Causal Cartographer: From Mapping to Reasoning Over Counterfactual Worlds

Gaël Gendron[1]*   **Jože M Rožanec**[2,3]   **Michael Witbrock**[1]   **Gillian Dobbie**[1]

[1]NAOInstitute, University of Auckland
[2]Jožef Stefan Institute
[3]University of Twente

Abstract

Causal world models are systems that can answer counterfactual questions about an environment of interest, i.e. predict how it would have evolved if an arbitrary subset of events had been realized differently. It requires understanding the underlying causes behind chains of events and conducting causal inference for arbitrary unseen distributions. So far, this task eludes foundation models, notably large language models (LLMs), which do not have demonstrated causal reasoning capabilities beyond the memorization of existing causal relationships. Furthermore, evaluating counterfactuals in real-world applications is challenging since only the factual world is observed, limiting evaluation to synthetic datasets. We address these problems by explicitly extracting and modeling causal relationships and propose the Causal Cartographer framework. First, we introduce a graph retrieval-augmented generation agent tasked to retrieve causal relationships from data. This approach allows us to construct a large network of real-world causal relationships that can serve as a repository of causal knowledge and build real-world counterfactuals. In addition, we create a counterfactual reasoning agent constrained by causal relationships to perform reliable step-by-step causal inference. We show that our approach can extract causal knowledge and improve the robustness of LLMs for causal reasoning tasks while reducing inference costs and spurious correlations.

1 Introduction

Learning to infer causal relationships and making causal predictions about the world is an important task for general reasoning systems [12, 40]. In particular, predicting how an environment would have evolved under a different policy (i.e. counterfactual questions) is a challenging and crucial question when evaluating how an artificial system understands the world. While an agent with low causal knowledge can make predictions about observed distributions, generalizing to arbitrary distributions and counterfactuals requires one to learn a causal world model [1, 36]. Studies on large language models (LLMs) have shown that they do not perform robust causal discovery or causal inference [48, 16, 17, 5, 18, 19] and fail to generalize to unseen distributions [44, 9, 10, 2]. This is a challenging task as causal knowledge is notoriously hard to collect [39, 31] and counterfactual data is generally not available as only the factual world is observed [14].

We tackle these problems by explicitly modeling causal relationships and enhancing LLM agents with a causal reasoning framework. We introduce the **Causal Cartographer**: a twofold system composed of a *graph retrieval-augmented generation* (graphRAG [22, 7, 35]) *agent* tasked to retrieve causal relationships from real-world news articles and a *counterfactual reasoning agent* performing reliable and efficient step-by-step causal inference while respecting causal relationships. We take advantage of the first component to build CausalWorld, a causal network that maps causal knowledge of the world,

---

*Corresponding author. Email: `ggen187@aucklanduni.ac.nz`

which we use to *provably* access counterfactual knowledge and build real-world causal questions for our second reasoning agent. We use the extracted knowledge to guide our causal reasoning agent on real-world counterfactual reasoning tasks. We show that our proposed method allows for the extraction of causal knowledge from the data and the evaluation of agents in real-world counterfactual situations. We summarize our contributions as follows:

- We introduce a causal extraction method for unstructured natural language data and a graph retrieval-augmented generation (Graph-RAG) agent;
- We use our method to build CausalWorld, a network of causal relationships from real-world news events published in 2020, containing 975 variables and 1337 causal relationships;
- We use this network as a repository of causal knowledge to evaluate large language models on counterfactual reasoning with real-world natural language data;
- We introduce the theoretical notion of causal blankets and prove that, assuming identifiability of the blanket, our method allows estimating true counterfactuals;
- We introduce an auxiliary step-by-step causal inference agent that can perform counterfactual reasoning respecting causal constraints;
- We show that causal conditioning achieves competitive performance for counterfactual reasoning while greatly reducing the LLM's context window and output length, decreasing the inference cost up to 70%.

Our code is available at: `https://github.com/ggendro/causal-cartographer`.

2   Background

Structural Causal Models   Recovering causal relationships in the real world is a long-standing problem of science [31]. Structural Causal Models (SCMs) [31, 33, 45] are graphical models that allow representing causal knowledge and performing causal inference. SCMs can be represented as Directed Acyclic Graphs (DAGs) where nodes are causal variables and edges are causal relationships. Causal variables can be distinguished between *endogenous* (observed) and *exogenous* (unobserved) variables. Exogenous variables $\mathbf{U}$ do not have parents and are represented by a probability distribution. Endogenous variables $\mathbf{V}$ are defined by functions that links them to their causal parents, i.e. $V_i \leftarrow f_{V_i}(\mathbf{pa}(V_i))$. SCMs are Markovian processes, which implies that if the full set of exogenous variables is known, then the values of every endogenous variable can be deterministically computed by iteratively inferring children values from their parents [8].
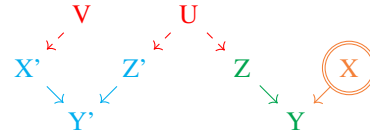

Figure 1: Counterfactual twin graph for three endogenous variables $\{X, Y, Z\}$ and two exogenous variables $\{U, V\}$, under an intervention over X. A counterfactual world is identical to the factual world, except for the intervention and its effects. Exogenous variables are shared by both worlds, connecting the two graphs. X is being intervened upon and so has no incoming edges. The counterfactual value of Y given by $P(Y \mid do(X), X', Y')$ is obtained by estimating the value of U from the factual observations, then deducing Z from U (left unchanged) and Y from Z and X.

While very powerful tools to estimate causal effects, SCMs typically cannot be fully retrieved as causal information is challenging to obtain.

Counterfactual Inference   Even without an SCM, some causal questions can be answered. Causal inference queries can typically be divided into three categories: *observations*, *interventions* and *counterfactuals*, forming *Pearl's Causal Hierarchy* [1]. Each category is harder to answer than the previous one, as it typically requires a better understanding of the causal relationships underlying the system of interest [1]. While multiple Markov-equivalent causal graphs can account for the same set of observations, answering interventions and counterfactuals requires one to know the true local causal structure [1, 36]. Moreover, the result of a counterfactual is generally not accessible in real-world scenarios. This is the *fundamental problem of causal inference*: only the factual world is observed [14].

(a) Causal extraction process.

India Oil Diversification Strategy
(*boolean*: [True, False])

India's strategic initiative to diversify its oil supply sources away from the traditional reliance on the Middle East.

world_188   world_226
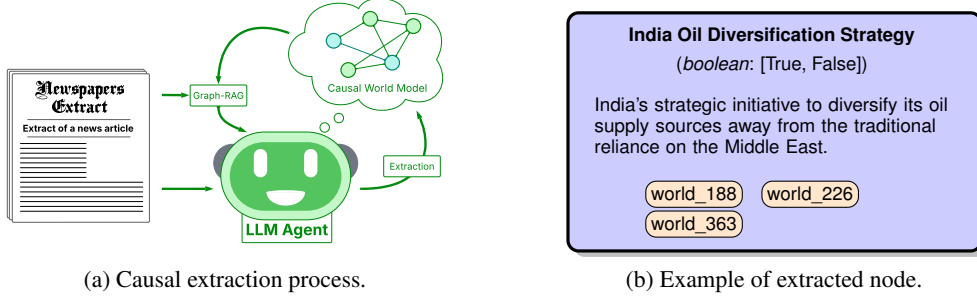world_363

(b) Example of extracted node.

Figure 2: Overview of the *Causal Cartographer Extraction agent* (CTG-Extract). (a) Illustration of the extraction pipeline. The title, content, and metadata from a news source are provided to the agent context. It then generates causal variables and their relationships and adds them to the causal world model. A graphRAG system further compares article data and agent node suggestions and adds to the agent context the nodes and causal relationships that seem related. (b) Example of a node extracted with the method. Its name, description, type, possible values, and the worlds it appears in are shown.

Observational queries are represented as conditional probabilities $P(Y \mid X)$. Interventions are represented as $P(Y \mid do(X))$ using the *do-operator* [31, 32]. An intervention alters the causal structure by forcing the value of the variable X, regardless of its prior probability, effectively cutting the parents of X. An intuitive example is a randomized control trial where the probability of obtaining treatment is randomized to ensure that it cannot correlate with other factors. Counterfactuals correspond to "what if?" questions, asking how would a world evolve under an intervention, given the outcome in the factual world. They can be represented with the following equation:

$$P(Y \mid do(X), X', Y') = \sum_{U \in \mathbf{U}} P(Y \mid do(X), U) P(U \mid X', Y') \tag{1}$$

X' and Y' represent the factual observations. X and Y correspond to the variables in the counterfactual world. $\mathbf{U}$ is the set of exogenous variables shared between the two worlds. Figure 1 shows a graphical expression with twin graphs corresponding to the factual and counterfactual worlds.

3 Graph Retrieval-Augmented Causal Extraction

This work introduces a method for causal extraction and reasoning based on LLM agents. This section focuses on the *Causal Cartographer Extraction agent* (CTG-Extract). We use it to recover existing causal knowledge about the world from news sources. The resulting causal network, insights we gather from it, and its application to counterfactuals are discussed in Sections 4 and 5.

3.1 CTG-Extract

We extract causal variables and causal relationships from news sources. Figure 2a illustrates the causal extraction process. While causal structure discovery methods attempt to infer the causal structure from data, causal extraction tasks recover *stated* causal relationships in text data [8, 46]. Large language models have shown limited performance in discovering causal relationships beyond domain knowledge [19, 18, 48, 17]. Acknowledging this limitation, we rely on human sources and only use the LLM agent to perform causal extraction. Causal information can either be *explicitly* or *implicitly* stated in the text, within a single sentence or across multiple sentences of paragraphs. Previous methods based on pattern-matching or statistical modeling do not capture the latter well [46]. Attention in LLMs allows them to aggregate information across a document, mitigating this problem.

Agent Description   We construct an LLM agent following the ReAct framework [47] and using the Smolagents library [2]. We prompt the LLM agent to conduct step-by-step causal extraction as a sequence of *thoughts*, *code* and *observations*. A thought step is a draft of the step to execute while

_____

[2]https://huggingface.co/docs/smolagents/

the code step is its implementation. The agent is provided with a Python interpreter. The standard output and errors of the interpreter are returned to the agent in the observation step. The agent then adapts its next step based on this feedback. It is tasked to execute the following plan:

1. Define the causal variables existing in the input text;
2. Define the *confounders*, i.e. causal variables that are not observed or mentioned in the text but have a direct effect on the observed variables;
3. Match the new variables with ones existing in the causal graph if possible (with graphRAG);
4. Define the causal relationships between the variables based on the text, relationships already existing in the graph should not be created;
5. Add the new causal variables and relationships to the graph.

Every requirement can be executed in a single thought-code-observation cycle but the agent can perform more steps if it did not immediately succeed. The prompts are provided in Appendix J. Each variable contains the following attributes: `name` and `description`, `type` and `values` describe the domain of the variable, `current_value` is the current instantiation of the variable, `contextual_information` provides additional context to the variable instantiation and `supporting_text_snippets` is the text extract justifying the response. Each relationship contains `cause` and `effect` variables, and similar `description` and `contextual_information` attributes. Their implementation is provided in Appendix J. An example is also provided in Figure 2b.

Grounding in Worlds To later perform counterfactual matching (in Section 5), we save multiple *worlds* per node. Each document describes a world, i.e. an instantiation of the observed subgraph of the world graph. For instance, two nodes A and B can be described in several documents. The concept they represent (described by attributes `name`, `description`, `type` and `values`) is invariant to world changes. However, their instantiations (described by attributes `current_value`, `contextual_information`, and `supporting_text_snippets`) can differ from one world to the next. They are saved as coming from different worlds, as illustrated by Figure 2b.

NxGraphRAG In addition to the Python interpreter, we introduce a novel graph retrieval-augmented generation (graphRAG) method [22, 7, 35] that we equip the agent with. The system, NxGraphRAG, generates an embedding of the input document and of each node of the causal graph using an auxiliary LLM and keeps them in a vector database. Before providing the document to the agent, NxGraphRAG is called and returns the top-K nodes with the highest cosine similarity to the input document. It also traverses the graph through P steps from each retrieved node and adds each neighbor node and edge to ensure that no relevant context is omitted. The embedding-based retrieval focuses on semantically relevant nodes while the traversal takes advantage of structural knowledge. After this initial call, the NxGaphRAG pipeline can be called again by the agent when creating new variables to verify that they do not already exist in the graph. An overview of the pipeline and additional details are provided in Appendix D.

3.2 Data

Our effort to build a causal graph from phenomena observed in the real world is part of a broader effort to understand the factors that influence the investment in and use of green energy [38, 11]. Therefore, when building the dataset, we constrained ourselves to media news related to the *oil price* topic, given research has shown that oil price fluctuations affect the consumption of renewable energy sources [42], and that positive shocks in oil prices reduce CO_2 emissions [15]. We constructed a dataset of 500 media events from 2020 by extracting them from EventRegistry [21], a platform sourcing, processing, and clustering media news based on events reported over time. For each media event, they provide a summary and information regarding the piece of news considered the most representative. We exploited this information to build the causal graph described in Section 4.

4 The CausalWorld Network

The CausalWorld network is a causal graph generated using the methodology described in Section 3. It contains 975 nodes and 1337 edges. In this section, we detail the findings obtained from the
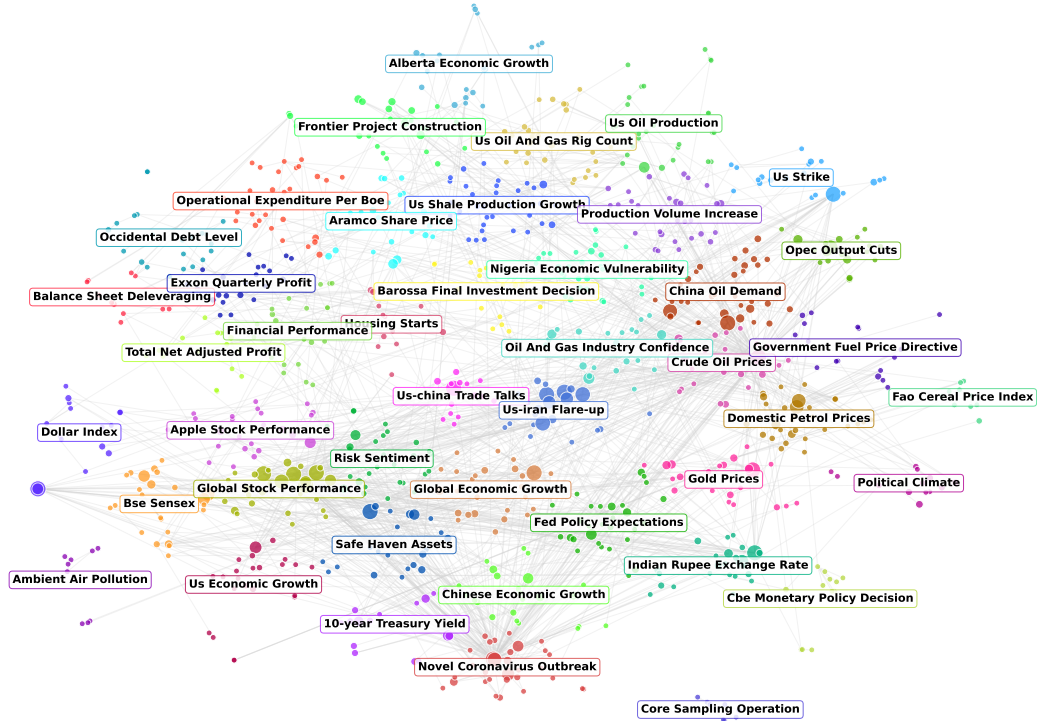
Figure 3: CausalWorld graph structure. Colors show the cluster in which the node belongs. Node size is based on the number of worlds a node appears in: the more a node appears, the bigger it is on the visualization. Labels correspond to the name of the most representative node of the cluster.

network. Figure 3 shows a visualization of the network. The resulting structure of CausalWorld is sparse, with a graph density of ~ 0.001. This is in accordance with the Sparse Mechanisms Shift (SMS) hypothesis, which states that a causal mechanism only sparsely affects other mechanisms [40]. Figure 11 in Appendix E further shows the distribution of strongly and weakly connected components, highlighting that the majority of the network is connected. Despite this structure, few feedback loops are observed. The network encompasses diverse knowledge and can be divided into 44 topic clusters like `Gold Prices` or `US Oil Production`, as shown in Figure 3. Furthermore, 109 structural communities can be found, communicating with other communities with a small set of bridge nodes, e.g.: `Crude Oil Prices`, `Novel Coronavirus Outbreak`, `Global Economic Growth` and `US-China Trade Talks`. These nodes are key elements of the graph that enable the propagation of information from one community to the next. More details are provided in Appendix E.

4.1 The CausalWorld Network combines Many Overlapping Worlds

The main difference between CausalWorld and other repositories of causal relationships is the presence of **worlds**, as described in Section 3. Each world instantiates a variable by providing it with a value and context. It is the key to match worlds and perform counterfactuals (discussed in Section 5). However, worlds must not be isolated and communicate via nodes shared by multiple worlds. 37% of the nodes in CausalWorld share two worlds or more, allowing information to be propagated across worlds. Since the majority of the nodes are linked by bridge nodes, most nodes can be used to compute counterfactuals.

4.2 Causal Insights from Structure: "Biofuel Demand Impacts Food Prices"

Interesting findings that can be extracted from the network structure are causal paths established by the causal extraction process. We provide an example of a causal chain in Figure 4. We can observe causal relationships between nodes extracted from different documents, e.g. "Palm oil Prices" belongs to world 119 but "Food Prices" connects it to worlds 58, 68, and 70. It illustrates a major
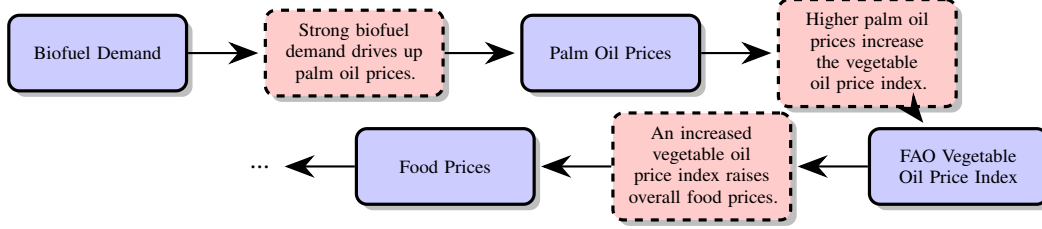
Figure 4: Illustration of a direct causal path in the CausalWorld graph. Nodes are blue boxes. Arrows represent causal dependencies. The description of the dependency is shown in dashed red boxes. Note that all nodes except for the root can have additional causal parents not shown in the chain and that the strength and function related to the causal relationships are not shown.
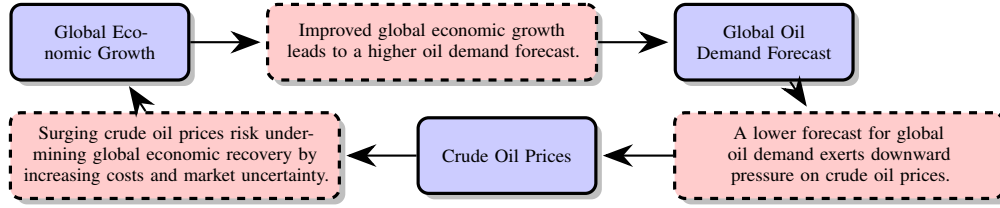

Figure 5: Illustration of a cycle in the CausalWorld graph. The legend is the same as in Figure 4.

property of CausalWorld: it allows inference to be performed between long-range dependencies across multiple sources while maintaining information sparsity. CausalWorld also allows for cycles in the graph and can, thus, represent feedback loops, as illustrated in Figure 5. More examples are provided in Appendix F.

5   Counterfactual Inference in the Real World

We now investigate how to take advantage of the CausalWorld graph to build a counterfactual reasoning evaluation dataset (CausalWorld-CR). We also introduce the *Causal Cartographer Reasoning agent* (CTG-Reason) and evaluate it on the generated causal inference tasks.

(a) Overview of the counterfactual matching and evaluation.

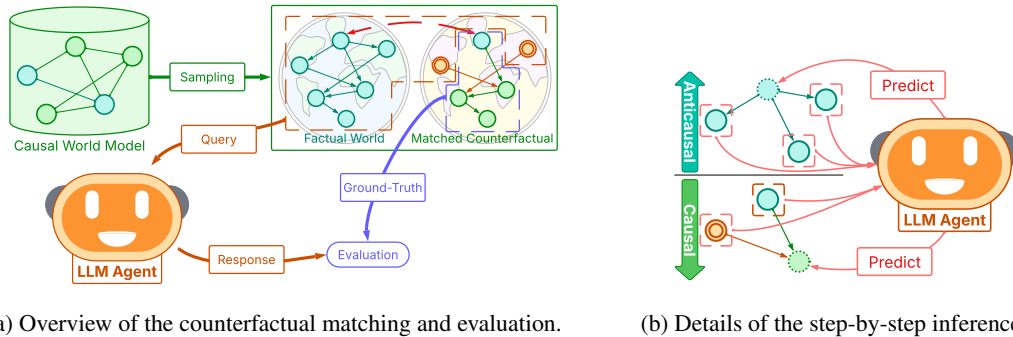(b) Details of the step-by-step inference.

Figure 6: Overview of the *Causal Cartographer Reasoning agent* (CTG-Reason) and the evaluation pipeline. (a) A query is sampled from the CausalWorld graph using counterfactual matching: values in the counterfactual world that match those in a factual world are abducted and replaced with information from the factual world. The agent performs counterfactual inference from the resulting graph. (b) The agent performs step-by-step reasoning by predicting the value of a child variable given its parents (causal direction), resp. predicting the value of a parent given its children (anticausal).

5.1 Counterfactual Matching

We aim to infer the value of a target causal variable given observations or counterfactual evidence. To build such queries from CausalWorld, we use **counterfactual matching**. In this section, we define this concept and show how it allows building real world counterfactuals. First, we must establish a **causal blanket**, i.e. a set of variables that, if known, fully determine the target variable. The causal blanket differs from a Markov blanket [34] because it only includes direct paths to the target variable and does not necessarily require the nodes in the blanket to be the target's parents. For example, in the chain $A \rightarrow B \rightarrow C \rightarrow D$, A forms a causal blanket for C. We assume that we can construct causal blankets from the CausalWorld graph. We formally define this concept in Definition 1.

Definition 1 (Causal Blanket). *Let $\mathcal{G}$ be a directed graphical model over a set of random variables $\mathcal{V}$, and let $T \in \mathcal{V}$ be a target variable. $\mathbf{anc}_{\mathcal{G}}(T)$ is the set of ancestors of T with respect to $\mathcal{G}$. A set of variables $\mathcal{B} \subseteq \mathcal{V} \setminus \{T\}$ is called a* causal blanket *for T in $\mathcal{G}$ iff, $\mathcal{B} \subseteq \mathbf{anc}_{\mathcal{G}}(T)$ and conditioned on $\mathcal{B}$, the target variable T is fully determined; that is, there exists a deterministic function f such that*

$$T = f(\mathcal{B}).$$

Equivalently, knowing $\mathcal{B}$ renders T conditionally independent of all other variables in $\mathcal{V} \setminus (\mathcal{B} \cup \{T\})$. Unlike the Markov blanket, a causal blanket requires that $\mathcal{B}$ contains only variables with direct causal paths to T, but these need not be limited to the parents of T. It follows that a variable can have multiple causal blankets.

It is not generally possible to obtain the true value of a counterfactual in the real world. Therefore, we focus on a subset of counterfactual queries accessible via **K-Matching**: finding two worlds within our observations that can act as factual and counterfactual worlds. I.e. observing the factual world and intervening on K observed variables of the counterfactual world is equivalent to observing the counterfactual world. Figure 6a illustrates the idea. We define this concept more formally below:

Definition 2 (K-Matching). *Let $\mathcal{O}_o$ and $\mathcal{O}_c$ denote two sets of observations sampled from $\mathcal{V}$. Suppose that a target variable T is present in both sets and that there exists a subset of shared observed observations $\mathcal{O}_s = \mathcal{O}_o \cap \mathcal{O}_c$ with $\mathcal{O}_s \neq \emptyset$. Let $\mathcal{B}_c \subseteq \mathcal{O}_c$ denote a causal blanket for T, with $|\mathcal{B}_c| = N$. We say that $\mathcal{B}_c$ can be K-matched with $\mathcal{O}_o$ over T if it is possible to build a new causal blanket for T with K interventions from $\mathcal{B}_c \setminus \mathcal{O}_s$ and $N - K$ observations from $\mathcal{O}_s$.*

Theorem 1 (K-Matching Equivalence). *Suppose that a causal blanket $\mathcal{B}_c$ is K-matched with $\mathcal{O}_o$ over a variable T and that $\mathcal{O}_o \setminus \mathcal{O}_s$ forms a causal blanket over each variable of $\mathcal{O}_s$. Then, observing $\mathcal{B}_c$ or observing $\mathcal{O}_o \setminus \mathcal{O}_s$ and intervening on $\mathcal{B}_c \setminus \mathcal{O}_s$ yields the same distribution for T; that is,*

$$P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_o \setminus \mathcal{O}_s) = P(T \mid \mathcal{B}_c).$$

The proof is given in Appendix C. This theorem implies that, if we can find a causal blanket for the counterfactual world and determine the value of $N - K$ variables of the blanket from the factual world, then we can build a counterfactual with K interventions over the remaining variables.

5.2 Matching with the CausalWorld-CR Dataset

The CausalWorld-CR dataset is divided into two query types: *observations* and *counterfactuals*. For both subsets, the inference task to solve consists of inferring the value of a target causal variable given causal ancestors. **Observation queries** are created by sampling target variables and their causal blanket from a single world. The reasoning model is evaluated by inferring the value of the target variable given the values of the blanket variables. Since the query is built from a single world, the ground-truth is provided by the source document from which the world is built. The model must answer a query of the type $P(\text{Target} \mid \mathcal{B}_o)$. We use K-matching to build **counterfactual queries**. The model must answer a query of the type $P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_o \setminus \mathcal{O}_s)$. We focus on 1-matching in our experiments.

By default, the strategy defined above generates highly unbalanced datasets. The nodes that have a high degree and are present in multiple worlds are over-represented. We balance the dataset to reduce these effects and generate a dataset of 400 samples (see Appendix G for details).

5.3 CTG-Reason

We build upon the methodology proposed by Gendron et al. [11] to create a graph-enhanced causal inference agent. The agent follows the steps of causal inference described in Section 2. It performs

step-by-step causal reasoning by computing the value of a variable from its direct causal parents only, or when anticausal reasoning is needed, from its direct children. At each step, the LLM's context only has access to the required parent/children variables. This approach, illustrated in Figure 6b, respects causal constraints and ensures that the agent only uses causal information for its reasoning, increasing efficiency and robustness by alleviating dependencies on non-causal and spurious correlations. An complete overview with the extraction agent is also shown in Figure 9 of Appendix B. The agent executes the plan described on the next page.

1. **Abduction:** If a twin graph is provided (with factual and counterfactual worlds), the agent performs abduction in an anticausal manner and computes the value of the exogenous variables of the factual world from their children. If their children are not observed, they are inferred from their own children *recursively*. The values for the exogenous variables are transferred to the counterfactual world.

2. **Intervention:** The counterfactual world is intervened upon: the incoming edges to all intervened variables are removed and the node values are fixed by the intervention value.

3. **Prediction:** The target value is inferred from its parents. If they are not observed, they are inferred from their own parents *recursively*. Since the set of observed and intervened variables forms a causal blanket, the target value is fully explained.

5.4 Evaluation

We perform experiments with widely used LLMs tailored for reasoning (o3-mini [28]) and general purpose tasks (GPT-4.1 [29] and LLaMA-3.1-8B [25]). We compare our step-by-step strategy against CausalCoT [16], a chain-of-thought prompting strategy [43] for causal inference. Since the evaluation queries are built from automatically scrapped real-world events, they express a diversity of types. The target variables to be predicted can be boolean, numerical or qualitative assessments of a trend. Moreover, the LLM may not provide a response corresponding to the same type. For instance, we find that LLMs tend to favor qualitative to numerical assessments (see Appendix I for more details).

Performance We look at the accuracy for boolean and trend queries as they correspond to categorical variables (true/false and increasing/decreasing/stable, respectively). Figure 7a shows that both strategies yield similar results, although CTG-Reason achieves slightly better accuracy. We also note that LLaMA-3.1 was not able to complete the queries with the CausalCoT strategy. We discuss this aspect in the next paragraph. Figure 7b shows the distribution of the relative error between the ground truth and the prediction for numerical counterfactual queries (in % of the ground truth value), for o3-mini and GPT-4.1. We look at the relative error instead of the absolute difference to take into account the unit difference between queries. We also excluded outliers ($\sim$4% of the answers were nonsensical numbers). We observe that CTG-Reason and CausalCoT strategies yield similar distributions for GPT-4.1 but that CTG-Reason achieves much lower error with o3-mini (curiously, for observation queries, CoT achieves better performance, see Figure 23b of Appendix I). Finally, we take a look at the semantic similarity between predicted and ground-truth answers in Figure 7c. Since CTG-Reason does not have access to non-causal contextual information, its answer wording greatly differs from the ground-truth, although this is not indicative of an incorrect answer.

Efficiency We study the efficiency of our proposed model in Figure 8. While CTG-Reason decomposes the problem into multiple steps and requires several LLM calls to solve a problem, this is balanced by a significantly lower quantity of retries. It can be explained by the restriction of the context to causal components, which reduces the scope of the problem and the size of the context window, making the problem *tractable*. As seen in the previous paragraph, this does not come at the cost of a reduced performance and can greatly help small models, as illustrated with LLaMA-3.1-8B: its context size is reduced by 72% and the output size is reduced by 91%.

6 Related Work

Traditional causal extraction methods rely on knowledge-based or statistical methods which can be brittle when causal relationships spread across multiple sentences or paragraphs [46]. More recent approaches rely on deep learning architectures and transformers, notably CausalBERT [20] which relies on BERT [6], but the use of larger LLMs remains under-explored. CauseNet [13] is
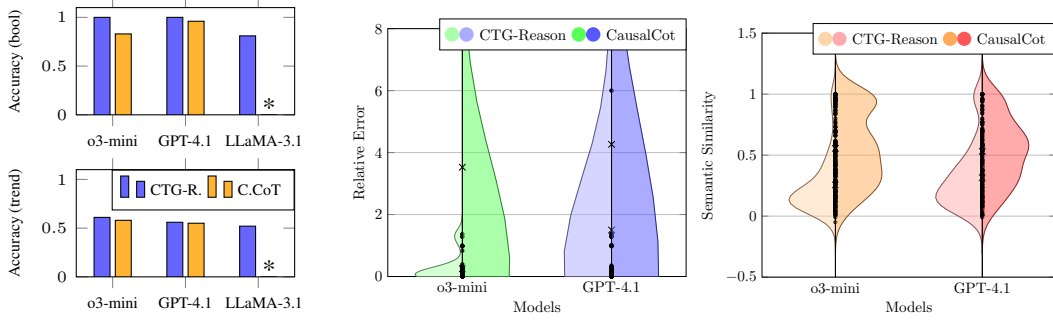
(a) Accuracy for bool/trend queries. (b) Relative error for num. queries. (c) Cosine similarity for text queries.

Figure 7: (a) Results on the boolean and trend subsets of CausalWorld-CR. Results are shown for o3-mini, GPT-4o and LLaMA-3.1-8B, using CTG-Reason (left bar/half) and CausalCoT (right bar/half). (b) and (c) Violin plots of the relative error (in %) and semantic (cosine) similarity between numerical/text ground truth and predicted answers on the counterfactual set. (*) The majority of queries with LLaMA-3.1-8B-CausalCoT returned with a timeout.
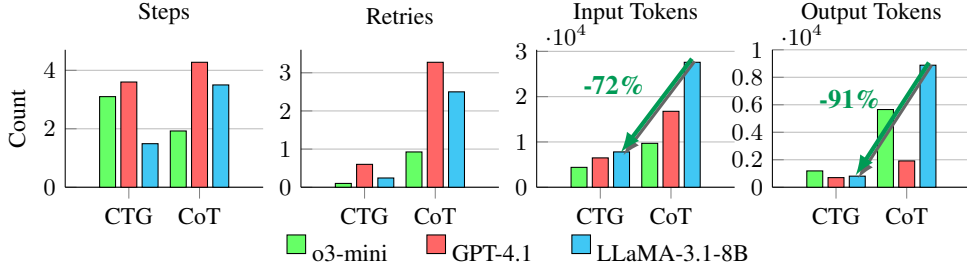
Figure 8: Statistics on the model answers with CTG-Reason and CausalCoT. (left) The average number of steps (i.e. model calls) required to solve a query. (middle left) The average number of retries after a failure to give a response (i.e. incorrect code formatting). (middle right) The average number of input tokens for the model. (right) The average number of output tokens.

a database of causal relationships extracted from the web by mining linguistic patterns, but it does not make the distinction between variable and value that allows us to do counterfactual matching. Our NxGraphRAG method also differs from existing graph-RAG systems (e.g. GraphRAG [7]) as we do not use knowledge graphs but causal graphs. Similarly to Li et al. [23] and Buehler [4], we integrate the graph-RAG approach to iteratively build a graph [35]. Studies have evaluated the causal reasoning abilities of LLMs, notably [48], [17] and Cladder [16]: a comprehensive benchmark for evaluating causal reasoning in language. Unlike in our work however, Cladder is built synthetically.

7 Limitations

The iterative nature of the causal extraction implies that the processing order of the documents has an impact on the final causal network. This is not a desired behavior as the causal relationships should be order-agnostic, potentially leading to a suboptimal configuration of the network. Moreover, since our counterfactual reasoning framework relies on causal blankets, we must assume knowledge of the full causal graph. It is not possible to guarantee that this is the case in the real world. Numerical estimations are also challenging for LLMs, which often output biased results [26], limiting the interpretation of a direct comparison. While our method aims to reduce bias and improve robustness by retaining causal relationships instead of (potentially spurious) correlations, we rely on the accuracy and honesty of the source data and are sensitive to adversarial attacks and misinformation injections. We discuss further the broader impact and ethical considerations of our work in Appendix A.

8 Conclusion

We introduce the **Causal Cartographer**, a twofold framework composed of causal extraction and inference agents that learn causal knowledge from natural language. We use this framework to build a network of causal knowledge and prove that it allows the estimation of real-world counterfactuals. We show that our proposed step-by-step inference agent can outperform chain-of-thought baselines on counterfactual reasoning and greatly reduce the inference cost by alleviating the impact of non-causal information. We hope that our work will inspire the creation of more robust and efficient reasoning agents based on causal principles. In particular, allowing agents to learn from counterfactual information is a promising direction towards building systems of higher cognition.

References

[1] Elias Bareinboim, Juan D. Correa, Duligur Ibeling, and Thomas Icard. On pearl's hierarchy and the foundations of causal inference. In Hector Geffner, Rina Dechter, and Joseph Y. Halpern (eds.), *Probabilistic and Causal Inference: The Works of Judea Pearl*, volume 36 of *ACM Books*, pp. 507–556. ACM, 2022. doi: 10.1145/3501714.3501743. URL `https://doi.org/10.1145/3501714.3501743`.

[2] Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on "a is b" fail to learn "b is a". In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=GPKTIktA0k`.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[4] Markus J. Buehler. Agentic deep graph reasoning yields self-organizing knowledge networks. *CoRR*, abs/2502.13025, 2025. doi: 10.48550/ARXIV.2502.13025. URL `https://doi.org/10.48550/arXiv.2502.13025`.

[5] Sirui Chen, Mengying Xu, Kun Wang, Xingyu Zeng, Rui Zhao, Shengjie Zhao, and Chaochao Lu. CLEAR: can language models really understand causal graphs? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pp. 6247–6265. Association for Computational Linguistics, 2024. URL `https://aclanthology.org/2024.findings-emnlp.363`.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1423. URL `https://doi.org/10.18653/v1/n19-1423`.

[7] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *CoRR*, abs/2404.16130, 2024. doi: 10.48550/ARXIV.2404.16130. URL `https://doi.org/10.48550/arXiv.2404.16130`.

[8] Gaël Gendron, Michael Witbrock, and Gillian Dobbie. A survey of methods, challenges and perspectives in causality. *CoRR*, abs/2302.00293, 2023. doi: 10.48550/ARXIV.2302.00293. URL `https://doi.org/10.48550/arXiv.2302.00293`.

[9] Gaël Gendron, Qiming Bao, Michael Witbrock, and Gillian Dobbie. Large language models are not strong abstract reasoners. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pp. 6270–6278. ijcai.org, 2024. URL `https://www.ijcai.org/proceedings/2024/693`.

[10] Gaël Gendron, Bao Trung Nguyen, Alex Yuxuan Peng, Michael J. Witbrock, and Gillian Dobbie. Can large language models learn independent causal mechanisms? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference*

on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024, pp. 6678–6701. Association for Computational Linguistics, 2024. URL `https://aclanthology.org/2024.emnlp-main.381`.

[11] Gaël Gendron, Joze M. Rozanec, Michael Witbrock, and Gillian Dobbie. Counterfactual causal inference in natural language with large language models. *CoRR*, abs/2410.06392, 2024. doi: 10.48550/ARXIV.2410.06392. URL `https://doi.org/10.48550/arXiv.2410.06392`.

[12] Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *CoRR*, abs/2011.15091, 2020. URL `https://arxiv.org/abs/2011.15091`.

[13] Stefan Heindorf, Yan Scholten, Henning Wachsmuth, Axel-Cyrille Ngonga Ngomo, and Martin Potthast. Causenet: Towards a causality graph extracted from the web. In Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (eds.), *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pp. 3023–3030. ACM, 2020. doi: 10.1145/3340531.3412763. URL `https://doi.org/10.1145/3340531.3412763`.

[14] Paul W Holland. Statistics and causal inference. *Journal of the American statistical Association*, 81(396):945–960, 1986.

[15] Jinyan Hu, Kai-Hua Wang, Chi Wei Su, and Muhammad Umar. Oil price, green innovation and institutional pressure: A china's perspective. *Resources Policy*, 78:102788, 2022.

[16] Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng Lyu, Kevin Blin, Fernando Gonzalez Adauto, Max Kleiman-Weiner, Mrinmaya Sachan, and Bernhard Schölkopf. Cladder: A benchmark to assess causal reasoning capabilities of language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL `http://papers.nips.cc/paper_files/paper/2023/hash/631bb9434d718ea309af82566347d607-Abstract-Conference.html`.

[17] Zhijing Jin, Jiarui Liu, Zhiheng Lyu, Spencer Poff, Mrinmaya Sachan, Rada Mihalcea, Mona T. Diab, and Bernhard Schölkopf. Can large language models infer causation from correlation? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=vqIH0ObdqL`.

[18] Thomas Jiralerspong, Xiaoyin Chen, Yash More, Vedant Shah, and Yoshua Bengio. Efficient causal graph discovery using large language models. *CoRR*, abs/2402.01207, 2024. doi: 10.48550/ARXIV.2402.01207. URL `https://doi.org/10.48550/arXiv.2402.01207`.

[19] Nitish Joshi, Abulhair Saparov, Yixin Wang, and He He. Llms are prone to fallacies in causal inference. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 10553–10569. Association for Computational Linguistics, 2024. URL `https://aclanthology.org/2024.emnlp-main.590`.

[20] Vivek Khetan, Roshni R. Ramnani, Mayuresh Anand, Shubhashis Sengupta, and Andrew E. Fano. Causal BERT: language models for causality detection between events expressed in text. In Kohei Arai (ed.), *Intelligent Computing - Proceedings of the 2021 Computing Conference, Volume 1, SAI 2021, Virtual Event, 15-16 July, 2021*, volume 283 of *Lecture Notes in Networks and Systems*, pp. 965–980. Springer, 2021. doi: 10.1007/978-3-030-80119-9_64. URL `https://doi.org/10.1007/978-3-030-80119-9_64`.

[21] Gregor Leban, Blaz Fortuna, Janez Brank, and Marko Grobelnik. Event registry: learning about world events from news. In Chin-Wan Chung, Andrei Z. Broder, Kyuseok Shim, and Torsten Suel (eds.), *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume*, pp. 107–110. ACM, 2014. doi: 10.1145/2567948.2577024. URL `https://doi.org/10.1145/2567948.2577024`.

[22] 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 Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual*

Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html`.

[23] Dawei Li, Shu Yang, Zhen Tan, Jae Young Baik, Sukwon Yun, Joseph Lee, Aaron Chacko, Bojian Hou, Duy Duong-Tran, Ying Ding, Huan Liu, Li Shen, and Tianlong Chen. DALK: dynamic co-augmentation of llms and KG to answer alzheimer's disease questions with scientific literature. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pp. 2187–2205. Association for Computational Linguistics, 2024. URL `https://aclanthology.org/2024.findings-emnlp.119`.

[24] Chengzhi Mao, Kevin Xia, James Wang, Hao Wang, Junfeng Yang, Elias Bareinboim, and Carl Vondrick. Causal transportability for visual recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 7511–7521. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00737. URL `https://doi.org/10.1109/CVPR52688.2022.00737`.

[25] Meta. Introducing llama 3.1: Our most capable models to date, 2024. URL `https://ai.meta.com/blog/meta-llama-3-1/`. Accessed: 2025-05-15.

[26] Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

[27] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. Text and code embeddings by contrastive pre-training. *CoRR*, abs/2201.10005, 2022. URL `https://arxiv.org/abs/2201.10005`.

[28] OpenAI. Openai o3-mini, 2025. URL `https://openai.com/index/openai-o3-mini/`. Accessed: 2025-05-05.

[29] OpenAI. Introducing gpt-4.1 in the api - openai, 2025. URL `https://openai.com/index/gpt-4-1/`. Accessed: 2025-05-13.

[30] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pp. 311–318. ACL, 2002. doi: 10.3115/1073083.1073135. URL `https://aclanthology.org/P02-1040/`.

[31] Judea Pearl. *Causality*. Cambridge university press, 2009.

[32] Judea Pearl. The do-calculus revisited. In Nando de Freitas and Kevin P. Murphy (eds.), *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012*, pp. 3–11. AUAI Press, 2012. URL `https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2330&proceeding_id=28`.

[33] Judea Pearl. The deductive approach to causal inference. *Journal of Causal Inference*, 2(2): 115–129, 2014.

[34] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[35] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *CoRR*, abs/2408.08921, 2024. doi: 10.48550/ARXIV.2408.08921. URL `https://doi.org/10.48550/arXiv.2408.08921`.

[36] Jonathan Richens and Tom Everitt. Robust agents learn causal world models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=pOoKI3ouv1`.

[37] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[38] Jože M Rožanec, Beno Šircelj, Peter Nemec, Gregor Leban, and Dunja Mladenić. Building a causality graph for strategic foresight. 2023.

[39] Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.

[40] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proc. IEEE*, 109 (5):612–634, 2021. doi: 10.1109/JPROC.2021.3058954. URL https://doi.org/10.1109/JPROC.2021.3058954.

[41] Sentence Transformers. sentence-transformers/all-mpnet-base-v2 · hugging face, 2024. URL https://huggingface.co/sentence-transformers/all-mpnet-base-v2. Accessed: 2025-05-05.

[42] Victor Troster, Muhammad Shahbaz, and Gazi Salah Uddin. Renewable energy, oil prices, and economic activity: A granger-causality in quantiles analysis. *Energy Economics*, 70:440–452, 2018.

[43] 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 Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

[44] Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 1819–1862. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.NAACL-LONG.102. URL https://doi.org/10.18653/v1/2024.naacl-long.102.

[45] Kevin Xia, Kai-Zhan Lee, Yoshua Bengio, and Elias Bareinboim. The causal-neural connection: Expressiveness, learnability, and inference. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 10823–10836, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/5989add1703e4b0480f75e2390739f34-Abstract.html.

[46] Jie Yang, Soyeon Caren Han, and Josiah Poon. A survey on extraction of causal relations from natural language text. *Knowl. Inf. Syst.*, 64(5):1161–1186, 2022. doi: 10.1007/S10115-022-01665-W. URL https://doi.org/10.1007/s10115-022-01665-w.

[47] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

[48] Matej Zecevic, Moritz Willig, Devendra Singh Dhami, and Kristian Kersting. Causal parrots: Large language models may talk causality but are not causal. *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=tv46tCzs83.
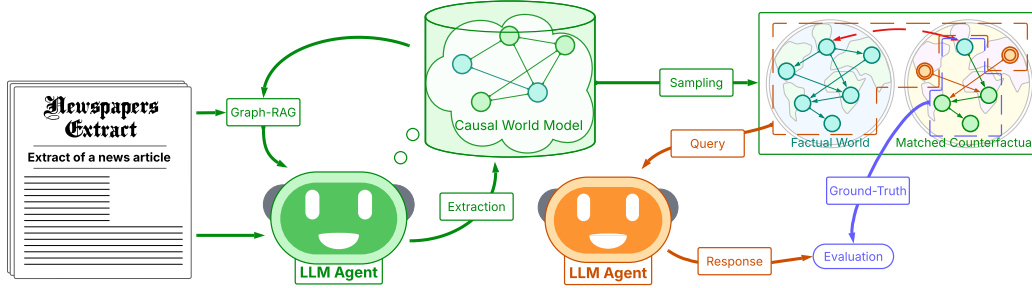
Figure 9: Overview of the complete Causal Cartographer pipeline with extraction and reasoning agents. The title, content and metadata from a news source is provided to the context of the extraction agent. It then generates causal variables and their relationships and adds them to the CausalWorld network, making explicit the causal knowledge of the framework. A graphRAG system further compares article data and agent node suggestions and adds them to the extraction agent's context to guide the extraction process. Then, ground-truth counterfactuals are sampled from the CausalWorld graph using matching: observations in the counterfactual world that match those in a factual world are removed and replaced with the factual world for abduction. The reasoning agent is evaluated on the generated real-world counterfactual queries.

A Broader Impact and Ethical Considerations

This work presents a method for building a repository of causal knowledge from real-world data in natural language, more specifically from news articles, and estimating counterfactual outcomes. We expect that it can help build better reasoning systems able to perform causal inference in natural language. The grounding in real-world data can particularly help improve the abilities of large language models on many downstream real-world tasks. Causal inference has been argued as a promising direction to reduce bias and increase fairness, trustworthiness and safety of AI systems [31, 12, 40, 1, 24, 8, 9, 10, 36] and we hope that this work can foster research in this direction and help bridge the gap with real-world applications.

However, we acknowledge that our work can also have negative impact. First, as mentioned in the limitations (Section 7), we rely on the accuracy and honesty of the source data and are not robust to adversarial attacks. Our method gives equal importance to all causal relationships, which means that a small amount of adversarial data can have a significant impact on the extraction and inference, exacerbating harmful trends and heavily affecting the downstream predictions. As a consequence, we do not recommend using our method on potentially unreliable sources. Second, our work can be used to make counterfactual predictions about real-world situations. While this work is only a first step in this direction, we expect that it will lead to economical and societal prediction engines. For instance, companies could make predictions about the economic impact of a businesses decision, or government organizations could predict the future impact of a policy on a population. Such usage can have very high impact over many areas and populations. It could provide guidance for enforcing better policies improving the well-being of populations but can also be used by malicious actors for harm. Even without the intervention of such actors, an over reliance on such prediction tools without critical analysis could similarly lead to disastrous effects in case of errors or misinterpretation of the prediction.

In its current stage, our work does not presents any such risks of misuse. However, we would like to emphasize the ethical considerations and risks links to the pursuit of this research direction and argue that future work should keep them into consideration.

B Overview of the Entire Causal Cartographer Pipeline

Figure 9 shows an unified representation of the Causal Cartographer with both extraction and reasoning agents. The CausalWorld network is the intermediate representation connecting the two aspects of the framework: representing the extracted knowledge and sampling counterfactual questions.

C Proof of Theorem 1

In this section, we prove the **K-Matching** theorem (Theorem 1) stated in Section 5.

Proof. We suppose that a causal blanket $\mathcal{B}_c$ is *K-matched* with $\mathcal{O}_o$ over a variable T and that $\mathcal{O}_o \setminus \mathcal{O}_s$ forms a causal blanket over each variable of $\mathcal{O}_s$. We aim to show that $P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_o \setminus \mathcal{O}_s) = P(T \mid \mathcal{B}_c)$, i.e. that a counterfactual query following this set of assumptions can be rewritten as an observational query.

We first rewrite the counterfactual query as a probability distribution over the set of exogenous variables $\mathbf{U}$, following Equation 1 [31]:

$$P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_o \setminus \mathcal{O}_s) = \sum_{U \in \mathbf{U}} P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), U) P(U \mid \mathcal{O}_o \setminus \mathcal{O}_s)$$

$\mathcal{B}_c = (\mathcal{B}_c \setminus \mathcal{O}_s) \cup \mathcal{B}_c$ forms a causal blanket over T. Therefore, $T \perp\!\!\!\perp \mathbf{U} \mid \mathcal{B}_c$. The only paths from $\mathbf{U}$ to T must go through the variables instantiating $\mathcal{O}_s$. We call them $\mathcal{S}$. We infer that $T \perp\!\!\!\perp \mathbf{U} \setminus \mathcal{S} \mid \mathcal{B}_c \setminus \mathcal{S}$. We can then rewrite the equation as follows:

$$P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_o \setminus \mathcal{O}_s) = \sum_{\mathcal{S} \in \mathbf{U}} P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{S}) P(\mathcal{S} \mid \mathcal{O}_o \setminus \mathcal{O}_s)$$

We now show that $\mathcal{O}_s$ is the only possible instantiation of $\mathcal{S}$. It is assumed that $\mathcal{O}_o \setminus \mathcal{O}_s$ forms a causal blanket over $\mathcal{S}$. Therefore, $\mathcal{O}_s$ can be deterministically computed from $\mathcal{O}_o$ and the probability $P(\mathcal{S} \mid \mathcal{O}_o \setminus \mathcal{O}_s)$ will return zero probability except for the values $\mathcal{O}_s$. the equation can then be rewritten as follows:

$$P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_o \setminus \mathcal{O}_s) = P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_s)$$

Rule 2 of *do-calculus* [31, 32] states that an intervention can be reduced to an observation if no backdoor path connects the intervened variable to the variable of interest. It can be written as follows:

$$P(y|do(x), do(z), w) = P(y|do(x), z, w) \text{ if } (Y \perp\!\!\!\perp Z|X, W)_{\mathcal{G}_{\overline{X}\underline{Z}}} \tag{2}$$

Since $do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_s$ forms a causal blanket over T, rule 2 applies and the equation can be further simplified into the desired quantity:

$$\begin{aligned} P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_o \setminus \mathcal{O}_s) &= P(T \mid do(\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_s) \\ &= P(T \mid (\mathcal{B}_c \setminus \mathcal{O}_s), \mathcal{O}_s) \\ &= P(T \mid \mathcal{B}_c) \end{aligned}$$

□

D Details on the NxGraphRAG Architecture

This section provides additional details regarding the graph retrieval-augmented generation pipeline introduced in Section 3. We introduce Networkx Graph Retrieval-Augmented Generation (Nx-GraphRAG), jointly used with the proposed Causal Cartographer Extraction agent (CTG-Extract). An overview of the method is shown in Figure 10. We use Langchain [3] to build the RAG pipeline. It is provided to the agent as a callable tool.
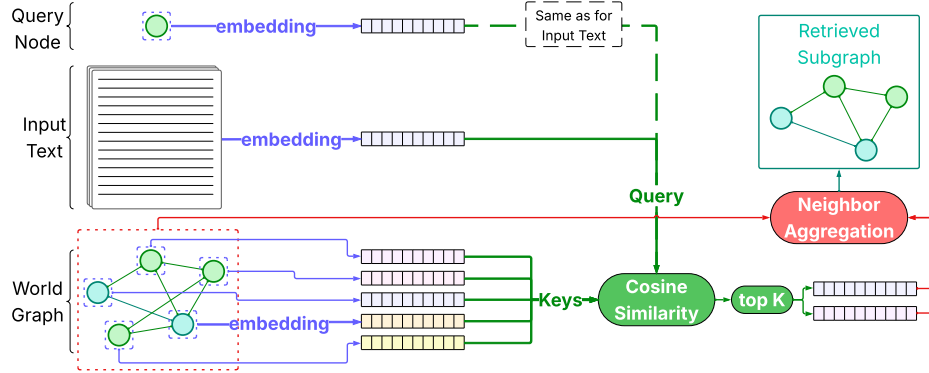
---

[3]https://www.langchain.com/

Figure 10: Overview of the NxGraphRAG pipeline. The attributes of the graph database are embedded and stored in a vector database. Depending on the use case, an input document or a candidate node are embedded as a query vector. The query vector is compared against the graph node (key) vectors using cosine similarity. The K closest embeddings are selected. For each node, its P-level neighbors are also selected. The subgraph is returned and added to the agent context.

NxGraphRAG generates embeddings for the world graph nodes, candidate nodes proposed by the agent, and input documents, that can be compared using cosine similarity. We use OpenAI Embeddings [27] to generate the embeddings. For the nodes, all the attributes are provided as a list of key-value character chains. The embeddings of the world graph represents the keys against which the queries, i.e. the candidate nodes and input documents, are compared with. While query vectors are generated on the fly, key vectors are stored in an vector database. We use an in-memory database because the generated world graph is small enough to fit in the memory (see Section 4 for more details). NxGraphRAG returns the top-K most similar nodes with the query. It also returns the neighbors of these nodes, up to a level P neighborhood. We use $K = 3$ and $P = 2$ to balance exhaustiveness and efficiency.

E   Details on the Structure of CausalWorld

Figure 11 shows the distribution of strongly and weakly connected components in the CausalWorld graph, highlighting that the majority of the network is connected and forms a single component. Despite this structure, few feedback loops are observed.

We now investigate the topics extracted from the news sources and represented in CausalWorld. We create clusters based on semantic embeddings. We only use the node attributes invariant to the current world. Due to the large scale of the graph, we use a smaller embedding model than in the NxGraphRAG pipeline. Since semantic clustering is only used for visualizing the graph content and is not connected to a downstream task, a downgraded performance would have no effect on the rest of our pipeline. We use all-mpnet-base-v2 [41] with the SentenceTransformers library [4] to generate the embeddings and K-Means to attribute clusters. We use the Silhouette score [37] to determine the optimal number of clusters and find that the network can be best divided into 44 clusters. They can be observed in Figure 3 of the main paper.

We further use the Louvain method [3] to detect communities within the graph based on structural information only. We discover 109 communities within the network. Eight nodes are bridge nodes separating four communities or more: `Crude Oil Prices`, `Crude Oil Prices`, `Novel Coronavirus Outbreak`, `Global Economic Growth`, `US-China Trade Talks`, `Middle East Unrest`, `US Strike`, `US-Iran Flare-up` and `US-Iran Tensions Easing`. These nodes are key elements of the graph that enable the propagation of information from one community to the next. Figure 13 illustrates the division of the network into communities.

Figure 14 shows the distribution of nodes among the worlds extracted from the input documents. We can observe that the majority of the nodes belong to a single world but that 37% of the nodes share

---

[4]https://www.sbert.net/

16

(a) Weakly Connected Components

(b) Strongly Connected Components

(c) Distribution of node in-degree
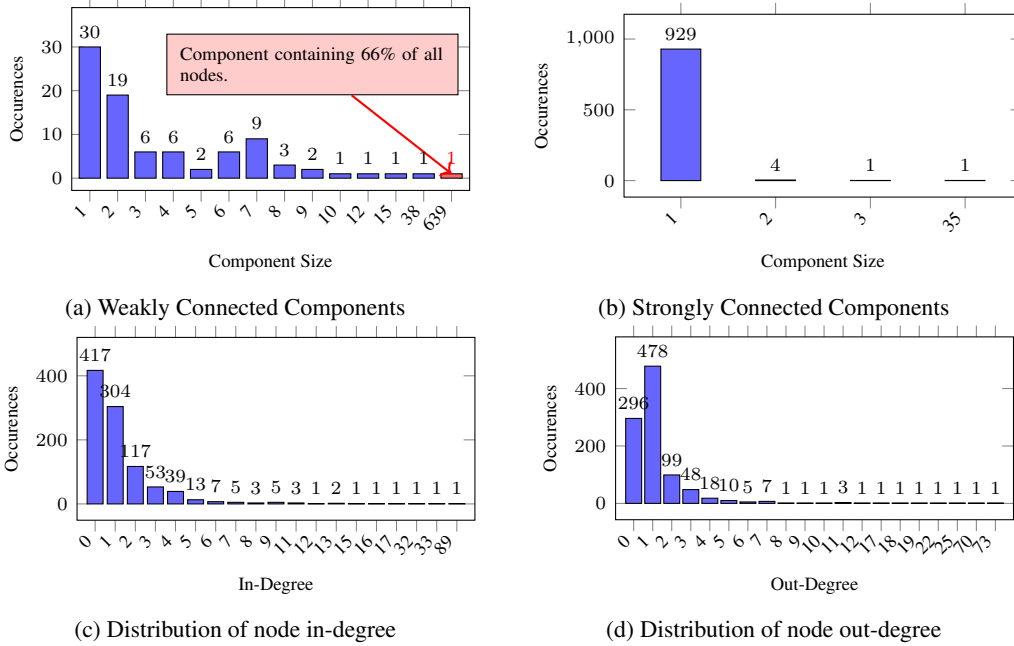
(d) Distribution of node out-degree

Figure 11: Distribution of weakly and strongly connected components and node degrees in Causal-World. The majority of nodes (66%) belong to one connected component (highlighted in red) while 3% are isolated and the others are in components of intermediate size. 5% of vertices are in non-trivial strongly connected components and are part on feedback loops. The majority of nodes have less than five neighbors, further highlighting the sparsity of the graph.
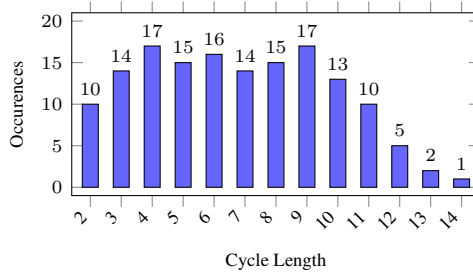
Figure 12: Distribution of cycles in CausalWorld. There are 149 cycles in the graph, with lengths comprised between two and 14.

two worlds or more, allowing information to be propagated across worlds. Since the majority of the nodes belong to the same component, most nodes can be used to compute counterfactuals.

F Additional Causal paths in CausalWorld

Figure 15 shows the longest causal chain that can be extracted from CausalWorld. Additional simplified chains are shown in Figure 16.

G Details on the CausalWorld Counterfactual Reasoning Dataset

The CausalWorld network allows matching counterfactuals and building samples for causal queries. However, as shown in Figure 11, the degree distribution in the graph is imbalanced, resulting in an over-representation of some nodes over others. We balance the dataset to mitigate this issue. In addition, in the case of long dependencies, some queries may present many causal paths between the observations and the target variable. We remove queries with too many possible causal paths
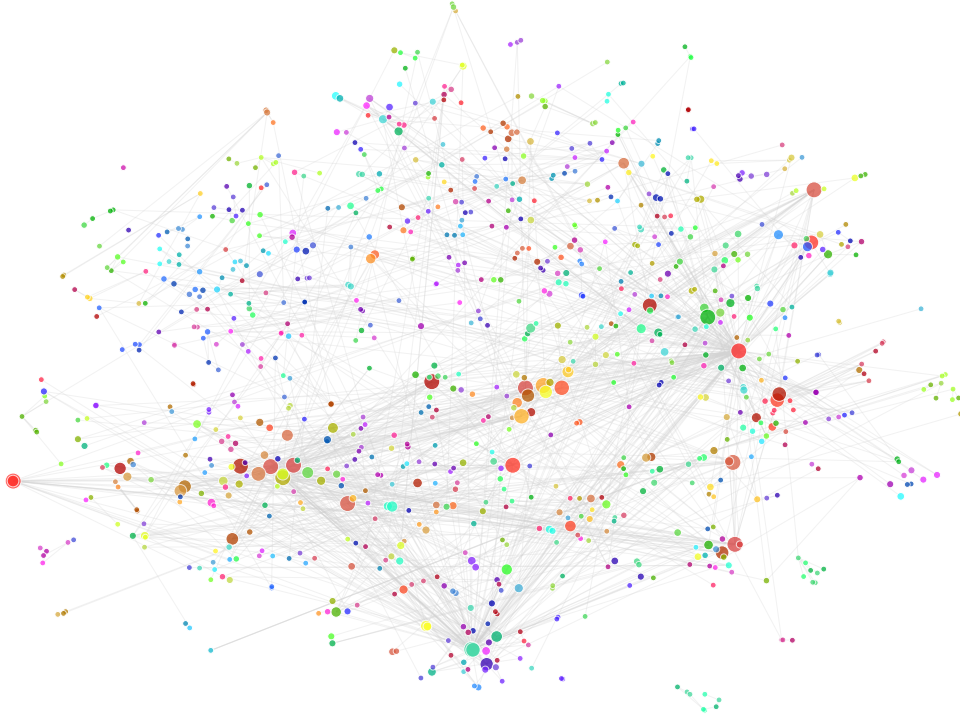
17

Figure 13: CausalWorld graph structure divided by Louvain communities. Colors correspond to the Louvain community in which belongs the node. Node size is based on the amount of worlds in which the node appear: more often a node appears, the bigger it is on the visualization. A total of 109 communities exists in the graph, therefore colors can correspond to several communities. This figure is for illustration purpose only.

(i.e. ≥ 50) to allow the problem to remain tractable. Figure 17 shows the distribution of query types and target domains in the CausalWorld-CR dataset. Figure 18 shows the number of nodes per query graph.

H   Implementation Details

In this section, we detail the implementation details used for our experiments.

For the causal extraction pipeline, we use OpenAI o3-mini-2025-01-31 [28] as our base LLM. Specific details for NxGraphRAG are given in Section D. We use the Smolagents library[5] as the interface with the LLM agents. We perform calls to o3-mini-2025-01-31 and gpt-4.1-2025-04-14 [29] using the LiteLLM interface with default hyperparameters. The order of magnitude for the total cost of the OpenAI API calls is $\sim$ USD 100. We run meta-llama/Llama-3.1-8B-Instruct [25] locally using the Transformers interface. Local experiments are run on eight NVIDIA A100-SXM4-80GB GPUs. The data analysis performed in Sections 4 and 5 involves semantic embedding. We perform the embedding using the all-mpnet-base-v2 [41] Transformer model. We use the SentenceTransformers library [6]. Data analysis experiments are performed on a single laptop with a 3.20 GHz AMD Ryzen 7 5800H CPU, 16GB RAM and a NVIDIA GeForce RTX 3070 Laptop GPU.
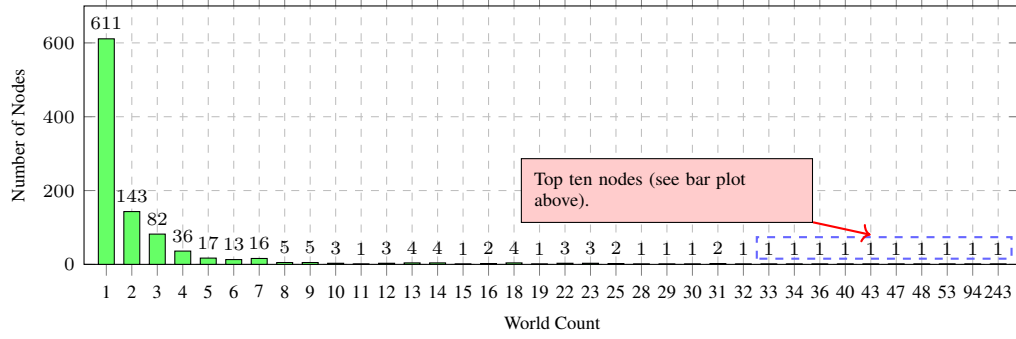
I   Additional Results on CausalWorld-CR

Figures 20 and 21 show the detailed results on the counterfactual and observation queries of CausalWorld-CR. The main results can be found in Figure 7. Figure 24 further shows the con-

---

[5]https://huggingface.co/docs/smolagents/

[6]https://www.sbert.net/
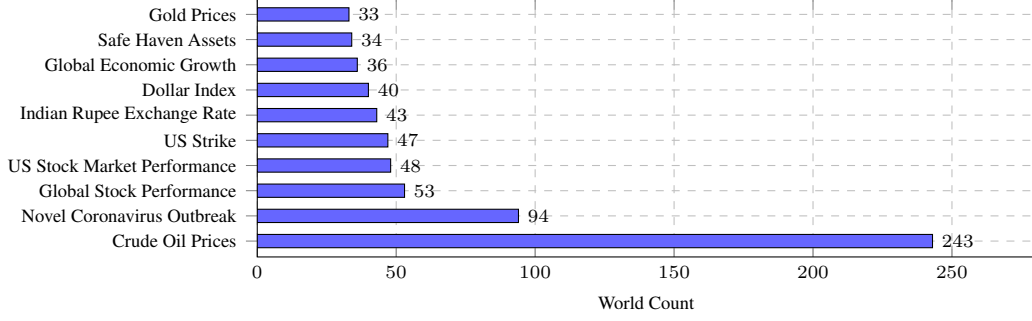
(a) Top ten nodes with the highest representation among saved worlds.

(b) Distribution of the nodes based on their number of appearances in worlds.

Figure 14: Distribution of the nodes among the worlds extracted from the source documents. The higher the world count, the most document a node appears in. The top plot shows the amount of worlds for the top ten nodes and the botom plot shows the distribution for the entire graph.

fusion matrices for the boolean and trend queries while Figures 22 and 23 show the distributions of the cosine similarity and relative error between the ground truth and the predictions, respectively. Figure 25 shows the BLEU scores [30] for o3-mini and GPT-4.1 on the counterfactual and observational sets. Higher scores indicates that model responses contains n-grams similar to the ones appearing in ground truth answers. While BLEU is suited for translation, in this settings, it indicates how close the answer formulation is compared to the original. A low score does not indicate that the answer is incorrect but that the grammatical elements that are used differ. As expected, score are indeed low but observational answers obtain slightly higher scores.

J   Agent Prompts

This section describes the system and user prompts used for the LLM agents in the extraction and inference systems, as well as the syntax for the causal variables and relationships in the causal network.

J.1   Causal Information Syntax

The syntax for the causal variables and relationships is provided in the LLMs' prompts as shown in the next sections. Additionally, it is also enforced using Pydantic.

J.1.1   Causal Variables

Causal variables are defined in the causal network as a dictionary of attributes. They are described to the LLM agents as follows:

```
{
    "name": <string>, # The name of the variable
    "description": <string>, # The description of the variable
    "type": <string>, # The type of the variable (boolean, integer, float, string, etc.)
```

19

Figure 15: Illustration of the longest direct causal path in the CausalWorld graph. It contains 18 nodes, represented in blue boxes. Arrows represent causal dependencies. The description of the dependency is shown in dashed red boxes. Note that all nodes except for the root can have additional causal parents not shown in the chain and that the strength and function related to the causal relationships are not shown.
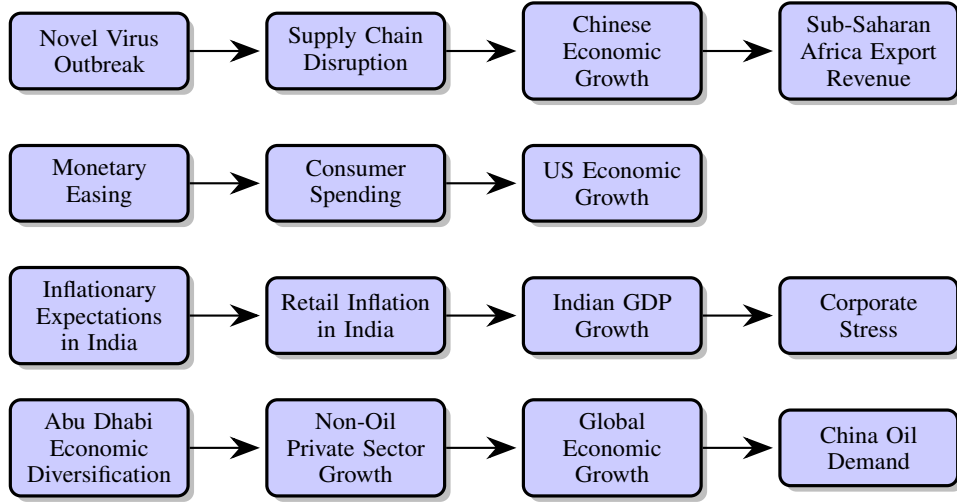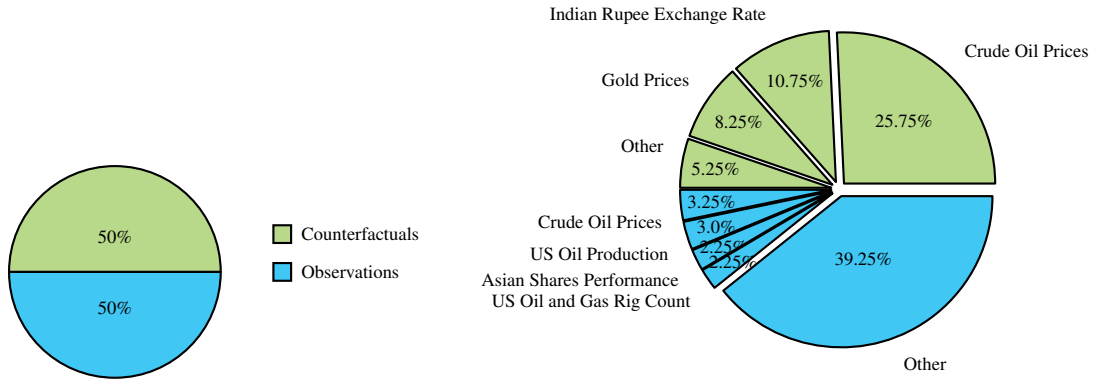
Figure 16: Additional direct causal paths in CausalWorld graph.

(a) Distribution of observation and counterfactual queries.

(b) Distribution of target variables in the observation and counterfactual splits.

Figure 17: Distribution of the query types and target domains in the CausalWorld-CR dataset. Counterfactual queries are more concentrated around specific domains than observations because they require to match several worlds for building the query.
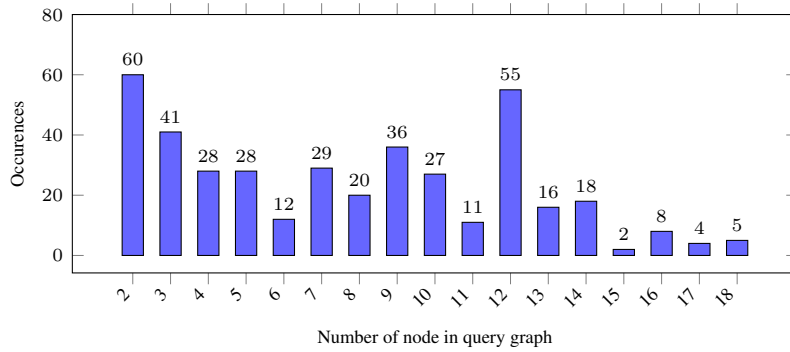

Figure 18: Distribution of the number of nodes in the query graphs of the CausalWorld-CR dataset.
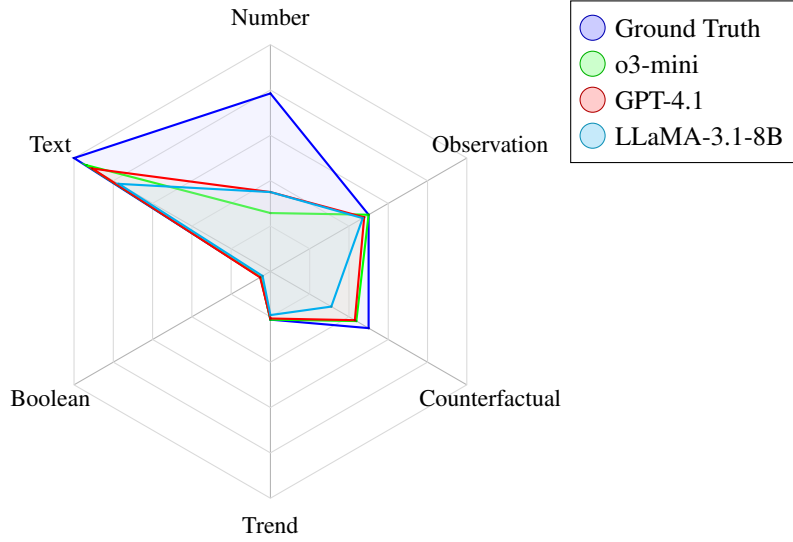
Figure 19: Distribution of query types and answer types from CTG-Reason. Types correspond to a category a sample can be evaluated in. Types are: *number* (answer is a number), *text* (answer is textual, matches all samples), *boolean* (true/false), *trend* (the qualitative assessment of a trend: increasing, decreasing, stable). *Observation* and *counterfactual* correspond to thequery types as defined i Section 5. A sample can cumulate multiple types and the model answer may not correspond to the same type as the ground truth. For instance, while the ground-truth has a numerical value, the model may provide a qualitative answer.
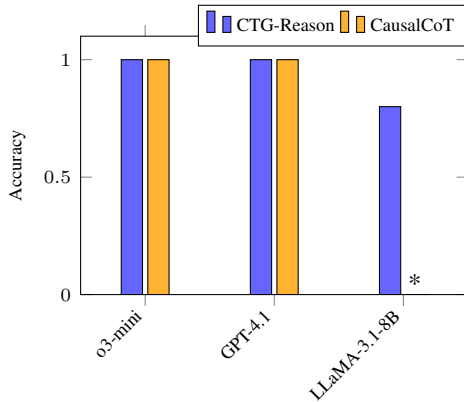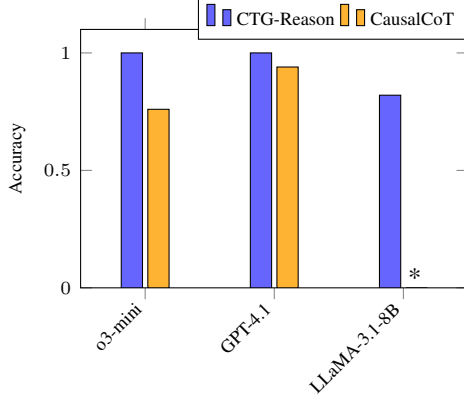

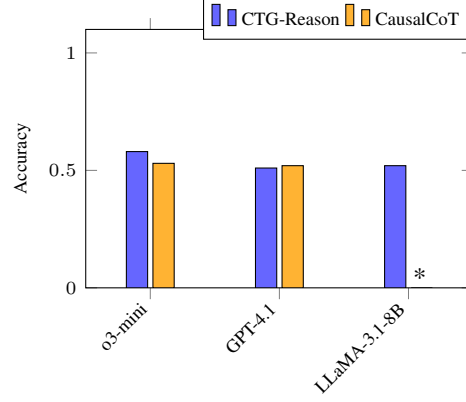(a) Accuracy for boolean queries.
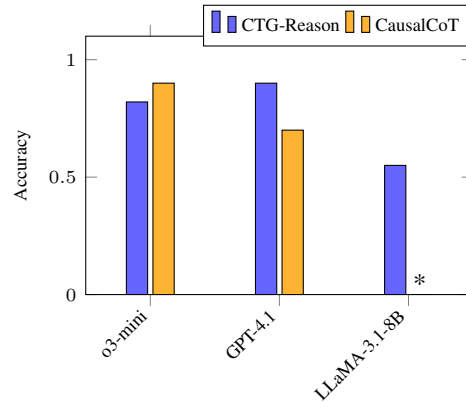
(b) Accuracy for trend queries.

Figure 20: Results on the *counterfactual* set for the boolean and trend queries of CausalWorld-CR. Results are shown for o3-mini, GPT-4o and LLaMA-3.1-8B, using CTG-Reason and CausalCoT. (*) The majority of queries with LLaMA-3.1-8B-CausalCoT returned with a timeout.
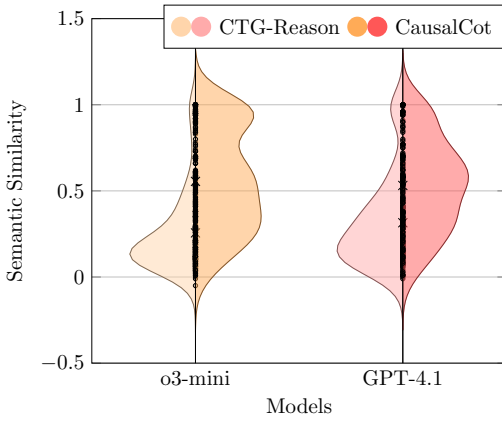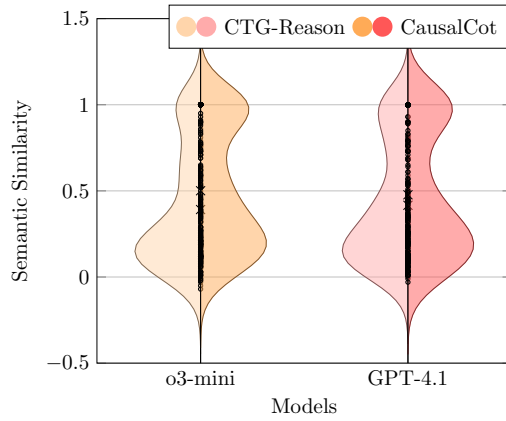
22

(a) Accuracy for boolean queries.

(b) Accuracy for trend queries.

Figure 21: Results on the *observational* set for the boolean and trend queries of CausalWorld-CR. Results are shown for o3-mini, GPT-4o and LLaMA-3.1-8B, using CTG-Reason and CausalCoT. (*) The majority of queries with LLaMA-3.1-8B-CausalCoT returned with a timeout.

(a) Counterfactual set.

(b) Observation set.

Figure 22: Violin plots of the semantic (cosine) similarity between ground truth and model answers. Results are shown for o3-mini and GPT-4.1.

```
    "values": <List[str]>, # The set or range of possible values for the variable ([1, 2, 3], 'range(0,10)
        ', ['low', 'medium', 'high'], 'True/False', 'natural numbers', etc.)
    "causal_effect": <string>, # The inferred causal effect of the variable
    "supporting_text_snippets": <Optional[List[str]]>, # The supporting text snippets in which the variable
        is mentioned
    "current_value": <Optional[string]>, # The observed current value of the variable
    "contextual_information": <Optional[string]>, # The contextual information associated with the current
        value of the variable
}
```

See Section 3 in the main paper for more details.

J.1.2 Causal Relationships

Similarly to causal variables, causal relationships are defined in the causal network as a dictionary of attributes. They are described to the LLM agents as follows:

```
{
    "cause": <string>, # The name of the cause variable
    "effect": <string>, # The name of the effect variable
    "description": <string>, # The description of the causal relationship between the variables
    "contextual_information": <Optional[string]>, # The contextual information associated with the causal
        relationship for the specific observed values of the variables
```
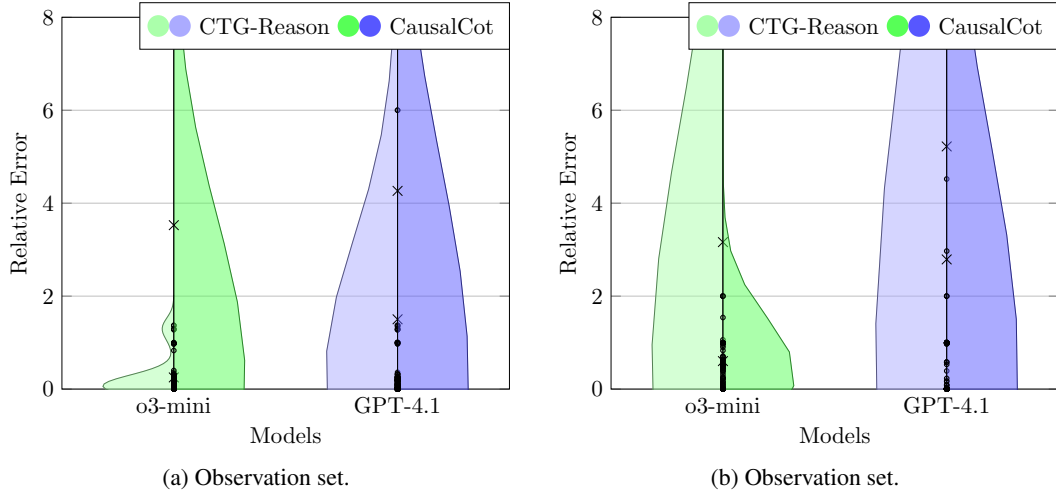
(a) Observation set.

(b) Observation set.

Figure 23: Violin plots of the relative error numerical between ground truth and prediction. The results are shown for o3-mini and GPT-4.1 for samples where both ground truth and predicted values are numerical. The plots show the distribution of the error in % of the ground truth value.

(a) Confusion matrix for boolean queries.

(b) Confusion matrix for trend queries.
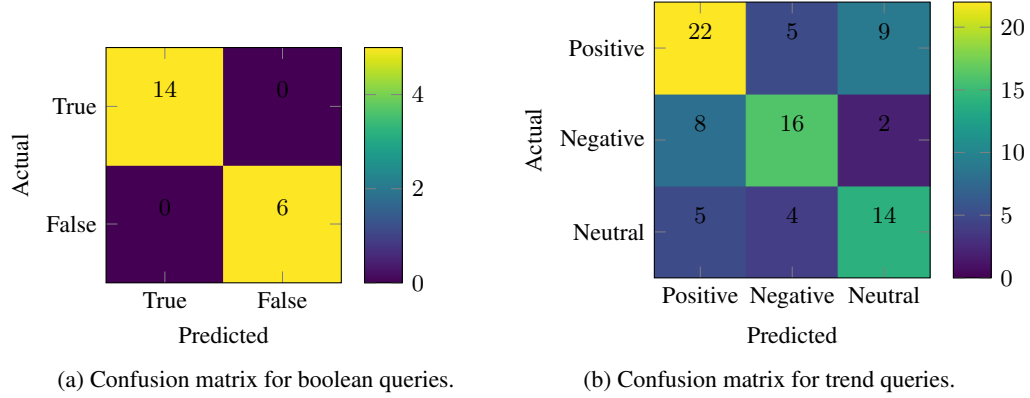
Figure 24: Confusion matrices for boolean and trend queries of CausalWorld-CR for o3-mini using CTG-Reason.
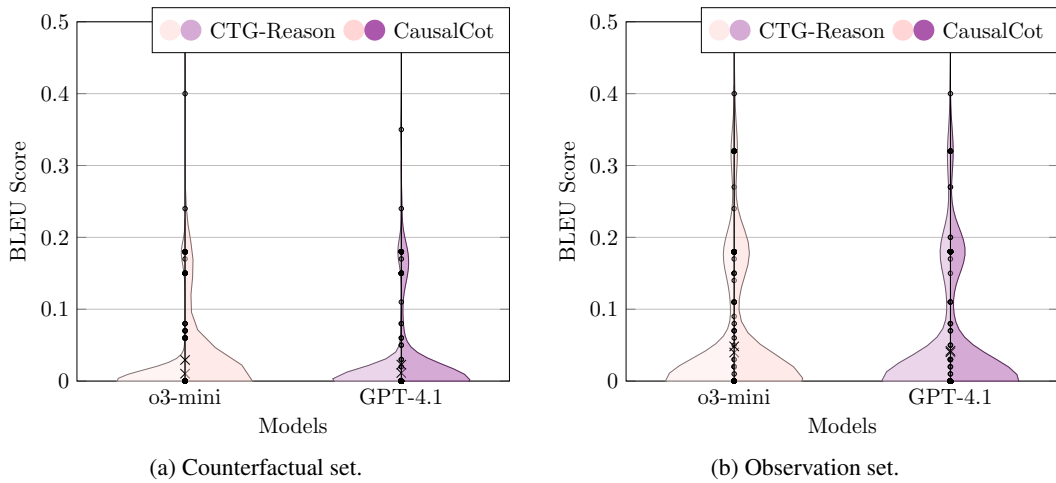

(a) Counterfactual set.

(b) Observation set.

Figure 25: Violin plots of the BLEU score for o3-mini and GPT-4.1 using CTG-Reason on counterfactual and observational query sets. A higher score indicates that the model prediction contains n-grams similar to the ones appearing in the ground truth answer.

```
    "type": <string>, # The type of the causal relationship (direct, indirect, etc.)
    "strength": <Optional[string]>, # The strength of the causal relationship
    "confidence": <Optional[string]>, # The confidence level in the existence of the causal relationship
    "function": <Optional[Callable]>, # The function that describes the causal relationship, if available.
}
```

J.2   Causal Extraction Agent Prompt

We show below the start of the causal extraction agent system prompt. It describes to the agent the extraction task to be performed and how to solve it. It is built upon on the ReAct framework [47] and the prompts provided by the Smolagents library [7].

```
You are an expert assistant who can solve any task using code blobs. You specialize into causal
    extraction tasks.
You will be given a text snippet and an initial causal graph. Your task will consist of finding the
    causal variables described in the text, the causal relationships that link them and adding them to
    the causal graph.
You will solve the task as best you can. To do so, you have been given access to a Python interpreter
    with the standard library and the networkx package.
You will also have access to an optional list of tools: these tools are basically Python functions which
    you can call with code.
You will use your expert reading comprehension, commonsense reasoning and coding skills to tolve the
    problem.
To solve the task, you must plan forward to proceed in a series of steps, in a cycle of 'Thought:', 'Code
    :', and 'Observation:' sequences.

At each step, in the 'Thought:' sequence, you should first explain your reasoning towards solving the
    task and the tools that you want to use.
Then in the 'Code:' sequence, you should write the code in simple Python. The code sequence must end with
    '<end_code>' sequence.
During each intermediate step, you can use 'print()' to save whatever important information you will then
    need.
These print outputs will then appear in the 'Observation:' field, which will be available as input for
    the next step.
In the end, you have to return a final answer using the `final_answer` tool. The output provided to the `
    final_answer` tool should be the networkx causal graph.

Each node should have the following dictionary of attributes:
{{variable}}
Some variables will have a value provided in the text, while others will be confounders that need to be
    estimated. Provide a current~value and contextual information whenever possible.

The causal relationships should be represented as directed edges between the nodes. Each edge should have
     the following dictionary of attributes:
{{causal_relationship}}

Your plan should be as follows:
1. Define the causal variables observed in the text. Use the variables provided when possible or create
    new ones when no variable matches.
2. Define the confounders that are not observed in the text or for which a value is not given, and that
    affect one or several of the causal variables defined in step 1.
3. Verify if the new variables have correspondance in the causal graph database. Use the `{{
    retrieval_tool_name}}` tool to assess if the variable is already in the database.
If it is, use it instead of creating a new one. It may have a different name in the database, the tool
    returns the top-k matching variables.
THIS IS A MANDATORY STEP. The variables provided in the input are only a subset of the variables in the
    database, you should always check if the variable already exist before creating new ones.
For each variable, use the one matching the most or create a new one if none matches.
4. Define the causal relationships between the variables, based on the text and common sense knowledge.
    Do not create causal relationships that already exist in the causal graph.
5. Build the full causal graph as a networkx DiGraph object.
Each step should be a separate 'Thought:', 'Code:', and 'Observation:' sequence.

The code MUST be executed in two code blocks minimum. After step 3, use <end_code> to indicate the end of
     the code block and retrieve the output of the tool call.
Then, use the observation to execute steps 4 and 5 and complete the task. An empty list of variables in
    the user input DOES NOT MEAN that there are no variables in the causal graph database.
Always check if the variable exists in the database before creating a new one.

[...]
```

The `variable` and `causal_relationship` placeholders are replaced with the prompts described in the previous section. Few-shots are also provided as part of the system prompt. Here is one example:

```
[...]

Here are a few examples tasks to solve.
---
Task:
"""
Increased Air Pollution Leads to Higher Respiratory Issues During Summer Months

A recent study has shown that cities experiencing high levels of air pollution are seeing a rise in
     respiratory issues among their populations, particularly during the summer months.
Experts attribute the increase to a combination of higher temperatures and pollution from industrial
     activities, which intensify during hotter months.
Dr. Laura Simpson, a pulmonologist, noted that individuals with pre-existing conditions, such as asthma,
     are at a greater risk of suffering from exacerbated symptoms.

Retrieved nodes:
<empty>

Retrieved edges:
<empty>
"""

Thought: I will first define the list of causal variables extracted from the text. These include:
1. Air Pollution - representing high levels of air pollution in cities.
2. Temperature - higher temperatures during the summer months.
3. Industrial Pollution - arising from intensified industrial activities during hotter months.
4. Respiratory Issues - the rise in respiratory issues among the population.
5. Pre-existing Respiratory Conditions - representing vulnerable individuals (e.g., those with asthma) at
     greater risk.

I will verify if these variables exist in our causal graph database using the tool graph_retriever for
     each variable by calling it with a query that includes the variable name. Then I will build the
     networkx DiGraph with these nodes and add the appropriate directed edges based on cause-effect
     relationships:
- Air Pollution → Respiratory Issues ("Increased air pollution leads to higher respiratory issues.")
- Temperature → Industrial Pollution ("Hotter temperatures intensify industrial activities' pollution.")
- Industrial Pollution → Respiratory Issues ("Pollution from industrial activities contributes to
     respiratory issues.")
- Pre-existing Respiratory Conditions → Respiratory Issues ("Individuals with pre-existing conditions are
      more vulnerable to respiratory issues.")

I will now call graph_retriever to check for each variable.

Code:
```py
import networkx as nx

# Define initial variables extracted from the text
causal_variables = [
{
    "name": "Air Pollution",
    "description": "The level of air pollutants present in the city environment.",
    "type": "float",
    "values": "range(0,100)",
    "supporting_text_snippets": ["cities experiencing high levels of air pollution"],
    "current_value": None,
    "contextual_information": "High in some cities as noted in the study."
},
{
    "name": "Temperature",
    "description": "Ambient temperature, particularly during the summer months.",
    "type": "float",
    "values": "range(20,50)",
    "supporting_text_snippets": ["higher temperatures", "summer months"],
    "current_value": None,
    "contextual_information": "Elevated during summer."
},


(truncated)


]

# Query the causal graph database for each variable using graph_retriever
for variable in causal_variables:
query_text = f"{variable['name']}: {variable['description']}"
result = graph_retriever(query=query_text)
print(f"Query for variable '{variable['name']}':", result, '\n')
```

```

```
print("End of variable verification.")
```<end_code>
Observation:
Query for variable 'Air Pollution': Retrieved nodes:
<empty>

Retrieved edges:
<empty>

Query for variable 'Temperature': Retrieved nodes:
<empty>

Retrieved edges:
<empty>


(truncated)


End of variable verification.

Thought: None of our causal variables exist in the database, so I will now build the full networkx DiGraph
        using our defined variables (as nodes) and add the causal relationships (as directed edges).
I will add the following edges:
- Air Pollution --> Respiratory Issues: "Increased air pollution leads to higher respiratory issues."
- Temperature --> Industrial Pollution: "Higher temperatures intensify industrial pollution from
        industrial activities."
- Industrial Pollution --> Respiratory Issues: "Pollution from industrial activities contributes to
        respiratory issues."
- Pre-existing Respiratory Conditions --> Respiratory Issues: "Individuals with pre-existing respiratory
        conditions are more vulnerable to respiratory issues."

I will now add these nodes and edges to the graph and return the final causal graph.

Code:
```py
# Build the full causal graph as a networkx DiGraph
G = nx.DiGraph()

# Add nodes for each variable and update their attributes separately
for variable in causal_variables:
var_name = variable["name"]
G.add_node(var_name)
G.nodes[var_name].update(variable)

# Define causal relationships as edges with descriptive attributes
causal_relationships = [
{
    "cause": "Air Pollution",
    "effect": "Respiratory Issues",
    "description": "Increased air pollution in cities leads to higher respiratory issues among populations
        .",
    "contextual_information": "Study shows a rise in respiratory issues in high pollution cities during
        summer.",
    "type": "direct",
    "strength": "high",
    "confidence": "high",
    "function": None
},
{
    "cause": "Temperature",
    "effect": "Industrial Pollution",
    "description": "Higher temperatures intensify industrial activities, leading to increased industrial
        pollution.",
    "contextual_information": "Pollution from industrial activities intensifies during hotter months.",
    "type": "direct",
    "strength": "moderate",
    "confidence": "high",
    "function": None
},


(truncated)


]

# Add edges from the causal_relationships list
for edge in causal_relationships:
cause = edge["cause"]
effect = edge["effect"]
```

```
G.add_edge(cause, effect)
G[cause][effect].update(edge) # /!\ This is how edge attributes should be added. They must be added after
    the edge is created.

# Output the final graph details for verification
print("Causal Graph Nodes:")
print(list(G.nodes(data=True)))
print("\nCausal Graph Edges:")
print(list(G.edges(data=True)))

final_answer(G)
'''<end_code>

[...]
```

Three examples are provided to the LLM. The rest of the prompt is identical to the default system prompt of Smolagents.

J.3 Counterfactual Inference Agent Prompt

We show below the start of the counterfactual inference agent system prompt, it is built similarly to the causal extraction agent.

```
You are an expert assistant who can solve any task using code blobs. You specialize into causal inference
    tasks.
You will be given causal variables and causal relationships. Your task will consist of computing the
    causal effect of a target causal variable given the values of its direct parents or children in the
    causal graph.
You will solve the task as best you can. To do so, you have been given access to a Python interpreter with
     the standard library.
You will also have access to an optional list of tools: these tools are basically Python functions which
    you can call with code.
You will use your expert reading comprehension, commonsense reasoning and coding skills to tolve the
    problem.
To solve the task, you must plan forward to proceed in a series of steps, in a cycle of 'Thought:', 'Code
    :', and 'Observation:' sequences.

At each step, in the 'Thought:' sequence, you should first explain your reasoning towards solving the task
    and the tools that you want to use.
Then in the 'Code:' sequence, you should write the code in simple Python. The code sequence must end with
    '<end_code>' sequence.
During each intermediate step, you can use 'print()' to save whatever important information you will then
    need.
These print outputs will then appear in the 'Observation:' field, which will be available as input for the
    next step.
In the end, you have to return a final answer using the 'final_answer' tool. The output provided to the '
    final_answer' tool should be the networkx causal graph.

The attributes of the target variable are provided as arguments with the name 'target_variable'.
The parent variables attributes are provided as a list of dictionaries with the name 'parent_variables'.
The children variables attributes are provided as a list of dictionaries with the name 'children_variables
    '.
The descriptions of the causal relationships between the target variable and its parents are provided as a
     list of attribute dictionaries with the name 'causal_relationships'.
Return a dictionary with the updated attributes of the target variable. The fields to update are '
    current_value', 'contextual_information' and 'causal_effect'
Use the information from the given target variable, parent variables, children variables and causal
    relationships to compute the causal effect of the target variable.
When a function is provided in the causal relationship, execute it with the Python interpreter.

Each causal variable will be represented with the following dictionary of attributes:
{{variable}}

The causal relationships will be represented with the following dictionary of attributes:
{{causal_relationship}}
```

Five few-shots are also provided as part of the system prompt. Here is one example:

```
[...]

Here are a few examples tasks to solve.
---
Task:
"""
Compute the causal effect of the variables in the context of the oil price war
```

```
You have been provided with these additional arguments, that you can access using the keys as variables in
        your python code:
{'children_variables': [], 'target_variable': {'name': 'oil_prices', 'description': 'Global price per
        barrel of oil; recently slumped to its lowest level since 2009', 'type': 'float', 'values': 'USD per
        barrel'}, 'causal_relationships': []}.
"""

Thought: We have a target variable "oil_prices" with description indicating that oil prices have slumped
        to the lowest level since 2009.
There are no parent or children variables provided and no causal relationships specified.
In this context, we need to update the target_variable dictionary by setting its "current_value", "
        contextual_information", and "causal_effect".
Since the variable is operating in an environment described as an "oil price war", we can infer that the
        aggressive competition among oil producers is likely driving a strong negative causal impact on
        oil_prices.

I'll now update the target variable accordingly:
- current_value: "lowest level since 2009"
- contextual_information: "In the context of an oil price war, aggressive market competition among oil-
        producing nations has driven global oil prices down significantly."
- causal_effect: "lowest level since 2009"

I'll now write the code with these updates and then use final_answer to output the updated target_variable.


Code:
```py
# Update target_variable with computed values
target_variable['current_value'] = "lowest level since 2009"
target_variable['contextual_information'] = ("In the context of an oil price war, aggressive market
        competition among oil-producing nations has driven global oil prices down significantly.")
target_variable['causal_effect'] = "lowest level since 2009"

final_answer(target_variable)
```<end_code>

[...]
```