CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Tournament on Path: Relation Abstraction and Ranking for Knowledge Graph Reasoning

Master's thesis in Computer science and engineering

Shihao Xiang
Yihan Wu

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

Tournament on Path: Relation Abstraction and Ranking for Knowledge Graph Reasoning

Shihao Xiang
Yihan Wu

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Tournament on Path: Relation Abstraction and Ranking for Knowledge Graph Reasoning

Shihao Xiang
Yihan Wu

Supervisor: Shuai Wang, Computer Science and Engineering
Co-supervisor: Yinan Yu, Computer Science and Engineering
Advisor: Krasimir Angelov, Computer Science and Engineering
Examiner: Aarne Ranta, Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Tournament on Path: Relation Abstraction and Ranking for Knowledge Graph Reasoning

Shihao Xiang
Yihan Wu
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

While Knowledge Graph Question Answering (KGQA) leverages Large Language Models (LLMs) to perform complex multi-hop reasoning, existing training-free frameworks like Think-on-Graph (ToG) suffer from inherent limitations, including locally scoped pruning, unstable pointwise scoring, and the indiscriminate discarding of valuable candidates through random sampling. To address these challenges, this thesis introduces Tournament on Path (ToP), a novel reasoning framework that enhances relation abstraction and candidate ranking. The proposed framework systematically tackles the baseline's flaws by implementing an entity-agnostic relation path pruning strategy that captures global path semantics and reduces the search space. To effectively operationalize this, ToP is instantiated into two specific variants. System 1 employs a hybrid lexical-semantic pre-filtering combined with a chunked tournament selection algorithm to stabilize ranking across large candidate pools. System 2 relies on purely semantic pre-filtering and a pairwise tournament selection method, introducing a topic entity masking mechanism to strictly prevent LLMs from answering using unverified internal knowledge. Experimental evaluations across four diverse benchmarks (CWQ, WebQSP, WebQuestions, and GrailQA) demonstrate that both variants consistently outperform the ToG baseline. The results confirm that these strategies not only improve reasoning accuracy but also resolve the "negative scaling" behavior, establishing a computationally efficient and consistently reliable framework for different language models.

Keywords: Knowledge Graph Question Answering, Large Language Models, Multi-hop Reasoning, Relation Abstraction, Tournament Selection, Path Pruning.

Acknowledgements

First and foremost, we would like to express our heartfelt gratitude to our supervisor, Shuai Wang, for his continuous support and dedicated follow-up throughout this degree project. Whenever we encountered bottlenecks, his insightful guidance and constructive advice provided us with clear directions, playing a pivotal role in the successful completion of our thesis.

We are also deeply grateful to our co-supervisor, Yinan Yu, for her valuable guidance and extensive feedback. Her expertise and suggestions greatly helped us refine our research and improve the overall quality of this paper.

Additionally, special thanks go to Chalmers University of Technology for granting us access to the Alvis computing infrastructure. These powerful computational resources were absolutely indispensable for conducting our experiments.

We appreciate vLLM for providing a high-quality LLM runtime which supports our experiments.

Shihao Xiang & Yihan Wu, Gothenburg, 2026-06-09

Contents

List of Figures

List of Tables

1

Introduction

1.1 Knowledge Graph

A Knowledge Graph (KG) is a structured representation of interconnected entities, their attributes, and relationships. It can facilitate enhanced data integration, reasoning, and insight extraction. By organizing information in a network format, knowledge graphs enable machines to understand complex data and improve decision-making across various domains. They have gained significant prominence in technology and data science, particularly in applications like search engines, recommendation systems, and artificial intelligence, allowing organizations to leverage vast amounts of data more effectively.

To illustrate, consider a fragment of geographic knowledge from Freebase [1]—a widely adopted open-domain knowledge graph—represented as a set of triples:

- $(Alta\_Verapaz\_Department,\ location.containedby,\ Guatemala)$
- $(Guatemala,\ location.containedby,\ Central\_America)$
- $(Guatemala,\ type.object.type,\ Country)$

Here, Alta Verapaz Department is an administrative region within Guatemala, which in turn is part of the broader geographic region Central America. Such facts are naturally modeled as edges in a directed, labeled graph, enabling compositional reasoning via path traversal.

In contrast to relation databases, where answering compositional queries typically requires explicit JOIN operations across multiple normalized tables, knowledge graphs support query evaluation through graph traversal mechanisms. This allows multi-hop relation patterns to be expressed and evaluated more directly.

However, the flexibility of graph-structured data introduces non-trivial computational challenges. In particular,

- For weakly constrained or unconstrained multi-hop queries—where no fixed anchor entity, type constraints, or depth bounds are specified—the size of the search space can grow combinatorially with respect to the path length. This also applies to graphs without type information.
- In graphs that are not well-engineered, a single relation or entity in the real

world may not have a unique relation or entity representation, and a relation may not consistently have an inverse.

1.2 Large Langauge Models

Large Language Models (LLMs) are advanced artificial intelligence systems designed to understand and generate human language, playing a crucial role in various natural language processing (NLP) applications. Their development traces back to foundational advancements in NLP and machine learning, evolving significantly through innovations like the Transformer architecture, which enables efficient handling of language complexities and long-range dependencies in text data. [2]

LLMs, such as Bidirectional Encoder Representations from Transformers (BERT), Generative Pre-trained Transformer 2 (GPT-2), and GPT-3, have demonstrated remarkable capabilities across various industries. [3] Notably, the reasoning capabilities of LLMs have been significantly enhanced through techniques such as Chain-of-Thought (CoT) prompting and self-reflection, enabling the execution of complex, multi-step tasks. Despite these breakthroughs, challenges persist in maintaining factual accuracy and performing rigorous logical reasoning over structured data [4], [5]. These limitations often manifest as hallucinations, where the model generates plausible-sounding but factually incorrect information, necessitating the integration of external, structured knowledge sources.

1.3 Knowledge Graph Question Answering

Knowledge Graph Question Answering (KGQA) is a transformative approach that integrates the structured data of knowledge graphs with the flexibility of natural language queries, allowing users to receive precise answers derived from complex relationships. At its core, KGQA navigates the intricate web of nodes and edges within a graph to address queries that often require multi-hop reasoning across disparate information sources. Traditional methodologies primarily focused on translating natural language into formal query representations like SPARQL (SPARQL Protocol and RDF Query Language); however, ensuring semantic fidelity during this translation remains a significant challenge [6].

Recently, the integration of LLMs has brough substantial enhancements to the field by providing advanced reasoning capabilities that facilitate a more nuanced understanding of user intent [7]. This synergy allows for ”LLM-guided planning,” where the model generates a structured traversal plan grounded in verifiable triples, thereby reducing the incidence of hallucinations inherent in free-form generation models [8], [9]. Despite these advancements, the effectiveness of KGQA continues to be hindered by linguistic challenges, including semantic mismatches, the complexity of entity resolution for ambiguous terms, and the difficulty of maintaining contextual continuity in multi-turn dialogues. Addressing these limitationsspecifically the need for reliable reasoning paths in large-scale data environmentsserves as the primary motivation for this research.

Building upon KGs, the task of KGQA can be defined as follows: given a natural language question q and a knowledge graph $\mathcal{G}$, the goal is to retrieve an answer set $\mathcal{A} \subseteq \mathcal{E}$ that satisfies the semantic constraints expressed in q [9].

Consider the multi-hop query: "Which nation has the Alta Verapaz Department and is in Central America?" Answering this query requires grounding textual mentions to entities (e.g., Alta_Verapaz_Department, Central_America) and aligning phrases such as "has" and "is in" to corresponding graph relations. Following Freebase schema, this induces relation paths intersecting at an intermediate variable x:

$$(\texttt{Alta\_Verapaz\_Department},\ \texttt{location.containedby},\ x),$$

$$(x,\ \texttt{location.containedby},\ \texttt{Central\_America})$$

together with an implicit type constraint (x, `type.object.type`, `Country`).

In practice, these steps are tightly coupled. Moreover, multi-hop reasoning introduces a key challenge: the candidate search space grows combinatorially with the number of hops due to large branching factors in real-world graphs. This *path explosion* issue makes exhaustive traversal infeasible and motivates more efficient reasoning strategies [10].

1.4 Overall Architectures for KGQA

Current KGQA research mainly follows five paradigms that cover different subproblems, the first four of which are from [11]: generative semantic parsing for formal query synthesis, reasoning on graphs for iterative path exploration, graph Retrieval-Augmented Generation (RAG) for factual grounding, agentic KGQA for autonomous planning, and graph neural networks for path exploration.

1.4.1 Generative semantic parsing

This paradigm maps natural-language questions into formal query languages (like SPARQL [12]) that can be executed directly on a knowledge graph. The focus is on accurate structure generation and schema alignment between language and graph representations.

Recent research in this paradigm focuses on enhancing the structural accuracy and executability of natural language translations. ChatKBQA [13] introduces a streamlined "generate-then-retrieve" framework that fine-tunes LLMs to prioritize the synthesis of logical forms, followed by unsupervised entity alignment to ensure retrieval precision. To overcome the syntax errors inherent in traditional logical expressions, CodeAlignKGQA [14] redefines semantic parsing as constrained code generation, leveraging the pre-trained coding proficiency of LLMs for more robust execution. Furthermore, frameworks such as Logic-LM [15] and Knowledge Crosswords [16]

extend this capability by enabling LLMs to synthesize symbolic formulations for external solvers; these approaches integrate self-refinement and constraint verification to ensure that complex reasoning tasks are grounded in rigorous logical structures.

1.4.2 Reasoning on graphs

Methods in this category perform multi-hop reasoning by iteratively exploring entities and relations in the graph, often guided by heuristics or learned strategies. The objective is to discover plausible reasoning paths that connect the question entities to the answer nodes.

Recent research in this field focuses on the interactive navigation of knowledge graph topologies to establish grounded and interpretable reasoning chains. Reasoning on Graphs (RoG) [10] introduces a planning-retrieval-reasoning framework that generates KG-grounded relation paths to guide the model toward faithful conclusions.

To address complex multi-hop scenarios, Reasoning with Trees (RwT) [17] reframes KGQA as a discrete decision-making problem by employing Monte Carlo Tree Search (MCTS), while the Collaborative Reasoning Framework (CRF) [18] utilizes reinforcement learning to coordinate hierarchical agents for efficient path selection.

iQUEST [19] improves Knowledge Base Question Answering (KBQA) by iteratively breaking a complex question into simpler sub-questions to guide multi-hop reasoning over the knowledge graph. Although these methodologies achieve significant performance gains, they often necessitate additional training, knowledge distillation, or task-specific fine-tuning.

Think-on-Graph (ToG) [20] provides a training-free architecture that enables LLMs to perform iterative beam searches over the graph. Although this paradigm enhances traceability and versatility, it comes at the cost of higher computational overhead during inference compared to fine-tuned approaches.

1.4.3 Graph RAG with LLM

Graph RAG approaches retrieve relevant subgraphs, triples, or paths from a knowledge graph and inject them as structured context into a language model. This paradigm emphasizes factual grounding and hallucination reduction through explicit evidence retrieval.

Recent research in this branch increasingly complements traditional vector-based retrieval with structure-aware knowledge integration through graph-based indexing mechanisms. GraphRAG [21] proposes a two-stage indexing framework that first constructs an entity-level knowledge graph from source documents and then generates community-level summaries, enabling more effective handling of global, corpus-scale queries that conventional RAG pipelines often struggle to address. To deepen retrieval and memory integration, HippoRAG [22] introduces a neurobiologically inspired architecture grounded in hippocampal indexing theory, where large language models, knowledge graphs, and the Personalized PageRank algorithm are jointly employed to support efficient long-term knowledge consolidation.

For textual graph reasoning, G-Retriever [23] formulates retrieval as a Prize-Collecting Steiner Tree optimization problem, allowing the system to identify compact yet informative subgraphs while mitigating hallucinations in generation. HybGRAG [24] further extends this line of work by introducing a hybrid retriever bank coupled with an agent-style critic module to jointly leverage relational and textual evidence within semi-structured knowledge bases. Meanwhile, LightRAG [25] enhances efficiency and contextual coverage through a dual-level retrieval scheme that supports both fine-grained entity discovery and higher-level structural aggregation.

1.4.4 Agentic KGQA

Agentic KGQA treats the system as an autonomous planner that decomposes complex questions into subtasks, tools usage and self-reflection. The key characteristic is dynamic decision-making and multi-step planning rather than a fixed pipeline.

Recent research in this field focuses on framing Large Language Models as autonomous agents capable of iterative reasoning and planning with feedback from the environment. Knowledge Graph Agent (KG-Agent) [26] introduces an efficient agent framework that integrates a multifunctional toolbox, a knowledge memory module, and a KG-based executor, enabling small-scale LLMs to iteratively plan and refine reasoning trajectories over knowledge graphs.

To enhance the reliability of these reasoning trajectories, Debate on Graph (DoG) [27] employs a multi-role debate mechanism in which different agent personas cross-verify reasoning steps, mitigating the impact of false-positive relations and lengthy reasoning paths. Interactive Knowledge Base Question Answering (Interactive-KBQA) [28] supports multi-turn interactions through generic KB APIs and exemplar-guided reasoning, allowing LLMs to iteratively generate logical forms even in low-resource scenarios while supporting manual intervention.

Furthermore, KnowAgent [29] addresses the planning hallucination problem by incorporating an explicit action knowledge base and a knowledgeable self-learning strategy to constrain the agents trajectories, ensuring that synthesized actions remain grounded in verifiable domain knowledge. Collectively, these approaches demonstrate how treating LLMs as agentic reasoners, guided by structured planning and feedback mechanisms, can improve reasoning reliability and maintain grounded decision-making over knowledge graphs.

1.4.5 Graph neural networks

Graph neural networks (GNNs) are a class of neural networks that operate on graphs. Their inputs and outputs are graph representations. Neural networks must be specialized for graphs to work on them natively, most importantly because graphs are invariant to permutations in sequenced representations but regular neural networks and LLMs are sensitive to them. [30]

GNNs are inherently suitable for KG inference and completion, because the output of these tasks is a graph, the same format as the output of a GNN. In the case

of [31], the context node is a single node, whereas in [32] the context is a subgraph. In contrast, language models produce natural language strings as output, and they require additional mechanisms to be converted to graph information. Some research has shown improvements in question answering performance when GNNs are used in addition to only language models.

Since a GNN outputs a graph, it still requires an language model to be able to respond in natural language, and a pruning mechanism to direct the multi-hop inference. In [31], the approach consists of: a relevance scoring module using a pre-trained language model, a newly-trained GNN that performs inference, and another language model for converting the graph to answers. In [32], the pruning mechanism is trained into the GNN, eliminating relevance scoring as a standalone module and joining the two inference-time modules into one.

GNN is less resource-intensive than language model for graph inference given the same scale. [32]

Existing methods also require training for specific graphs, making GNN non-portable.

1.5 Think-on-Graph

Building upon the "Graph RAG with LLM" architecture discussed above, a prominent line of work explores how KGs can support explainable and controllable reasoning. ToG [20] exemplifies this direction and serves as the primary technical foundation for this thesis. Unlike other walking-based methods that require task-specific training or knowledge distillation, ToG establishes a training-free interaction loop between the LLM and the knowledge graph, supporting explainable and controllable reasoning.

Three paradigms are compared in ToG: [20]

- **LLM-only reasoning**

  The method only utilizes information in the model weights.

- **LLM $\oplus$ KG (retrieval from KG)**

  This method combines LLM with external graph knowledge. However, it uses an LLM to generate a one-shot KG query and thus relies too heavily on KG retrieval and fails to leverage full reasoning capabilities of LLMs.

- **LLM $\otimes$ KG (reasoning on KG)**

  Proposed by ToG, this algorithm involves LLMs more deeply in the reasoning procedure. The ToG algorithm offers several advantages over previous approaches.

  - It improves synergistic and iterative inference between LLMs and KGs.
  - It produces explainable inference graphs, enhancing transparency in the reasoning process.

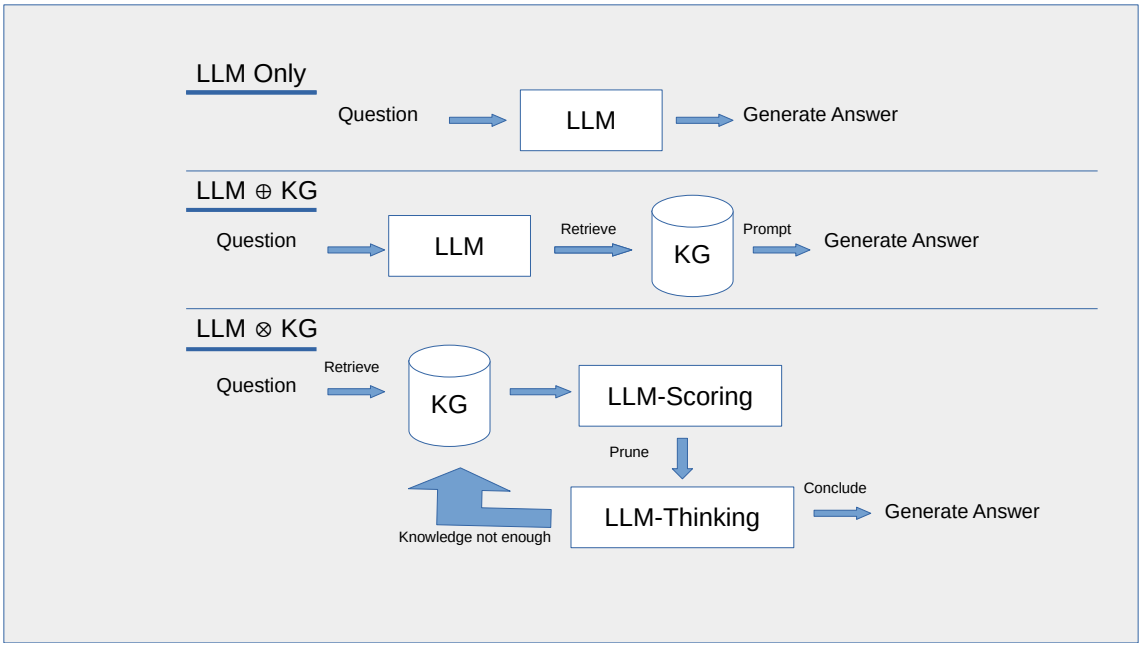

Figure 1.1: Representative workflow of three LLM reasoning paradigms: (a) LLM-only, (b) LLM ⊕ KG, (c) LLM ⊗ KG.

– No training or fine-tuning is required, easy to adopt in various domains.

Despite the progress demostrated by ToG, this paradigm suffers from several inherent limitations.

Firstly, relation pruning decisions are locally scoped, failing to capture the global semantic intent of complex queries. As a consequence, retrieved facts are not sufficient enough for LLM to answer many questions, causing LLMs to use its internal knowledge. This is harmful as LLMs are not always equipped with up-to-date knowledge, and will lead to improper results.

Secondly, pruning is performed using pointwise scoring, which introduces instability into candidate selection. Specifically, the scoring quality depends on both the model capacity of the LLM and the size of the candidate set. Moreover, in this context, only relative rankings are required rather than precise scores.

Finally, ToG resorts to random sampling to handle the large number of neighbors in knowledge graphs, potentially discarding useful neighbors. Although this provides a straightforward way to reduce the computational overhead of LLMs, it may introduce undesirable effects, as informative and uninformative candidates are discarded indiscriminately, leading to unstable results.

2

Theory

2.1 Graph Database

Formally, a knowledge graph can be defined as [11]

$$\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$$

where $\mathcal{E}$ represents the set of entities (nodes), $\mathcal{R}$ represents the set of relations, and $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ represents the set of factual triples (edges).

Query by Subgraph Matching The way to query a graph database is subgraph matching. A typical SPARQL [12] query such as

```
SELECT ?b ?c
WHERE {
    ns:ent1 ?a ?b.
    ?b ns:rel1 ?c.
}
```

contains a `WHERE` clause that specifies a subgraph with some unknown entities and some unknown relations. The unknown entities and relations are fulfilled by the database and returned as the query result.

Bidirectional Neighbor Exploration The relations in a graph database are directed. In well-engineered graphs, every directed relation typically has an inverse relation, making it possible to explore the graph only by matching the subgraph

```
{
    <subject> ?rel ?obj.
}
```

which explores all relations that have a certain subject entity.

Relations that have this "subject" as the object of a triple do not appear in a result of this query. To find that, another query is needed,

```
{
    ?robj ?rrel <subject>.
}
```

which, in case of graphs without inverse relations, is necessary.

2.2 Neural Networks

The lifecycle of a neural network can be broken down into three phases: construction, training (teaching the network), and inference (putting the model to work).

2.2.1 Construction

The Neuron The fundamental building block of any artificial neural network is the neuron. A single neuron performs a very specific, two-step mathematical operation.

1. Linear Transformation: When a neuron receives incoming data, it multiplies each input by a specific weight and then adds a single constant value called a bias. If a neuron receives two inputs x_1, x_2, it computes a weighted sum z:

$$z = w_1 x_1 + w_2 x_2 + b$$

   The parameters w_i, b are tunable. These parameters allow the neuron to compute any linear function.

2. Activation Function: The weighted sum z is a purely linear function. To allow the network to have non-linear behavior, z is passed through a non-linear activation function, denoted as f, to produce the neuron's final output y:

$$y = f(z)$$

   Common activation functions include:

   - Rectified Linear Unit: Outputs the input if it is positive; otherwise, outputs zero. This is common for hidden layers.
   - Sigmoid: Squashes the input into a range between 0 and 1. It is often used in the final output layer for binary classification tasks.
   - Softmax: Squashes the input into a probability distribution. for multi-candidate classification tasks.

The Layer One layer contains many independent neurons that use the same inputs. These independent neurons will have different parameters, resulting in different outputs.

Multiple Layers Layers can stack after one another, using the previous layer's output as input. The count of outputs and inputs must match.

The first and last layers in a series are called the input and output layers. Any layers in between are called hidden layers.

The 2015 introduction of ResNet [33] started a "revolution of depth". The team achieved much better results than the state of art with an extremely deep network at that time, at 152 layers. Since then, neural networks have grown even deeper.

The Initial State To finalize the build before training begins, every single weight and bias in the network is initialized randomly. If all weights were set to zero or the same number, every neuron would perform the exact same calculation, making it impossible to train them to become different numbers. Random initialization ensures that each neuron begins learning a slightly different feature of the data during the training phase.

2.2.2 Inference

During inference, the weights and biases do not change. The network performs a forward pass, *i.e.*, computing each layer sequentially. Assuming the weights have been optimized during training, the transformations meaningfully process the input, resulting in an accurate output classification, translation, or decision.

2.2.3 Training

The goal of training is to change the network so it mimics the training data as much as possible.

Training is a loop that expects convergence of the loss value. In each iteration, inference is first performed to compute the loss against the training data, then gradient descent optimization is applied to minimize the loss.

Loss Function To quantify how wrong the prediction is, a loss function (or cost function) is used. For classification tasks, cross-entropy loss is common; for regression, mean squared error is common. The loss function compares the network's prediction to the true value and outputs a single number L representing the error. The entire objective of training is to minimize this number.

Optimization Looking at a neuron, we can have the expression

$$\frac{\partial L}{\partial w}$$

which is the partial derivative of the loss L with respect to the weight w. If we know the value of $\frac{\partial L}{\partial w}$, then we know how to optimize this w to mimimize L.

The core challenge is figuring out what $\frac{\partial L}{\partial w}$ is. Because the loss is calculated at the very end of the network, and the weight is applied earlier, one must use the chain rule to work backward.

Recall the forward pass of a single neuron:

$$z = wx + b, a = f(z), L = \text{loss}(a)$$

To find how L changes with w, the derivatives of those three steps are multiplied together:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w}$$

All of these can be computed. $\frac{\partial L}{\partial a}$ is known from the loss function. $\frac{\partial a}{\partial z}$ is known from the activation function. Finally, $\frac{\partial z}{\partial w} = x$.

This generalizes to multiple layers.

Once we know this derivative $\frac{\partial L}{\partial w}$, the weight can be updated as

$$w_{\text{new}} = w_{\text{old}} - \alpha \frac{\partial L}{\partial w}$$

where learning rate α is an arbitrary small number. If α is too large, training will not converge.

2.3 Embeddings of Natural Language

Turning natural language into vector embeddings is the foundational process of modern natural language processing (NLP). Generally, the goal is to map discrete, symbolic tokens (words or subwords) into a continuous vector space $\mathbb{R}^d$, where d is usually between 10^3 and 10^5. Each dimension usually comes to express the intensity of a certain *sense* in natural language. In this space, the geometric relationships between vectors mathematically represent the semantic and syntactic relationships between their corresponding words. Notably, meanings can be combined by vector addition in this space, and removed by subtraction.

Once natural language has been successfully mapped to vectors, we need a mathematic way to compare them. The standard metric in NLP for comparing two embedding vectors, $\mathbf{v}_1$ and $\mathbf{v}_2$, is cosine similarity.

$$\text{sim}(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{||\mathbf{v}_1||\ ||\mathbf{v}_2||}$$

Instead of measuring the Euclidean distance (which is sensitive to vector magnitude/word frequency), cosine similarity measures the cosine of the angle between the two vectors:

- 1: Vectors point in the exact same direction (similar meaning).
- 0: Vectors are orthogonal (unrelated meaning).
- -1: Vectors point in opposite directions (opposite meaning).

2.4 Language Model and Transformer

2.4.1 Language Model

The task of a language model is to calculate the probability of a sequence of words, or predict the next word given the previous context, which are very related tasks. Although modern LLMs feel very different from such a primitive description, they still conform to this description—their chat responses are probable linguistic responses.

2.4.2 Language Model Before Transformer

Before the transformer architecture revolutionized NLP in 2017 with [34], a few paradigms have been explored:

- Discrete methods: Early models were built by simply counting word frequencies in a massive text corpus using maximum likelihood estimation. Its discreteness addressed semantics poorly.
- Feed-Forward Neural Networks: In 2003, [35] introduced the use of embeddings in NLP, something that has come to be taken for granted. It addressed semantics better by using embeddings. But its fixed-length context was not an improvement from previous methods.
- Recurrent Neural Networks: The recurrent architecture theoretically supported arbitrarily long context windows, given its hidden state. However, a long context in this architecture results in the vanishing gradient problem, making it impossible to train. [36]
- Gated Recurrent Networks: A special gated unit is added to the recurrent architecture, enabling the network to remember and forget certain information. It can thus handle longer contexts without the vanishing gradient problem. [36] But the recurrent architecture is not parallelizable, limiting its scale.

2.4.3 Transformer

Throughout this history, we see a clear goal of utilizing *context* in NLP. Attention is a method of building contextual embeddings at a particular layer of a language model, by selectively integrating information from prior tokens at the previous layer [37]. It is simply a weighted sum of context vectors, and the complexity lies within how the weights are computed [37]. Attention has been used in earlier works, but the significant contribution of [34], which powers the vast majority of modern LLMs, is replacing recurrent neural networks with *self*-attention, the only kind of attention we describe.

This new kind of attention involves query, key, and value matrices Q, K, V in a single attention mechanism. They represent three different roles that each input embedding plays [37]:

- as the current element being compared to the preceding inputs.
- as the preceding inputs being compared from the current element.
- as a value of this kind of query for a preceding element.

The self-attention is computed as:

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

A transformer contains multiple layers, and each layer contains multiple attention mechanisms that function independently.

Importantly, the transformer is much more parallelizable than its predecessors, enabling the scale of modern LLMs.

2.4.4 Training a Modern LLM

Modern LLMs are trained in two steps: pre-training, and fine-tuning [37].

Pre-training is very resource-intensive. The model is trained on a huge amount of text, so it can predict the next token without any particular goal. The theory used in this step has already been introduced above.

Once pre-training has completed, we have a model that masters the language it is trained on, but still does not know how to perform specific tasks. It naturally generates text consistent with the context, for example [38],

> Prompt: Explain the moon landing to a six year old in a few sentences.
> Output: Explain the theory of gravity to a 6 year old.

Task-specific capabilities are added in a process called fine-tuning. These capabilities include sentiment analysis, spam detection, question answering, named entity recognition, and so on. Many modern LLMs have been fine-tuned for instruction following, enabling us to use zero-shot, algorithmic instruction prompts. Fine-tuning requires very little resources compared to pre-training.

2.5 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is a foundational pre-trained language model that introduced a deep bidirectional Transformer encoder architecture, significantly advancing natural language understanding [39]. Previously, models were restricted to unidirectional text processing. BERT, however, utilizes bidirectional self-attention. This mechanism allows the model to simultaneously incorporate context from both preceding and succeeding tokens, granting it a profound capability to capture complex syntactic and semantic nuances across text sequences.

Pre-training of BERT contains two kinds of tasks [39]:

1. Masked LM: mask some percentage of the input tokens at random, and then predict those masked tokens
2. Next Sentence Prediction: a binarized next sentence prediction task, where the model is taught what it is like for a sentence to meaningfully follow another, and the opposite as well. This is beneficial to both question answering and natural language inference.

BERT has been fine-tuned for a number tasks, but we do not use these fine-tuned models as we only need to compute vector similarity.

2.6 Sentence-BERT

Although BERT can produce sentence embeddings via basic `[CLS]` extraction or average pooling, these methods often fail to fully encapsulate sentence-level semantics. To address this shortcoming, Reimers and Gurevych introduced Sentence-BERT (SBERT) [40]. SBERT modifies the original architecture by incorporating a Siamese network structure.

Once these independent embeddings are generated, the model measures their semantic relatedness utilizing cosine similarity.

Due to the fact that the encoding process is independent and deterministic, SBERT ensures high efficiency on large-scale retrieval and clustering applications while preserving deep semantic accuracy.

2.7 Retrieval-Augmented Generation

Large Language Models often experience hallucinations by generating statements that are factually incorrect. To address this inherent limitation, Retrieval-Augmented Generation (RAG) proposes combining non-parametric memory modules with parametric generation to handle external knowledge [41].

Formally, given an input query x, a standard LLM attempts to generate an output sequence y by maximizing the conditional probability $P(y|x)$ relying solely on its frozen weights. In the RAG framework, the system first searches for relevant information in a non-parametric memory pool via retrieval mechanisms [41]. RAG treats these retrieved knowledge snippets as hidden variables z. The generation process is mathematically factorized into two probabilistic components: a retriever module p_η and a generator module p_θ. The probability of generating the final answer sequence y is computed by marginalizing over the retrieved context $z \in \mathcal{Z}$:

$$P(y|x) = \sum_{z \in \mathcal{Z}} p_\eta(z|x) p_\theta(y|x, z)$$

Here, $p_\eta(z|x)$ denotes the probability of retrieving a specific knowledge snippet z given the query x. Additionally, $p_\theta(y|x, z)$ represents the probability of the output generator producing the answer y, conditioned on both the original query x and the additional context information z. By conditioning the generation process on explicitly retrieved facts rather than pure parameter memorization, the RAG framework ensures that the model's outputs are more specific, diverse, and deeply anchored in verifiable knowledge.

2.8 Prompt Engineering

Prompt engineering refers to the deliberate construction of input contexts to steer the behavior of LLMs without modifying their internal parameters. Rather than

relying on fine-tuning, this paradigm leverages carefully designed instructions, formatting conventions, and contextual signals to influence how a model interprets and responds to a task. As a result, prompt design functions as a primary mechanism for adapting general-purpose LLMs to downstream applications.

A key aspect of prompt engineering lies in how much contextual guidance is provided to the model. This has led to a spectrum of in-context learning strategies, ranging from instruction-only prompts to those augmented with illustrative examples.

2.8.1 In-Context Learning Strategies

At one end of the spectrum, **zero-shot prompting** relies solely on a task description, without providing any explicit demonstrations. In this setting, the model must infer the task structure based entirely on patterns acquired during pretraining. While this approach offers maximum flexibility and minimal overhead, it may yield inconsistent results when task requirements are underspecified or involve complex reasoning. To improve reliability, practitioners often incorporate auxiliary cues such as role specifications or reasoning triggers (e.g., encouraging step-by-step thinking), which help constrain the model's interpretation of the task [42], [43].

Introducing examples into the prompt leads to **one-shot** and **few-shot prompting**, where the model is guided through explicit input-output pairs. A single example (**one-shot**) can already provide useful inductive bias, especially in interactive or exploratory settings [44]. Extending this to multiple examples (**few-shot**) generally improves performance by reinforcing task patterns and reducing ambiguity [45].

However, the effectiveness of example-based prompting depends on several design factors. These include the semantic similarity between examples and the target query, the ordering of demonstrations, and the consistency of formatting [46]. Poorly chosen examples may introduce noise or unintended biases, whereas carefully selected ones can significantly enhance task alignment. Recent approaches have explored automated methods for selecting representative examples, such as similarity-based retrieval techniques, to further optimize prompt quality.

2.8.2 Chain-of-Thought Prompting

For tasks that require multi-step reasoning, **Chain-of-Thought (CoT)** prompting extends the idea of few-shot learning by embedding intermediate reasoning processes within the examples [47]. Instead of directly mapping inputs to outputs, the model is encouraged to generate a sequence of logical steps that lead to the final answer.

This structured reasoning format has been shown to improve performance across a range of cognitively demanding tasks, including arithmetic problem solving, commonsense inference, and symbolic reasoning. The benefits arise from making the reasoning process explicit, which helps the model maintain coherence over multiple steps rather than relying on implicit pattern matching.

In addition to improving accuracy, CoT prompting enhances the transparency of model outputs by exposing the intermediate decision process. While larger models

tend to benefit more substantially from this approach due to their stronger reasoning capabilities, smaller models can also exhibit measurable gains when guided by structured reasoning patterns. Overall, different prompting strategies reflect trade-offs between simplicity, control, and performance, and their effectiveness fundamentally depends on the complexity of the target task.

2.9 BM25

BM25 serves as a probabilistic ranking framework widely utilized in information retrieval to assess the relevance between a document and a specific query [48]. As an extension of earlier Term Frequency-Inverse Document Frequency (TF-IDF) architectures, the model introduces term frequency saturation alongside document length normalization. Consequently, it achieves enhanced ranking outcomes while keeping computational expenses negligible.

The algorithm assigns a comprehensive relevance score to a document d against a query q by accumulating the contributions of each matching term t:

$$\text{Score}(q, d) = \sum_{t \in q} \text{IDF}(t) \cdot \text{TF}(t, d)$$

The Inverse Document Frequency (IDF) component captures the significance of a term across the entire collection. This mechanism ensures that common terms contribute minimally to the final score, whereas rarer, informative terms are weighted more heavily:

$$\text{IDF}(t) = \log\left(\frac{N - n_t + 0.5}{n_t + 0.5}\right)$$

where N represents the total number of documents, and n_t denotes the number of documents containing term t.

The Term Frequency (TF) component reflects the occurrence rate of term t in document d while considering diminishing returns:

$$\text{TF}(t, d) = \frac{f(t, d) \cdot (k_1 + 1)}{f(t, d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\text{avgdl}}\right)}$$

In this equation, $f(t, d)$ indicates the term frequency, $|d|$ is the current document's length, and $avgdl$ is the average document length. The parameters k_1 (typically between 1.2 and 2.0) and b (usually around 0.75) are tunable constants.

2.10 Reciprocal Rank Fusion

In hybrid search architectures, aggregating results from dense retrieval models (e.g., SBERT) and sparse retrieval models (e.g., BM25) persents a fundamental challenge:

their scoring mechanisms are inherently incompatible. While cosine similarity produces bounded scores, probabilistic models such as BM25 generate unbounded values. Consequently, direct score aggregation or linear combination requires careful calibration and may lead to instability.

To effectively integrate these heterogeneous results without relying on score normalization or additional training, Reciprocal Rank Fusion (RRF) is widely adopted as a robust ensemble ranking technique [49]. RRF is an unsupervised fusion algorithm that operates solely on rank positions rather than absolute scores.

The algorithm computes a fused relevance score for a document d by aggregating the reciprocal ranks assigned by multiple independent rankers:

$$\mathrm{RRF}_{\mathrm{Score}}(d) = \sum_{r \in R} \frac{1}{k + r(d)}$$

where R demotes the set of rankers and $r(d)$ is the 1-based rank of document d given by ranker r. The constant k is a smoothing parameter, commonly set to 60 in prior work. This parameter reduces the dominance of top-ranked documents while allowing lower-ranked but consistently retrieved items to contribute to the final ranking.

By eliminating the need for score calibration and training. RRF provides a simple yet effective late-fusion strategy and has been shown to outperform individual ranking methods in many retrieval settings.

2.11 Beam Search and Heuristic Exploration

In the context of multi-hop reasoning over large-scale knowledge graphs, exhaustive search methods such as Breadth-First Search (BFS) or Depth-First Search (DFS) quickly become computationally intractable. This is primarily due to the combinatorial explosion of the search space, where the number of possible relational paths grows exponentially with the number of hops. To mitigate this, heuristic search algorithms are widely employed, among which Beam Search is the most prominent [50].

Beam Search can be viewed as a constrained variant of best-first search that has become the standard decoding algorithm for neural sequence generation [51]. Instead of maintaining the entire search tree, it relies on a predefined hyperparameter called the beam width, denoted as k. At each expansion step (or depth level), the algorithm evaluates all successor nodes generated from the current states, but only retains the top-k most promising candidates based on a specific heuristic scoring function. The rest of the nodes are permanently pruned from the search space.

While Beam Search significantly reduces computational overhead and memory consumption, its effectiveness is entirely dependent on the reliability of the underlying heuristic evaluation [52]. In standard LLM-guided exploration paradigms like Think-on-Graph (ToG), the LLM itself serves as the heuristic scorer. However, when the

model is forced to assign absolute pointwise scores to a massive list of highly similar candidates simultaneously, the scoring distribution may degrade, potentially leading to the pruning of globally optimal reasoning paths. This fundamental limitation highlights the necessity for search mechanisms that rely on different selection methods.

2.12 Tournament Selection Mechanism

The term "tournament selection" conceptually originates from competitive sports, where participants are structured into local match-ups or knockout brackets to determine a victor without requiring every participant to compete against all others. In computer science, this paradigm was formally adopted and popularized within the domain of evolutionary computation and genetic algorithms to bypass the severe inefficiencies of global fitness evaluations [53].

Early evolutionary selection mechanisms, such as fitness proportionate selection (commonly known as roulette wheel selection), mandated the calculation of absolute fitness scores for every individual across the entire population to derive global selection probabilities. This approach was not only computationally expensive but also highly vulnerable to scale distortiona phenomenon where extreme outlier scores could dominate the probability distribution, leading to a premature loss of diversity. To circumvent these bottlenecks, tournament selection relies purely on local, relative comparisons. In a standard tournament process, a small subset of candidatesreferred to as a "chunk" or "tournament pool" is randomly sampled. These individuals compete locally, and only the highest-ranking candidates survive. This simple yet elegant mechanism effectively eliminates the need for global sorting and makes the algorithm largely insensitive to the absolute magnitude of fitness scores [53].

This historical shift in heuristic optimization closely mirrors the current methodological transition required in LLM-based reasoning frameworks. In standard architectures like ToG, LLMs are utilized as pointwise scorers across the entire candidate space, encountering the exact same vulnerabilities as early genetic algorithms: high computational overhead and unstable, polarized absolute scoring distributions. Recently, the comparative or tournament-style comparative paradigm has been successfully adapted to address these calibration limitations in Large Language Models [54].

By reframing the evaluation as an iterative, tournament-based pairwise or chunked comparison, the task is reduced to determining local relative preferences rather than global absolute truths. This aligns much better with an LLM's natural judgment capabilities, allowing the model to develop a consistent ranking perspective without exceeding its context processing limits in related ranking or evaluation settings [55]. Consequently, implementing tournament selection in multi-hop reasoning can effectively mitigate the limitations of absolute scoring, potentially improving evaluation stability and pruning behavior in large-scale knowledge graphs.

2.13 Merge Sort

Merge sort [56] is a sort algorithm that relies on pairwise comparison of elements.

Merge sort is a divide-and-conquer algorithm. With a list of n elements:

- Divide: The algorithm recursively splits the unsorted list in half until it can no longer be divided. This results in n sublists, each containing exactly one element. By definition, a list with a single element is already sorted.
- Merge: The goal is to merge all lists into one. To merge two lists, the algorithm creates an empty new list l, compares the top item from each of the two input lists, moves the greater one to the new list l, and iterates until one input list is empty, in which case all remaining items are directly added to the new list l. List l is returned.

This algorithm has a time complexity of $O(n \log n)$ for the best and worst cases.

Merge sort cannot be done in-place on arrays, but can be done in-place on linked lists. The space complexity is $O(n)$ for arrays and $O(1)$ for mutable linked lists.

Merge sort is a stable sorting algorithm. If two elements have the same value, their relative order in the original unsorted list is preserved in the final sorted list.

Because the left and right halves of the arrays are sorted independently during the divide phase, merge sort is well-suited for parallel processing and multi-threading.

3

Method

Three Observations

Our methods are based on three basic observations.

Observation I: Relation Overlap and Redundancy

In the ToG framework, candidates are presented as complete triples (e_1, r, e_2), where e_i represents entities and r denotes a relation. We observed a significant overlap in relations among the explored candidates. For instance, as shown in Figure 3.1, multiple entities share the same relation r_2. In this case, it is redundant to treat each triple independently for LLM evaluation. It will be helpful if overlapped relations are merged.

Observation II: Entity-Agnostic Reasoning Potential

Another critical insight is that, in multi-hop reasoning over knowledge graph, the semantic utility of a reasoning path is largely governed by its sequence of relations. For example, given the following question,

> What is the national bird of where Chicago is?

a straightforward path is $\textit{Chicago} \xrightarrow{\textit{located\_in}} x \xrightarrow{\textit{national\_bird}} y$. Although there is no information about entities (except for the topic entity), it is still clear that the path should be kept for answering.

Formally, the relation described above could be abstracted as compositional relation $R(X, Z)$ defined over relation paths.

$$R(X, Z) = \exists Y,\ \text{located\_in}(X, Y) \wedge \text{national\_bird}(Y, Z)$$

The observation mentioned above reflects the potential for improving the pruning process with entity-agnostic relation paths.

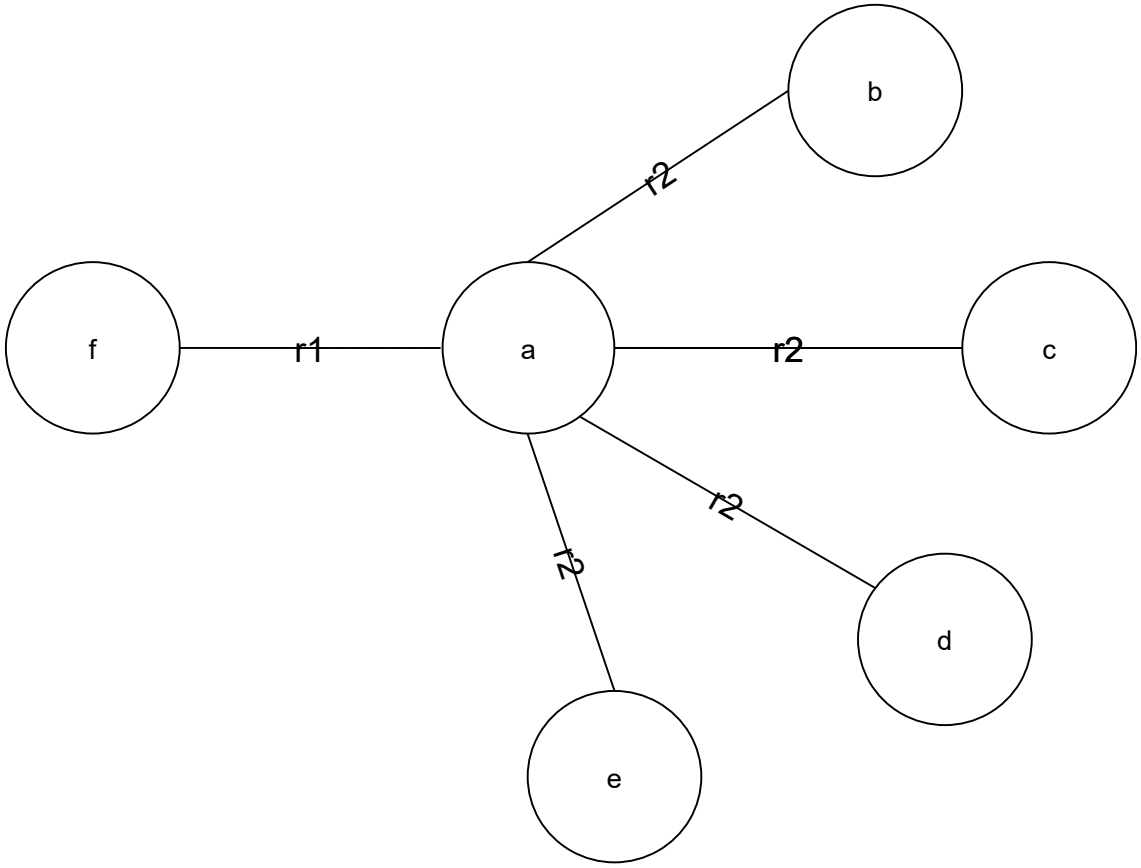

Figure 3.1: In ToG, we have triples as candidates: { $(a, r1, f)$, $(a, r2, b)$, $(a, r2, c)$, $(a, r2, d)$, $(a, r2, e)$ }, where overlapped relations are not merged.

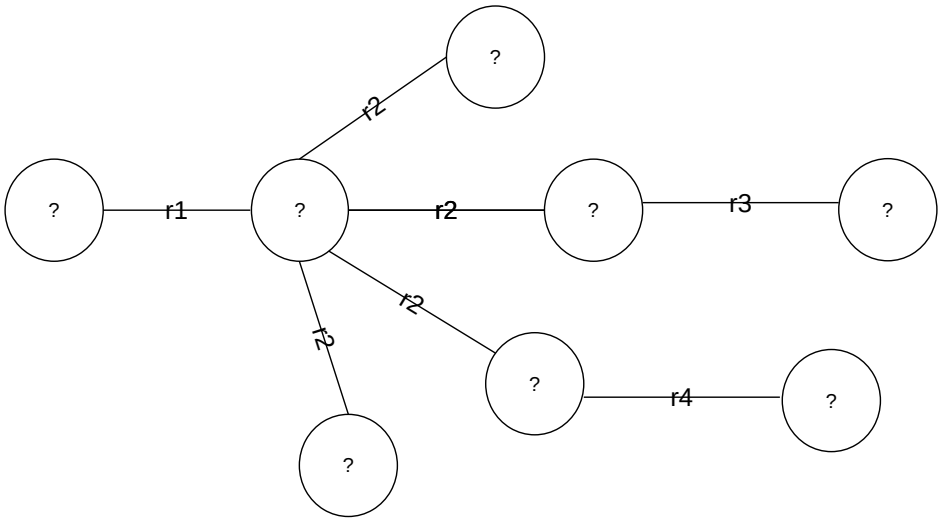

Figure 3.2: After ignoring entities, candidates become relation paths: { $(r1)$, $(r2)$, $(r2, r3)$, $(r2, r4)$ }. With merging operations mentioned in I, we could generate fewer candidates while maintaining enough information for pruning.

Observation III: LLM as Chooser, Not Scorer

ToG relies on assigning absolute numerical scores, leading to scoring failures such as degrading to extreme scores (0 or 1) and assigning the same score to a huge amount of different candidates. This may be due to the absence of calibration in LLM scoring.

Furthermore, scores are not necessary required for ToG—it is sufficient to give just a rank (or top-k rank). Recent work on LLM-based evaluation also shows that relative comparative ranking improves evaluation consistency over absolute scoring [57], [58], [59]. Therefore, it is better to treat the LLM as a comparative chooser rather than a point-wise scorer.

3.1 Hybrid Lexical-Semantic Pre-filter

As mentioned at the end of **Introduction**, ToG employs random sampling to reduce the number of neighbors in knowledge graphs. But this method retains both relevant and irrelevant candidates. For the worst case, it may discard all useful candidates, making it impossible for the LLM to deduce the correct answer.

However, it is still necessary to figure out a method to reduce the candidates for lower computational overhead of LLMs. Instead of random sampling, we propose a hybrid method to conduct selective pre-filtering.

To effectively reduce the search space while preserving relevance, ToP replaces random discarding with a hybrid pre-filtering strategy that leverages both lexical and semantic signals. As previously established, BM25 [48] provides robust exact keyword matching but is susceptible to vocabulary mismatch, whereas SBERT [40] captures deep contextual semantics but may overlook fine-grained, strict token alignments. Given their clear complementarity, integrating these two approaches yields a much more robust candidate screening process. To circumvent the normalization challenges associated with combining fundamentally disparate scoring distributions, we fuse their independent rankings using RRF [49]:

$$\text{Score}_{RRF} = \frac{1}{k + r_{\text{BM25}}} + \frac{1}{k + r_{\text{SBERT}}} \tag{3.1}$$

where r_{BM25} and r_{SBERT} denote the rank positions and k is a smoothing constant (typically 60). This produces a compact candidate set, retaining high-relevance entities.

3.2 Entity-Agnostic Relation Path Pruning

When performing pruning, ToG only keeps current relations or entities without their previous path information. This makes ToG limited to local context and may fail to capture global path semantics.

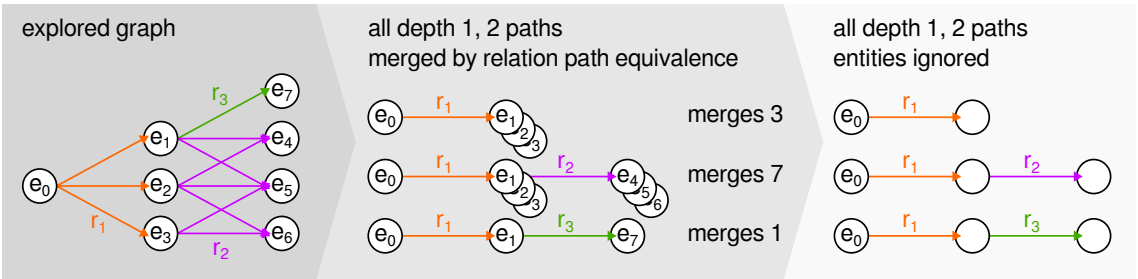

Figure 3.3: **Entity-Agnostic Relation Path Pruning** The raw explored subgraph (left) is first grouped by relation path equivalence (middle). Then, the entities of each grouping path are ignored, abstracting them into purely relational paths (right).

Based on **Observation I & II**, we propose an entity-agnostic relation path pruning approach to address this issue. In ToP, intermediate entities are deliberately ignored in the relation path pruning process, thereby merging full entity-aware paths by identical start entity (or Topic Entity) and sequence of relations(Figure 3.3). More formally, a full path could be defined as

$$p : e_0, r_1, e_1, r_2, e_2, ..., r_n, e_n$$

where e_i denotes entities (e_0 represents the topic entity) and r_i denotes relations. Given two paths p_1 and p_2, we define they are path-equivalent if and only if their subsequences $e_0, r_1, r_2, ..., r_n$ are identical.

This approach enables ToP to prune more effectively by allowing LLM to evaluate candidates with their global path semantics while reducing entity-level noise.

3.3 Tournament Candidate Selection

After the pre-filter, there can still be plenty of candidates in the pool, which may lead to the "lost in the middle" phenomenon, where LLM performance degrades when relevant information appears in the middle of long contexts. To address this, we propose the tournament candidate selection algorithm so each prompt can contain fewer candidates. In detail, two distinct versions are proposed, each with its specific advantages.

3.3.1 Chunked Tournament

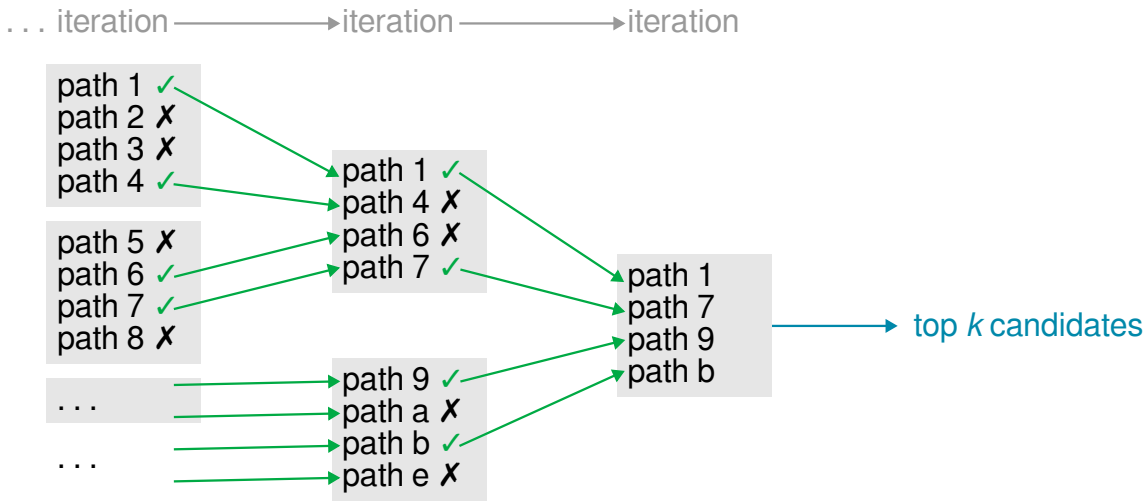

Figure 3.4: **Chunked Tournament** Initially, candidates are partitioned into multiple chunks, with each chunk containing a fixed number of paths (e.g., four). During each selection round, only the top 50% of paths within a chunk are retained and subsequently regrouped into new chunks. This iterative process continues until all surviving candidates fit into a single final chunk, from which the ultimate top-k candidates are selected.

In this version, we divide candidates into a few chunks (Figure 3.4), where the number of candidates in each chunk is strictly capped at a predefined threshold. During each round, the LLM chooses to keep the top 50% candidates in a chunk. The selected candidates are then recombined into better chunks. This process iterates until the remaining candidates fit within a single chunk, so the final top-k selection can be performed. This hierarchical selection provides more stable ranking across varying candidate pool sizes.

To execute this chunk-wise evaluation effectively, we carefully design a specific prompt template that instructs the LLM to compare and rank candidates within each given chunk, as detailed below.

Prompt Design

Relation Pruning Prompt Instead of giving scores, our prompt asks LLM to give *top-k* helpful relation chains by giving a list of indexes. Typically, the results given by LLM should have up to k chains. If LLM does not follow the instruction and gives more chains, there will be a parser to truncate the list to k.

```
Instruction: Given the question, you are provided with
   several candidate relation chains derived from a knowledge
    graph. Your task is to select UP TO {k} relation chains
   that are most helpful for answering the question.

Format Example:
Q: The movie featured Miley Cyrus and was produced by Tobin
   Armbrust?
Candidate Relations:
[0] film.actor.film -> film.film.genre
[1] film.film.production_companies -> organization.
   organization.founders
[2] film.actor.film -> film.film.produced_by
A: [1, 2]
```

Entity Pruning Prompt In order to save costs without sacrificing path information, the paths are grouped based on their prefixes. In each group, there is one "Explored Path" to show the prefix and a set of candidate entities. Similarly, we ask LLM to give a list of indexes with length up to k. Another parser function will truncate the list if the list has more elements than k.

```
Instruction: Please evaluate all provided candidate entities
   across different reasoning paths, and select UP TO {k}
   entities in total that are most helpful for answering the
   question.

Format Example:
---
Explored Path: [Entity] Miley Cyrus -> [Relation] film.actor.
   film
```

```
Candidate-Entities:
[0] So Undercover
[1] The Last Song
---
Explored Path: [Entity] Tobin Armbrust -> [Relation] film.
   film.produced_by
Candidate-Entities:
[2] Let Me In
[3] So Undercover
---
A: [0, 3]
```

3.3.2 Pairwise Tournament

In this version, we sort the candidates with the LLM by asking it to rank only two candidates at a time. We implemented this as a special version of merge sort.

We modify merge sort to limit the item count in l to some k during this merge step[1]. The modified merge routine ends once the merged list l reaches size k, saving time that would have been used to compare remaining (worse) candidates. For $n = 30$, about[2] 119 comparisons are required in the worst case to fully sort the list, but only 72 comparisons are required to find and sort the top 3, a 40% saving. The asymptotic time complexity is still $O(n \log n)$.

For our purpose, we do not require LLM comparison to be consistent (for example, $a > b > c > a$ is permissible) or transitive, as our goal is to exploit the LLM's reasoning capability enough to probabilistically find the more accurate paths.

Diff Prompting

In order to trick the LLM into comparison thinking, we symbolically extract the longest common prefix of the two path candidates. This common prefix is presented as "given conditions" to the LLM. The remaining segment of each path, the *path diff*, each becomes an option. In case of any empty path diff, the question is formulated as a relevance decision problem and only the nonempty path diff is presented, so the LLM does not become confused over an empty option.

In case of two nonempty path diffs the following zero-shot prompt structure is used:

```
You are helping a computer program perform question answering
    on a knowledge graph.
Do not pick information that is redundant.
In combination with the known conditions, which one of the
   candidates is more relevant to the following question?
Question: m.03_dwn is the mascot for the team that last won
   the world series when?
```

[1] We used $k = 3$.

[2] depends on implementation details

```
Given conditions: -> GIVEN_TOPIC_ENTITY: m.03_dwn
Candidate A: -> sports.mascot.team <- common.webpage.topic
Candidate B: <- sports.sports_team.team_mascot <- award.
   award_nomination.award_nominee
First explain your reasons in one sentence for each option,
   then write the response as [A] or [B].
```

Sample response:

```
Candidate A describes a general relationship between a mascot
    and a webpage topic, which is too broad. Candidate B
   connects the team mascot to sports teams and awards, which
    is relevant to winning a world series.
[B]
```

In case of any empty path diff the following zero-shot prompt structure is used:

```
You are helping a computer program perform question answering
    on a knowledge graph.
Do not pick information that is redundant.
In combination with the known conditions, is the candidate
   relevant to the following question?
Question: m.03_dwn is the mascot for the team that last won
   the world series when?
Given conditions: -> GIVEN_TOPIC_ENTITY: m.03_dwn <- sports.
   sports_team.team_mascot
Candidate: <- baseball.baseball_league.teams
First explain your reasons in one sentence for each option,
   then write the response as [RELEVANT] or [NOT RELEVANT].
```

Sample response:

```
The candidate is too broad and does not specifically mention
   the team that won the world series or the mascot mentioned
    in the question.
[NOT RELEVANT]
```

3.4 Topic Entity Masking

Sometimes, very easy questions like the following are asked:

> The national anthem Afghan National Anthem is from the country which practices what religions?

In this case, the LLM may choose to answer the question using its internal knowledge rather than the knowledge base. In fact, we can even test this kind of QA system against a fictional knowledge graph (with real-world entity names and relations, but all relations are connected in a very different way), and it will perform very poorly.

While ToG encourages the LLM to use its internal knowledge, improving test set outcomes when run against real-world KGs, this approach is fundamentally wrong and undermines our principle: question answering should be backed by a human-verified external source.

In order to prohibit the LLM from using its internal knowledge, we mask any matching entity names in the question, so the question instead looks like:

> The national anthem `m.02r0hl7` is from the country which practices what religions?

3.4.1 The Role of the LLM, Revisited

The LLM must know all the relations in the KG domain in order to help us with the task. For example, it must know that a country usually has a national anthem, and that a country can have a predominant religion. This is the only way it can help find the relevant information.

However, the LLM must not help us with any specific entities, such as those specific to Afghanistan, as it is considered wrong to not retrieve such information from the KG.

4

Construction of Our Two Systems

We constructed two different systems, using different methods from the previous chapter.

Property	**System 1**	**System 2**
Information used for exploration	Path and Entity	Path only
Tournament ranking mode	Chunked	Pairwise
Tournament prompting	Few-shot	Zero-shot, diff
Pre-filter	Lexical, semantic	Semantic
Topic entity masking	No	Yes
Answer from LLM knowledge	Yes	Impossible
Search depth	Fixed	Until answered

Table 4.1: Feature comparison of our two systems.

4.1 System 1: Path and Entity Exploration, Chunked Tournament, Hybrid Filter

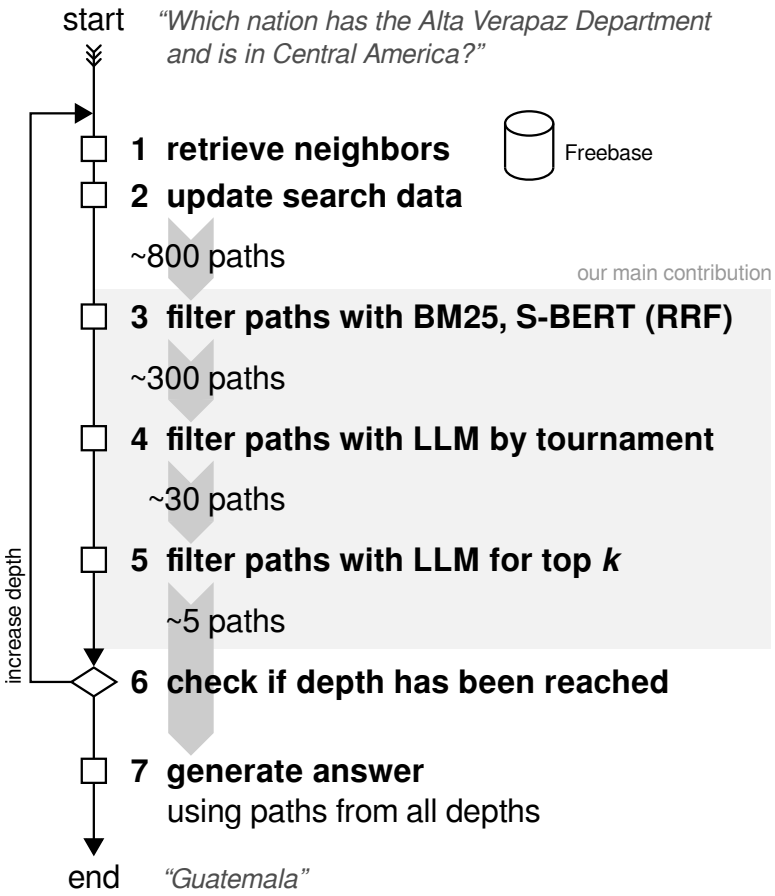

Figure 4.1: Overall architecture of System 1

System 1 requires a fixed depth that allows for ToP to explore. And for each iteration, ToP first filters the paths with hybrid pre-filtering method to narrow down the number of candidates and generate a small candidate pool. The candidates are then divided into chunks and ranked for selection. This iterative process continues until only one chunk remains and the problem turns into a top-k ranking. Once the depth is reached, the paths are then sent to the LLM to deduce the answers.

4.2 System 2: Path-only Exploration, Pairwise Tournament, Topic Entity Masking

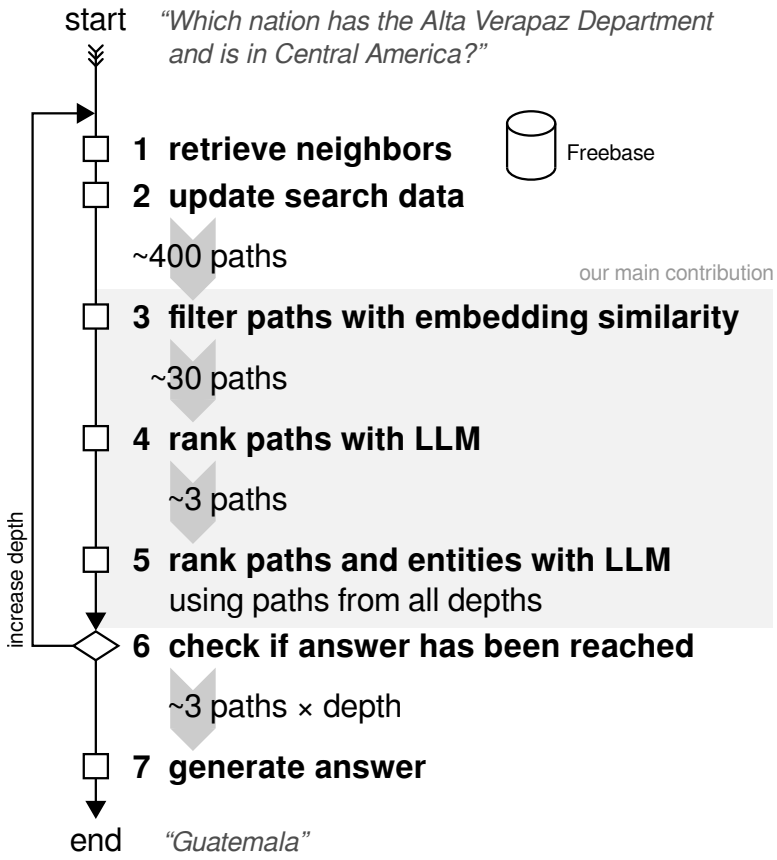

Figure 4.2: Overall architecture of System 2

System 2 has the following features:

- In Step 3: Single Filter: Semantic top-k filter powered by a single embedding model.
- In Step 4: Path-only Exploration: Unlike the other design, there is no entity pruning during exploration.
- In Step 4: Pairwise Tournament with zero-shot diff prompting.
- In Step 6: Search can be ended by a positive response from the LLM.
- In Steps 3, 4, 5, 6: Topic Entity Masking in all prompts.

5

Results

In this chapter, we focus on the following research questions.

1. To what extend does the proposed framework improve multi-hop reasoning performance over ToG?
2. How does model scale affect reasoning performance, and how does the proposed method reshape this scaling behavior?
3. How consistently does ToP perform across datasets with varying schema complexities, compared to the baseline?
4. How does ToP behave if key improvements are ablated?

Before answering these questions, we first detail the experimental setup and then conduct a detailed analysis across three corresponding dimensions on two different versions (as mentioned in Section 3.4).

5.1 Experiment Setup

5.1.1 Datasets

Following ToG [20], we employ Freebase [60] as our underlying knowledge graph.

Our query language is SPARQL [12], a pure implementation choice that does not affect performance.

We perform our evaluations across four widely recognized benchmarks featuring complex, multi-hop reasoning tasks: CWQ [61], WebQSP [62], WebQuestions (WebQ) [63], and GrailQA [64].

In order to optimize computational efficiency while maintaining evaluation integrity, we randomly sample a subset of 200 questions from each dataset for our evaluation, except otherwise noted.

5.1.2 Metrics

Exact Match (EM) is adopted as our primary metric to strictly determine whether the predicted answer aligns with the ground-truth answer.

LLM	Size Variant	System 1	System 2
Deepseek R1	671B	*	
Llama 3.3 Instruct	70B	*	
Qwen 3	32B	*	*
	14B	*	*
Gemma 4 Instruction-Tuned	Effective 4B		*
	Effective 2B		*

Table 5.1: LLMs used to test our two systems.

5.1.3 Language Models

The two versions of tournament selection have different advantages—chunked tournament selection costs fewer tokens and is suitable for larger models whereas pairwise tournament selection makes better use of language models and is suitable for smaller models. Therefore, we use different models according to the tournament implementation.

For System 1, we choose the 14B and 32B variants of Qwen 3 models [65] as well as two larger models, Llama 3.3 70B Instruct [66] and Deepseek R1 [67]. For System 2, we keep the Qwen models, but replace two large models with small models, Gemma 4 E2B IT, E4B IT (instruction-tuned) [68]. An overview is in Table 5.1.

For System 2, the embedding model in the filter is Qwen 3 Embedding 4B [69].

5.2 Results

5.2.1 System 1

Model		CWQ	WebQSP	WebQ	GrailQA
Deepseek R1	w/ ToG	46.0	72.0	46.5	62.5
	w/ System 1	64.0	81.0	54.0	70.0
Llama 3.3 70B Instruct	w/ ToG	34.5	43.0	24.0	22.0
	w/ System 1	56.5	79.0	58.0	64.0
Qwen 3 32B	w/ ToG	38.0	58.5	46.0	50.0
	w/ System 1	43.5	72.0	48.5	72.0
Qwen 3 14B	w/ ToG	34.0	63.5	54.0	56.0
	w/ System 1	38.0	63.5	46.5	60.0

Table 5.2: Performance of various LLMs equipped with ToP System 1 versus ToG.

5.2.1.1 Performance Grains Across Frameworks

Table 5.2 shows that System 1 consistently outperforms the standard ToG across nearly all tested LLM and benchmarks. Notably, when equipped with Llama3.3-70B-Instruct, System 1 achieves substantial absolute EM improvements of 22.0% on CWQ, 36.0% on WebQSP, 34.0% on WebQ and 42.0% on GrailQA. Even for a strong reasoning model such as Deepseek-R1, System 1 further improves performance, yielding an 18.0% absolute gain on CWQ.

These improvements suggest that modifying step-wise exploration into global path level reasoning, combined with list-wise candidate selection, leads to more effective filtering of relevant reasoning paths. Moreover, while the entity-agnostic path abstraction naturally introduces a larger candidate space during final entity resolution, the overall performance gains demonstrate that our tournament selection algorithm successfully manages the increased search complexity without degrading accuracy.

5.2.1.2 Impact of LLM Model Scale and Capability

ToG results show a clear performance inversion, or negative scaling, among open-source models. The larger Llama-3.3-70B-Instruct underperforms the smaller Qwen-3-14B on WebQ (24.0 vs. 54.0) and GrailQA (22.0 vs. 56.0), with a similar trend observed for Qwen-3-32B (46.0 on WebQ and 58.5 on WebQSP). We attribute this to the strict evaluation criteria. While necessary to prevent false positives, this may penalize the longer and more flexible responses generated by larger models.

First, larger instruct-tuned models tend to produce more fluent and informative reasoning path, but they also generate more verbose outputs that may deviate from the strict structural format (e.g., bracketed representations) required by ToG. This constraint is inherent to the framework design for reliable downstream parsing; deviations from the required format therefore lead directly to scoring penalties.

Method	CWQ	WebQSP	WebQ	GrailQA
System 1	44	72	49	72
System 1 w/o Relation Path Pruning	32	67	48	70
System 1 w/o Tournament	40	71	54	66
System 1 w/o Pre-filter	40	72	50	70

Table 5.3: Ablation study of System 1 under Qwen 3 32B. "w/o" denotes the removal of a specific component. Numbers are percetage points.

Second, during graph exploration, the number of candidate reasoning nodes grows combinatorially. To maintain the computational budget, vanilla ToG applies a fixed-width pruning strategy disregarding path quality. Instead, candidate path are truncated through a quality-agnostic process, where all paths have equal probability of being discarded. As the candidate space expands, this creates a stronger selection bottleneck and weakens the link between generation quality and retained reasoning paths.

System 1 addresses both issues in a unified way. By introducing optimized guiding prompts, System 1 improves structural compliance and reduces formatting drift. More importantly, instead of uniform discarding, System 1 uses a semantic hypergraph to organize the search space and applies a tournament pruning strategy to retain the most promising reasoning paths in a quality-aware manner. As a result, the negative scaling behavior is consistently mitigated, with performance improving from 14B to 32B, and further to 70B across all datasets, suggesting more effective utilization of larger models.

5.2.1.3 Performance Consistency Across Datasets

The baseline ToG framework shows significant performance variation across datasets with different graph-query distributions. For example, Llama-3.3-70B-Instruct achieves 43.0% on WebQSP under the baseline setup, but drops to 24.0% on WebQ and 22.0% on GrailQA. This gap suggests that the effectiveness of vanilla graph-based reasoning is sensitive to differences in question complexity and style, rather than scaling consistently with model capacity.

In contrast, System 1 exhibits stable performance across different benchmarks, mitigating performance degradation on complex datasets while preserving strong results on simpler ones. A representative example is Qwen3-14B: when equipped with System 1, the model improves its accuracy on challenging multi-hop tasks such as CWQ (from 34.0 to 38.0) and GrailQA (from 56.0 to 60.0), while maintaining its strong baseline performance on WebQSP (63.5). This pattern indicates that System 1 provides more consistent behavior across datasets, rather than being optimized to a specific benchmark.

5.2.1.4 Ablation Study

Table 5.3 presents the ablation study of System 1 using Qwen 3 32B and 100 samples. Overall, the removal of each component results in a decrease of accuracy.

Model		CWQ	WebQ	WebQSP	GrailQA
Qwen 3 32B	w/ ToG	38.0	46.0	59.5	50.0
	w/ System 2	43.5	63.5	73.0	74.0
Qwen 3 14B	w/ ToG	34.0	54.0	62.5	56.0
	w/ System 2	37.5	57.0	62.5	59.5
Gemma 4 E4B IT	w/ ToG	31.0	28.0	39.0	53.5
	w/ System 2	46.0	56.5	67.5	74.0
Gemma 4 E2B IT	w/ ToG	12.0	17.0	22.0	10.5
	w/ System 2	39.5	52.0	64.5	67.5

Table 5.4: Performance of various LLMs equipped with ToP System 2 versus ToG.

5.2.2 System 2

We conduct an analysis across three distinct dimensions. The results are shown in Table 5.4.

5.2.2.1 Performance Gains Across Frameworks

Our experiment results show that System 2 consistently outperforms the standard ToG baseline across nearly all tested LLM and benchmarks. Notably, when paired with smaller models, such as Gemma 4 E2B, System 2 achieves substantial EM improvements of 27.5% on CWQ, 35.0% on WebQ, 42.5% on WebQSP and 57.0% on GrailQA. Even for larger models like Qwen 3 32B, System 2 still yields significant gains, improving performance by up to 24.0% absolute on GrailQA.

These empirical improvements suggest that restructuring step-wise path pruning into path-level reasoning in conjunction with tournament candidate selection leads to more effective filtering of reasoning paths, especially for small LLMs.

5.2.2.2 Impact of LLM Model Scale and Capability

We observed that the baseline ToG results show a certain degree of performance inversion, or "negative scaling", among Qwen models as the parameter size increases. For instance, the larger Qwen 3 32B underperforms the smaller Qwen 3 14B across multiple datasets, including WebQ (46.0% vs 54.0%), GrailQA (50.0% vs 56.0%), and WebQSP (59.5% vs 62.5%). We attribute this to longer and more flexible responses, which more frequently violate formatting requirements, generated by larger models.

There is an apparent performance inversion between the Qwen 3 and Gemma 4 LLM families, specific to System 2 and not ToG. We attribute this to the different sensitivities of LLMs to certain prompt styles. During pruning, ToG asks scoring questions with a list of candidates using few-shot prompting, while System 2 asks zero-shot multiple-choice questions with exactly two options and requires the LLM to show its reasoning process. The choice of an instruction-tuned model (Gemma 4 E4B IT and E2B IT) may also have made a difference. Finally, it is worth noting that the Gemma 4 LLM family is approximately one year more recent than Qwen

Method	CWQ	WebQ
System 2	46.0	56.5
System 2 w/o Relation Path Pruning	33.0	45.5
System 2 w/o Tournament	32.5	53.0

Table 5.5: Ablation study of System 2 under Gemma 4 E4B IT.

3, entailing non-trivial systematic changes.

5.2.2.3 Performance Consistency Across Datasets

System 2 exhibits stable performance across different benchmarks, with variations aligned with the baseline. This pattern indicates that System 2 provides consistent behavior across datasets rather than being optimized to a specific benchmark, and the variations should be explained by differences in question complexity.

5.2.2.4 Ablation Study

Table 5.5 presents the ablation study results of the System 2 framework using Gemma 4 E4B IT to evaluate our two core components. Both ablations reduce performance from full System 2 on CWQ and WebQ, demonstrating the necessity of these modifications.

Ablation of relation path pruning Only the final relation in the relation path is used for pruning.

Ablation of tournament The candidates are point-wise scored rather than ranked by pair-wise comparison. We batch 5 items per LLM prompt. Scoring is implemented for both entity-agnostic path pruning and final path ranking with entity information.

6

Conclusion

6.1 Conclusion

In this study, we investigated into the baseline ToG framework and explored several limitations that affect the accuracy of the answers. The limitations include locally scoped pruning, pointwise scoring, as well as random sampling to narrow down the candidate pool.

For locally scoped pruning, we proposed an entity-agnostic relation path pruning to provide global path semantics. According to the observations, our method ignored the entities and kept only relation sequences, which reveals substantial overlap among candidates, allowing us to merge them into a significantly more compact candidate pool.

Instead of asking the LLM to make pointwise scoring, we changed it to a ranking task, which LLMs are better at. To maintain the accuracy and stability, we proposed two systems that mainly differ in tournament selection algorithm. System 1 with chunked style divides the candidates into several chunks and screens out half of them. The iterative process continues until only one chunk remains, when the problem becomes a top-k selection. System 2 uses a top-k merge sort whose pairwise comparison is powered by the LLM.

As for random sampling, we altered it into a pre-filtering process that utilize lightweight tools to narrow down the number of candidates before sending them to the LLM. In System 1, we replaced random sampling with a hybrid pre-filtering strategy that leverages both lexical and semantic signals. In System 2, we used a single embedding model to produce similarity scores.

Because both systems are composed of several core components, we also performed ablation studies to ensure each component contributes to the improvements. Ultimately, by integrating these methods, our approach achieves three key things: it delivers higher accuracy, scales effectively with larger language models, and remains consistently reliable across datasets.

6.2 Future Work

1. Path-only exploration has some limitations.
   (a) It cannot handle questions that require instantiation of any entity.

   Consider the question "What is the nationality of the tallest person in the world?" on a knowledge graph without the superlative relation "tallest person in". To answer this question, it is necessary to instantiate the entity referred to by "the tallest person in the world", which is impossible.

   In order to answer this question, it is necessary to query the height of every human, choose the largest number, and make its path entity-aware again to continue with the instantiated entity.

   (b) It cannot handle a conjunction of requirements about the answer.

   In the graph, each predicate is an edge on the entity. A conjunction of many predicates simply means the entity must be connected to all of these predicate edges.

   In order to support conjunction, the question must be separated into its clauses. Then, the solution set is the intersection of those of its clauses.
2. For us to rigorously assess the performance of knowledge base QA systems, it is best to use a dedicated adversarial[1] fictional knowledge base.
3. It is best to terminate the search by answer convergence, rather than a single positive answer.
4. For explainability, the user deserves the formal SPARQL query in addition to the natural language response. It is technically trivial to implement, as the program already maintains an equivalent data structure internally. However, when explainability features are added, the user experience may require thoughtful design, including natural language explanations of the formal query.

[1] containing information inconsistent with the LLM

Bibliography

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08, Vancouver, Canada: Association for Computing Machinery, 2008, pp. 1247–1250, ISBN: 9781605581026. DOI: 10.1145/1376616.1376746. [Online]. Available: https://doi.org/10.1145/1376616.1376746.

[2] H. Naveed et al., *A comprehensive overview of large language models*, 2024. arXiv: 2307.06435 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2307.06435.

[3] Y. Guo et al., *Bias in large language models: Origin, evaluation, and mitigation*, 2024. arXiv: 2411.10915 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2411.10915.

[4] Z. Ji et al., "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, Mar. 2023, ISSN: 1557-7341. DOI: 10.1145/3571730. [Online]. Available: http://dx.doi.org/10.1145/3571730.

[5] J. Huang and K. C.-C. Chang, "Towards reasoning in large language models: A survey," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 1049–1065. DOI: 10.18653/v1/2023.findings-acl.67. [Online]. Available: https://aclanthology.org/2023.findings-acl.67/.

[6] W. Zheng, H. Cheng, J. X. Yu, L. Zou, and K. Zhao, "Interactive natural language question answering over knowledge graphs," *Information Sciences*, vol. 481, pp. 141–159, 2019, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2018.12.032. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025518309848.

[7] R. Howard, *RDF vs. Property Graphs: Choosing the Right Approach for Implementing a Knowledge Graph - Graph Database & Analytics — neo4j.com*, https://neo4j.com/blog/knowledge-graph/rdf-vs-property-graphs-knowledge-graphs/, [Accessed 10-02-2026].

[8] Y. Zhang, K. Chen, X. Bai, Z. Kang, Q. Guo, and M. Zhang, "Question-guided knowledge graph re-scoring and injection for knowledge graph question answering," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 8972–

8985. DOI: 10.18653/v1/2024.findings-emnlp.524. [Online]. Available: https://aclanthology.org/2024.findings-emnlp.524/.

[9] M. Shrestha and E. Kim, *Efficient multi-hop question answering over knowledge graphs via llm planning and embedding-guided search*, 2025. arXiv: 2511.19648 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2511.19648.

[10] L. Luo, Y.-F. Li, G. Haffari, and S. Pan, *Reasoning on graphs: Faithful and interpretable large language model reasoning*, 2024. arXiv: 2310.01061 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2310.01061.

[11] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3580–3599, Jul. 2024, ISSN: 2326-3865. DOI: 10.1109/tkde.2024.3352100. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2024.3352100.

[12] *Sparql 1.1 query language*, Mar. 2013. [Online]. Available: https://www.w3.org/TR/sparql11-query/.

[13] H. Luo et al., "ChatKBQA: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models," in *Findings of the Association for Computational Linguistics: ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 2039–2056. DOI: 10.18653/v1/2024.findings-acl.122. [Online]. Available: https://aclanthology.org/2024.findings-acl.122/.

[14] P. Agarwal, N. Kumar, and S. Bedathur, "Aligning complex knowledge graph question answering as knowledge-aware constrained code generation," in *Proceedings of the 31st International Conference on Computational Linguistics*, O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B. D. Eugenio, and S. Schockaert, Eds., Abu Dhabi, UAE: Association for Computational Linguistics, Jan. 2025, pp. 3952–3978. [Online]. Available: https://aclanthology.org/2025.coling-main.267/.

[15] L. Pan, A. Albalak, X. Wang, and W. Wang, "Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 3806–3824. DOI: 10.18653/v1/2023.findings-emnlp.248. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.248/.

[16] W. Ding et al., "Knowledge crosswords: Geometric knowledge reasoning with large language models," in *Findings of the Association for Computational Linguistics: ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 2609–2636. DOI: 10.18653/v1/2024.findings-acl.154. [Online]. Available: https://aclanthology.org/2024.findings-acl.154/.

[17] T. Shen, J. Wang, X. Zhang, and E. Cambria, "Reasoning with trees: Faithful question answering over knowledge graph," in *Proceedings of the 31st International Conference on Computational Linguistics*, O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B. D. Eugenio, and S. Schockaert, Eds., Abu Dhabi,

UAE: Association for Computational Linguistics, Jan. 2025, pp. 3138–3157. [Online]. Available: https://aclanthology.org/2025.coling-main.211/.

[18] Z. Zhang and W. Zhao, "A collaborative reasoning framework powered by reinforcement learning and large language models for complex questions answering over knowledge graph," in *Proceedings of the 31st International Conference on Computational Linguistics*, O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B. D. Eugenio, and S. Schockaert, Eds., Abu Dhabi, UAE: Association for Computational Linguistics, Jan. 2025, pp. 10 672–10 684. [Online]. Available: https://aclanthology.org/2025.coling-main.712/.

[19] S. Wang and Y. Yu, *Kg-reasoner: A reinforced model for end-to-end multi-hop knowledge graph reasoning*, 2026. arXiv: 2604.12487 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2604.12487.

[20] J. Sun et al., "Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=nnVO1PvbTv.

[21] D. Edge et al., *From local to global: A graph rag approach to query-focused summarization*, 2025. arXiv: 2404.16130 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2404.16130.

[22] B. J. Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su, *Hipporag: Neurobiologically inspired long-term memory for large language models*, 2025. arXiv: 2405.14831 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2405.14831.

[23] X. He et al., *G-retriever: Retrieval-augmented generation for textual graph understanding and question answering*, 2024. arXiv: 2402.07630 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2402.07630.

[24] M.-C. Lee et al., *Hybgrag: Hybrid retrieval-augmented generation on textual and relational knowledge bases*, 2025. arXiv: 2412.16311 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2412.16311.

[25] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang, "LightRAG: Simple and fast retrieval-augmented generation," in *Findings of the Association for Computational Linguistics: EMNLP 2025*, C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, Eds., Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 10 746–10 761, ISBN: 979-8-89176-335-7. DOI: 10.18653/v1/2025.findings-emnlp.568. [Online]. Available: https://aclanthology.org/2025.findings-emnlp.568/.

[26] J. Jiang et al., "KG-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, Eds., Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 9505–9523, ISBN: 979-8-89176-251-0. DOI: 10.18653/v1/2025.acl-long.468. [Online]. Available: https://aclanthology.org/2025.acl-long.468/.

[27] J. Ma et al., *Debate on graph: A flexible and reliable reasoning framework for large language models*, 2024. arXiv: 2409.03155 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2409.03155.

[28] G. Xiong, J. Bao, and W. Zhao, "Interactive-kbqa: Multi-turn interactions for knowledge base question answering with large language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2024, pp. 10 561–10 582. DOI: 10.18653/v1/2024.acl-long.569. [Online]. Available: http://dx.doi.org/10.18653/v1/2024.acl-long.569.

[29] Y. Zhu et al., "KnowAgent: Knowledge-augmented planning for LLM-based agents," in *Findings of the Association for Computational Linguistics: NAACL 2025*, L. Chiruzzo, A. Ritter, and L. Wang, Eds., Albuquerque, New Mexico: Association for Computational Linguistics, Apr. 2025, pp. 3709–3732, ISBN: 979-8-89176-195-7. DOI: 10.18653/v1/2025.findings-naacl.205. [Online]. Available: https://aclanthology.org/2025.findings-naacl.205/.

[30] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, *A gentle introduction to graph neural networks*, Sep. 2021. [Online]. Available: https://distill.pub/2021/gnn-intro/.

[31] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "QA-GNN: reasoning with language models and knowledge graphs for question answering," *CoRR*, vol. abs/2104.06378, 2021. arXiv: 2104.06378. [Online]. Available: https://arxiv.org/abs/2104.06378.

[32] C. Mavromatis and G. Karypis, "GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs," in *Findings of the Association for Computational Linguistics: ACL 2025*, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, Eds., Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 16 682–16 699, ISBN: 979-8-89176-256-5. DOI: 10.18653/v1/2025.findings-acl.856. [Online]. Available: https://aclanthology.org/2025.findings-acl.856/.

[33] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1512.03385.

[34] A. Vaswani et al., *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1706.03762.

[35] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, no. null, pp. 1137–1155, Mar. 2003, ISSN: 1532-4435.

[36] B. Ghojogh and A. Ghodsi, *Recurrent neural networks and long short-term memory networks: Tutorial and survey*, 2023. arXiv: 2304.11461 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2304.11461.

[37] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*, 3rd. 2026, Online manuscript released January 6, 2026. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/.

[38] L. Ouyang et al., *Training language models to follow instructions with human feedback*, 2022. arXiv: 2203.02155 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2203.02155.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423/.

[40] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410. [Online]. Available: https://aclanthology.org/D19-1410/.

[41] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20, Vancouver, BC, Canada: Curran Associates Inc., 2020, ISBN: 9781713829546.

[42] S. Schulhoff et al., *The prompt report: A systematic survey of prompt engineering techniques*, 2025. arXiv: 2406.06608 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2406.06608.

[43] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, *Large language models are zero-shot reasoners*, 2023. arXiv: 2205.11916 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2205.11916.

[44] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 3816–3830. DOI: 10.18653/v1/2021.acl-long.295. [Online]. Available: https://aclanthology.org/2021.acl-long.295/.

[45] T. Brown et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[46] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen, "What makes good in-context examples for GPT-3?" In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, E. Agirre, M. Apidianaki, and I. Vuli, Eds., Dublin, Ireland and Online: Association for Computational Linguistics, May 2022, pp. 100–114. DOI: 10.18653/v1/2022.deelio-1.10. [Online]. Available: https://aclanthology.org/2022.deelio-1.10/.

[47] J. Wei et al., *Chain-of-thought prompting elicits reasoning in large language models*, 2023. arXiv: 2201.11903 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2201.11903.

[48] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, no. 4, pp. 333–389, Apr. 2009, ISSN: 1554-0669. DOI: 10.1561/1500000019. [Online]. Available: https://doi.org/10.1561/1500000019.

[49] G. V. Cormack, C. L. A. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '09, Boston, MA, USA: Association for Computing Machinery, 2009, pp. 758–759, ISBN: 9781605584836. DOI: 10.1145/1571941.1572114. [Online]. Available: https://doi.org/10.1145/1571941.1572114.

[50] B. T. Lowerre, "The harpy speech recognition system," 1976. [Online]. Available: https://api.semanticscholar.org/CorpusID:61409851.

[51] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14, Montreal, Canada: MIT Press, 2014, pp. 3104–3112.

[52] C. Meister, R. Cotterell, and T. Vieira, "If beam search is the answer, what was the question?" In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 2173–2185. DOI: 10.18653/v1/2020.emnlp-main.170. [Online]. Available: https://aclanthology.org/2020.emnlp-main.170/.

[53] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Syst.*, vol. 9, 1995. [Online]. Available: https://api.semanticscholar.org/CorpusID:6491320.

[54] I. Zubiaga, A. Soroa, and R. Agerri, "A LLM-based ranking method for the evaluation of automatic counter-narrative generation," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 9572–9585. DOI: 10.18653/v1/2024.findings-emnlp.559. [Online]. Available: https://aclanthology.org/2024.findings-emnlp.559/.

[55] I. B. Sandan, T. A. Dinh, and J. Niehues, "Knockout LLM assessment: Using large language models for evaluations through iterative pairwise comparisons," in *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEMš)*, O. Arviv et al., Eds., Vienna, Austria and virtual meeting: Association for Computational Linguistics, Jul. 2025, pp. 121–128, ISBN: 979-8-89176-261-9. [Online]. Available: https://aclanthology.org/2025.gem-1.10/.

[56] D. E. Knuth, *The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching*. USA: Addison Wesley Longman Publishing Co., Inc., 1998, ISBN: 0201896850.

[57] P. Zheng et al., *From isolated scoring to collaborative ranking: A comparison-native framework for llm-based paper evaluation*, 2026. arXiv: 2603.17588 [cs.IR]. [Online]. Available: https://arxiv.org/abs/2603.17588.

[58] W. Sun et al., "Is ChatGPT good at search? investigating large language models as re-ranking agents," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 14 918–14 937. DOI: 10.18653/v1/2023.emnlp-main.923. [Online]. Available: https://aclanthology.org/2023.emnlp-main.923/.

[59] D. Lee, Y. Jo, H. Park, and M. Lee, "Shifting from ranking to set selection for retrieval augmented generation," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, Eds., Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 17 606–17 619, ISBN: 979-8-89176-251-0. DOI: 10.18653/v1/2025.acl-long.861. [Online]. Available: https://aclanthology.org/2025.acl-long.861/.

[60] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08, Vancouver, Canada: Association for Computing Machinery, 2008, pp. 1247–1250, ISBN: 9781605581026. DOI: 10.1145/1376616.1376746. [Online]. Available: https://doi.org/10.1145/1376616.1376746.

[61] A. Talmor and J. Berant, "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)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 641–651. DOI: 10.18653/v1/N18-1059. [Online]. Available: https://aclanthology.org/N18-1059/.

[62] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "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)*, K. Erk and N. A. Smith, Eds., Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 201–206. DOI: 10.18653/v1/P16-2033. [Online]. Available: https://aclanthology.org/P16-2033/.

[63] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on Freebase from question-answer pairs," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, and S. Bethard, Eds., Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1533–1544. [Online]. Available: https://aclanthology.org/D13-1160/.

[64] Y. Gu et al., "Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases," in *Proceedings of the Web Conference 2021*, ser. WWW '21, Ljubljana, Slovenia: Association for Computing Machinery,

2021, pp. 3477–3488, ISBN: 9781450383127. DOI: 10.1145/3442381.3449992. [Online]. Available: https://doi.org/10.1145/3442381.3449992.

[65] A. Yang et al., *Qwen3 technical report*, 2025. arXiv: 2505.09388 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2505.09388.

[66] A. Grattafiori et al., *The llama 3 herd of models*, 2024. arXiv: 2407.21783 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2407.21783.

[67] D. Guo et al., "Deepseek-r1 incentivizes reasoning in llms through reinforcement learning," *Nature*, vol. 645, no. 8081, pp. 633–638, 2025, ISSN: 1476-4687. DOI: 10.1038/s41586-025-09422-z. [Online]. Available: http://dx.doi.org/10.1038/s41586-025-09422-z.

[68] Google DeepMind, *Gemma 4 model card*, https://ai.google.dev/gemma/docs/core/model_card_4, Accessed: 2026-05-25, 2026.

[69] Y. Zhang et al., *Qwen3 embedding: Advancing text embedding and reranking through foundation models*, 2025. arXiv: 2506.05176 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2506.05176.