A Framework for Explaining LLM Reasoning with Knowledge Graphs

by

Moein Shirdel

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2025

© Moein Shirdel 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Large Language Models (LLMs) have demonstrated remarkable question-answering (QA) capabilities, yet their decision processes and outputs often remain opaque and prone to factual inconsistencies. While existing methods evaluate or ground LLM outputs after generation, they typically lack mechanisms for aligning LLM reasoning with external knowledge sources. This thesis introduces AprèsCoT, a lightweight model-agnostic framework that validates LLM reasoning by grounding it in an external knowledge graph (KG). AprèsCoT operates through three main components: Subgraph Retrieval, which extracts a KG subgraph relevant to a given query; Triple Extraction and Parsing, which converts the LLM's output into factual triples; and Matching, which aligns these triples with entities and relations in the extracted KG subgraph. The integration of these modules enables alignment between LLM reasoning and structured knowledge, producing traceable and structured explanations alongside model outputs. We evaluate alternative retrieval and matching strategies, analyze their trade-offs, and demonstrate how AprèsCoT helps users surface reasoning gaps, hallucinations, and missing facts. Experiments across multiple domains, including large-scale KGs, highlight AprèsCoT's effectiveness in advancing trustworthy and explainable AI.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Lukasz Golab, for his invaluable support, patient mentorship, and guidance throughout my research. His feedback and encouragement were pivotal to the development and completion of this thesis.

I am also deeply grateful to Professor Parke Godfrey, Professor Jaroslaw Szlichta, and Dr. Divesh Srivastava for their advice and continued guidance during various stages of this project. Their expertise and support helped shape both the direction and depth of this work.

Special thanks to Joel Rorseth for generously sharing his experience with similar research and for his help in developing the web interface for the system.

I would also like to thank Professor Charles Clarke and Professor Paulo Alencar for dedicating their time and attention to reading my thesis.

Finally, I extend my heartfelt thanks to my family and friends. Their encouragement and unwavering support helped me stay grounded and motivated throughout this journey.

Dedication

To my parents, Mansour and Raha, and to my dearest younger brother, Farbod, for their steady support, positivity, and belief in me, no matter the miles between us.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures ix

List of Tables x

1 Introduction 1

1.1 Motivation 1

1.2 Contributions 2

2 Background and Related Work 6

2.1 Background 6

2.1.1 Language Models and the Rise of LLMs 6

2.1.2 Retrieval-Augmented Generation (RAG) 7

2.1.3 Knowledge Bases and Knowledge Graphs 8

2.1.4 Constructing KGs from Text 8

2.1.5 Matching and Retrieval in KGs . . . 9
2.1.6 Aligning LLM Responses to KGs . . . 10
2.2 Related Work . . . 11
2.2.1 LLM Explanations . . . 11
2.2.2 LLM Fact Checking . . . 12
2.2.3 KG-Augmented QA with LLMs . . . 12
2.2.4 KG Information Retrieval . . . 13

3 Methodology **14**
3.1 System Overview . . . 14
3.2 Subgraph Retriever . . . 15
3.3 Prompting . . . 22
3.4 Triple Extraction . . . 23
3.5 Output Matching . . . 23

4 Experiments **27**
4.1 Experimental KGs and LLMs . . . 27
4.2 Subgraph Retrieval Evaluation . . . 28
4.2.1 Setup and Retrieval Methods . . . 28
4.2.2 Evaluation Metrics . . . 30
4.2.3 Results . . . 30
4.2.4 Analysis . . . 31
4.3 Embedding Model Selection . . . 33
4.4 Use Cases . . . 35
4.4.1 Consistent Answers and Reasoning . . . 35
4.4.2 KG Data Quality . . . 35
4.4.3 Inconsistent Answers and CoT . . . 36

5 Conclusions and Future Work **41**

5.1 Final Remarks 41

5.2 Future Direction 41

References **43**

List of Figures

1.1 An explanation produced by AprèsCoT. 5

3.1 System architecture and workflow of AprèsCoT. 15
3.2 Prompt used to obtain a hypothetical answer for a question. 17
3.3 Path-aware context in describing KG edges. 19
3.4 Illustration of Algorithm 1 over a KG neighborhood. 21
3.5 Instructions given to the LLM in the QA process. 25
3.6 An example of a prompt created for a question asked in RAG mode. . . . 26

4.1 MiniLM vs. DistilBERT: Darker green indicates higher similarity, reflecting better separation of relevant and irrelevant facts by MiniLM. 34
4.2 An explanation showing consistency in LLM answers and reasoning. 38
4.3 An explanation showing missing information in the KG. 39
4.4 An explanation showing inconsistent LLM Answers and Reasoning. 40

List of Tables

4.1 Dataset characteristics. 28
4.2 Sample of questions used in the subgraph retrieval experiments. 29
4.3 Aggregated experiment results for all questions. 31
4.4 Aggregated experiment results for 3-hop questions. 31

Chapter 1

Introduction

1.1 Motivation

Large language models (LLMs) have brought major advances to natural language processing, demonstrating strong generative performance on tasks such as question answering (QA). However, despite their capabilities, these models can suffer from *hallucination*, generating confident yet inaccurate outputs, which poses challenges for their adoption in sensitive domains such as healthcare and education.

To address hallucination, systems such as Google Gemini,[1] Perplexity,[2] and Microsoft Copilot[3] adopt retrieval-augmented generation (RAG) [23], where relevant documents (from the web or a private knowledge base) are retrieved and added to the LLM's context to ground its response. This allows the model to reason over external information during inference and combine its pretrained knowledge with retrieved content via in-context learning [5].

However, RAG does not guarantee that the retrieved information will actually influence the model's response. Tracing the provenance of a generated answer often requires additional effort, either through a secondary model that attempts post-hoc citation recovery [24], which itself may hallucinate, or by repeatedly querying the LLM with subsets of the retrieved content to infer which sources affected the output [8, 32]. Both approaches can be unreliable and computationally expensive.

[1] https://gemini.google.com/
[2] https://www.perplexity.ai/
[3] https://copilot.microsoft.com/

Beyond provenance tracing, recent work in fact checking aims to assess whether an LLM's output aligns with external evidence. Here, existing approaches focus on verifying the final output rather than the reasoning process leading to it. However, ensuring valid reasoning is equally important in high-stakes domains to build trust in LLMs. For example, in our use cases (Section 4.4.3), we analyze inference differences across LLMs and highlight instances of implausible reasoning steps and unsupported inference. Such cases can undermine trust in a model's decision-making, even when the final answer seems plausible.

To enable reasoning-level verification, we build on two observations:

- that modern LLMs can generate their reasoning steps, e.g., via *chain of thought* (CoT) prompting [45], by appending the sentence "Let's think step by step" to the query; and
- that these steps, if plausible, can be mapped to facts stored in a *knowledge graph* (KG) to produce a data provenance trail leading to a structured explanation.

These observations motivate a framework that grounds an LLM's reasoning process in structured knowledge, enabling both interpretability and verifiable reasoning.

1.2 Contributions

Leveraging the above observations and addressing the gaps identified in existing work, we design APrèsCoT as a post-hoc tool applicable to any LLM or QA system that exposes its reasoning process, provided that a corresponding KG exists or can be constructed from a suitable document corpus. It allows users to pose questions across various domains by selecting an LLM of their choice along with a corresponding KG for verification. APrèsCoT then aligns both the model's answers and its reasoning steps against the KG. Optionally, users may enable RAG mode, where APrèsCoT supplements the LLM with contextual information drawn from the KG to facilitate question answering. Examples showing both RAG and non-RAG modes are included within the Experiments section (Section 4.4).

We make the following contributions.

1. **LLM-KG Reasoning Alignment:** We present APrèsCoT, a framework that explains LLM reasoning by aligning their generated answers and intermediate steps

with entities and relations in a KG. The idea is inspired by nearest neighbor LLMs [19]; i.e., looking up documents most similar to the prompt and using the content of these documents to either modify next-token probabilities or as a form of citation. However, our use of LLM reasoning to trace structured inference paths over a KG is novel.

2. **Modular Framework:** AprèsCoT follows a modular architecture consisting of three main components.

   (a) The *Subgraph Retriever* extracts a KG subgraph relevant to the user's question.

   (b) The *Triple Extractor and parser* converts the LLM's response into final answers and factual triples.

   (c) The *Matcher* aligns the processed LLM responses with entities and relations in the retrieved subgraph.

   Proposing new strategies to design these modules and their integration is the key contribution of this work, enabling reasoning alignment and explanation over large-scale KGs in an interactive setting.

3. **Validation:** The interpretability and quality of AprèsCoT's explanations depend on how each system component is implemented. We validate these design choices in two ways: first, through experiments that evaluate alternative implementations and justify our selected configurations; and second, through use cases that demonstrate AprèsCoT's effectiveness as an explanation tool.

   (a) *Module-level evaluation:* We evaluate different implementations of the main components, particularly the **Subgraph Retrieval** and **Matcher** modules, through studies that compare alternative KG retrieval strategies (Section 4.2) and embedding models (Section 4.3), assessing their respective trade-offs. The results show that our enhancements improve the ability of subgraph retrieval methods to extract question-relevant subgraphs.

   (b) *Use-cases:* We illustrate how AprèsCoT assists users in interpreting model reasoning through three representative scenarios: **consistent inference**, where the response is fully supported by the KG; **incomplete inference**, where the model's response is only partially grounded; and **KG incompleteness**, where AprèsCoT helps with identifying missing or insufficient knowledge within the knowledge graph.

Figure 1.1 shows AprèsCoT in action.[4,5] The user selects from a set of preloaded LLMs (ChatGPT 3.5 with RAG) and the KG (MetaQA Movies), and asks for the release years of movies starring actor Jean Rochefort. The LLM's answer (1972 and 1990), along with the parsed chunks of information used in its reasoning, are displayed both in tabular form and on the corresponding KG. Dark blue KG nodes represent entities in the question, blue nodes correspond to answers, and green edges highlight the inference paths that AprèsCoT matched to the CoT. This example illustrates full alignment between the LLM output and the KG: the answers correspond to existing nodes, and the reasoning steps map onto connected inference paths and demonstrate traceable reasoning.

This thesis is organized as follows. In Chapter 2, we provide the necessary background and review related work, highlighting the gaps that AprèsCoT aims to address. Chapter 3 outlines how AprèsCoT is designed and describes the implementation of each component. Chapter 4 presents our experimental setup, results, and analysis for evaluating alternative configurations of these components. This is followed by three use cases to demonstrate how users can explore inference gaps to discover instances of LLM hallucination or KG incompleteness. Lastly, Chapter 5 concludes with a summary of findings and directions for future work.

[4] AprèsCoT is available at http://lg-research-2.uwaterloo.ca:8055/aprescot
[5] The source code is also available at https://github.com/moeinsh78/ApresCoT

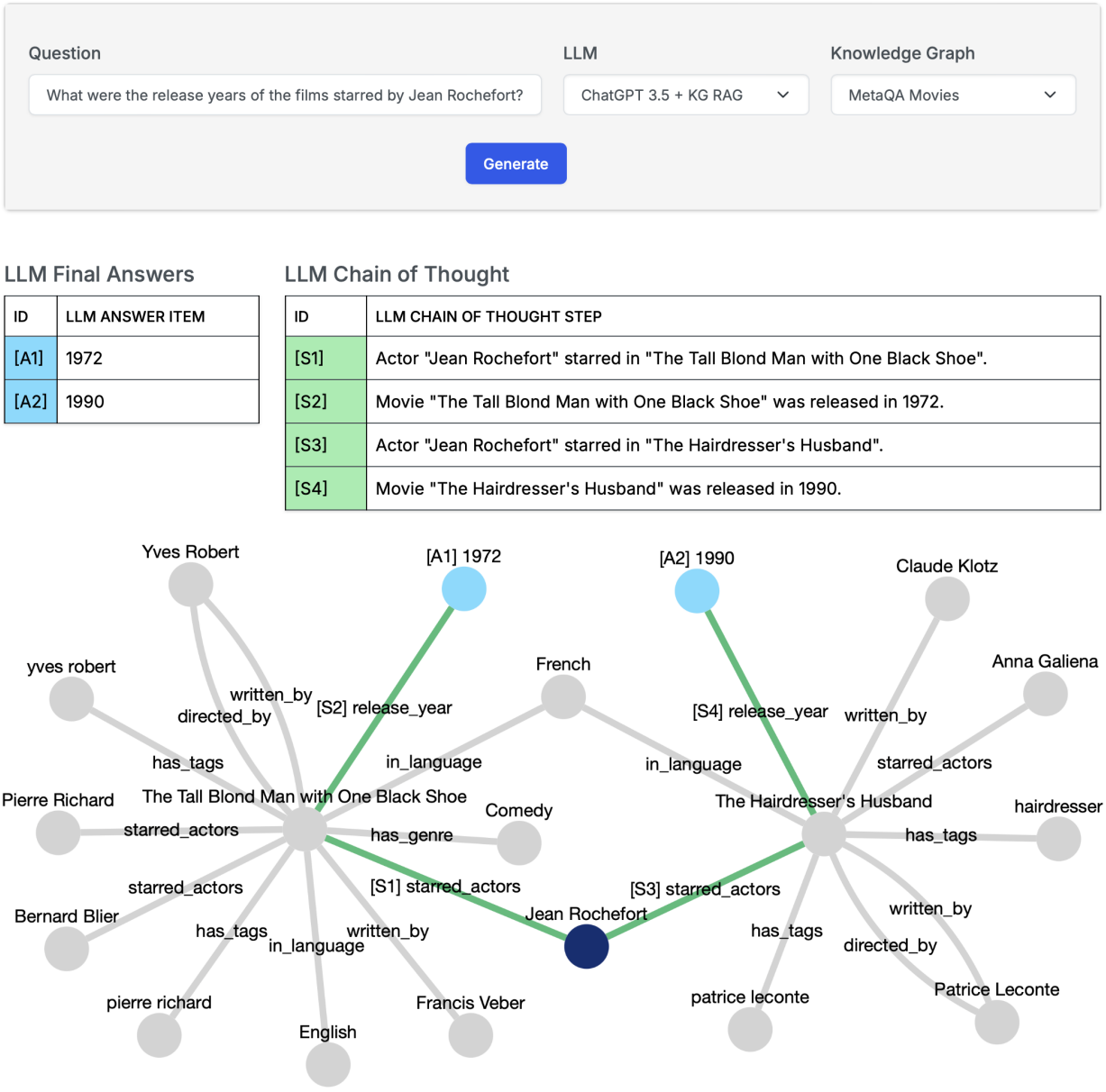


Figure 1.1: An explanation produced by ApròsCoT.

Chapter 2

Background and Related Work

2.1 Background

This section provides the foundational background necessary to contextualize the design and objectives of AprèsCoT. It begins by tracing the evolution of language models and the emergence of large-scale architectures capable of reasoning through context (Section 2.1.1). It then introduces retrieval-augmented generation, a paradigm that enhances factual grounding in LLMs by incorporating external information (Section 2.1.2). It then continues to discuss structured knowledge sources as popular sources of information in RAG paradigm, describing how knowledge bases and knowledge graphs represent and organize factual information (Section 2.1.3), and how such graphs can be constructed from unstructured text (Section 2.1.4). Lastly, we examine methods for matching and retrieving relevant information from these graphs (Section 2.1.5), and motivate the alignment of LLM reasoning with structured knowledge to promote interpretability and factual reliability (Section 2.1.6).

2.1.1 Language Models and the Rise of LLMs

The evolution of language models has progressed rapidly from context-limited systems to general-purpose reasoning models. The Transformer architecture introduced by [41] replaced sequential processing with a self-attention mechanism, allowing each word to directly attend to others in a sequence of text and capture long-range dependencies across text efficiently. Building on this foundation, BERT [9] showed that a bidirectional Transformer encoder, pre-trained on large text corpora and later fine-tuned for specific tasks,

could achieve strong language understanding. Soon after, GPT-3 [5] demonstrated that scaling Transformer-based decoder-only models to hundreds of billions of parameters enabled in-context learning: the ability to perform new tasks from examples given in natural language prompts without additional training. Extending this trend, PaLM [7] combined massive model size with distributed training infrastructure and introduced chain-of-thought prompting, where the model generates intermediate reasoning steps to improve multi-step problem solving. Together, these advances mark the shift from task-specific training to general-purpose large language models capable of flexible reasoning through context, an ability that also helps mitigate some limitations of earlier systems by allowing them to incorporate guidance and clarification directly within their prompts.

2.1.2 Retrieval-Augmented Generation (RAG)

Despite their impressive reasoning ability, large language models are limited by the knowledge embedded in their weights during training, which cannot be updated or verified at inference time. This often leads to factual errors or outdated responses. Retrieval-Augmented Generation addresses this limitation by allowing models to retrieve external information and integrate it into their responses.

A RAG system normally consists of a retriever that searches an external corpus for relevant documents and a generator that conditions its output on both the query and the retrieved text, which typically is an LLM with the ability of in-context learning. This architecture enables grounding in real evidence and improves factual accuracy and interpretability. The idea was introduced in [23], showing that coupling retrieval with generation enhances performance on knowledge-intensive tasks such as open-domain question answering.

Later work refined this paradigm: [14] proposed REALM, which learns to retrieve documents during pre-training, and [17] developed ATLAS, jointly optimizing retrieval and generation at scale. While these systems mostly rely on information unstructured text, retrieval can also draw from structured sources, such as knowledge bases (KBs) and knowledge graphs, where information is stored as entities and relations. Leveraging these structured forms of knowledge provides greater semantic precision and transparency, motivating the next section on knowledge-based retrieval.

2.1.3 Knowledge Bases and Knowledge Graphs

To complement retrieval from unstructured text, many systems leverage structured knowledge sources, where information is explicitly represented as entities and relations. A knowledge base is a general term for any system that represents information about the world in a structured form, typically through entities, attributes, and relations. Traditional KBs organize information in a schema-driven manner. Each entity or fact is stored according to a predefined schema (e.g., a relational database), where attributes and relations are fixed in advance. This structure enables precise queries using formal languages such as SQL but limits flexibility when representing complex or heterogeneous relationships.

A knowledge graph can be viewed as a graph-structured form of a knowledge base, in which facts are represented as triples (subject, predicate, object). Here, entities correspond to nodes, and relations between them correspond to edges labeled by predicates. This representation captures heterogeneous and semantically rich relationships between entities, allowing for multi-hop reasoning and contextual inference. Beyond their representational advantages, KGs also support expressive query capabilities through languages such as SPARQL, allowing structured retrieval, inference, and integration with external data sources.

Some widely used knowledge graphs include DBpedia [22], which extracts structured facts from Wikipedia, ConceptNet [34], that encodes general human knowledge about everyday concepts and their relationships, and YAGO [35], which automatically combines Wikipedia's broad scope with a structured hierarchy to build a large-scale, highly accurate semantic knowledge base.

2.1.4 Constructing KGs from Text

While large-scale knowledge graphs are often built from curated or semi-structured sources, many domains rely on extracting knowledge directly from unstructured text. The process of transforming natural language into structured triples, and eventually KGs, is known as information extraction (IE). A typical IE pipeline consists of several stages:

1. **Named Entity Recognition (NER)** identifies mentions of entities such as people, organizations, or locations within text.
2. **Entity Linking (EL)** resolves these mentions to entries in a knowledge base, ensuring consistency across references.

3. **Relation Extraction (RE)** detects semantic relationships between entities (e.g., *directed_by*, *born_in*).

Together, these steps produce triples of the form *(subject, predicate, object)* that represent factual statements (e.g., {*Inception, directed_by, Christopher Nolan*}) which can then be added as nodes and edges to a knowledge graph.

Both early and recent works have contributed to advancing these components, either by implementing the full pipeline or improving specific stages. [47] applied convolutional neural networks (CNNs) to relation classification, advancing RE beyond manually crafted features toward automatically learned sentence representations. Stanford OpenIE [1] exemplifies open-domain information extraction, extracting relational tuples from unrestricted text without relying on a fixed ontology. More recently, EAGER [6] introduced a framework that integrates KG construction with explainable question answering. EAGER builds or enriches a knowledge graph from text, links question terms to graph entities, and retrieves relevant subgraphs. By explicitly exposing these paths, the system provides transparent and verifiable explanations. Once constructed, however, these knowledge graphs must be navigated and queried. The next section discusses methods for matching entities and retrieving relevant subgraphs, which are fundamental to enabling reasoning and interpretability in systems that leverage information in KGs.

2.1.5 Matching and Retrieval in KGs

In knowledge graphs, matching and retrieval refer to the process of identifying relevant portions of the graph that correspond to a query or context. These concepts are defined as follows:

- **Seed node:** the entity node that serves as the starting point for graph exploration, typically derived from the input query or a recognized mention in text.
- **Hop distance:** the number of edges traversed from a seed node to reach another node within the graph.
- **Subgraph:** a subset of the KG containing the seed node and its neighboring nodes and edges, representing the retrieved region of interest.

Retrieval approaches can generally be divided into two categories: **structural** and **semantic**. Structural methods, such as **breadth-first retrieval**, iteratively expand from

the seed node by visiting neighbors up to a specified hop limit. While this approach ensures complete local coverage, it may also retrieve large, irrelevant neighborhoods as the number of hops increases.

Alternatively, **similarity-based retrieval** employs semantic representations to identify entities and relations that are contextually related to the query without relying solely on structural distance. This approach often relies on knowledge graph embedding models that map entities and relations into continuous vector spaces using their textual features, enabling retrieval based on vector similarity measures such as cosine or Euclidean distance. Representative knowledge embedding models include TransE [4], DistMult [46], and ComplEx [40], which encode relational dependencies so that semantically related entities are positioned closely in the embedding space. Embedding-based retrieval for multi-hop reasoning has seen significant advances. For example, [18] proposes a model that tightly integrates retrieval and reasoning in a single architecture and parameter-space to better handle large search spaces of knowledge graph-based QA. More recently, [16] demonstrates that smaller language models can serve as efficient subgraph retrievers when equipped with relational sequence embeddings, improving performance in multi-hop KG reasoning.

Despite these advancements, effective retrieval remains challenging due to the expanding search space in multi-hop reasoning and semantic noise in embedding-based similarity. Designing reliable ranking and filtering mechanisms is thus essential to balance precision and coverage. Accurate retrieval forms the foundation for aligning structured knowledge with generative reasoning. The following section explores how these retrieved subgraphs can be integrated with large language models to enhance factual grounding and interpretability.

2.1.6 Aligning LLM Responses to KGs

Large language models can generate coherent reasoning chains and outputs, yet these remain ungrounded in verifiable knowledge. Aligning their reasoning steps with **structured information** from KGs enables explanations that are both interpretable and factually traceable. Such alignment enhances transparency, by revealing the factual basis of reasoning; verifiability, by linking claims to explicit relations; and trust, by exposing the evidence supporting each step. This motivation underlies the design of systems like AprèsCoT, which integrate KGs to ground LLM reasoning and outputs in structured, inspectable knowledge bases.

2.2 Related Work

To situate AprèsCoT within existing research, this section reviews four areas that address interpretability and factual grounding in LLMs. The first area, *LLM Explanations*, explores how model behavior can be interpreted through either mechanistic analysis of internal activations or attribution-based methods that link outputs to influential inputs, both relevant to improving LLM transparency (Section 2.2.1). The second, *LLM Fact Checking*, focuses on verifying the factual correctness of model outputs or intermediate steps, which aligns with AprèsCoT's goal of ensuring factual grounding and mitigating hallucination (Section 2.2.2). The third, *KG–Augmented LLMs for Question Answering*, shows how recent works integrate structured knowledge directly into reasoning to enhance QA systems (Section 2.2.3). Lastly, *KG Information Retrieval* reviews methods for retrieving relevant subgraphs from large knowledge bases (Section 2.2.4), a critical step in our framework, where the KG is incorporated after reasoning rather than during it.

2.2.1 LLM Explanations

Research on explaining LLMs includes a range of approaches aimed at improving transparency. Among these, one line of work, referred to as *mechanistic interpretability*, examines how transformer models internally compute and store information by analyzing their parameters and activations. Studies have shown that transformer feed-forward layers contribute to predictions by learning semantically meaningful concepts that align with output tokens [12], that groups of attention heads form interpretable circuits implementing algorithmic behaviors [44], and that specific multilayer perceptron modules can store factual associations [29]. Other work has examined how token predictions evolve across layers by analyzing the model's hidden representations and their alignment with output tokens [2]. While these methods provide insight into model internals, they depend on parameter-level access that is typically unavailable in modern, closed-source LLMs.

A complementary set of methods focuses on *attribution* and *factual grounding*, linking generated statements to the contextual evidence that most influenced them, often in retrieval-augmented settings. Several approaches do this through counterfactual or perturbation-based analysis: RAGE identifies minimal supporting subsets of the retrieved text by pruning pieces of the context and observing output changes [32], while RAG-Ex applies model-agnostic perturbations to measure how variations in retrieval affect generation [36]. Others rely on learned surrogate models to approximate causal effects: ContextCite fits a sparse linear model over ablation experiments to estimate each context unit's contribution [8] and AttriBoT introduces a set of techniques that accelerate this process and enable

more efficient attribution [27]. These methods enhance transparency by linking generated claims to unstructured textual evidence. AprèsCoT offers a complementary perspective by grounding both answers and reasoning steps in structured KGs, where attribution occurs over entities and relations rather than text spans.

2.2.2 LLM Fact Checking

Recent studies have explored ways to make large language models more trustworthy by checking the accuracy of their generated content. Early works such as FEVER [39] and SciFact [43] treat fact-checking as a retrieval and comparison task, testing whether a given claim can be supported or contradicted by evidence from external text. FActScore [30] extended this idea to long-form answers by breaking them into smaller factual statements and measuring how many are verifiable against a trusted source. *Chain of Verification* (CoVe) [10] further reduced factual errors by prompting a model to ask and answer its own verification questions about its output.

However, these methods mainly assess the correctness of the final answer rather than the reasoning process that led to it. *Deductive Verification of Chain-of-Thought Reasoning* [26] takes a step toward reasoning verification by checking whether each step in a model's explanation logically follows from its stated premises, but it still depends on another language model's judgment, which may itself be unreliable. In contrast, AprèsCoT verifies both the reasoning steps and the final answer by grounding them in a trusted external source (a knowledge graph). Each step is linked to concrete entities and relations, allowing the full reasoning path to be checked and visualized, thus reducing the risk of self-verification errors or hallucinated logic.

2.2.3 KG-Augmented QA with LLMs

Another line of research has focused on *early fusion* of KGs and LLMs to enhance reasoning and factual precision in question answering. The concept of early fusion refers to integrating structured graph reasoning directly into the LLM's inference process. For instance, GreaseLM [48], KG-CoT [50], and Think-on-Graph [38] incorporate KG structure into the model's reasoning loop, either through joint graph–text encoding, CoT prompts grounded in KGs, or iterative graph exploration by an LLM agent. G-Retriever [15] and KELDaR [25] follow a related idea but formalize retrieval as a subgraph-selection or question-decomposition process, allowing the LLM to reason and generate over a compact and

structured evidence set. These systems build more accurate QA pipelines by fusing KGs and LLMs before or during reasoning.

In contrast, AprèsCoT is not a QA model but an explanation framework that operates after reasoning. It performs post-hoc alignment between an LLM's reasoning process and a structured KG, producing grounded and interpretable traces of reasoning. When a QA system satisfies AprèsCoT's requirements such as CoT-compatibility, our framework can function as an explanation layer built on top of it. Our approach is therefore complementary to early-fusion models: rather than using KGs to guide reasoning, we use them to explain it. In this *late-fusion* design, the quality of retrieval and alignment becomes critical to explanation accuracy, introducing new challenges in identifying and extracting the most relevant portions of the KG.

2.2.4 KG Information Retrieval

Because AprèsCoT integrates the KG after reasoning, it must first retrieve a concise yet representative subgraph that supports the model's reasoning steps. Unlike some early-fusion models that interact with the KG throughout inference, AprèsCoT retrieves information only once, requiring a trade-off between completeness and efficiency. Existing methods fall into two categories: **structure-based** approaches, such as breadth-first search (BFS), which expand a node's neighbors to a fixed depth; and **semantic similarity-based** approaches, such as SRTK [33], which retrieve nodes based on label similarity. These approaches will be discussed in more detail in Section 3.2.

However, these methods were mostly developed for simpler, single-hop questions and often struggle with complex, multi-hop reasoning.[1] To overcome these limitations, we build on top of these existing methods to achieve better coverage and scalability, ensuring that the extracted subgraph captures the relevant knowledge needed for accurate grounding and explanation in AprèsCoT.

[1]Single-hop questions can be answered using a single KG fact; e.g., "Where was Einstein born?" Multi-hop questions on the other hand require combining multiple facts and step-by-step reasoning; e.g., "What language is spoken in the country where Einstein was born?"

Chapter 3

Methodology

This section describes the components of, and design decisions behind, AprèsCoT. We begin with an overview of the full workflow in Section 3.1. This is followed by individual sections describing the subgraph retriever (Section 3.2), the prompting strategy (Section 3.3), the parsing and triple extraction process (Section 3.4), and the semantic matcher (Section 3.5). In addition to the presentation of the framework, we highlight specific methodological contributions, including new subgraph retrieval strategies that extend existing similarity-based approaches.

3.1 System Overview

AprèsCoT is a lightweight, post-hoc framework to analyze how LLMs arrive at answers and to verify their reasoning against knowledge graphs. Its workflow follows a modular pipeline depicted in Figure 3.1. Given a natural language question and a selected knowledge graph, AprèsCoT (1) identifies relevant KG facts via the subgraph retriever, (2) queries the LLM with reasoning prompts and optional context information about the question, extracted from the KG, (3) parses the LLM output into final answer items and pieces of reasoning, extracted as textualized triples, (4) aligns these outputs with nodes and edges in the retrieved subgraph using the matcher module, and (5) produces a structured visual explanation by overlaying reasoning paths onto the KG. The architecture is model-agnostic, meaning that AprèsCoT requires only inference access to the LLM and a suitable KG. General-purpose or domain-specific KGs usually exist or can be constructed from a suitable document corpus using existing tools [6].

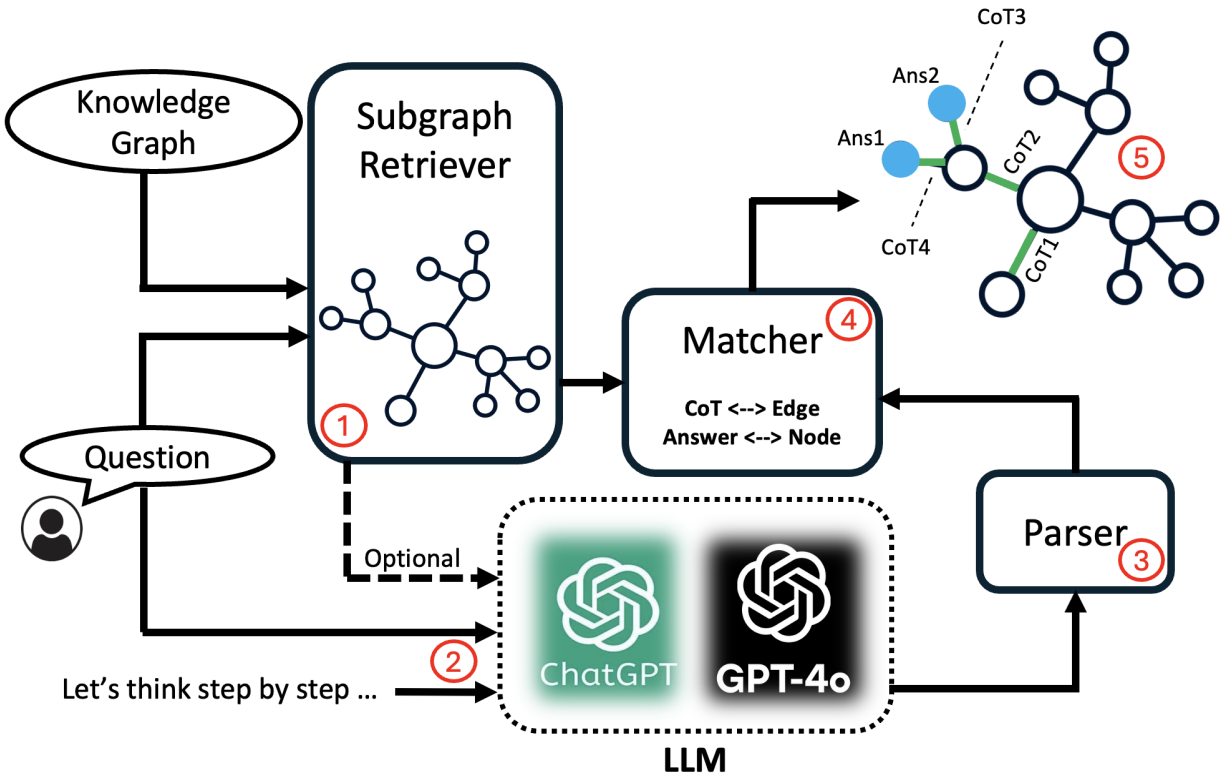


Figure 3.1: System architecture and workflow of AprèsCoT.

3.2 Subgraph Retriever

We model a knowledge graph as a labeled graph:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}),$$

where $\mathcal{V}$ is the set of entities (nodes), $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ is the set of edges labeled with relation types $\mathcal{R}$, and $\mathcal{L}$ is a labeling function assigning textual descriptions to nodes and edges.

A natural language question is denoted as q. The goal of the subgraph retriever is to extract a candidate subgraph $\mathcal{G}_q \subseteq \mathcal{G}$ that contains the entities and relations most relevant to answering q. We define:

$$\mathcal{G}_q = \textsc{Retrieve}(\mathcal{G}, q),$$

where RETRIEVE is an algorithm that selects a subset of nodes and edges from $\mathcal{G}$ based on the given question, q. The subgraph retrieval method should balance three factors: *recall* (including all facts needed for reasoning in $\mathcal{G}_q$), *precision* (avoiding irrelevant facts that may overwhelm the matcher), and *retrieval time* (to support interactive explanations).

We explore several retrieval algorithms, each implementing the RETRIEVE$(\cdot)$ function differently[1]. The first two serve as standard baselines, while the remaining methods are novel contributions introduced in this work. We benchmark all methods in Section 4.2.

I. Breadth-First Search (BFS) The simplest method, BFS, expands from seed entities detected in the question q (via named entity recognition). BFS covers local neighborhoods around seed entities, but may include many irrelevant edges, especially in large-scale knowledge graphs.

II. Semantic Similarity–based Retrieval (SSR) This method is inspired by the semantic similarity–based graph search used in systems such as *EAGER* [6] and *SRTK* [33]. Similar to the traversal component in these works, we begin with seed entities identified in the question q and expand the search over their local neighborhoods.

However, instead of exploring edges purely by structure or depth, we prioritize edges whose textual descriptions are semantically similar to the question. Each candidate edge—represented as a short natural language description obtained from its subject, relation, and object labels—is compared to the question embedding, and the top-scoring edges at each stage are kept for further expansion. The number of retained paths is configured via a "beam size" parameter. The process continues iteratively, expanding from the most relevant nodes until a stopping condition such as a maximum number of edges or hops is reached.

This method serves as a bridge between structural and semantic retrieval. Compared to a structural method such as BFS, which may collect many irrelevant facts in large graphs, similarity-based retrieval focuses on the subset of relations that are most likely to express the semantics of the question.

However, its performance depends heavily on how the question is phrased: if the wording of the question diverges from the embedding space of the possible answers, even relevant edges may receive low similarity scores. This limitation motivates the introduction of the

[1]Terminology such as *seed entity*, *hop distance*, *subgraph*, and *named entity recognition* follows the definitions introduced in Sections 2.1.4 and 2.1.5

```
Please write a passage to answer the question.

Question:
Which countries have land borders with Germany's neighbours?
Passage:
```

Figure 3.2: Prompt used to obtain a hypothetical answer for a question.

next retrieval variant, which uses a *hypothetical answer* rather than the question itself to guide similarity scoring.

III. Semantic Similarity with Hypothetical Answer (SSR+HyDE) Building on the previous similarity-based method, we adapt the idea of *Hypothetical Document Embeddings* (HyDE) [11] to improve retrieval robustness when the question wording differs from pieces of information that will assist with answering the question.

The core idea is to guide similarity scoring using a *hypothetical answer* rather than the question itself. To find this hypothetical answer, we perform a separate inference call to an LLM to generate a short, plausible answer $\hat{a}$ for each question q, and then compare edge descriptions to $\hat{a}$ instead of q. For this, we use a standard chat-completion interface and restrict the model output to a maximum number of tokens to keep the answer concise. The rest of the retrieval process (e.g., edge textualization, traversal, and top-k selection) remains the same as the previous method. In our implementation of AprèsCoT, we use the prompt shown in Figure 3.2 to ask gpt-4o-mini for a hypothetical answer to a given question:

This modification is expected to help the retriever better capture the semantics of the target subgraph, as the generated answer typically contains relational cues that are closer in embedding space to relevant KG facts. Notably, the factual correctness of the generated answer is not required; its purpose is solely to provide a semantically meaningful embedding. As such, even a lightweight or less accurate LLM can be used to reduce computational cost.

To the best of our knowledge, this is the first application of utilizing a hypothetical answer in the context of KGs and subgraph retrieval, as the original HyDE framework [11] was developed for retrieving text documents rather than graph edges.

IV. Path-Aware Similarity Retrieval (SSR+PASR) While the previous retrieval methods treat each candidate edge independently, this assumption can be limiting in multi-hop questions, where the relevance of an edge often depends on the context gathered by the traversal path so far. To address this, we extend the similarity-based retrieval by accumulating the textual descriptions of previously visited edges to form a running context. At each expansion step, instead of comparing the question (or the hypothetical answer) to the candidate edge only, we compare it to the concatenation of the current path description.

Figure 3.3 illustrates this idea. The intuition is as follows. Consider the question, "Which countries have land borders with Germany's neighbors?" If we only consider individual edges, a fact such as "France shares border with Spain" may seem irrelevant. However, when combined with the context accumulated from *Germany* → *France*, the path "Germany shares border with France; France shares border with Spain" becomes highly relevant to the original question. This approach, which we refer to as *Path-Aware Similarity Retrieval (PASR)*, can be applied on top of both the question-based and HyDE-based variants, resulting in SSR+PASR or SSR+HyDE+PASR, respectively.

Algorithm 1 describes the logic of the *Path-Aware Similarity Retrieval (PASR)* procedure. The algorithm takes as input a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$, a natural language question q, a set of seed entities extracted from q, and parameters controlling the retrieval depth (H), beam size (b), and subgraph size (N). The goal is to return a set of relevant edges $\mathcal{E}_q$ and the nodes they connect, based on their semantic similarity to the question (or its hypothetical answer).

Lines 3–7 handle the query setup. The question itself is used as the query, unless the *use_hyde* flag is enabled. In that case, the algorithm invokes an LLM to generate a hypothetical answer $\hat{a}$ to the input question, and uses that as *query* to retrieve similar KG edges[2]. Lines 8-9 initialize the main data structures: the set of collected edges, the set of visited nodes, and the initial *frontier*, which holds the seed nodes along with their cumulative path descriptions. Each item in the frontier is a pair consisting of a node and a textual trace representing the path taken to reach it.

The core loop (lines 10–24) expands the frontier for H hops. For each frontier node, the algorithm examines its outgoing edges (line 13), skips those already seen, and builds a new cumulative path description by appending the current edge's label to the prior context (line 14). Each edge is scored based on the cosine similarity between the query

[2]In our implementation, we embed the query once at the beginning and pass its vector to $\text{SIM}(\cdot)$ to avoid repeated encoding. The pseudocode has a slight difference for simplicity and to stay consistent with the definition of $\text{SIM}(\cdot)$ in the Matcher section.

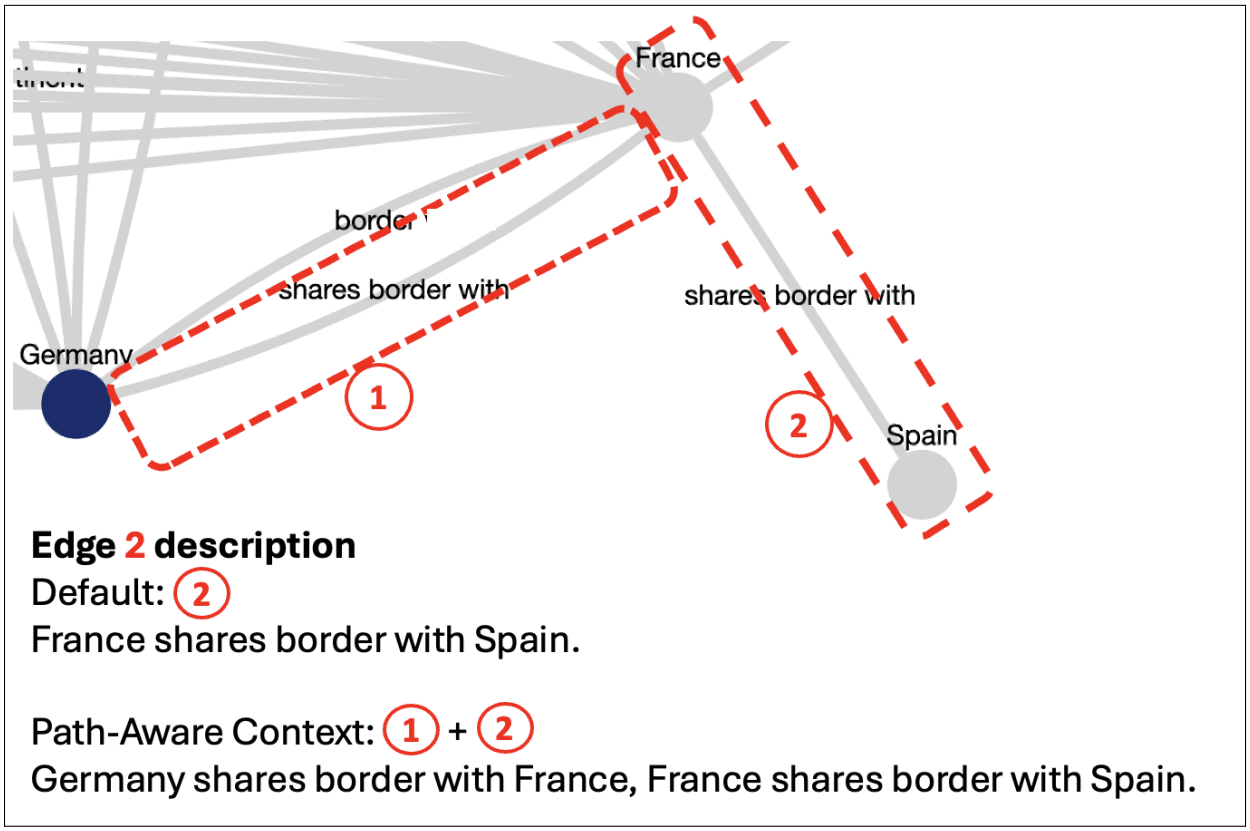


Figure 3.3: Path-aware context in describing KG edges.

and this cumulative text (line 15) and added to the set of collected edges. The SIM$(\cdot)$ function returns the similarity score between its input texts using an embedding model. This function is identical to the one used in the Matcher module described later, as they use the same underlying embedding model.

The top-b candidates are retained after expanding every node in the frontier (lines 19), and the collected edges are added to the result set (line 20). Their target nodes are used to construct the next-hop frontier. The loop terminates early if no more candidates are found or if the subgraph has reached the desired size (lines 21-23). The final result includes the retrieved edge set and all nodes seen during traversal.

This pseudocode represents the general case of our similarity-based retrieval methods. If the frontier stores only node labels and similarity is computed using individual edge descriptions, the algorithm reduces to *SSR* (with *use_hyde* = false) or *SSR + HyDE* (with

Algorithm 1 Path-Aware Similarity Retrieval (PASR)

Input: Knowledge Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$, seed_entities, question q, max_hops H, beam_size b, max_nodes N, flag *use_hyde*
Output: Retrieved edges $\mathcal{E}_q$, visited nodes $\mathcal{V}_q$
$query \leftarrow q$
if *use_hyde* **then**
    $\hat{a} \leftarrow Generate_hypothetical_answer(q)$
    $query \leftarrow \hat{a}$
end if
Initialize $\mathcal{E}_q \leftarrow \emptyset$, $seen_edges \leftarrow \emptyset$, $seen_nodes \leftarrow$ seed_entities
$frontier \leftarrow \{(s, -) \mid s \in \text{seed_entities}\}$ ▷ (node, cumulative description)
for $h = 1$ to H **do**
    $candidates \leftarrow \emptyset$
    for all (n, d_{cum}) in *frontier* **do**
        for all neighboring edges $e = (u, r, v)$ of n not in seen_edges **do**
            $d_{\text{new}} \leftarrow d_{\text{cum}}$; $\mathcal{L}(e)$
            $s \leftarrow \text{SIM}(query, d_{\text{new}})$
            Add (s, e, d_{new}) to candidates
        end for
    end for
    Sort candidates by score and keep top-b
    Add selected edges to $\mathcal{E}_q$ and expand frontier with their targets
    if $candidates = \emptyset$ **or** $|\mathcal{E}_q| \geq N$ **then**
        break
    end if
end for
return $(\mathcal{E}_q, seen_nodes)$

use_hyde = true). PASR extends these by incorporating cumulative path descriptions during scoring. We benchmark these retrieval methods in Chapter 4 on several KGs.

To illustrate Algorithm 1, consider the question: "Which countries have land borders with Germany's neighbors?" The seed node is *Germany*, the search depth is set to two hops, and the retriever uses a hypothetical answer as the similarity query. Figure 3.4 visualizes a small region of the KG relevant to this query, with edges labeled as `neighbors` and `capital`. Paths deemed more relevant to the query are shown in blue.

1. The LLM is first queried with the input question to generate a hypothetical answer, which is embedded and used as the retrieval query.

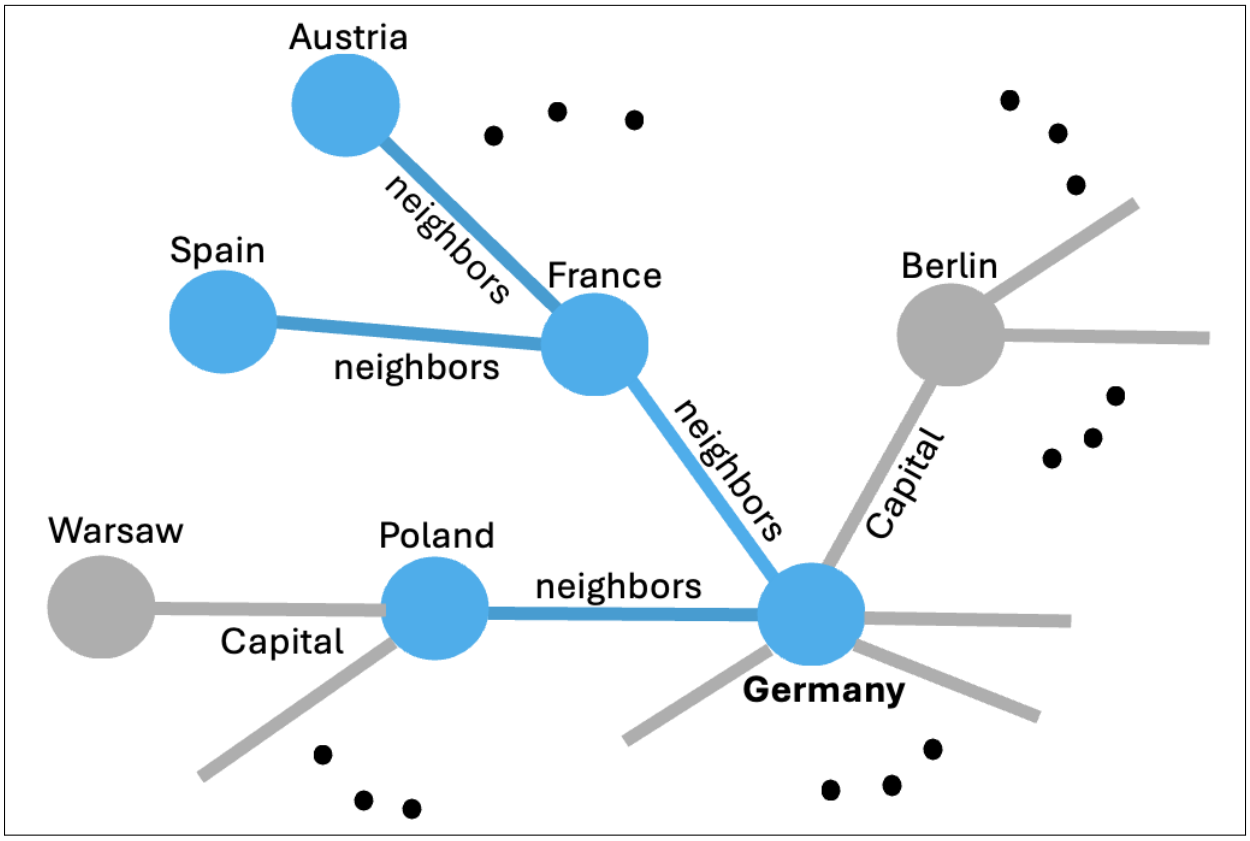


Figure 3.4: Illustration of Algorithm 1 over a KG neighborhood.

2. The frontier is initialized with the seed node (*Germany*) and an empty path description.
3. First hop:
   - Outgoing edges from Germany are expanded, including:
     $Germany \xrightarrow{\text{neighbors}} France$,
     $Germany \xrightarrow{\text{neighbors}} Poland$,
     $Germany \xrightarrow{\text{capital}} Berlin$
   - Each edge is converted into a short sentence such as "Germany neighbors France" or "Germany capital Berlin" and compared to the query using the similarity function.

- Assume the edges to *France* and *Poland* score highly, while the capital relation to *Berlin* does not qualify amongst top-ranked paths. Then, only *France* and *Poland* are added to the frontier for the next hop.

4. Second hop:
   - Frontier nodes including *France* and *Poland* are expanded similarly. For example:
     $France \xrightarrow{\text{neighbors}} Spain$,
     $France \xrightarrow{\text{neighbors}} Austria$,
     $Poland \xrightarrow{\text{Capital}} Warsaw$
   - Each two-step path is aggregated into a cumulative description, such as: "Germany neighbors France; France neighbors Spain.", and scored based on their relevance to the query.
5. After reaching the hop or node limit, the top-ranked paths are retained to form the retrieved subgraph $\mathcal{G}_q$. Here, $Poland \xrightarrow{\text{Capital}} Warsaw$ is pruned due to low similarity score to the query.

3.3 Prompting

For each question asked, AprèsCoT instructs the LLM to answer the question and share its reasoning process alongside final answers. Figure 3.5 shows the instruction template used to guide the language model during both base and RAG QA to specify the desired output structure.

Given a natural language question q, we define the full LLM prompt as:

$$q' = \textsc{Prompt}(q, C),$$

where C is an optional set of contextual facts derived from the retrieved subgraph. In *RAG mode*, C consists of textual sentences derived from the retrieved subgraph $\mathcal{G}_q$. If the user selects the non-RAG mode, the model answers only based on its pretrained knowledge and $C = \varnothing$. Specifically, in order to build C, we convert each edge $(u, r, v) \in \mathcal{G}_q$ into a natural language fact using its labels, and the resulting sentences are concatenated to form C. Figure 3.6 illustrates how a subgraph of the MetaQA knowledge graph is transformed into a textual format. MetaQA is introduced in detail in the next section.

The LLM's output containing its free-form reasoning text and final answers to the question is passed to the extractor module to facilitate matching free-form natural language to a KG.

3.4 Triple Extraction

This module initially parses out the LLM's reasoning and final answers based on the formatting instructions given to the LLM, discussed in Section 3.3. After parsing, two textual components are obtained: the final answer text and the reasoning text. The final answers are first collected into a set

$$A = \{a_1, a_2, \ldots, a_n\},$$

and the reasoning text is passed to a triple extraction module built using the *spaCy* library. We leverage spaCy's syntactic dependency parser and rule-based relation extractors to identify subject–predicate–object tuples from the reasoning text. This yields a set of extracted triples

$$T = \{T_1, T_2, \ldots, T_m\}, \quad T_i = (\text{subject}_i, \text{relation}_i, \text{object}_i).$$

Each extracted triple is then converted to a simple natural language sentence of the form *"subject relation object"*. The resulting set of textualized reasoning statements is denoted as

$$S = \{S_1, S_2, \ldots, S_m\},$$

where each S_i corresponds to the sentence representation of triple T_i.

Although we use *spaCy*'s dependency-based relation extraction in our current implementation, this component is modular and can be replaced with any off-the-shelf triple extraction method. These sets, A and S, are then passed to the matcher module, which aligns them with the retrieved subgraph.

3.5 Output Matching

The matcher module aligns the LLM outputs with the corresponding elements of the retrieved subgraph $\mathcal{G}_q$. To perform alignment, we project textual items from both the LLM output and the subgraph into a shared vector space using an embedding model $\phi(\cdot)$. Specifically, we embed the following:

- answer items $a_i \in A$ and reasoning steps $s_j \in S$,
- node labels $\mathcal{L}(v)$ for each node $v \in \mathcal{V}_q$, and
- edge labels $\mathcal{L}(e)$ for each edge $e \in \mathcal{E}_q$.

Similarity between two textual items x and y is computed as the cosine distance of their embeddings:

$$\text{SIM}(x, y) = \frac{\phi(x) \cdot \phi(y)}{\|\phi(x)\|\|\phi(y)\|}.$$

Reasoning steps are matched to edges via:

$$M_S = \{(s_j, e^*) \mid e^* = \arg\max_{e \in \mathcal{E}_q} \text{SIM}(s_j, \mathcal{L}(e)),\ \text{SIM}(s_j, \mathcal{L}(e^*)) \geq \tau_S\},$$

and answer items are matched to nodes analogously:

$$M_A = \{(a_i, v^*) \mid v^* = \arg\max_{v \in \mathcal{V}_q} \text{SIM}(a_i, \mathcal{L}(v)),\ \text{SIM}(a_i, \mathcal{L}(v^*)) \geq \tau_A\},$$

where τ is a similarity threshold distinguishing valid matches from non-matches. We set $\tau_S = 0.7$ and $\tau_A = 0.9$, determined via grid search on a curated dataset of valid matches and non-matching answers and sentences. That is, the matcher module returns the top match if its similarity is above the threshold and no match otherwise.

As we will show in the use cases presented in Section 4.4, failed matches correspond to gaps in inference paths, due to model hallucination or KG incompleteness. Items that fall below the threshold are considered unmatched, indicating potential hallucination or KG incompleteness.

In our implementation of AprèsCoT, we employ the *all-MiniLM-L6-v2* encoder as $\phi(\cdot)$, balancing runtime efficiency with semantic accuracy. The choice of embedding model is further evaluated experimentally in Section 4.3. The resulting alignment sets M_A and M_S form the basis for constructing structured explanations in the next step of the pipeline.

Finally, AprèsCoT overlays the matched reasoning path onto the retrieved subgraph. Entities in the question are highlighted, answers are marked, and reasoning steps are traced through the KG. This visualization enables users to investigate inconsistencies, missing links, or hallucinations, and it reveals differences across retrieval strategies or LLM backbones.

```
You are a reasoning assistant for question answering.
You will be given a question, and in some cases, a block
of context information retrieved from a knowledge base.
For each question, follow these steps:

1. Think step by step to answer the question. Use the
context information, if given, or use your knowledge
otherwise, and share your reasoning and every bit of
information you might use.

2. After you finish the reasoning, include a final section
that begins exactly with: ---FINAL ANSWERS---
Then list the final answers, one per line. Do not include
bullets or numbering.

Output Format:
<your reasoning here>
---FINAL ANSWERS---
<one answer per line>
```

Figure 3.5: Instructions given to the LLM in the QA process.

<Context>
Movie "The Shawshank Redemption" was directed by "Frank Darabont".
Movie "The Shawshank Redemption" was released in 1994.
Movie "The Mist" was directed by "Frank Darabont".
Movie "The Green Mile" was directed by "Frank Darabont".
Movie "The Mist" was released in 2007.
Movie "The Green Mile" was released in 1999.
Actor "Morgan Freeman" starred in "The Shawshank Redemption".
Actor "Tim Robbins" starred in "The Shawshank Redemption".
Movie "The Mist" was written by "Stephen King".
Movie "The Green Mile" has genre Drama.
Movie "The Green Mile" has genre Fantasy.
</Context>

<Question>
What were the release years of the films directed by the director of The Shawshank Redemption?
</Question>

Let's answer the Question by reasoning step by step.

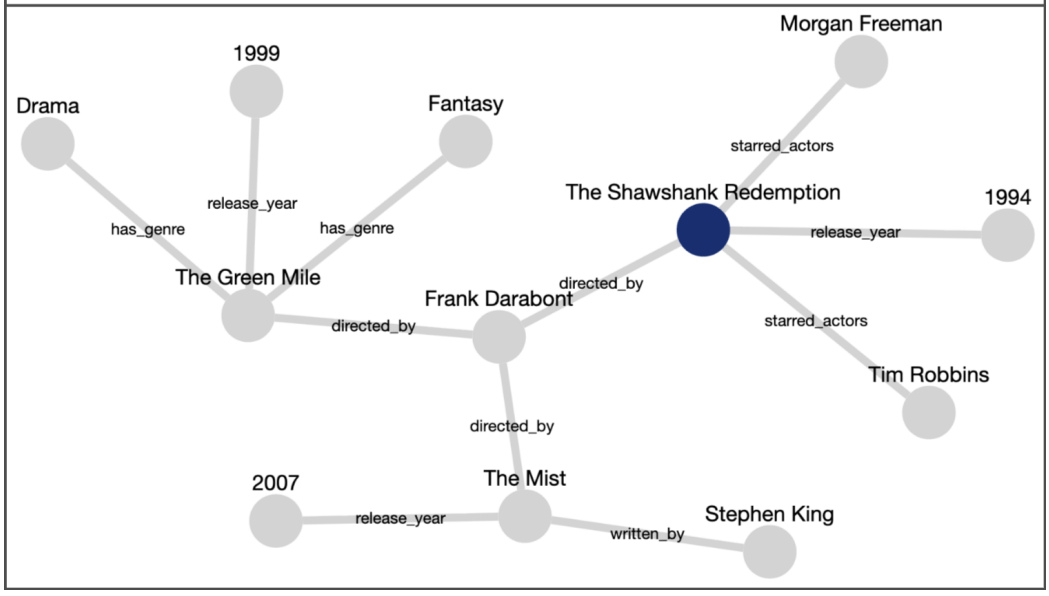


Figure 3.6: An example of a prompt created for a question asked in RAG mode.

Chapter 4

Experiments

This section evaluates the components of AprèsCoT through both benchmark experiments and illustrative use cases. We begin by describing the LLMs and knowledge graphs used in this study (Section 4.1). We then detail our subgraph retrieval experiments by describing each method under test, defining the evaluation metrics, and presenting the results in Section 4.2. Furthermore, we describe our experiments for selecting an embedding model that fits matching requirements in interactive settings (Section 4.3). Finally, we show use cases that demonstrate AprèsCoT's ability to explain and verify LLM reasoning across various application domains in Section 4.4.

4.1 Experimental KGs and LLMs

Our experiments use three knowledge graphs covering a range of domains and scales: MetaQA, UMLS, and WikiData. This diversity allows us to evaluate retrieval and matching performance in both focused and open-domain settings. Table 4.1 summarizes their key characteristics.

- MetaQA is a movie-focused KG used widely in KGQA benchmarks [28, 37]. It contains entities such as actors, directors, and movie titles, connected by relations such as starred_actor and directed_by. This is a suitable KG for complex multi-hop reasoning questions.
- UMLS (Unified Medical Language System) is a compact but dense medical KG, well-suited for analyzing inference alignment in a specialized domain. It contains rich

knowledge graph	domain	#nodes	#edges
WikiData [42]	*general*	90.4M	748.9M
MetaQA [49]	*movies*	43.2K	134.7K
UMLS [3]	*medical*	127	6.1K

Table 4.1: Dataset characteristics.

semantic relationships between medical concepts, including hierarchical and associative relations such as "treats" and "is a," and supports concept grouping by semantic types (e.g., all diseases or pharmacologic substances) [3].

- WikiData is a large, heterogeneous, open-domain KG with millions of entities and relations. It introduces scalability challenges for subgraph retrieval and matching, making it a suitable benchmark setting for assessing whether AprèsCoT remains responsive in interactive scenarios.

For LLMs, we experiment with two widely used models via OpenAI's API: ChatGPT 3.5 and gpt-4o-mini. Both models are capable of generating reasoning steps via chain-of-thought prompting.

4.2 Subgraph Retrieval Evaluation

4.2.1 Setup and Retrieval Methods

To evaluate the effectiveness of subgraph retrieval in AprèsCoT, we construct a benchmark of natural language questions that vary in structure and complexity. Each question is paired with a ground-truth subgraph, manually identified by us as the minimal set of KG edges leading to ground truth answer entities that are semantically necessary to answer the question and justify its reasoning trace. These ground-truth examples are used to compute precision and recall for each retrieval method.

The benchmark questions cover a range of properties, including the number of correct answers, the number of reasoning steps required (i.e., the number of ground-truth edges), the number of hops from the seed entities, the local density of the KG around those seed entities, and the lexical paraphrasing and similarity to the answer space. Table 4.2 illustrates a sample of the questions and their characteristics used in subgraph retrieval experiments.

Question	KG	Seed nodes	#hops
(Q1) What were the release years of the films directed by the director of The Shawshank Redemption?	MetaQA	The Shawshank Redemption	3
(Q2) Which countries have land borders with Germany's neighbors?	WikiData	Germany	2
(Q3) Where is the hometown of the clubs Arjen Robben has played for?	WikiData	Arjen Robben	2
(Q4) Which countries have football clubs where both Thierry Henry and Eden Hazard have played during their careers?	WikiData	Thierry Henry, Eden Hazard	2
(Q5) In which continents are the clubs that Edinson Cavani has played for based?	WikiData	Edinson Cavani	3
(Q6) Which continents contain the countries of the clubs that Edinson Cavani has played for?[1]	WikiData	Edinson Cavani	3
(Q7) What were the significant events happened in the capital of Germany?	WikiData	Germany	2
(Q8) In which languages has the director of The Pianist directed movies?	MetaQA	The Pianist	3
(Q9) Which countries are neighbors with both Argentina and Peru?	WikiData	Argentina, Peru	1
(Q10) What were the release years of the films starring both Jean Reno and Emily Mortimer?	MetaQA	Jean Reno, Emily Mortimer	2

Table 4.2: Sample of questions used in the subgraph retrieval experiments.

The goal is to compare five subgraph retrieval strategies discussed in Section 3.2. The first two methods are existing baselines, and the addition of HyDE and PASR to each variant are amongst the contributions of this work.

1. **BFS** (Breadth-First Search) (*Baseline*)
2. **SSR** (Semantic Similarity-based Retrieval) (*Baseline*)
3. **SSR + HyDE**: Comparing edge similarity to a hypothetical answer rather than the question in SSR.

[1]Notably, Q5 and Q6 are practically similar, only with different wordings. Q6 has a more straightforward wording to observe the effect of adding HyDE.

4. **SSR + PASR**: Comparing path similarity rather than edge similarity in SSR.
5. **SSR + HyDE + PASR**: Comparing path similarity to a hypothetical answer rather than edge similarity to the question in SSR.

4.2.2 Evaluation Metrics

We evaluate each subgraph retrieval method using four metrics that reflect both the quality of the retrieved subgraph and the system's runtime constraints:

1. **Recall** measures the proportion of ground-truth edges successfully retrieved. High recall is essential to ensure that reasoning paths and answer-supporting facts are available for matching and explanation. However, recall must be balanced against system requirements such as latency; overly large subgraphs, while complete, may hinder interactive analysis and overwhelm downstream components (i.e., the Matcher). Otherwise, low recall may lead to false no-match cases, which could be misinterpreted as KG incompleteness or LLM hallucination (see related use cases in Section 4.4).
2. **Precision** reflects the proportion of retrieved edges that belong to the ground-truth subgraph. While AprèsCoT is designed to tolerate irrelevant context during matching, higher precision leads to faster and more and accurate alignment.
3. **Subgraph Retrieval Time**: We measure the total time required to retrieve the subgraph for a given question. Since AprèsCoT is intended to operate in interactive settings, retrieval latency is a key constraint in selecting the final algorithm.
4. **Matching Time** (per match): To isolate the impact of retrieval on downstream processing, we measure the time taken to find the closest match for a single reasoning step and a single answer candidate with the retrieved subgraph. This metric captures how the size and quality of the subgraph affect overall explanation latency, independent of the LLM's output length.

4.2.3 Results

We now compare the performance of the five retrieval methods across the benchmark questions described earlier. Each method is evaluated using the metrics defined above. For each benchmark question, we compute the metrics individually and report averages

SR Method	Recall	Precision	SR time	Matching time
BFS	1.00	0.019	0.016 s	9.62 s
SSR	0.495	0.114	7.86 s	3.33 s
SSR+PASR	0.609	0.138	8.47 s	3.40 s
SSR+HyDE	0.708	0.164	10.39 s	3.44 s
SSR+HyDE+PASR	0.823	0.185	11.27 s	3.19 s

Table 4.3: Aggregated experiment results for all questions.

SR Method	Recall	Matching time	#edges
BFS	1.00	9.90 s	15K
SSR	0.26	3.66 s	130
SSR+PASR	0.34	3.66 s	130
SSR+HyDE	0.53	3.83 s	130
SSR+HyDE+PASR	0.72	3.45 s	130

Table 4.4: Aggregated experiment results for 3-hop questions.

across the full set of questions. This allows us to assess how each method performs in capturing relevant subgraphs while maintaining the responsiveness required by AprèsCoT.

Table 4.3 summarizes the results across all questions. Rows correspond to the five retrieval strategies, and columns report the average values of each evaluation metric across the benchmark. Matching time is reported per alignment operation. Recall is prioritized in our analysis, as missing critical edges may directly undermine the system's ability to verify reasoning traces.

Additionally, since multi-hop questions tend to amplify retrieval differences, we report an aggregated view of results specific to 3-hop questions in Table 4.4. While these cases are already included in the overall averages in Table 4.3, this focused view allows us to compare retrieval methods under greater retrieval complexity, including their impact on recall, matching latency, subgraph size, and visualizability.

4.2.4 Analysis

In this section, we summarize the observations from the results depicted in Tables 4.3 and 4.4:

BFS Impracticality While BFS achieves perfect recall by design, its exhaustive nature makes it impractical for AprèsCoT's responsiveness and explainability objectives. First, as shown in Table 4.3, BFS yields a significantly higher average matching time—9.62 seconds—more than 6 seconds slower than any other method. This latency scales linearly with the number of reasoning steps or answers, quickly compounding in multi-step explanations.

Second, as Table 4.4 illustrates, BFS retrieves an average of 15,000 edges for 3-hop questions, a scale that overwhelms the user interface and obscures the intended reasoning paths. In contrast, similarity-based methods yield significantly smaller, more focused subgraphs that remain suitable for interactive analysis and reasoning path tracing. Together, these limitations highlight why BFS, despite its coverage, is not a viable retrieval strategy for a system such as AprèsCoT.

Comparing Similarity-Based Retrievals Among the four similarity-based retrieval methods, we observe relatively consistent performance in **matching time** based on Table 4.3. This is expected due to the similar nature of these variants, since they only differ in the way they judge and rank KG edge similarity. **Subgraph retrieval time** slightly increases in HyDE-based variants due to the additional LLM call required to generate a hypothetical answer. However, the time gap does not increase as this inference step is performed only once per question, and the added cost does not scale with output size, remaining acceptable for interactive use. Besides, experiments are conducted with complex multi-hop queries, requiring deeper search than most common scenarios. In practice, this cost can be further reduced by substituting the remote LLM call with a local model. In addition, **precision** alone provides limited insight in this context as all methods retrieve some extraneous edges by design, and **recall** is the key differentiator.

Recall Improvements with HyDE and PASR Starting from a baseline average recall of 0.495 with SSR, we observe improvements with both HyDE and PASR individually, and a substantial boost when combined. The hybrid SSR+HyDE+PASR method achieves an average recall of 0.823, representing a 66% relative improvement over the SSR baseline. This confirms that both hypothetical answer guidance and path-aware context enhance the retriever's ability to recover relevant reasoning edges, and that their effects are complementary when used together.

To better understand the individual contributions of HyDE and PASR, we examine recall trends across methods. HyDE's benefit is evident in Table 4.3: adding a hypothetical answer significantly boosts recall when comparing SSR to SSR+HyDE, and similarly when comparing SSR+PASR to the combined SSR+HyDE+PASR. This suggests that providing

the retriever with a more answer-oriented guidance vector consistently improves its ability to surface relevant facts.

PASR's contribution, while more subtle in the aggregate, becomes clearer when focusing on 3-hop questions. The motivation behind PASR was to improve retrieval for long reasoning chains by incorporating contextualized path information rather than treating each edge in isolation. As shown in Table 4.4, this benefit is reflected in the results: the combined SSR+HyDE+PASR method achieves an average recall of 0.72 on 3-hop questions, compared to 0.53 for SSR+HyDE—a relative improvement of 36.5%. This indicates that PASR plays a critical role in deeper inference settings, where single-edge similarity is insufficient to capture the semantics of multi-step reasoning paths.

Interestingly, while recall is our primary optimization target, we also observe modest improvements in precision alongside recall as we move from SSR to SSR+PASR, SSR+HyDE, and ultimately SSR+HyDE+PASR. This is notable because precision and recall often trade off, but here both improve. This suggests that the additions of HyDE and PASR not only help retrieve more relevant facts, but also guide the retriever toward higher-quality candidates, reducing irrelevant edge selection. These trends further support the joint use of both techniques.

In conclusion, the SSR+HyDE+PASR strategy demonstrates the strongest performance and best aligns with AprèsCoT's requirements in terms of both recall and efficiency, and is thus selected as our default subgraph retriever.

4.3 Embedding Model Selection

The matcher module in AprèsCoT relies on sentence embeddings to compare LLM-generated statements with textualized KG facts. To ensure responsiveness in interactive settings, we prioritize embedding models that offer both semantic precision and fast inference.

We selected all-MiniLM-L6-v2, a lightweight transformer-based encoder producing 384-dimensional vectors amongst off-the-shelf embedding models. The choice was based on two factors: (1) encoding speed, to support efficient interaction, and (2) its ability to produce meaningful separation between semantically similar and dissimilar facts in the embedding space. Without proper separation, unrelated facts may score highly and be incorrectly matched during alignment.

Since mismatches between reasoning and KG facts can mislead interpretation, we prioritize semantic separation when selecting the embedding model. To evaluate this, we compared all-MiniLM-L6-v2 and a larger encoder with 768-dimensional vectors, DistilBERT,

- Movie 'The Tall Blond Man with One Black Shoe' was released in 1972.		
Sentence	MiniLM Similarities	DistilBERT Similarities
Actor 'Jean Rochefort' starred in 'The Tall Blond Man with One Black Shoe'.	0.6463	0.958
Actor 'Jean Rochefort' starred in 'The Hairdresser's Husband'.	0.2193	0.9442
Movie 'The Tall Blond Man with One Black Shoe' was directed by 'Yves Robert'.	0.7726	0.912
Movie 'The Tall Blond Man with One Black Shoe' was written by 'Yves Robert'.	0.731	0.9115
Movie 'The Tall Blond Man with One Black Shoe' was written by 'Francis Veber'.	0.7213	0.9496
Actor 'Bernard Blier' starred in 'The Tall Blond Man with One Black Shoe'.	0.6731	0.9533
Actor 'Pierre Richard' starred in 'The Tall Blond Man with One Black Shoe'.	0.6717	0.9605
Movie 'The Tall Blond Man with One Black Shoe' was released in 1972.	0.9897	0.988
Movie 'The Tall Blond Man with One Black Shoe' is in English language.	0.778	0.8923
Movie 'The Tall Blond Man with One Black Shoe' is in French language.	0.7135	0.9131
Movie 'The Tall Blond Man with One Black Shoe' is in French language.	0.7804	0.9086
Movie 'The Tall Blond Man with One Black Shoe' is described with 'yves robert' tag.	0.6614	0.9361
Movie 'The Tall Blond Man with One Black Shoe' is described with 'pierre richard' tag.	0.6839	0.9406
Movie 'The Hairdresser's Husband' was directed by 'Patrice Leconte'.	0.2972	0.935
Movie 'The Hairdresser's Husband' was written by 'Patrice Leconte'.	0.2402	0.9339
Movie 'The Hairdresser's Husband' was written by 'Claude Klotz'.	0.2756	0.9461
Actor 'Anna Galiena' starred in 'The Hairdresser's Husband'.	0.286	0.9264
Movie 'The Hairdresser's Husband' was released in 1990.	0.3925	0.9776
Movie 'The Hairdresser's Husband' is in French language.	0.2916	0.9023
Movie 'The Hairdresser's Husband' is described with 'patrice leconte' tag.	0.2097	0.9466
Movie 'The Hairdresser's Husband' is described with 'hairdresser' tag.	0.2964	0.9169

Figure 4.1: MiniLM vs. DistilBERT: Darker green indicates higher similarity, reflecting better separation of relevant and irrelevant facts by MiniLM.

by measuring cosine similarity scores between a target sentence and a set of candidate KG facts. Figure 4.1 shows an example where the input sentence is: "The Tall Blond Man with One Black Shoe was released in 1972." Both models rank the correct KG fact highest, but DistilBERT yields a narrow spread of similarity scores, offering limited separation between relevant and irrelevant facts. In contrast, all-MiniLM produces a broader score distribution, reducing the risk of false alignments and improving match reliability.

Hence, all-MiniLM-L6-v2 is used as the default encoder in the experiments and the web interface.

4.4 Use Cases

In this section, we describe three use cases illustrating consistent inference, incomplete inference, and KG incompleteness, each on one of the pre-loaded KGs. We will analyze LLM answers over knowledge graphs such as WikiData for their general questions, MetaQA for movie inquiries, and UMLS for medical questions. Furthermore, we compare the outputs produced by several LLMs, both in standalone and RAG mode.

4.4.1 Consistent Answers and Reasoning

The first use case is illustrated in Figure 4.2. The user selects gpt-4o-mini as the LLM and WikiData as the knowledge graph, and submits a geography-related question: "Which countries have land borders with both Germany and the Czech Republic?". As shown in the parsed response tables, "Poland" and "Austria" are identified as final answers.

To verify the answers and reasoning steps, users can trace the corresponding search patterns starting from the two entities mentioned in the question, "Germany" and "Czech Republic." Edges labeled [S1] through [S9] capture all countries that share land borders with Germany, while [S10] through [S13] represent those sharing land borders with the Czech Republic.

The only two nodes discovered from both search paths are "Poland" and "Austria," forming two distinct reasoning chains connecting "Germany" and "Czech Republic." This use case shows consistency between answers and reasoning steps, as the two completed reasoning paths yield two correct final answers, and it demonstrates full alignment of the LLM response with the facts in WikiData.

4.4.2 KG Data Quality

In this use case, AprèsCoT surfaces potential data quality issues in the KG. The user submits the same query as in Figure 1.1: "What were the release years of the films starred by Jean Rochefort?", but this time uses gpt-4o-mini in standalone mode, without access to any external context. The model generates an answer based on its pretrained knowledge and produces both an answer and intermediate reasoning steps via chain-of-thought prompting. Since the query pertains to movies, the MetaQA knowledge graph is selected as the grounding source.

As shown in Figure 4.3, gpt-4o-mini produces five answer candidates, [A1] through [A5], supported by ten reasoning steps [S1] through [S10]. However, only two of the predicted answers and their corresponding reasoning paths (those already seen in the example from the Introduction) can be matched to facts in the KG by AprèsCoT. The remaining answers (1995, 1967, and 1974) and most of the reasoning steps ([S1]–[S3], [S8]–[S10]) cannot be aligned with any KG content. Upon inspection, [S1] is a generic biographical statement, while [S2] and [S3] refer to a film released in 1995 starring Jean Rochefort that is not present in MetaQA. Similarly, [S8] through [S10] cite additional films from 1967 and 1974. These mismatches prompt a further investigation: if such films do not exist, the LLM has likely hallucinated. But if they do exist, the KG is incomplete. In this case, a web search confirms that the missing films are real, highlighting incompleteness in MetaQA.

Repeating the experiment with ChatGPT 3.5 yields a similar pattern; some additional correct answers are surfaced, but others are omitted. Notably, the reasoning traces from ChatGPT 3.5 occasionally include procedural instructions (e.g., "Identify films starred by Jean Rochefort", "check their release years") rather than factual content, reflecting stylistic differences in LLM behavior across models.

4.4.3 Inconsistent Answers and CoT

We conclude with an example from the UMLS medical KG that illustrates inconsistencies between an LLM's answers and its chain of thought. In Figure 4.4, the user asks, "What types of animals are affected by dysfunctions caused by Fungus?" using ChatGPT 3.5 with RAG enabled. The model produces three final answers, [A1] *Mammal*, [A2] *Reptile*, and [A3] *Bird*, along with five reasoning steps [S1] through [S5], all of which are matched against a KG subgraph retrieved by AprèsCoT.

Using the visual explanation, the user can verify that only two answers, *Mammal* and *Reptile*, are logically grounded in the KG via complete reasoning paths beginning at *Fungus*, aligned with CoT steps [S1], [S2], and [S5]. However, the third answer, *Bird*, lacks any supporting path from the source entity. This absence is shown in the graph via dashed red edges, marking a break between the CoT and the KG evidence. Additionally, CoT steps [S3] and [S4], while valid KG facts, do not contribute to answering the question as they reference non-animal entities outside the scope of the query.

Switching to gpt-4o-mini with RAG reveals a different inference pattern: while the final answers remain the same, the reasoning steps align fully with the answers, including a valid path to *Bird*. At the same time, the irrelevant steps observed in the earlier output

([S3] and [S4]) are absent. This example again shows AprèsCoT's value in understanding the inference differences between LLMs.

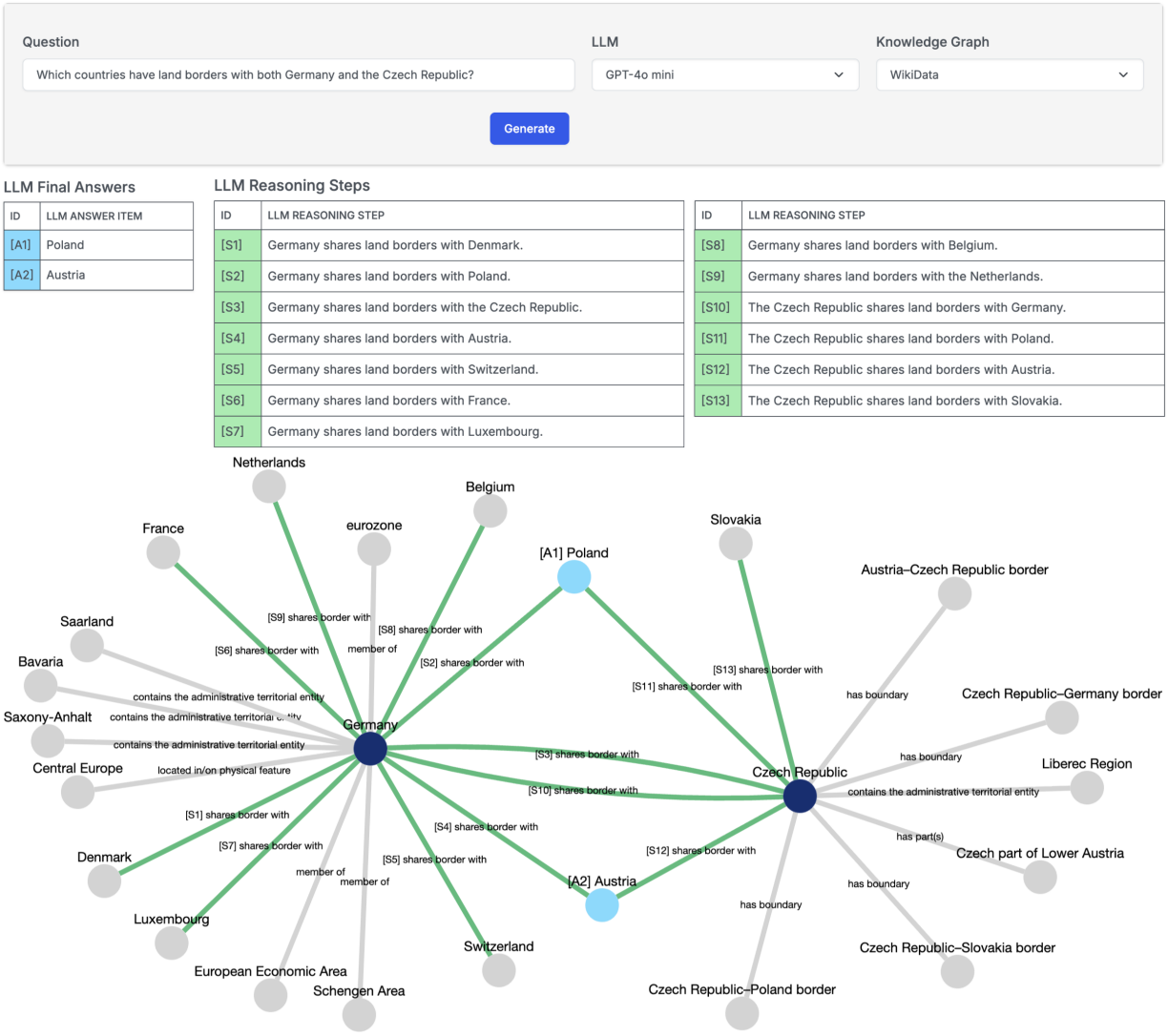


Figure 4.2: An explanation showing consistency in LLM answers and reasoning.

Question

What were the release years of the films starred by Jean Rochefort?

LLM

GPT-4o Mini

Knowledge Graph

MetaQA Movies

Generate

LLM Final Answers

ID	LLM ANSWER ITEM
[A1]	1995
[A2]	1972
[A3]	1990
[A4]	1967
[A5]	1974

LLM Chain of Thought

ID	LLM CHAIN OF THOUGHT STEP
[S1]	Jean Rochefort was a prominent French actor known for his work in various films.
[S2]	Jean Rochefort starred in the film 'The Horseman on the Roof'.
[S3]	The Horseman on the Roof' was released in 1995.
[S4]	He also starred in 'The Tall Blond Man with One Black Shoe'.
[S5]	'The Tall Blond Man with One Black Shoe' was released in 1972.
[S6]	Another notable film he starred in is 'The Hairdresser's Husband'.
[S7]	'The Hairdresser's Husband' was released in 1990.
[S8]	Rochefort appeared in 'The Last Adventure'.
[S9]	Movie 'The Last Adventure' which was released in 1967.
[S10]	He was also in 'The Return of the Tall Blond Man' released in 1974.

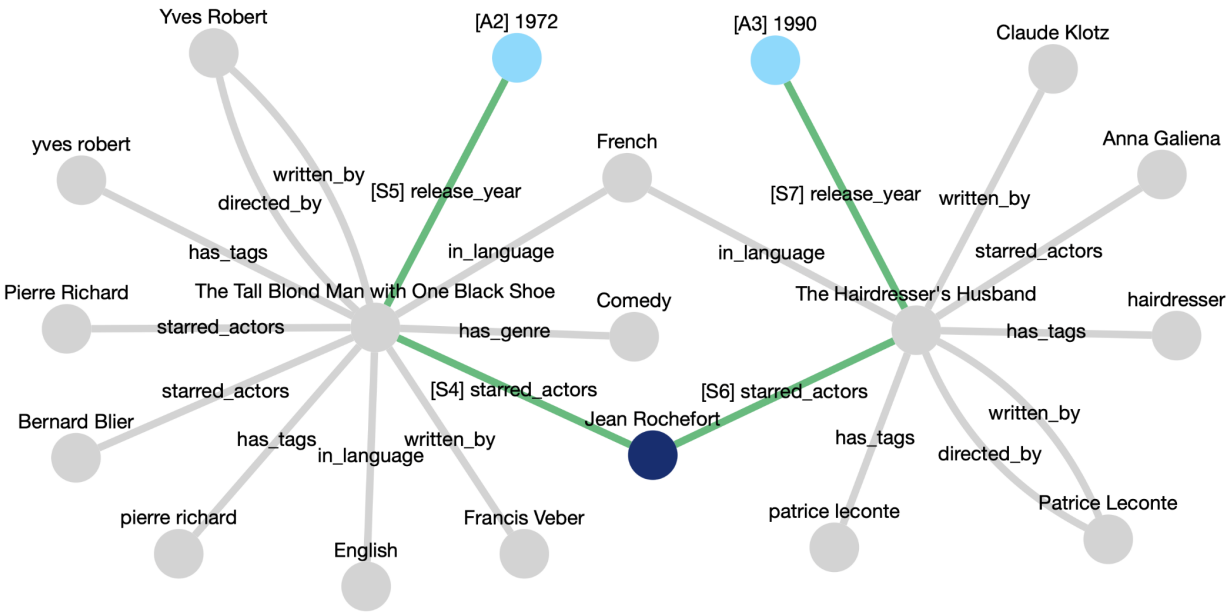


Figure 4.3: An explanation showing missing information in the KG.

Question

What types of animals are affected by dysfunctions caused by Fungus?

LLM

ChatGPT 3.5 + KG RAG

Knowledge Graph

UMLS Relations

Generate

LLM Final Answers

ID	LLM ANSWER ITEM
[A1]	Mammal
[A2]	Reptile
[A3]	Bird

LLM Chain of Thought

ID	LLM CHAIN OF THOUGHT STEP
[S1]	Fungus causes Mental or Behavioral Dysfunction.
[S2]	Mental or Behavioral Dysfunction affects Mammal.
[S3]	Mental or Behavioral Dysfunction affects Mental Process.
[S4]	Mental or Behavioral Dysfunction affects Neoplastic Process.
[S5]	Mental or Behavioral Dysfunction affects Reptile.

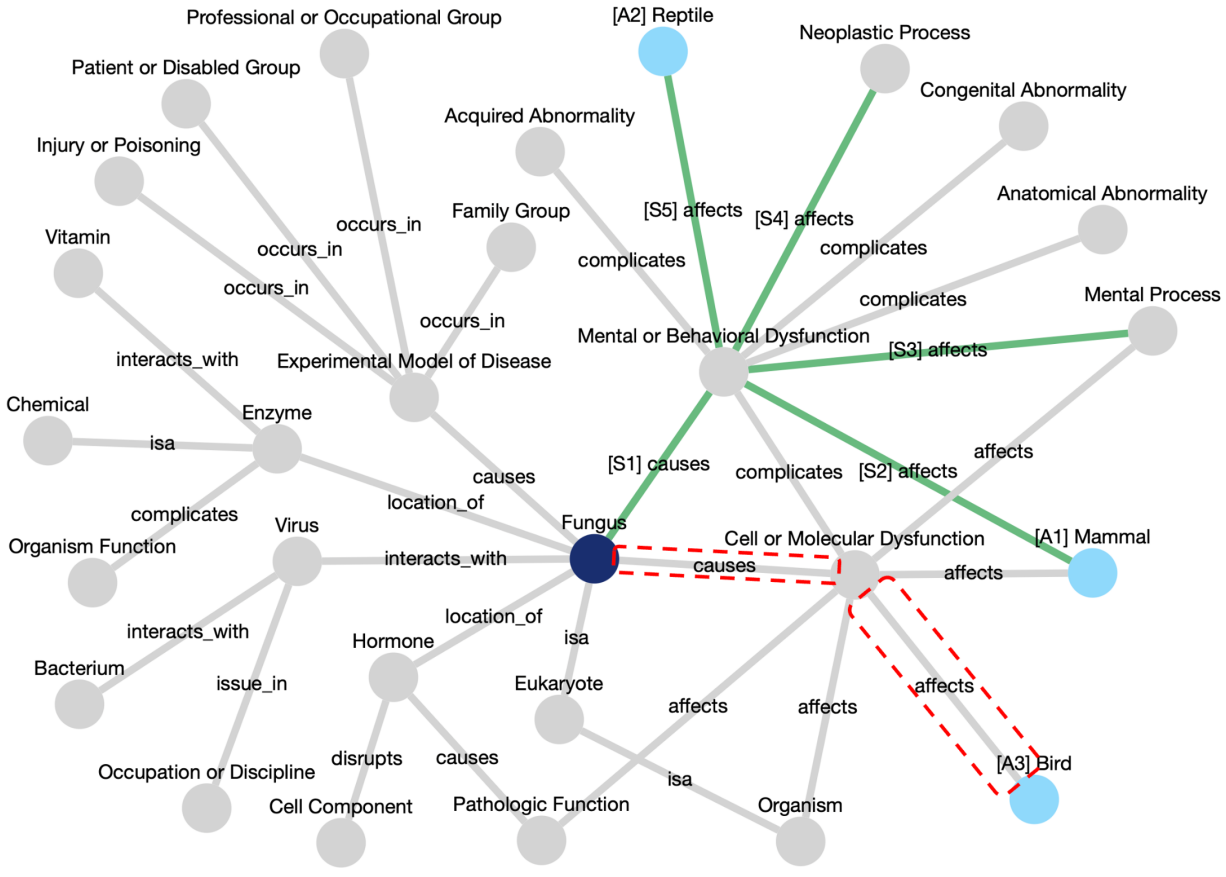


Figure 4.4: An explanation showing inconsistent LLM Answers and Reasoning.

Chapter 5

Conclusions and Future Work

5.1 Final Remarks

We introduce AprèsCoT, a lightweight, modular framework for explaining LLM answers and reasoning in a post-hoc manner, aimed at aligning model-generated answers and reasoning steps with structured knowledge from external knowledge graphs. AprèsCoT provides interpretable traces of reasoning on KGs across a range of domains by decomposing the process into three components: subgraph retrieval, LLM response parsing, and semantic matching. These modules are implemented and evaluated with multiple design alternatives to understand their trade-offs and effectiveness.

Through systematic experiments, we benchmarked different KG retrieval algorithms and embedding models to assess their impact on retrieval coverage, matching precision, and computational cost. Notably, we proposed and validated novel extensions to standard similarity-based retrieval methods, namely Hypothetical Document Embeddings (HyDE) and Path-Aware Similarity Retrieval (PASR), which improved recall significantly without sacrificing runtime efficiency. We also demonstrated AprèsCoT's practicality through curated use cases demonstrating LLM consistencies in reasoning, model hallucination, and KG incompleteness, highlighting how structured visualizations can aid users in interpreting and auditing LLM behavior.

5.2 Future Direction

Building on the current framework, several promising extensions can be explored:

First, AprèsCoT could be adapted to identify and diagnose adversarial behavior in RAG systems. In such settings, malicious actors may inject misleading or false information into the retrieved context to influence the LLM's output. By aligning reasoning steps with a trusted KG, AprèsCoT can help surface contradictions or mismatches, providing a structured view of where the model's reasoning diverges from the ground truth. This opens opportunities to integrate AprèsCoT into AI auditing tools or safety protocols.

Second, as knowledge graphs grow in size and complexity, performance bottlenecks emerge during subgraph retrieval and matching. To address this, future work can focus on infrastructure-level improvements, such as using native graph data stores with support for efficient multi-hop neighborhood indexing or in-memory caching strategies. These optimizations would allow AprèsCoT to remain responsive even when operating on large-scale KGs like WikiData, enabling interactive use in more demanding environments.

Third, the subgraph retrieval process itself can be further optimized. Currently, similarity comparisons are performed at query time, which may be computationally expensive. Future versions of AprèsCoT could incorporate precomputed semantic indices or leverage learned retrieval models trained on question-answer pairs to reduce runtime overhead. Query-time pruning heuristics may also help limit the search space without sacrificing relevance, thereby improving both speed and accuracy of the subgraph selection.

These directions offer natural next steps for extending AprèsCoT toward broader impact and stronger real-world utility. We hope this work contributes to more transparent, trustworthy, and interpretable language model applications.

References

[1] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, Volume 1: Long Papers*, pages 344–354. The Association for Computer Linguistics, 2015.

[2] Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *CoRR*, abs/2303.08112, 2023.

[3] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[4] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*, pages 2787–2795, 2013.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[6] Andrew Chai, Alireza Vezvaei, Lukasz Golab, Mehdi Kargar, Divesh Srivastava, Jaroslaw Szlichta, and Morteza Zihayat. EAGER: explainable question answering using knowledge graphs. In *Proceedings of the 6th Joint Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, pages 4:1–4:5. ACM, 2023.

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

[8] Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. Contextcite: Attributing model generation to context. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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, NAACL-HLT 2019*, pages 4171–4186. Association for Computational Linguistics, 2019.

[10] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics, ACL 2024*, pages 3563–3578. Association for Computational Linguistics, 2024.

[11] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, pages 1762–1777. Association for Computational Linguistics, 2023.

[12] Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 30–45. Association for Computational Linguistics, 2022.

[13] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[14] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020.

[15] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

[16] Wenyu Huang, Guancheng Zhou, Hongru Wang, Pavlos Vougiouklis, Mirella Lapata, and Jeff Z. Pan. Less is more: Making smaller language models competent subgraph retrievers for multi-hop KGQA. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15787–15803. Association for Computational Linguistics, 2024.

[17] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43, 2023.

[18] Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net, 2023.

[19] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[20] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[21] Leslie Lamport. *LATEX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[22] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[23] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[24] Weitao Li, Junkai Li, Weizhi Ma, and Yang Liu. Citation-enhanced generation for llm-based chatbots. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, pages 1451–1466, 2024.

[25] Yading Li, Dandan Song, Changzhi Zhou, Yuhang Tian, Hao Wang, Ziyi Yang, and Shuhao Zhang. A framework of knowledge graph-enhanced large language model based on question decomposition and atomic retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11472–11485. Association for Computational Linguistics, 2024.

[26] Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. Deductive verification of chain-of-thought reasoning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, 2023.

[27] Fengyuan Liu, Nikhil Kandpal, and Colin Raffel. Attribot: A bag of tricks for efficiently approximating leave-one-out context attribution. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*. OpenReview.net, 2025.

[28] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024*. OpenReview.net, 2024.

[29] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*

35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, 2022.

[30] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pages 12076–12100. Association for Computational Linguistics, 2023.

[31] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

[32] Joel Rorseth, Parke Godfrey, Lukasz Golab, Divesh Srivastava, and Jaroslaw Szlichta. RAGE against the machine: Retrieval-augmented LLM explanations. In *40th IEEE International Conference on Data Engineering, ICDE*, pages 5469–5472. IEEE, 2024.

[33] Yuanchun Shen. SRTK: A toolkit for semantic-relevant subgraph retrieval. *CoRR*, abs/2305.04101, 2023.

[34] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4444–4451. AAAI Press, 2017.

[35] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007*, pages 697–706. ACM, 2007.

[36] Viju Sudhi, Sinchana Ramakanth Bhat, Max Rudat, and Roman Teucher. Rag-ex: A generic framework for explaining retrieval augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024*, pages 2776–2780. ACM, 2024.

[37] Haitian Sun, Tania Bedrax-Weiss, and William W. Cohen. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pages 2380–2390. Association for Computational Linguistics, 2019.

[38] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth*

International Conference on Learning Representations, ICLR 2024. OpenReview.net, 2024.

[39] James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. The fact extraction and verification (FEVER) shared task. In *Proceedings of the First Workshop on Fact Extraction and VERification, FEVER@EMNLP 2018*, pages 1–9. Association for Computational Linguistics, 2018.

[40] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 5998–6008, 2017.

[42] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[43] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 7534–7550. Association for Computational Linguistics, 2020.

[44] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[45] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022.

[46] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[47] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*, pages 2335–2344. ACL, 2014.

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

[49] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 6069–6076. AAAI Press, 2018.

[50] Ruilin Zhao, Feng Zhao, Long Wang, Xianzhi Wang, and Guandong Xu. Kg-cot: Chain-of-thought prompting of large language models over knowledge graphs for knowledge-aware question answering. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024*, pages 6642–6650. ijcai.org, 2024.