not designed for knowledge intensive question answering tasks. As a result, we introduce some modifications: incorporation of actions via the ASM, a slightly different search procedure and stopping condition to better handle QA, and a different sampling procedure for improved diversity when doing constrained sampling.

Algorithm 1 presents the Tree-of-Traversals tree search algorithm. Given a query q it begins with initialization using `initialize(q)` as described in Section 2.1. After initialization, it searches by (1) choosing the unexplored node to expand based on the node's value assigned by the value function (Best First→ Depth First as tie-breaker), (2) sampling k actions from the LLM (k is branching factor) using the prompt associated with the selected node's state in the ASM, (3) for each of the sampled action, applying the transition function, and (4) evaluating the value of the resulting nodes with the LLM value function. The search stops when an answer is found with a node value exceeding the threshold τ. To bound the search space we add two hyper-parameters: *max depth* of the tree search, after which the algorithm is forced to transition to the *done* state (i.e., answer the query), and *max expansions* after which the model halts exploration and returns "Cannot find answer".

Algorithm 1 Tree-of-Traversals

Input: $q \leftarrow$ query, $k \leftarrow$ branching factor, $\tau \leftarrow$ answer threshold, $P \leftarrow$ LLM, $\mathcal{F} \leftarrow$ ASM
Output: Answer to q

1: **procedure** ToACTIONS(q)
2:     $T \leftarrow \phi$ ▷ Empty tree
3:     $Y \leftarrow \phi$ ▷ Empty answer set
4:     $s_0 \leftarrow$ `initialize`(q)
5:     T.add(s_0)
6:     **while** $!finished$ **do**
7:        $s \leftarrow$ `choose_node`(T)
8:     ▷ Sample k actions from P using the action state and associated prompt of s from $\mathcal{F}(s)$
9:        $\mathbf{a} \leftarrow$ `sample_actions`($s, k, \mathcal{F}, P$)
10:       **for** $a \in \mathbf{a}$ **do**
11:      ▷ Apply action a to state s according to $\mathcal{F}$
12:         $s' \leftarrow$ `transition`($s, a, \mathcal{F}$)
13:      ▷ Evaluate the resulting state s' using P
14:         $s'.value \leftarrow$ `evaluate`(s', P)
15:         $T.add(s')$
16:         **if** $s'.action\_state = done$ **then**
17:           $Y.add(s')$
18:           **if** $s'.value > \tau$ **then**
19:    ▷ When answer exceeds threshold, stop search
20:             $finished \leftarrow True$
21:     **return** $\text{argmax}_{y \in Y}\ y.value$

Value Function Guidance. Tree-of-Traversals computes the value of a node to determine its utility. This value is created by the LLM using evaluation prompts (step `evaluate` in Algorithm 1). The value can be between 0 to 1 where 1 indicates highest utility. We use two types of evaluation prompts: one for intermediate states and one for answer states. The prompts include the original query, the local KG subgraph, the trajectory of previous actions, followed by instructions for evaluating the node (see Appendix F.4 and F.5). These values are then used to guide the exploration of the action space. Specifically, `choose_node` returns the unexplored node with the highest value (Best First). If there are nodes with the same value, Depth First Search is used.

Chain-of-Traversals. In some cases we can find the answer to a query with a single sequence of thoughts and actions using the ASM and KG interface. This is equivalent to Tree-of-Traversals with a branching factor of $k = 1$. We refer to this as Chain-of-Traversals. While useful for compari-

son, experiments show the benefit of considering multiple branches.

2.4 Tree-of-Traversals with Multiple KGs

Augmenting an LLM with more than one KG mainly involves building KG interfaces for each of the added KGs. There are a few other changes to the algorithm. (1) The entities extracted in `initialize(q)` are matched with each KG interface. (2) When presenting the options of relations during *selecting-relation*, `get_relations(E_{selected})` is called on each KG interface. (3) When adding a new entity to the local KG subgraph we call an entity linking function for the other KG interfaces. We allow for the entity linking function to be a separate function in the KG interface or to fallback on `initialize(o)` where o is the text label of the entity that has just been added. This makes allows the use of explicit links between common KGs if available while still functioning without.

3 Experiments

We evaluate Tree-of-Traversals using three different models available on Amazon Bedrock[3]: Claude-Instant (`claude-instant-v1`), Llama2 70b (`llama2-70b-chat-v1`), and Llama2 13b (`llama2-13b-chat-v1`). AWS Bedrock provides on-demand access through a single API to various Foundation Models, including both open source and black box ones. This is precisely a use case Tree-of-Traversals is designed for.

3.1 Tasks and Datasets.

We first evaluate the Tree-of-Traversals algorithm on two common tasks used for evaluating an LLM's knowledge: 2WikiMultiHop and QALD-10. To allow testing on complex questions requiring knowledge from multiple KGs we create a new dataset that requires knowledge from multiple KGs.

2WikiMultiHop Dataset (Ho et al., 2020) was constructed by extracting multi-hop templates from HotPotQA (Yang et al., 2018), combining templates to get complex reasoning questions, generating candidate questions with Wikidata, and then confirming that the entity mentions for each edge appear in the Wikipedia paragraphs. Answers to these questions can be derived from both Wikipedia and Wikidata (Vrandečić and Krötzsch, 2014). We

---

| Which event was hosted in a venue with a larger maximum capacity, *Fearless Tour: Indianapolis* or *Fearless Tour: Omaha*? |

1. Get venue for *Fearless Tour: Indianapolis* → *Gainbridge Fieldhouse*
2. Get venue for *Fearless Tour: Omaha* → *CHI Health Center Omaha*
3. Get maximum capacity of the respective venues → (18165, 18975)
4. Answer: *Fearless Tour: Omaha was hosted in a venue with a larger maximum capacity.*

Figure 4: Example question from MusicBrainz-x-Wiki and how Tree-of-Traversals arrives at its final answer. Red indicates entities and and relationships belonging to MusicBrainz. Green indicates relationships only in Wikidata. Blue indicates entities linked to both KGs.

---

subsample 500 questions from the test set following the approach used in ReAct (Yao et al., 2022), including the sampling seed value of 233.

QALD-10 Dataset (Usbeck et al.) is a multilingual Knowledge Graph Question Answering (KGQA) dataset with 395 questions created by humans, translated into other languages, and then constructed as a SPARQL query over Wikidata[4]. These questions are more varied in terms of reasoning structure than the questions in 2WikiMultiHop (e.g. requiring multiple answers or aggregations). We used the questions in English language.

MusicBrainz-x-Wikidata Dataset One novel use-case of Tree-of-Traversals is synthesizing and reasoning over multiple knowledge graph sources. There is no existing dataset that requires synthesizing information from multiple KGs to answer individual questions. Therefore, to test the reasoning ability using multiple KGs, we create a new dataset, MusicBrainz-x-Wikidata, containing 109 questions that require reasoning with information from both MusicBrainz and Wikidata. Unlike Wikidata which contains general knowledge, MusicBrainz is a deep domain-specific database on the music industry. The vast majority of this information is unlikely to be known by large language models. We construct the MusicBrainz-x-Wikidata dataset with human annotators who were provided with instructions to find reasoning paths between these two KGs, the question types to focus on, and some example questions. The curated questions were checked for ambiguity and sensibility. By design, each question

---

[3]https://aws.amazon.com/bedrock

[4]https://github.com/KGQA/QALD-10

| | 2WikiMultiHop (↑) | | |
Algorithm	Claude-Instant	Llama2-70b	Llama2-13b
Chain-of-Thought	25.2	30.4	26.8
ReAct	5.8	30.4	6.8
ReAct→ CoT	27.0 (84.8%)	41.6 (41.6%)	23.6 (71.8%)
FLARe	35.0	45.4	34.8
Chain-of-Traversals	42.6	45.0	31.0
Tree-of-Traversals	**63.0**	**56.6**	**37.3**
	QALD-10 (↑)		
Algorithm	Claude-Instant	Llama2-70b	Llama2-13b
Chain-of-Thought	47.4	40.0	**42.6**
ReAct	3.1	23.7	8.7
ReAct→ CoT	47.9 (72.6%)	43.4 (44.4%)	40.8 (79.5%)
FLARe	39.2	38.8	23.1
Chain-of-Traversals	55.7	56.9	39.6
Tree-of-Traversals	**64.3**	**61.6**	39.2

Table 1: EM-in on 2WikiMultiHop dataset. →CoT indicates falling back to Chain-of-Thought when no answer is given. (%) indicates the number of such cases.

requires extracting information from both knowledge graphs in order to answer it successfully. In addition to the reasoning types in 2WikiMultiHop, this dataset contains questions that involve aggregations, comparisons of aggregations, qualifications, and complex combinations of these. An example can be seen in Figure 3.1. Detail on instructions, question types, and examples are in Appendix A.

3.2 Metric.

We use Exact Match Included (EM-in) as the evaluation metric. EM-in is 1 if the ground truth answer appears with an exact match anywhere in the answer and 0 otherwise. This accounts for the propensity of an LLM to output answers in a sentence with varying syntax. This is a common metric but is often referred to interchangeably with Exact Match (EM) (Sun et al., 2023). When there are multiple answers, we compute average EM-in over all labels.

3.3 Comparison Baselines.

We experiment against three related approaches that can work with any black-box LLMs: (1) Chain-of-Thought (CoT) prompting (Wei et al., 2022) (2) ReAct (Yao et al., 2022) which iterates between generating thoughts and generating actions for searching and retrieving from a text knowledge base like Wikipedia, and (3) Forward Looking Ac-

tive Retrieval (FLARe) (Jiang et al., 2023b) which iterates between generating thoughts and then retrieving from a knowledge base to correct inaccuracies.

3.4 Implementation details.

For all models, we use a sampling temperature of 0.0 for the LLM when multiple samples are not required and a temperature of 1.0 when diverse samples are required. We test our approach in two settings: (1) with a branching factor of $k = 1$ termed as Chain-of-Traversals, and (2) with a branching factor of $k = 3$ termed as Tree-of-Traversals. In both setups, we set maximum depth to 7 which means that upon reaching the *default* action state beyond depth 7, the only available action is to answer the question. For Tree-of-Traversals, we set the maximum total expansions to 20. The answer threshold τ is set to 0.8 which corresponds to high confidence answers supported by the KG according the `evaluate` prompt (Appendix F.5).

For 2WikiMultiHop and QALD-10, we use Wikidata as the knowledge graph (Vrandečić and Krötzsch, 2014). For MusicBrainz-x-Wikidata, we use MusicBrainz in addition to Wikidata. We implement the KG interface for Wikidata using Wikidata SPARQL queries[5], and implement the KG interface for MusicBrainz KG using the MusicBrainz
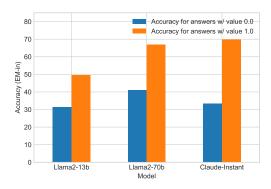
---
[5] https://query.wikidata.org/

Figure 5: The EM-in accuracy for answers with model assigned value 0.0 or 1.0 and the corresponding true EM-in score of the answer. This includes all proposed answers, not just the final answer returned by the model.
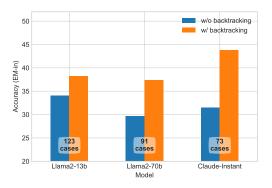

Figure 6: Comparison of results with and without backtracking on 2WikiMultiHop questions that needed backtracking. There would be a major degradation in performance if backtracking was not allowed.

API[6].

4 Results and Discussion

Table 1 presents the results of our experiments on 2WikiMultiHop and QALD-10. For all the models, Tree-of-Traversals outperforms the baselines on 2WikiMultiHop, setting state-of-the-art results for these tasks in the zero-shot setting. We hypothesize that much of this gain is due to Tree-of-Traversals' access to the knowledge base via proposed KG interface and its thought-action procedure guided by the ASM. This is evident, as even Chain-of-Traversals, which does not perform a tree traversal (including multiple thoughts/actions, backtracking, and node value computation), significantly outperforms ReAct's knowledge grounding: 8.7% higher accuracy than ReAct→ CoT on 2WikiMultiHop when averaged over all models. Compared to Chain-of-Traversals, Tree-of-Traversals further improves performance. It gives on average a 12.8% absolute accuracy increase for 2WikiMultiHop and 4.3% absolute improvement for QALD-10. We note that the better performing the model is, the more it stands to gain from Tree-of-Traversals as noted by the difference between Llama-70b and Llama-13b.

On MusicBrainz-x-Wikidata (Table 2), which contains challenging reasoning questions requiring access to two KGs, there is an average of 37.4% relative improvement from Tree-of-Traversals over Chain-of-Traversals as shown in Table 2. Both Chain-of-traversals and Tree-of-Traversals outperform Chain-of-Thoughts on this dataset. We only

compare with Chain-of-Thoughts as other retrieval methods have low coverage over the MusicBrainz knowledge base.

4.1 Effect of the Value Function.

Tree-of-Traversals relies on signal from the value function to pick the final tree trajectory. If the values were arbitrary, then Tree-of-Traversals would not do any better than Chain-of-Traversals. Figure 5 shows that there is a meaningful signal from the value function for all models. There is an average performance difference of 31.0% between the accuracy for answers valued at 1.0 vs 0.0. This represents an average relative improvement of 83.2% when selecting answers valued at 1.0 over those valued at 0.0. See Appendix C for individual profiles of each model's value function.

4.2 Effect of Backtracking.

To determine the effect of backtracking in Tree-of-Traversals, we ask the counter-factual of how would the model perform if it could not backtrack (Figure 6). We limit the analysis to 2WikiMultiHop questions in which Tree-of-Traversals generates answers on a subtree, and the model eventually backtracks on that subtree (i.e., the cases where we have a counter-factual). As a result, these questions are generally more challenging than the overall distribution of questions. In these cases, we compare the result if the highest-valued answer was taken from the first searched subtree compared to the ultimate answer after backtracking. We find that the ability to backtrack gives Tree-of-Traversals a significant accuracy increase ranging from 4.1% to 12.3%.

	Music-x-Wiki ($\uparrow$)		
Algorithm	Claude-Instant	Llama2-70b	Llama2-13b
Chain-of-Thoughts	11.9	10.1	13.8
Chain-of-Traversals	22.0	21.9	14.7
Tree-of-Traversals	**40.4**	**23.4**	**16.5**

Table 2: EM-in on MusicBrainz-x-Wikidata dataset.

4.3 Performance on MusicBrainz-x-Wikidata.

For MusicBrainz-x-Wikidata dataset, we only compare against Chain-of-Thought since the implementations for other baseline algorithms do not have access to similar music-specific knowledge bases. Despite this, we observed some interesting results. Chain-of-Thought does far worse on MusicBrainz-x-Wikidata than on the more general datasets based exclusively on Wikipedia/Wikidata (10.1%-13.8% vs. 25.2%-47.4%). This could be because LLMs are trained on vast amount of general knowledge such as that found in Wikipedia, but they are not trained with as much data from specific domains such as music. Besides the presence of KG interface and ASM with Tree-of-Traversals, this could be an additional reason why Tree-of-Traversals does 2.2 times as well as Chain-of-Thought on MusicBrainz-x-Wikidata. This demonstrates the importance of augmenting LLMs with domain-specific KGs and/or multiple KGs which the proposed Tree-of-Traversals is capable of.

4.4 Analysis of Baselines.

ReAct underperforms Chain-of-Thought in most cases. This phenomenon was seen in the original ReAct paper which noted that their approach significantly reduced hallucinations despite lower accuracy (Yao et al., 2022). Therefore, falling back on Chain-of-Thought results in an accuracy improvements for for Llama2-70b and Claude-Instant. We note that `claude-instant` vastly underperforms Llama2-70b on ReAct. The primary reason for this is `claude-instant` refuses to "search" people and apologizes after performing invalid actions. Despite this, ReAct→CoT still improves over CoT on both 2WikiMultiHop and QALD-10. FLARe improves model performance on 2Wiki-MultiHop but not on QALD-10. This may be due to better overlap between the text knowledge base for 2WikiMultiHop.

5 Related Works

Knowledge Base Question Answering There is a large history of work that has looked into answering questions using a knowledge graph (Wu et al., 2019; Lan et al., 2021). The approaches can broadly be broken into those that attempt to parse the question into a logical form (Berant and Liang, 2014; Luo et al., 2018; Zhu et al., 2022), and information retrieval approaches (Bordes et al., 2015; Chen et al., 2019)

Knowledge Enhancement. Recently, researchers have looked into enhancing pretrained language models and LLMs using knowledge graphs. Many works have focused on incorporating knowledge graph data into LLMs through training, often with architectural changes to the model (Zhang et al., 2019; Wang et al., 2019; Peters et al., 2019; Yamada et al., 2020; He et al., 2021). Some of these have found success using specialized graph encoder layers (Yasunaga et al., 2021; Sun et al., 2021). Some have sought to mix this process with pretraining of the language model (Yasunaga et al., 2022). The limitations of including KGs during LLM training are: the models are incapable of incorporating KG updates without retraining, are not able to change KG source (e.g., a domain-specific one), and may have low explainability resulting from learning knowledge in model weights rather than explicit retrieval. In addition, these methods add complexity to the training process, increase cost, and do not work with LLM without access to model weights. As a result, scaling these methods has often been seen as too risky.

Other methods treat a KG as a more complete source of truth and use the LLM to generate structured queries for the KG. Tianle et al. (2023) and Choudhary and Reddy (2023) teach the LLM to generate a logical KG query which then returns matching entities from the KG. These methods share similarities with the semantic parsing based

approaches mentioned earlier. However, by returning a logical query for the KG, these methods lose the reasoning and commonsense abilities of the LLM.

Some approaches have looked at injecting KG or knowledge base data into LLM prompts. Many methods do a single round of retrieval based on the query (Lewis et al., 2020; Li et al., 2023) using a retrieval mechanism, such as dense passage retrieval (Karpukhin et al., 2020). These methods cannot answer more complex multi-hop questions as the initial retrieval is unlikely to contain the secondary or tertiary information that will ultimately be required. Later methods have made the retrieval process iterative. E.g. FLARe (Jiang et al., 2023b) uses prediction of the upcoming sentence to retrieve relevant documents and regenerates until the sentence contains high-confidence tokens. In Wang et al. (2023) multi-round QA format is used for multi-round retrieval. ReAct (Yao et al., 2022) does multiple rounds of thoughts and actions to query a text-based knowledge base. Jiang et al. (2023a) repeatedly interfaces with a KG. In other parallel works, (Sun et al., 2023; Wen et al., 2023) explore methods for building KG context for LLMs using path and neighborhood based search methods. The above methods do not incorporate tree search or forms of advanced reasoning beyond the LLM's innate ability. They cannot explore multiple reasoning paths or solutions and cannot backtrack if making a mistake. This results in lesser capabilities. Additionally, none of the above methods explore reasoning over multiple KGs.

Multiple Knowledge Base QA. Some works studied QA on questions derived from differing domains. These works learn to pick between different domain-specific QA models, each trained on a single knowledge base (Puerto et al., 2021; Geigle et al., 2021; Puerto et al., 2023). Besides requiring training, such systems cannot answer questions requiring the synthesis of information from different domains (MusicBrainz-x-Wikidata).

6 Conclusion

Tree-of-Traversals is a powerful algorithm that enables LLMs to utilize KG reasoning with zero examples, no schema dependency, and no training. We demonstrate its efficacy experimentally on multiple LLMs and datasets. We hope Tree-of-Traversals continues to be developed and see value in studying integration with personalized user KGs

as well as with other domain-specific datasets.

7 Limitations

Tree-of-Traversals is slower than simpler retrieval alternatives. Improving the value function would reduce the number of LLM and KG API calls by avoiding incorrect paths along the tree to explore. Other engineering solutions, such as hosting graph servers, could be implemented to accelerate LLM and KG access. In terms of the token cost, the KG text representation is token efficient compared to many retrieval methods using unstructured knowledge bases as the latter tend to maximize usage of the LLM context window. However, compared to non-retrieval baselines, there is significant increase in token usage cost.

The types of KG questions that can be answered are limited by the context window, search depth, and LLM's reasoning ability. For instance, a very large aggregation ('*How many mountains have elevation above 3500 meters?*') would require more entities than what fits in the LLM context. Future work could look at adding other actions to the ASM, such as aggregations, that could distill the local KG into more relevant information.

EM-in is also an imperfect metric. False negatives can arise from discrepancies in date and number formatting, text formatting, and aliasing. False positives can arise in cases in which the model does not definitively answer but includes the correct answer in its response.

8 Ethical Impact

We do not anticipate Tree-of-Traversals to introduce new areas of risk but it may have unstudied effects on existing risks of LLMs or KGs. We highlight the following areas. (i) Tree-of-Traversals gives new capabilities to LLMs after training. While positive in terms of accuracy, we have not evaluated its effect on wider safety metrics. (ii) Our evaluation is limited to only English versions of KGs and datasets. Tree-of-Traversals should be evaluated in other languages to ensure consistent and fair experience. (iii) We have not performed analysis under misleading or deceptive knowledge graphs. Using a publicly modifiable knowledge graph does come with the risk that information could be deceptively changed.

In terms of positive ethical impact, making this research public democratizes access to knowledge-augmented LLMs as this method does not require

a large training investment or owning the language model to build and customize.

References

Bilal Abu-Salih. 2020. Domain-specific knowledge graphs: A survey. *ArXiv*, abs/2011.00235.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *ArXiv*, abs/2302.04023.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Annual Meeting of the Association for Computational Linguistics*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *ArXiv*, abs/1506.02075.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. Uhop: An unrestricted-hop relation extraction framework for knowledge-based question answering. *ArXiv*, abs/1904.01246.

Wonjun Choi and Hyunju Lee. 2019. Inference of biomedical relations among chemicals, genes, diseases, and symptoms using knowledge representation learning. *IEEE Access*, 7:179373–179384.

Nurendra Choudhary and Chandan K. Reddy. 2023. Complex logical reasoning over knowledge graphs using large language models. *ArXiv*, abs/2305.01157.

Feroz Farazi, Maurin Salamanca, Sebastian Mosbach, Jethro Akroyd, Andreas Eibeck, Leonardus Kevin Aditya, Arkadiusz Chadzynski, Kang Pan, Xiaochi Zhou, Shaocong Zhang, Mei Qi Lim, and Markus Kraft. 2020. Knowledge graph approach to combustion chemistry and interoperability. *ACS Omega*, 5:18342 – 18348.

Gregor Geigle, Nils Reimers, Andreas Ruckl'e, and Iryna Gurevych. 2021. Tweac: Transformer with extendable qa agent classifiers. *ArXiv*, abs/2104.07081.

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. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ziniu Hu, Yichong Xu, W. Yu, Shuohang Wang, Ziyi Yang, Chenguang Zhu, Kai-Wei Chang, and Yizhou Sun. 2022. Empowering language models with knowledge graph reasoning for open-domain question answering. *ArXiv*, abs/2211.08380.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Wenliang Dai, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55:1 – 38.

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. *ArXiv*, abs/2305.09645.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Li-Yu Daisy Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. *ArXiv*, abs/2305.06983.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. *ArXiv*, abs/2105.11644.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, S. Auer, and Christian Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

Shiyang Li, Yifan Gao, Hao Jiang, Qingyu Yin, Zheng Li, Xifeng Yan, Chao Zhang, and Bing Yin. 2023. Graph reasoning for question answering with triplet retrieval. In *Annual Meeting of the Association for Computational Linguistics*.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *ArXiv*, abs/1909.02151.

Yang Liu, Qingguo Zeng, Joaquín B. Ordieres Meré, and Huanrui Yang. 2019. Anticipating stock market of the renowned companies: A knowledge graph approach. *Complex.*, 2019:9202457:1–9202457:15.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Q. Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Conference on Empirical Methods in Natural Language Processing*.

Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, Russa Biswas, Gerard de Melo, Angela Bonifati, Edlira Vakaj, Mauro Dragoni, and Damien Graux. 2023a. Large language models and knowledge graphs: Opportunities and challenges. *ArXiv*, abs/2308.06374.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023b. Unifying large language models and knowledge graphs: A roadmap. *ArXiv*, abs/2306.08302.

Matthew E. Peters, Mark Neumann, IV RobertL.Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Conference on Empirical Methods in Natural Language Processing*.

Haritz Puerto, Tim Baumgärtner, Rachneet Sachdeva, Haishuo Fang, Haotian Zhang, Sewin Tariverdian, Kexin Wang, and Iryna Gurevych. 2023. Ukp-square v3: A platform for multi-agent qa research. In *Annual Meeting of the Association for Computational Linguistics*.

Haritz Puerto, Gözde Gül Şahin, and Iryna Gurevych. 2021. Metaqa: Combining expert agents for multi-skill question answering. *ArXiv*, abs/2112.01922.

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

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, Weixin Liu, Zhihua Wu, Weibao Gong, Jianzhong Liang, Zhizhou Shang, Peng Sun, Wei Liu, Ouyang Xuan, Dianhai Yu, Hao Tian, Hua Wu, and Haifeng Wang. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *ArXiv*, abs/2107.02137.

Yiming Tan, Dehai Min, Y. Li, Wenbo Li, Na 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.

Li Tianle, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Annual Meeting of the Association for Computational Linguistics*.

Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, et al. Qald-10—the 10th challenge on question answering over linked data.

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

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

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juan-Zi Li, and Jian Tang. 2019. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2023. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *ArXiv*, abs/2308.09729.

Peiyun Wu, Xiaowang Zhang, and Zhiyong Feng. 2019. A survey of question answering over knowledge base. In *China Conference on Knowledge Graph and Semantic Computing*.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Conference on Empirical Methods in Natural Language Processing*.

Lin F. Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. 2023. Chatgpt is not enough: Enhancing large language models with knowledge graphs for fact-aware language modeling. *ArXiv*, abs/2306.11489.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset

for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing.*

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D. Manning, Percy Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. *ArXiv*, abs/2210.09338.

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 *North American Chapter of the Association for Computational Linguistics.*

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023. Extractive summarization via chatgpt for faithful summary generation. *ArXiv*, abs/2304.04193.

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

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Annual Meeting of the Association for Computational Linguistics.*

Yueqin Zhu, Wenwen Zhou, Yang Xu, Ji Liu, and Yongjie Tan. 2017. Intelligent learning for knowledge graph towards geological data. *Sci. Program.*, 2017:5072427:1–5072427:13.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023a. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *ArXiv*, abs/2305.13168.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji rong Wen. 2023b. Large language models for information retrieval: A survey. *ArXiv*, abs/2308.07107.

Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. 2022. Neural-symbolic models for logical queries on knowledge graphs. In *International Conference on Machine Learning*, pages 27454–27478. PMLR.

A MusicBrainz x Wiki

Additional information on the creation of the MusicBrainz x Wiki dataset.

A.1 Creation steps

The dataset was created by annotators in a semi-automatically supported manner. We developed an automated tool to support the following process:

- Find linking entities that are present in both MusicBrainz and Wikidata of various types (artist, label, place, event, etc.)

- Find related entities that could be used in a question to perfectly identify the entities above (e.g., The place owned by the Detroit Tigers → Tiger Stadium)

- Find related entities that could be used in a question to ambiguously identify a linking entity, so that a qualifier could be added to disambiguate the linking entity.

- Count the number of edges in Wikidata or MusicBrainz for each relationship type an entity has.

We use these tools to create an initial set of multi-hop questions falling into specific reasoning categories. We then have human annotators check each question for sensibility (grammatically correct and can be definitively understood) and ambiguity (one right answer in the KGs). If possible, the question is reworded or fixed to not contain ambiguities. We are then left with a final set of 109 questions.

A.2 Composition of Questions

The composition of questions for Musicbrainz x Wiki can be found in table 3.

A.3 Annotators

The annotators consisted of approximately 10 research scientists and engineers, all English speaking and based in the United States.

B Implementation Details

B.1 Diversity Oversampling.

One problem with Tree-of-Thoughts is that when there are constrained options to choose from, the LLM becomes repetitive and fails to produce diverse outputs (Yao et al., 2023). Their solution used a "propose prompt" with additional instructions for proposing multiple distinct thoughts. However, this

Question Type	Count	Description	Example
Bridge	30	Direct multi-hop questions	The father of Alexander Newley was the vocal arranger for what song?
Bridge Comparison	7	Comparison of values from two multi-hop reasoning chains	Which venue was opened more recently, the venue for The Eras Tour: Seattle (night 1) or the venue for Fearless Tour: Seattle?
Bridge Aggregation	42	Count of the number of entities satisfying a multi-hop condition (aggregation only over the second hop)	The place which is the filming location of Michelangelo Buonarroti, has how many songs recorded there?
Bridge Aggregation Comparison	7	Comparison of counts of entities between two different aggregation conditions	Who composed more songs, Marie Daulne's mother or Qubilah Shabazz's godparent?
Qualification Bridge	10	Multi-hop question in which one of the edges needs to be disambiguated with a qualifying statement	The filming location of Scream Awards that was built in 1929, has what recording engineered there?
Qualification Bridge Aggregation	9	Count of the number of entities satisfying a qualification bridge style condition	The sibling of Teppo Ruohonen that was born in 1949, has composed how many songs?
Qualification Bridge Aggregation Comparison	1	Comparison of qualification bridge aggregation type questions	Which of the children of Isabel Preysler that are singers has released more albums?
Double Qualification	3	Multi-hop question in which each hop requires a disambiguating qualification edge	The child of Guus Belinfante that was a singer, has a vocal feature on what recording by Doe Maar?

Table 3: Composition of questions for Musicbrainz x Wikidata

adds significant complexity to the prompt for purposes that are only tangential to the task being completed. A simpler approach we employ is to generate diverse actions by oversampling. Specifically, if the branching factor is k, we sample $2 * k$ possible actions from the LLM and then extract the first k unique ones to use. We only employ this for *selecting-entities* and *selecting-relation* as the action choices are limited and diversity is required. This change does not noticeably affect the computation cost of Tree-of-Traversals as multiple actions are sampled for the same prompt and the average prompt input size (100s-1000s of tokens) is generally much larger than the average generation length for these actions (<20 tokens). Latency remains similar as multiple actions are sampled in parallel. With this, we consistently generate diverse options when *selecting-entities* or *selecting-relation*.

B.2 Hyperparameters

Tree-of-Traversals does not have a huge number of hyperparamters. Most of the hyperparameters were chosen analytically, though some were chosen based on preliminary experiments. We share some information on choosing them.

τ was chosen analytically in conjunction with the prompt for `evaluate` on the *done* state (Appendix F.5). Chosen to be 0.8 so that Tree-of-Traversals only stops when it is confident that the answer is supported by the KG.

The temperature was chosen to be 1.0 to promote diversity. If implementing with other models which can have temperature >1.0, we recommend hyperparameter tuning or using a sample prompt of each type and analytically choosing the temperature such that diverse but sensible actions are output.

The branching factor $k = 3$ was chosen based on small scale experimentation. Larger k makes

the model spend longer searching, taking longer time and increasing expenses. Smaller k did not generate as much diversity and options. It is worth noting that even with $k = 3$ and diversity sampling, sometimes the actual number of unique outputs is less than 3.

The max depth was chosen analytically to be minimal while enabling answering of most questions. Some questions in QALD-10 and MusicBrainz-x-Wikidata would require a greater depth to answer.

The maximum expansions cutoff was chosen purely to ensure the model does not hang on a single question too long. The actual value is somewhat arbitrary and has minimal impact on accuracy.

B.3 KG Interface

We present some additional details of the KG Interface.

The `initialize` function is a three step process. First a LLM call and prompt is used to extract named entities. Second, for each extracted entity candidates are searched for using the KG API. Finally, another LLM call is used to match the extracted entity to the best candidate.

`get_relations` and `get_edges` are implemented exclusively using the respective KG API. For Wikidata, these are each one or two SPARQL query (forward and reverse edges). For MusicBrainz, they each require multiple API calls as the endpoints are separated for different entity types. E.g. there is a separate endpoint for Artists and Recordings.

C   Value Function Profiles

We profile each value model's value function to assess their characteristics. Figure 7 shows the individual answer value vs accuracy profiles for each model on 2WikiMultiHop. All models have a positive correlation between the answer value and the answer's accuracy. However, as is to be expected, there are differences with Llama2-13b having the lowest correlation and Claude-Instant having the highest correlation.

Figure 7 only look at the value function response for final answer states as those are the only ones with a directly associated accuracy score. While accurately evaluating answer states is more important, we also want to see some useful signal from the values for intermediary states. Figure 8 shows the average value for nodes on the search path to

the answer, separated by whether the answer scores as correct or not. We would expect actions that ultimately lead to a correct answer to have higher values than those that lead to incorrect ones. We clearly see this with the *correct* distribution being further right than the *incorrect* distribution. This is least pronounced for Llama2-13b which is to be expected. We also note that the distributions shape is different between the Llama2 models and Claude-Instant.
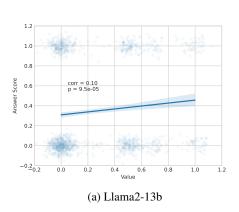
These results indicate that larger, stronger models, will likely see more improvement in the value function, and thus, more utility from Tree-of-Traversals .
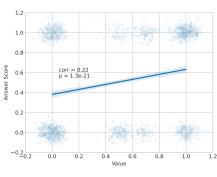
D   Prompt Engineering

Some prompt re-engineering was required when switching between models for both our methods and ReAct. This engineering was limited primarily to formatting of responses. For example, Llama2 would start responses with "So the next action should be" while we required the action to start with "THINK", "EXPAND_KG" or "ANSWER". Instead we incorporated "So the next action should be:" into the prompt so that the output began with the chosen action. For `selecting-entities` we added "You have selected the following entities to expand:". For `selecting-property` we added "I suggest selecting property:". These issues could also be solved by changing the parsing of the presented prompts.

E   Fallback to Chain-of-Thought ($\rightarrow$CoT)

We fallback to Chain-of-Thought for the ReAct baseline when either no answer is provided after depth 7 or when one of the following phrases appears in the answer: 'determine', 'unable', 'cannot', 'unknown', 'unsure', or 'not possible'. While these certainly could appear in intentional answers, we find that they occur consistently and exclusively in non-answer style responses.

(a) Llama2-13b
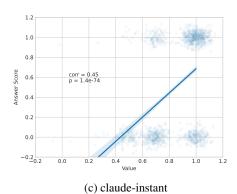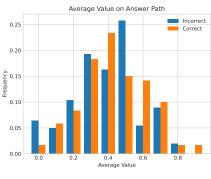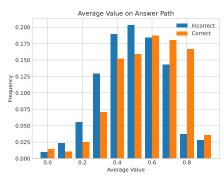

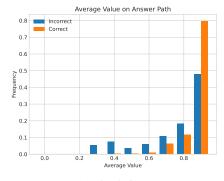(b) Llama2-70b

(c) claude-instant

Figure 7: EM-in Accuracy vs value function for all Tree-of-Traversals answers on 2Wiki. Points are jittered for visibility. Trendline indicates correlation and p-value.

(a) Llama2-13b

(b) Llama2-70b

(c) claude-instant

Figure 8: The average value function of intermediary steps along the path to each answer, separated by whether the answer would be scored as correct or incorrect. For each answer proposed by the model, the scores along the path are averaged, and the resulting distribution plotted. Data is from 2WikiMultiHop results.

F   Prompts

The following pages contains the prompts used in Tree-of-Traversals. The dynamic components that change based on the query, KG state, action history are shown in color.

F.1   Prompt for *default* action state

```
Original Query:
    Who is Bob Dylan's maternal grandmother?

Knowledge Graph Entities:
    Q392:  Bob Dylan - American singer-songwriter
    Q62519478:  Beatrice Stone
Knowledge Graph Edges:
    Bob Dylan:
        mother:
            Beatrice Stone

Previous Actions:
EXPAND KG: I should search for the mother of Bob Dylan
SELECT ENTITIES: Q392
SELECT RELATION: P25 - has mother

You are a superintelligent AI equipped with the ability to search a knowledge graph for definitive, up-to-
    date answers. Your task is to interface with the knowledge graph in order to answer the above query.
    You will be able to expand the knowledge graph until you have found the answer. Think in detail before
    acting or answering.

Available actions:
'THINK' - Generate relevant thoughts to solving the problem. This could include recalling well known facts
    from memory.
    e.g.,
        THINK: I should search for the movies directed by...
        THINK: I know that Biden is American, therefore...
        THINK: I see that John Cena is in both The Independent and Vacation Friends, therefore...
'EXPAND_KG' - Search for edges of entities in the knowledge graph using an external API. This is a useful
    function for getting to the correct answer.
    e.g., EXPAND_KG: I should search for the country of origin of Jonathon Taylor
'ANSWER' - Generate the final answer once the problem is solved. Just state the best answer, do not output a
    full sentence.
    e.g.,
        ANSWER: No
        ANSWER: Harry Potter
        ANSWER: [Harry Potter, Ron Weasley, Hermione Granger]
```

F.2   Prompt for *selecting-entities* action state

```
Original Query:
    Who is Bob Dylan's maternal grandmother?

Knowledge Graph Entities:
    Q392:  Bob Dylan - American singer-songwriter
    Q62519478:  Beatrice Stone
Knowledge Graph Edges:
    Bob Dylan:
        mother:
            Beatrice Stone

Previous Actions:
EXPAND KG: I should search for the mother of Bob Dylan
SELECT ENTITIES: Q392
SELECT RELATION: P25 - has mother
EXPAND KG: I should search for the mother of Beatrice Stone

Current task
EXPAND_KG takes two parameters. The first is the entity or entity group to get more information about.
    Select which entity or entities from the KG to expand.
Provide the QIDs. Options include [Q392, Q62519478]
SELECT ENTITIES:
```

F.3 Prompt for *selecting-relation* action state

```
Original Query:
    Who is Bob Dylan's maternal grandmother?

Knowledge Graph Entities:
    Q392:  Bob Dylan - American singer-songwriter
    Q62519478:  Beatrice Stone
Knowledge Graph Edges:
    Bob Dylan:
        mother:
            Beatrice Stone

Previous Actions:
EXPAND KG: I should search for the mother of Bob Dylan
SELECT ENTITIES: Q392
SELECT RELATION: P25 - has mother
EXPAND KG: I should search for the mother of Beatrice Stone
SELECT ENTITIES: Q62519478

Your current task is to select the property (PID) to expand along for the selected entities.
The selected entities are: [Q62519478]
The options of properties to choose from are:
P31 - has instance of
P21 - has sex of gender
P27 - has country of citizenship
P735 - has given name
...

Select exactly one property (PID e.g., P10) from those listed above
SELECT PROPERTY:
```

F.4 General Prompt for `evaluate`

```
Original Query:
    Who is Bob Dylan's maternal grandmother?

Knowledge Graph Entities:
    Q392:  Bob Dylan - American singer-songwriter
    Q62519478:  Beatrice Stone
Knowledge Graph Edges:
    Bob Dylan:
        mother:
            Beatrice Stone

Previous Actions:
EXPAND KG: I should search for the mother of Bob Dylan
SELECT ENTITIES: Q392
SELECT RELATION: P25 - has mother
EXPAND KG: I should search for the mother of Beatrice Stone
SELECT ENTITIES: Q62519478

Your current task is to evaluate the above knowledge graph and action history.
Based on the original query, the current knowledge graph, and the action history, give the likelihood that
    the model will correctly answer the question.
If the most recent action provided information towards the goal and followed the preceding thought, give a
    high score.
If the last action was unhelpful, give a low score.

The output should be a number between 0 and 1 with one decimal. Do not output anything else.

RATING [0.0-1.0]:
```

F.5 Prompt for `evaluate` on answer (*done*)

```
Original Query:
    Who is Bob Dylan's maternal grandmother?

Knowledge Graph Entities:
    Q392:  Bob Dylan - American singer-songwriter
    Q62519478:  Beatrice Stone
    Q62519478:  Florence Sara Stone
Knowledge Graph Edges:
    Bob Dylan:
        mother:
            Beatrice Stone
    Beatrice Stone:
        mother:
            Florence Sara Stone

Provided answer: The answer is Florence Sara Stone.

Your task is to score the correctness of the provided answer based on the original query, and the knowledge
     graph.
Give a pessimistic score from 0.0 to 1.0 on how likely the answer is to be correct.
0.0 if definitely wrong
0.0 if unable to answer based on the knowledge graph
0.5 if unsure
0.7 for probably correct but not confirmed in knowledge graph
1.0 for definitely correct and confirmed in knowledge graph.

Give reasoning to get to the correct answer. Then provide a score.
e.g.,
Reasoning...
So the score for the provided answer should be...
```