RECoRD: A Multi-Agent LLM Framework for Reverse Engineering Codebase to Relational Diagram

Anonymous Authors[1]

Abstract

Understanding the behavior and logical structure of complex algorithms is a fundamental challenge in industrial systems. Recent advancements in large language models (LLMs) have demonstrated remarkable code understanding capabilities. However, their potential for reverse engineering algorithms into interpretable causal structures remains unexplored. In this work, we develop a multi-agent framework, RECoRD, that leverages LLMs to *Reverse Engineering Codebase to Relational Diagram*. RECoRD uses reinforcement fine-tuning (RFT) to enhance the reasoning accuracy of the relation extraction agent. Fine-tuning on expert-curated causal graphs allows smaller specialized models to outperform larger foundation models on domain-specific tasks. The RFT-trained models significantly outperformed their foundation counterparts, improving F1 score from 0.69 to 0.97. RECoRD also exhibited strong generalization, with models fine-tuned on one use case improving performance on others. By automating the construction of interpretable causal models from code, RECoRD has wide-ranging applications in areas such as software debugging, operational optimization, and risk management.

1. Introduction

Understanding the behavior and logical structure of complex algorithms in large industrial systems is critical for debugging, performance optimization, and security assurance. However, modern software systems are often developed and maintained by multiple teams over extended periods, making it challenging to retain a clear and interpretable representation of the underlying logic.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

While traditional program analysis techniques provide some level of insight, they often fail to capture the causal relationships governing software behavior. In contrast, causal graph-based methods offer a principled approach to address this complexity and answer "why" (attribution) and "what-if" (counterfactual) questions at scale. However, constructing these causal graphs remains a perennial challenge.

A thoughtful approach to software development often starts with the creation of causal graphs, frequently in the form of high-level flowcharts and dependency diagrams, prior to implementing the actual code. These visual models serve as essential blueprints, clearly illustrating the logical structure and relationships between inputs and outputs within the program. Unfortunately, in reality, due to time constraints and rapidly evolving business requirements, developers often update the source code without maintaining the corresponding documentation and graphical models.

In this paper, we aim to address the aforementioned challenges by developing a novel framework called *Reverse Engineering Codebase into Relational Diagram* (RECoRD). Leveraging recent advances in large-language models (LLMs), RECoRD automates the extraction of causal dependencies from source code. RECoRD consists of two key agents: the *entity extraction agent*, which identifies key variables used in the code, and the *relation extraction agent*, which leverages reinforcement fine-tuned LLM to identify the relationships among the code entities. Our method systematically extracts causal relationships among input, intermediate and output variables, providing structured and interpretable representations of complex algorithms while substantially reducing human efforts required to produce and maintain causal graphs. Our key contributions include:

- We develop a scalable LLM-driven dual-agent framework, RECoRD, that automates the extraction of causal graphs from complex computer code, significantly reducing the manual effort required to produce and maintain such representations.
- We propose a novel approach for generating curated training datasets of causal graphs, enhancing pre-trained language models through *reinforcement fine-tuning* (RFT), and employing the fine-tuned LLM to extract causal relationships from source code.
- We conduct empirical evaluations on several public

codebases to validate the accuracy and efficiency of the RECoRD framework in producing interpretable causal structures from complex algorithms.

2. Related Work

Causal discovery and root cause analysis. Causal discovery is fundamental in fields like supply chain management, biology, and medicine. These methods aim to uncover causal structures to support root cause analysis (**??**) and counterfactual reasoning. While progress has been made with observational data-driven algorithms—such as the Peter-Clark test and kernel conditional independence test (**???**)—these approaches still depend heavily on domain expertise and manually curated data, limiting scalability to complex, evolving systems.

Generative knowledge graph construction. Melnyk et al. (**?**) proposed an end-to-end system that generates knowledge graphs from textual inputs. The emergence of powerful foundation models like GPT-4 and Claude 3 has transformed the field, allowing knowledge graph construction through zero/one/few-shot prompting. Jiralerspong et al. (**?**) present a framework that leverages LLMs combined with a breadth-first search approach for full causal graph discovery, achieving state-of-the-art results on real-world causal graphs. Furthermore, Darvariu et al. (**?**) demonstrate that LLMs can serve as effective priors for causal graph discovery, integrating background knowledge to improve performance on common-sense benchmarks, particularly in assessing edge directionality. The most recent work (**?**) showed LLM's capability in constructing knowledge graph without additional human efforts. However, these LLM-based methods often struggle with accurately capturing the structural semantics present in source code.

Code understanding and integration of program analysis with LLMs. Code-centric LLMs like Codex (**?**), CodeGen (**?**), and StarCoder (**?**) show strong performance in code generation and translation, but their token-level perspective can yield syntactically correct but semantically incorrect outputs. Hybrid systems address this by incorporating program structure: CodeLLM-Devkit injects abstract syntax tree and control/data-flow features (**?**), while LLMDFA uses symbolic tracing to check intermediate results (**?**). Still, even semantics-preserving rewrites can mislead models, highlighting the need for explicit structural grounding (**?**).

3. Methodology

Our proposed RECoRD framework automates the extraction of interpretable causal graphs directly from complex source code by integrating deterministic program analysis and advanced reasoning capabilities of RFTed LLMs. Our approach consists of three sequential yet complementary stages (see Algorithm 1 and Fig. 1 for an illustration). First,

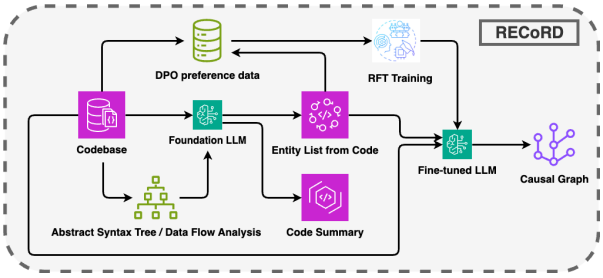

Figure 1. Three-stage causal graph generation flow. The process begins by ingesting and chunking code and documents, proceeds through entity identification, and culminates in relationship extraction that yields a final causal graph.

entities representing causal nodes are systematically identified through a hybrid process combining *abstract syntax tree* (AST) parsing, data flow analysis, and LLM-guided refinement, optionally followed by manual filtering to mitigate potential inaccuracies. Subsequently, using these refined entities, we employ a carefully crafted prompt engineering strategy to guide foundation LLMs in extracting causal relationships. Finally, recognizing limitations in foundational model outputs, we implement a RFT strategy utilizing DPO informed by expert-annotated preferences, enhancing the precision and reliability of inferred causal relationships. In the rest of this paper, we use the terms relational diagram, causal graph, and directed acyclic graph (DAG) all interchangeably.

3.1. Entity Extraction Agent

The first critical component of RECoRD is automated entity extraction. This process systematically identifies relevant variables within the source code, including inputs, intermediate states, and outputs, which collectively constitute the nodes of our causal graph. We combine program analysis techniques, such as AST parsing and data flow analysis, with LLM-based refinement to achieve high accuracy. The source code is parsed to extract fundamental properties, such as function signatures, parameter lists, return variables, class hierarchies, and object attributes. These properties are then processed by an LLM agent. This agent evaluates the contextual significance of each variable, refining the initial candidate set to ensure comprehensive identification of causal nodes.

3.2. Relation Extraction Agent – Prompt Engineering

The relation extraction agent is responsible for inferring the causal relationships between the entities identified by the entity extraction agent. Crafting effective prompts for this task is crucial, as it directly impacts the quality of the inferred causal structures. We employ a multi-stage prompt engineering approach to guide the relation extraction agent: **Entity**

Algorithm 1 RECoRD: Automated Causal Graph Extraction

Require: Source code $\mathcal{C}$, pre-trained LLM $\mathcal{M}$, parameters $\epsilon_{\text{upper}}, \beta, p$
Ensure: Causal graph $G = (V, E)$, adjacency matrix A
Stage 1: Entity Extraction (Sec 3.1)
Generate Abstract Syntax Tree from source code $\mathcal{C}$
Perform data flow analysis on AST to identify initial candidate nodes
Use LLM-based node-list generation agent (Claude 3.7 Sonnet) to refine candidate nodes
Optional: Manually filter nodes to remove obviously irrelevant entities
Finalize refined node list $V \leftarrow \{v_1, v_2, \ldots, v_n\}$
Stage 2: Relation Extraction via Prompt Engineering (Sec 3.2)
Construct prompt using refined nodes V and source code $\mathcal{C}$
Infer preliminary causal relationships $E' \subseteq V \times V$ using prompt-engineered inference with $\mathcal{M}$
Stage 3: Reinforcement Fine-Tuning via DPO (Sec 3.3)
for $t \leftarrow 1$ **to** T **do**
    Sample $m \sim \text{Uniform}(1, \lfloor \epsilon_{\text{upper}} \cdot n \rfloor)$ nodes
    Generate subgraph node set $I \subseteq V$, $|I| = m$
    Generate correct subgraph G_I based on ground truth edges E_I
    Generate incorrect subgraph $\hat{G}_I$ by randomly removing or adding edges proportional to $|E_I| \cdot p$
    Compute preference score $s(G_I, \beta)$ using Eq. (1)
    Collect pair $(G_I, \hat{G}_I)$ with preference score $s(G_I, \beta)$
end for
Fine-tune $\mathcal{M}$ using collected training pairs with DPO and LoRA adapters to obtain fine-tuned model $\mathcal{M}_{\text{RFT}}$
Final Graph Extraction
Use fine-tuned LLM $\mathcal{M}_{\text{RFT}}$ with node set V to infer final causal relationships E
Construct the final causal graph $G = (V, E)$ and derive adjacency matrix A
return G, A

Framing: The prompt begins by clearly stating the list of entities extracted from the code (e.g., entities identified in Section 3.1), framing them as the key variables or components in the system. **Causal Relationship Elicitation**: The agent is then instructed to analyze the relationships between these entities and identify the causal dependencies. **Directional Insights**: To enrich the causal understanding, the prompt further requests the agent to specify the directionality of each relationship (e.g., what variable is an input to another variable).

We start with prompting large size foundation models (FMs) using code and its corresponding instructions. The prompt instruction in Appendix A gives the best results. Although FMs can identify a set of key entities within code and infer causal relationships among them, their outputs often contain inaccuracies such as missing causal links and the inclusion of spurious relations not supported by the code. These limitations persist even in state-of-the-art models such as Qwen3-32B. To address these issues, we adopt an RFT approach to enhance the model's code comprehension and improve its reasoning accuracy.

3.3. Relation Extraction Agent – RFT

To enhance the model's reasoning accuracy, we adopt an RFT approach on smaller-scale models with 7B to 14B parameters. RFT explicitly optimizes model behavior through feedback derived from both correct and incorrect responses, guided by corresponding preference scores. We generate various subsets of the full causal graph, assign preference scores to each, and use them to fine-tune the model (see details in Section 3.3). This process allows the model to learn causal relationships between smaller sets of entities with a large number of examples, and generalize the reasoning to the full graph. For RFT, we employ DPO (**?**) in combination of the *Parameter-Efficient Fine-Tuning* (PEFT) (**??**) method with *Low-Rank Adaptation* (LoRA) (**?**) adapters. DPO directly updates the foundation model's parameters based on relative preferences rather than absolute labels, leading to a more stable and efficient learning process. LoRA further enhances efficiency and generalization ability cross a wide range of codebases by fine-tunes only a small subset rather than updating all parameters of the pre-trained model. In our framework, each adapter is specialized for a specific codebase, allowing us to deploy a single foundation model equipped with multiple codebase specific LoRA adapters. This design not only support diverse code understanding task, but also mitigates catastrophic forgetting by isolating task-specific knowledge within separate adapters.

RFT Dataset Generation. To generate sub-graphs for training, we set an upper threshold ϵ_{upper} on the proportion of nodes included in the fine-tuning examples given n number of entities (nodes). For each subsampled list of entities $\{N_1, N_2, \cdots, N_m\}$ where $m \sim Uniform([0, [\epsilon_{upper} \cdot n])$, we generate a pair of the correct sub-graph and an incorrect sub-graph, where the incorrect sub-graph is generated through sampling a DAG with the m entities where the DAG is not the same as the ground truth sub-graph.

The DPO preference score is calculated by comparing the edges between the generated sub-graph and the ground truth sub-graph. To allow for asymmetric penalty for missing edges versus wrong edges, we use a weighted sum of the two different penalties controlled by the parameter β as the score:

$$s(\mathcal{G}_I, \beta) = \max\left(0, \frac{\sum (A_I - \hat{A}_I) \cdot \mathbf{1}_{\{A_I - \hat{A}_I > 0\}}}{|I|^2} + \beta \frac{\sum -(A_I - \hat{A}_I) \cdot \mathbf{1}_{\{A_I - \hat{A}_I < 0\}}}{|I|^2}\right) \quad (1)$$

where I is the set of the nodes in the sub-graph, $\mathcal{G}_I$ refers to the sub-graph with the node list I, A_I and $\hat{A}_I$ are the adjacency matrices of the ground truth sub-graph and the generated sub-graph respectively.

In practice, when the full ground truth graph is unknown, we will generate pairs of sub-graphs for human domain experts to rank their preferences, which will then be used for DPO fine-tuning. The larger the threshold ϵ_{upper}, more nodes are involved in the sub-graphs and therefore more domain knowledge is required to label the human preference. For each prompt, we modify the instruction to let the agent extract causal graph based on the node list involved in each sub-graph only. The prompt, in combination with the positive/negative examples with their scores, constitute the DPO training dataset.

4. Experiments

We demonstrate RECoRD's performance using three real-world use cases that have causal interpretations (physical mechanisms) supporting the implementation: News Vendor, MiniSCOT, and Black-Scholes, each with a known ground truth DAG and readily available open-sourced code base (see Appendix B for details).

Across all three use cases we observe a clear precision–recall trade-off among the entity-extraction pipelines (Table 2). We first apply a lightweight human filtering step: a minimal manual review (typically 1–2 minutes by a non-expert) to remove obviously irrelevant items such as loop counters, framework artifacts, or generic placeholders. This step requires no domain-specific knowledge and serves as a practical compromise between fully automated extraction and exhaustive annotation. The code-only +H pipeline achieves perfect precision on the Newsvendor case and slightly higher recall than AST-assisted variants on MiniSCOT, confirming its effectiveness for compact, business-rule–driven code. In structurally richer settings such as Black-Scholes, however, many relevant variables are buried in deeper abstractions; here AST guidance, despite adding noise, captures subtle but critical terms (e.g., higher-order Greeks) and yields substantially better recall. This recall gain is particularly valuable because relation extraction performance is more sensitive to missing nodes than spurious ones, as missing variables preclude the generation of edges entirely. Overall, the AST +LLM +H pipeline strikes the best balance between coverage and precision, and its recall advantage translates to higher downstream edge.

We experiment with RECoRD's Relation Generation Agent on the above three use cases using the following RFT models: Qwen2.5-7B-Instruct-RFT, Qwen2.5-14B-Instruct-RFT, and Phi-4-RFT. The model performance is measured by the F1-score between the model generated DAG entities and relations and the ground truth annotated by human experts, as shown in Appendix F. The results demonstrate that RECoRD can effectively extract interpretable causal graphs from complex codebase in an automated manner. For the MiniSCOT use case, the RFT-based approach was able to accurately recover the causal structure without any wrongly identified nodes or edges (see Figure 3). Even with partial entity lists which is closer to practical settings, the RFT-trained models significantly boosted performance compared to the foundation models.

To demonstrate the generalization capability of RECoRD to a new codebase without human-labeled data, we use a RFT model trained using the News Vendor codebase to extract a causal graph from the MiniSCOT codebase. Figure 2 and Table 1 show the corresponding causal graph and F1 score.

Table 1. Graph relation F1-score of RECoRD with different configurations on MiniSCOT use case. The RFT model is fine-tuned by the training dataset from News Vendor.

Use Case	**Model**	**Full entity list**		**LLM + H + AST**		**LLM + H**	
		FM	RFT	FM	RFT	FM	RFT
MiniSCOT	Qwen3-32B	0.750	-	0.412	-	0.303	-
	Qwen2.5-7B	0.786	**0.857**	0.364	**0.452**	0.182	**0.323**
	Phi-4	0.690	0.733	0.375	0.438	0.121	0.250

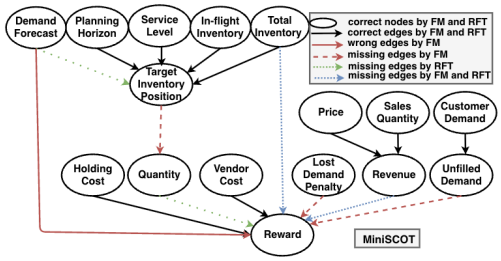

Figure 2. Comparison of ground truth and learned causal graphs for MiniSCOT with full entity list using Qwen 7B-Instruct FM and RFT trained on News Vendor. RFT significantly improves the causal discovery accuracy.

5. Conclusions

We presented RECoRD, a multi-agent pipeline that fuses deterministic program analysis with reinforcement fine-tuned LLMs to turn production code into accurate, interpretable causal graphs with minimal human input. There are several venues for future research. First, developing scalable training procedures to extract causal relationships from large-scale code repositories remains an under-explored territory. Second, designing advanced mechanisms to further enhance RECoRD's reasoning capabilities could improve its ability to uncover domain-transferable causal relationships, potentially leading to more robust and generalizable models. Finally, expanding RECoRD's applications beyond causal graph extraction to recursive algorithmic flowcharts, such as reinforcement learning, presents an intriguing opportunity to broaden its impact.

References

Arrow, K. J., Harris, T., and Marschak, J. Optimal inventory policy. *Econometrica*, 19(3):250–272, 1951. ISSN

00129682, 14680262. URL http://www.jstor.org/stable/1906813.

Budhathoki, K., Janzing, D., Bloebaum, P., and Ng, H. Why did the distribution change? In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 1666–1674. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/budhathoki21a.html.

Carta, S., Giuliani, A., Piano, L., Podda, A. S., Pompianu, L., and Tiddia, S. G. Iterative zero-shot llm prompting for knowledge graph construction. *arXiv preprint arXiv:2307.01128*, 2023.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Darvariu, V.-A., Hailes, S., and Musolesi, M. Large language models are effective priors for causal graph discovery. *arXiv preprint arXiv:2405.13551*, 2024.

Glymour, C., Zhang, K., and Spirtes, P. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524, 2019.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp, 2019. URL https://arxiv.org/abs/1902.00751.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jiralerspong, T., Chen, X., More, Y., Shah, V., and Bengio, Y. Efficient causal graph discovery using large language models. In *ICLR 2024 Workshop: How Far Are We From AGI*.

Krishna, R., Pan, R., Pavuluri, R., Tamilselvam, S., Vukovic, M., and Sinha, S. Codellm-devkit: A framework for contextualizing code llms with program analysis insights. *arXiv preprint arXiv:2410.13007*, 2024.

Li, R., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Jia, L., Chim, J., Liu, Q., et al. Starcoder: may the source be with you! *Transactions on Machine Learning Research*.

Maggiar, A., Song, I., and Muharremoglu, A. Multi-echelon inventory management for a non-stationary capacitated distribution network. *Optimization Online*, 2022.

Melnyk, I., Dognin, P., and Das, P. Knowledge graph generation from text. *arXiv*, 2023. URL https://arxiv.org/abs/2211.10511.

Nguyen, T.-T., Vu, T. T., Vo, H. D., and Nguyen, S. An empirical study on capability of large language models in understanding code semantics. *arXiv preprint arXiv:2407.03611*, 2024.

Nijkamp, E., Pang, B., Hayashi, H., Tu, L., Wang, H., Zhou, Y., Savarese, S., and Xiong, C. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

Oksendal, B. *Stochastic differential equations (3rd ed.): an introduction with applications*. Springer-Verlag, Berlin, Heidelberg, 1992. ISBN 3387533354.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: your language model is secretly a reward model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Shimizu, S., Hoyer, P. O., Hyvärinen, A., Kerminen, A., and Jordan, M. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10), 2006.

Singal, R., Michailidis, G., and Ng, H. Flow-based attribution in graphical models: A recursive shapley approach. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9733–9743. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/singal21a.html.

Wang, C., Zhang, W., Su, Z., Xu, X., Xie, X., and Zhang, X. Llmdfa: Analyzing dataflow in code with large language models. *Advances in Neural Information Processing Systems*, 37:131545–131574, 2024.

Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., and Wang, F. L. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023. URL https://arxiv.org/abs/2312.12148.

Zhang, K., Peters, J., Janzing, D., and Schölkopf, B. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, pp. 804–813, Arlington, Virginia, USA, 2011. AUAI Press. ISBN 9780974903972.

A. Prompt

(entity1), (entity2), ... represent the identified entity from Section 3.1.

Prompt Instruction

You are a causal relationship extractor and your task is to extract causal relationships from a code snippet and a technical document. Let's extract it step by step:

(1) Identify the entities in the text from the code snippet and the document. An entity must be a noun or a noun phrase that refers to a real-world object or an abstract concept.
(2) Identify the relationships between the entities. A relationship must describe as "is an input to" from the head entity to the tail entity.
(3) List the relationship in triplet format: ('head entity', 'is an input to', 'tail entity').

The code snippet is provided in (code) tags. The technical document is provided in (doc) tags. Only identify the causal relationship from the following nodes: (entity1), (entity2), ...
DO NOT put any preambles in the response. If you don't know the answer, just say I don't know.
DO NOT make up the answer.

B. Real-world Use Cases

- **News vendor:** The news vendor problem (**?**) is a classic operations research problem which determines how much of a perishable product to order under stochastic demand, aiming to minimize total expected cost. This cost comes from ordering too much (overage cost) or too little (underage cost). The optimal ordering quantity is a function of the demand distribution, whole sale price, retail price, and salvage cost (code).

- **MiniSCOT** is a simplified simulation platform developed by Amazon. It simulates multiple supply chain metrics evolution over time based on physics of the supply chain operation. In particular, we focus on its strategic buying decision which is one of the most critical and computationally expensive components in real-world supply chain systems (**?**). The MiniSCOT buying module optimizes for the optimal purchase order quantities and inbound routing based on demand forecast, inventory positions, planning period and sales patterns across different retail locations (code).

- **Black-Scholes:** The Black-Scholes model (**?**) is a classic problem in quantitative finance that determines the optimal price of an European option of certain asset (e.g., stock) which is assumed to evolve according to a geometric Brownian motion. The optimal option price is a function of key input parameters including the initial stock price, drift and volatility parameters, interest rate, strike price, and maturity time (code).

C. Fine-tuning Procedure

When conducting RFT with LoRA, we carefully tune the hyperparameters to streamline the efficient learning process and minimize the training loss. Specifically, we grid search over the learning rate, batch size, LoRA rank, and LoRA alpha, to select the optimal size of LoRA training parameter for lower training loss and faster convergence time. We set the alpha to four times the rank to improve the LoRA adapter's influence on the full model weights.

Our results suggest that smaller size models (7B and 14B parameters) fine-tuned with carefully curated domain-specific sample data outperform a large size SOTA reasoning model (Qwen3-32B) with higher accuracy, lower latency, and cost. The RFT model demonstrates improved reasoning over the foundation model, capable of explicitly identifying patterns within the code, such as variables that are input to a method or derived from other variables. These results confirm that entity recall plays a critical role in determining overall graph quality, as missing entities inherently limit the ability to recover corresponding edges. AST-guided methods, augmented by simple human filtering, strike an effective balance by maximizing recall without introducing excessive noise, resulting in the highest end-to-end graph F1 scores across all benchmarks.

Table 2. Entity-level precision (P) and recall (R).

Use Case	AST + Code		AST + Code + H		Code-only + H	
	P	R	P	R	P	R
Newsvendor	0.89	0.73	0.89	0.73	**1.00**	0.73
MiniSCOT	0.56	0.63	**0.83**	0.63	0.61	**0.69**
Black–Scholes	0.34	0.58	**0.37**	0.58	0.27	0.21

D. Entity List Evaluation

E. Learned Causal Graph

We show the learned causal graph in Figure 3 and Figure 2, with models fine-tuned with different use case.

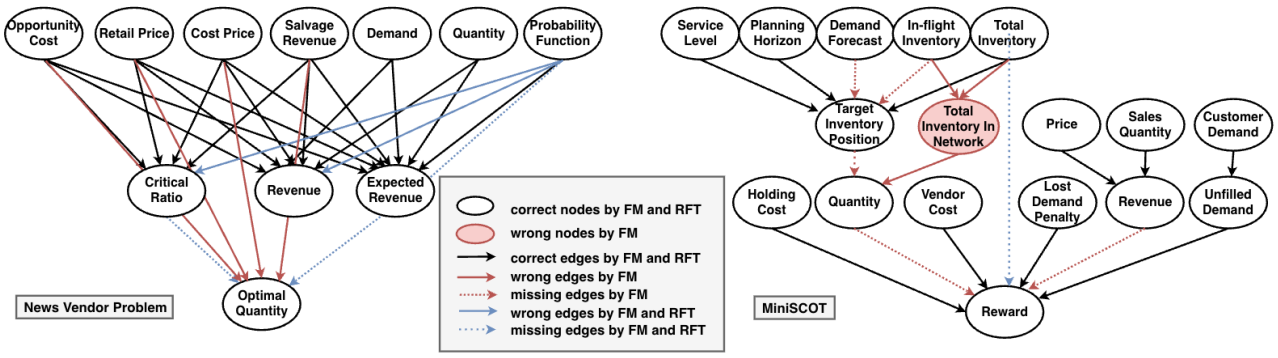

Figure 3. Ground truth and learned causal graphs with full entity list using Qwen 7B-Instruct models for News Vendor Problem (left); and Phi-4 models for MiniSCOT (right). RFT significantly improves the causal discovery accuracy.

F. Causal Graph Evaluation

We report the node F1 score in Table 3 and edge F1 score in Table 4 respectively.

Table 3. Graph node F1-score of **RECoRD** with different configurations on 3 different use cases. LLM+H+AST refers to AST + LLM generated entity list with human filtering; LLM+H refers to LLM generated entity list with human filtering.

Use Case	Model	Full entity list		LLM + H + AST		LLM + H	
		FM	RFT	FM	RFT	FM	RFT
Newsvendor	Claude 3.7 Sonnet	1.000	–	1.000	–	0.952	–
	Qwen-7B-Instruct	1.000	0.952	0.952	0.952	0.900	0.900
	Phi-4	0.917	1.000	0.870	0.909	0.952	1.000
MiniSCOT	Claude 3.7 Sonnet	0.968	–	0.500	–	0.722	–
	Qwen-7B-Instruct	0.914	1.000	0.530	0.690	0.629	0.632
	Phi-4	0.938	0.970	0.606	0.710	0.743	0.824
Black-Scholes	Claude 3.7 Sonnet	0.840	–	0.750	–	0.591	–
	Qwen-14B-Instruct	0.966	0.966	0.862	0.909	0.605	0.591
	Phi-4	1.000	1.000	0.847	0.885	0.545	0.545

Table 4. Graph relation F1-score of RECoRD with different configurations on 3 different use cases. The F1 score is computed against the set of edges in the ground truth causal graph.

Use Case	**Model**	**Full entity list**		**LLM + H + AST**		**LLM + H**	
		FM	RFT	FM	RFT	FM	RFT
Newsvendor	Claude 3.7 Sonnet	0.765	–	0.710	–	0.728	–
	Qwen-7B-Instruct	0.810	0.895	0.571	0.687	0.462	0.518
	Phi-4	0.737	0.800	0.647	0.647	0.500	0.533
MiniSCOT	Claude 3.7 Sonnet	0.645	–	0.400	–	0.487	–
	Qwen-7B-Instruct	0.667	0.857	0.425	0.519	0.182	0.364
	Phi-4	0.690	0.968	0.375	0.467	0.267	0.323
Black-Scholes	Claude 3.7 Sonnet	0.593	–	0.417	–	0.291	–
	Qwen-14B-Instruct	0.552	0.573	0.483	0.569	0.256	0.286
	Phi-4	0.556	0.562	0.412	0.497	0.284	0.284