Leveraging LLMs to Discover Causal Dependencies: A Case Study on a University Program

Leveraging LLMs to discover causal dependencies: a case study on a university program

Claudia Diamantini[1 [0000−0001−8143−7615]], Chiara Gobbi[1 [0009−0009−6589−5347]], Alessandro Mele[1 [0009−0009−7852−0528]], Domenico Potena[1 [0000−0002−7067−5463]], and Cristina Rossetti[1,2 [0009−0003−5243−4249]]

[1] DII, Università Politecnica delle Marche, via Brecce Bianche, Ancona, 60131, Italy
[2] DAUIN, Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, 10129, Italy

Abstract. Defining a process model is essential for understanding organizational workflows and guiding future activities. While process discovery techniques can derive models from event logs, designing a process model from scratch remains challenging, as it relies on human experts to interpret requirements. This work explores process model elicitation in the educational domain, where courses in a university program are treated as activities with prerequisite relationships. We propose a methodology leveraging Large Language Models to automatically infer causal dependencies among courses by analyzing their syllabi. Using pairwise prompting with step-back and chain-of-thought reasoning, the extracted dependencies are used to build a Directly-Follows Graph, subsequently evaluated by domain experts. Experimental results show that Large Language Models can effectively identify and justify course dependencies, providing valuable insights that can enhance curriculum design and provide students with clearer guidance on course sequencing. The approach also highlights gaps in prerequisite structures, suggesting a broader application in educational planning and dynamic learning environments.

Keywords: Process mining · Large language models · Educational process mining.

1 Introduction

Explicitly defining a process model is a crucial activity for understanding how an organization operates and for guiding future activities. A process model can be derived from the information contained in an event log using process discovery techniques, when an event log is available. However, during the process design phase, only the requirements are known, and is challenging to define a process model. In this case, human experts are required to interpret the requirements and formalize them into a process model.

We focus on the design of process models from scratch in the educational domain, considering the study of a set of available courses as activities to be performed in order to acquire the skills provided by a study program (i.e., the

process). In this scenario, process model elicitation corresponds to identifying the set of causal dependencies between available courses, meaning that fully understanding the content of a course requires the skills acquired in another course. In an academic scenario, this set of dependencies is useful, for instance, to verify the manifesto of a degree program, specifically the allocation of courses across semesters and years. Moreover, in contexts where students can attend courses in any order without any content-related prerequisites, causal dependencies provide a useful guideline for students, suggesting the order in which courses should be taken.

The issue becomes even more complex in dynamic online learning systems, where several courses are available and frequent change or addition of courses occur. In these environments, every time a new course is introduced, all instructors would need to reassess and redefine the prerequisite relationships for their courses in light of the new ones. This process is highly challenging as it requires continuous effort to keep the dependency structure up to date, making manual maintenance impractical.

In this work, we investigated upon the use of Large Language Models (LLMs) for automatically discovering dependencies among available courses, based on the topics covered in each course's syllabus. Specifically, we leverage the ability of LLMs to both analyze and establish relationships among course topics. The proposal uses a pairwise prompting supported by a step-back [20] and chain-of-thought [18] approach. The discovered dependencies are used to build a Directly-Follows Graph (DFG). The results are evaluated by domain experts and show that LLMs can both identify, with a good precision, and justify dependencies.

The work is organized as follows. Section 2 provides an overview of related work, Section 3 describes the proposed approach through a real-world case study, Section 4 discusses the experimental results, and Section 5 concludes the paper and delineates future work.

2 Related work

LLMs are advanced Artificial Intelligence (AI) systems that learn from vast collections of corpora to process and generate text. These models leverage deep learning techniques, with the transformer architecture being the most widely used [17]. This enables LLMs to recognize intricate linguistic patterns and relationships across words and sentences allowing despite the complexities of human language [19,10].

By acquiring broad knowledge across multiple domains, these models enhance language comprehension and open new possibilities for improving the identification of causal dependencies [14,8]. These advances in LLMs have sparked new interest in their application to causal discovery tasks. [11] explores how LLMs can initialize differentiable causal discovery models, improving robustness and scalability. Similarly, [9] analyzes factors influencing LLM performance in causal discovery. Furthermore, [12] investigates autonomous LLM-augmented causal reasoning, showcasing their ability to enhance causal inference tasks. In addi-

tion, [1] discusses how LLMs can be leveraged for causal discovery, providing insights into their potential advantages over traditional methods.

Most LLM-based approaches to causal dependencies discovery, predominantly rely on pairwise prompting [13,14,2] where the model is required to produce causal relations like "Does A cause B?" using different prompt formulations. Some other work proposes a triplet-based prompting strategy combined with chain-of-thought reasoning [18] that enhances accuracy and mitigates cyclic dependencies when inferring relationships [16].

Process mining has greatly benefited from LLM integration. [3] shows that some LLMs can effectively perform multiple tasks, such as anomaly detection, process model generation, and process model understanding. The application of LLMs in process mining is also discussed in [5], which provides a detailed analysis of their potential. Additionally, [6] proposes the analysis of business processes models through natural process querying language and [7] develops a LLM specialized in the interpretation, analysis and optimization of business processes.

Despite these advancements, several challenges persist. While LLMs have demonstrated potential in causal discovery and process mining, concerns regarding their interpretability, robustness, and reliability remain critical areas of investigation. [15] introduces a benchmark designed to evaluate LLMs in process mining contexts, addressing existing gaps in performance evaluation, while [4] investigates methods for evaluating LLM outputs in PM tasks, an essential aspect for real-world applicability.

In our approach, we investigate the use of LLMs to extract causal dependencies among courses available in a course set. In details, we use a pairwise prompting supported by a step-back [20] and chain-of-thought [18] approach.

3 Methodology

The proposed methodology aims at extracting causal dependencies among courses available in a university study plan, based on the topics covered in the syllabus of each course.

In order to provide an illustrative example, we focus on a 3-year Bachelor's Degree program in Computer and Automation Engineering from the Università Politecnica delle Marche, Italy[3]. In particular, we focus on courses of the 2023-2024 academic year[4]. Foundational courses, such as *Mathematical Analysis 1* and *2*, *Linear Algebra and Geometry*, *Physics 1*and *2*, *Fundamentals of Computer Science* and *Business Economics* are all concentrated in the first year. Starting from the second year, courses specific to the degree program are introduced, including *Fundamentals of Automation*,*Elements of Electronics*, *Electrotechnics*, *Automation Control* and *Object Programming*. In the second year, students have the opportunity to select two courses from the ones offered by the university, based on their area of interest. Finally, in the third year, there are

[3] https://www.univpm.it/Entra/Universita_Politecnica_delle_Marche_Home/L/1
[4] https://guide.univpm.it/guide.php?fac=ingegneria&lang=lang-eng

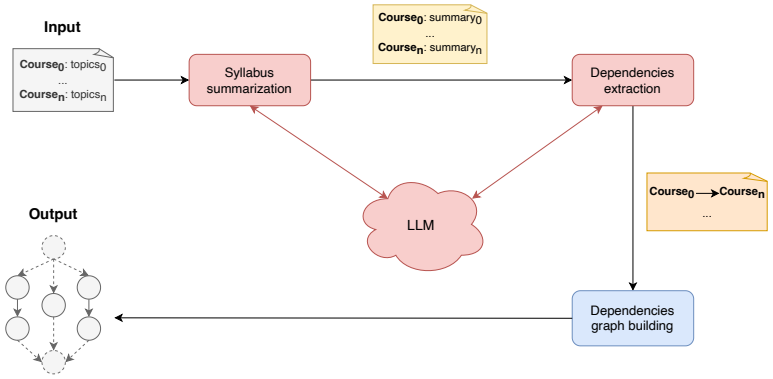


Fig. 1: The proposed methodology.

only two mandatory courses, and all the others are electives, allowing students to choose their specialization path. Examples of specialized courses include: *Web Technologies*, *Databases*, and *Technologies for Automation and Robotics*.

The methodology adopts a step-back [20] and chain-of-thought [18] prompting approaches, which consist of providing the LLM with an initial prompt and then refining the output through additional prompts to obtain the final result. This approach results useful to instruct the LLM to refine its reasoning process incrementally, ensuring a coherent final output. The steps of our methodology are detailed in Figure 1.

Syllabus summarization In this section, we define the steps to obtain the initial prompt provided to the LLM. The course-related information were extracted from the syllabus of courses. The syllabus provides details for each course of the degree program, including the teacher and the topics covered in the course.

The starting prompt consists on providing the LLM with list of courses, each of one is described by the name and the topics covered in the course. To mitigate biases introduced by course naming, we begin by assigning a neutral code to each course in the syllabus, thus redirecting the model's attention to the substantive content of the program. For the sake of simplicity, in the following, we still refer to courses with their names instead of their codes. The task assigned to the LLM is to summarize, for each course, the most important concepts to learn. This step is fundamental, since we provide the LLM with a knowledge base for future queries. Figure 2 shows an excerpt of the prompt to derive summarization.

Dependencies extraction In this step, we aim at deriving the set of *causal dependencies* among the courses listed in the degree program. A causal dependency between courses A and B ($A \rightarrow B$) means that in order to attend B, some topics covered in course A are required. This step is further divided in sub steps: we query the LLM (i) to derive the initial set of causal dependencies based on the

This is the list of courses from the syllabus:

Course: C1 {Mathematical analysis 1}
Program: sets, relations and functions. natural, integer, rational and real numbers. complex numbers, trigonometric and exponential representation.

Course: C2 {Mathematical analysis 2}
Program: functions of several variables. curves, line integrals, vector fields, differential forms. multiple integrals with applications. ordinary differential equations. laplace transform on r.

Course: C3 {Business Economics}
Program: economic and financial aspect of management, enterprise in the economic system, corporate legal forms.

. . .

Your task is to summarize their programs highlighting the most important concepts to learn.

Fig. 2: Excerpt of the starting prompt provided to the LLM to derive summarization. The course names are enclosed in curly brackets for simplicity only and have not been included in the prompt

provided knowledge base, (ii) to obtain the explanations that guided the LLM in defining them, and (iii) to determine whether, based on the explanations just provided, it would insert new dependencies or delete discovered ones, obtaining the final set.

The first sub step consist on providing the LLM with the definition of causal dependency. To achieve this, the LLM is provided with the following prompt: *Given two courses A and B, the causal dependency $A \rightarrow B$ means that in order to study B, it is essential to have previously studied A*. Then we assign the following task to the LLM: *Find causal dependencies $A \rightarrow B$, where A and B must be only in this list of given courses.* With this indications, the LLM is provided with a precise definition of causal dependency and is forced to adhere to the specified course names. Luckily, in our case study, the LLM has not discovered dependencies among non-existing courses.

The second sub step involves querying the LLM to obtain the reasons behind the definition of the dependencies. This is a crucial step because it forces the LLM to articulate its reasoning explicitly, rather than simply presenting a set of conclusions. Revealing these justifications allows verification of whether the dependencies align with the actual course content. Figure 3 shows the reasoning provided by the LLM for the dependency *Mathematical analysis 1* $\rightarrow$ *Mathematical analysis 2*.

The last sub step consists on querying the LLM to revisit the dependencies in light of its own justifications, reflecting on whether the previous dependencies it

Mathematical Analysis 1 → Mathematical Analysis 2

Explanation: mathematical analysis 1 covers foundational calculus concepts such as limits, continuity, differentiation, integration, and basic infinite series. Understanding these single-variable calculus principles is essential before advancing to mathematical analysis 2, which deals with functions of several variables, multiple integrals, and advanced topics like differential equations. The multi-variable calculus in mathematical analysis 2 builds directly upon the single-variable calculus learned in mathematical analysis 1.

Fig. 3: An example of causal dependency explanation.

articulated are correct and complete. In this step, the LLM detects inconsistencies or realizes that certain dependencies are misaligned with the actual course content and with the explanations it gave in the previous step and then it revises the dependencies accordingly. This ensures that the LLM not only clarifies how it arrived at particular prerequisites but also critically reevaluates its own reasoning, refining and strengthening the set of causal dependencies across the courses.

Dependencies graph building The resulting set of causal dependencies is then used to construct a DFG. Let $\mathcal{G} = (V, E)$ be a graph, where each node $v \in V$ represents a course, and each directed edge $\langle v_i, v_j \rangle \in E, v_i, v_j \in V$ represents the causal dependency $v_i \rightarrow v_j$. In order to obtain a connected graph, we added two artificial nodes, i.e., $START$ and END. For each course for which the LLM has not identified incoming dependencies, we add an artificial edge from the $START$ node and the course. For example, the *Business Economics* course, being from a different disciplinary field, does not require engineering-related skills. For all the courses for which the LLM has not identified outcoming dependencies, we add artificial edges to the END node.

4 Experiments

This section aims to illustrate (i) the experimental setup, (ii) the obtained dependencies, and (iii) the corresponding evaluation phase. The proposed methodology was implemented in Python and using the *gpt-o1-preview* LLM, accessible via OpenAI APIs[5]. Regarding the LLM, the `seed` shall aim at ensuring experiment reproducibility. For space reason, in this work we show only an excerpt of the dependency graph; the complete material is available at the following repository[6]. The evaluation step is conducted as follows: the graph generated by the LLM ($\mathcal{G}$) is compared with the reference graph ($\mathcal{G}^*$) built by asking course professors

[5] https://platform.openai.com/docs/overview
[6] https://github.com/KDMG/causal_dependencies_degree_courses/tree/main

to indicate which other courses a student should take before enrolling in their own course. Then, all dependencies discovered by the LLM that are also present in the baseline graph are evaluated by professors.

4.1 Results

The dependency graph derived from LLM ($\mathcal{G}$) contains 40 edges, 19 of which identify causal dependencies not present in $\mathcal{G}^*$. Due to space constraints, Figure 4 illustrates an excerpt of $\mathcal{G}$. The excerpt highlights (with dashed edges) the artificial links between the $START$ node and the courses for which the LLM has no discovered dependencies, i.e., *Linear Algebra and Geometry*, *Mathematical Analysis 1*, *Business Economics*, and *General Physics 1*. Note that the graph $\mathcal{G}$ obtained can be interpreted as a process model that provides students with guidance on how to build the required knowledge before progressing to more advanced topics in the study plan. Specifically, the dependencies in the graph highlight different types of prerequisite relationships: AND-gateways where multiple prerequisites are recommended before taking a course and OR-gateways, which indicate that completing a course enables students to access multiple subsequent courses for which they have acquired the necessary foundations.

Figure 4 also highlights (with blue edge) a *transitive* dependency. A transitive dependency is a dependency that can be derived from others. Given the dependency *Mathematical Analysis 1* $\rightarrow$ *Mathematical Methods for Automation Engineering*, we observe that can be derived from *Mathematical Analysis 1* $\rightarrow$ *Mathematical Analysis 2* and *Mathematical Analysis 2* $\rightarrow$ *Mathematical Methods for Automation Engineering*. Discussions with domain experts revealed that all transitive dependencies were intentionally omitted in the definition of $\mathcal{G}^*$. Consequently, before the evaluation stage, all transitive dependencies were removed from $\mathcal{G}$, resulting in a total of 37 edges, of which 17 dependencies (E') were identified by the LLM and missing in $\mathcal{G}^*$. Table 1 reports some examples of the resulting dependencies and the corresponding explanation provided by the LLM.

Table 1: Example of dependencies discovered by LLM not in $\mathcal{G}^*$.

Dependency: *Mathematical Analysis 1* $\rightarrow$ *Probability and Statistics*
Explanation: Calculus concepts from mathematical analysis 1, such as integration and differentiation, are essential in probability and statistics for working with probability density functions, calculating probabilities, and understanding continuous random variables. The ability to compute limits and work with infinite series is also important for topics like convergence in distributions and the Central Limit Theorem in probability and statistics

Table 1: Example of dependencies discovered by LLM not in $\mathcal{G}^*$.

Dependency: *Mathematical Analysis 1 → Fundamentals of Electronics* **Explanation**: Calculus concepts from mathematical analysis 1 are important in fundamentals of electronics for analyzing circuit behavior over time, especially when dealing with capacitors, inductors, and time-varying signals. Understanding differential equations and integrals is necessary for transient analysis and frequency response in electronics
Dependency: *Mathematical Analysis 2 → Automatic Control* **Explanation**: Automatic control requires strong calculus skills from mathematical analysis 2, as control systems analysis and design involve differential equations, Laplace transforms, and continuous-time system modeling. Understanding multi-variable calculus and the mathematical tools for analyzing system dynamics is crucial in automatic control
Dependency: *Linear Algebra and Geometry → Probability and Statistics* **Explanation**: The necessity of linear algebra for probability and statistics depends on the depth of topics like multivariate distributions, covariance matrices, and linear regression If probability and statistics covers these advanced topics extensively, linear algebra becomes essential However, for introductory probability and statistics focusing on univariate distributions and basic statistical methods, linear algebra may not be strictly required Therefore, the dependency should be clarified based on the course's emphasis
Dependency: *Linear Algebra and Geometry → Mathematical Methods for Automation Engineering* **Explanation**: mathematical methods for automation engineering involves functional analysis and linear operators, which are built upon linear algebra concepts from linear algebra and geometry, including vector spaces and linear transformations The abstract mathematical structures studied in mathematical methods for automation engineering require a deep understanding of linear algebra

4.2 Discussion

After the post-processing step in which the transitive edges were excluded, a total of 20 correct edges were identified by the LLM, compared to the 51 in $\mathcal{G}^*$; 31 dependencies are not recognized by the LLM and 17 are found by the LLM but are not present in $\mathcal{G}^*$. Here, we focus on these latter as they may not be incorrect. In fact, $\mathcal{G}^*$ cannot be considered 100% reliable for several reasons. First, human experts who contributed to its development may have omitted some dependencies for the sake of simplicity or because they considered them less important than others. Moreover, while the LLM used only the content of the syllabus, the expert also relied on its own knowledge and experience. This forces us to consider the dependencies in E' to avoid erroneously labeling them as

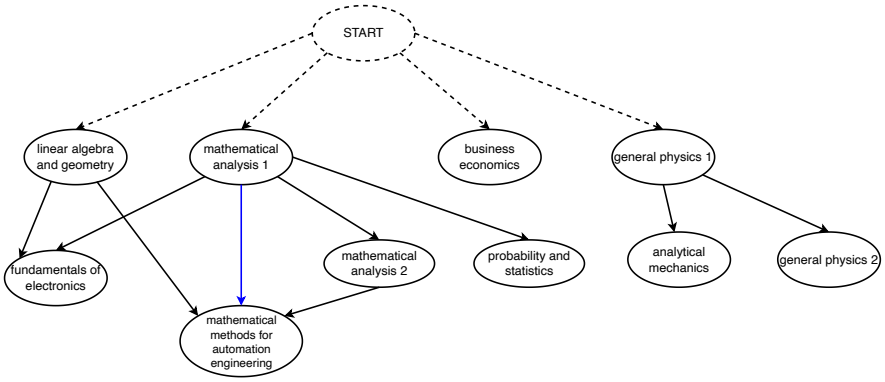
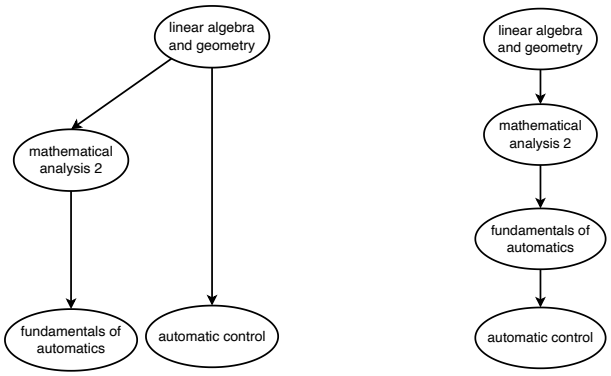


Fig. 4: Excerpt of $\mathcal{G}$.

wrong. In fact, some of these dependencies might be correct from the evaluator point of view, and this can be useful to possibly refine and adjust the source graph.

Before the actual evaluation, we identify among E' a subset that we call *indirect dependencies*. These dependencies represent those edges that are not explicitly present in $\mathcal{G}^*$, but can be found in an existing path. For example, assume $\{(A, B), (B, C), (C, D)\} \in E^*$ and $(A, D) \in E'$ a dependency returned by the LLM but not present in $\mathcal{G}^*$. The edge (A, D) cannot be considered wrong, because there is in fact in $\mathcal{G}^*$ a path connecting node A to node D, so this dependence exists but in an indirect way. We identified a total of 9 indirect dependencies in E'. For instance, consider the dependency *Linear Algebra and Geometry* $\rightarrow$ *Automatic Control* (Figure 5): in $\mathcal{G}^*$ the path *Linear Algebra and Geometry* $\rightarrow$ *Mathematical Analysis 2* $\rightarrow$ *Fundamentals of Automatics* $\rightarrow$ *Automatic Control* exists. It can be stated that, in order to fully comprehend *Automatic Control*, it is essential to have studied *Linear Algebra and Geometry*. As such, the indirect dependency *Linear Algebra and Geometry* $\rightarrow$ *Automatic Control* can be considered as correct.

These observations cause the total number of dependencies that we consider to be correct to rise from 20 to 29. It now remains to discuss the remaining 8 dependencies that are neither present in $\mathcal{G}^*$ nor indirect. Of these, 4 were assessed by the evaluators as incorrect. The other 4 are discussed in detail below. The dependencies *Numerical Analysis* $\rightarrow$ *Modeling and Identification of Dynamic Processes* and *Numerical Analysis* $\rightarrow$ *Fundamentals of Automatics*, which connect *Numerical Analysis* to automation courses, was positively assessed by the evaluators because methods to solve linear and non-linear systems as well as concepts of eigenvalues, derivatives and integrals studied in *Numerical Analysis* may serve as useful mathematical knowledge to better approach the design of dynamic control systems. As concerns the dependency *Object-Oriented Programming* $\rightarrow$ *Operating Systems*, evaluators believed that the implementation of



(a) Extract of dependencies in $\mathcal{G}$

(b) Extract of dependencies in $\mathcal{G}^*$

Fig. 5: Example of indirect dependency.

concurrent programming (present in the course's syllabus) may involve the use of constructs from object-oriented programming. In addition, the practice lectures require knowledge of Java. Thus, this dependency was evaluated as correct, although it may be considered a weak link. Finally, also the dependency *Algebra and Logics* $\rightarrow$ *Object-Oriented Programming* was labeled as correct, since the notions of logical reasoning and algebraic structured studied in *Algebra and Logics* can be useful, even if not strictly necessary, for a better approach to a programming course. On the basis of this discussion, the total number of correct dependencies in $\mathcal{G}$ are 33.

It is interesting to note that evaluators agreed with the explanations provided by the LLM. For instance, concerning the above-mentioned dependencies involving *Numerical Analysis*, both the LLM and the evaluators stated that the mathematical methods studied in the course helps the understanding, modeling and solving dynamic processes. In addition, the LLM seems capable of assigning an importance weight to dependencies with terms such as "necessary" or "not strictly required". For example, the LLM assessed the course *Algebra and Logics* as useful but not strictly necessary for *Object-oriented programming*, just as the evaluators.

An important consideration concerns the identification of clusters. In fact, in $\mathcal{G}^*$ we can find homogeneous groups of courses belonging to macro-areas, thus accounting for disciplines belonging to the same field of study. Experts identified 4 different clusters $c_1 =$ *Mathematical and physical foundations*, $c_2 =$*Computer science*, $c_3 =$*Automation* and $c_4 =$*other*. Courses {*Mathematical Analysis 1*, *Mathematical Analysis 2*, *Linear Algebra and Geometry*, *General Physics 1*, *General Physics 2*, *Numerical Analysis*, *Algebra and Logics*} belong to c_1; {*Fundamentals of Computer Science*, *Operating Systems*, *Computer Architecture and Cloud Computing*, *Web Technologies*, *Object-oriented Programming*, *Databases*, *Software Engineering*, *Mobile Programming*} belong to c_2; {*Fundamentals of*

Automatics, *Industrial Automation*, *Automatic Control*, *Automation Laboratory*, *Robotics and Automation Technologies*, *Modeling and Identification of Dynamic Processes*, *Methods and Techniques for Automation*, *Computer Aided Control Design*} belong to c_3. The LLM correctly finds the 66.66% of dependencies within each cluster, meaning that the majority of edges between courses of c_i are present in $\mathcal{G}$.

5 Conclusion and further works

This paper introduces a new approach to extract causal dependencies among courses, demonstrating its application to an Italian university degree program.

The study demonstrated that the DFG obtained using the LLM exhibits high precision but suffers from low recall, as some prerequisite relationships present in the reference graph are not detected by the LLM. This limitation is due to the varying and often insufficient level of detail in the course programs, making it challenging for the LLM to identify dependencies. On the other hand, professors, who have more vast and precise knowledge, proposed additional links. For this reason, providing to the LLM more comprehensive materials, such as more detailed syllabus, could help compensate for these limitations. Notably, the LLM also inferred additional dependencies that were not present in $\mathcal{G}^*$ but were reasonable. This suggests that LLMs can provide valuable insights that complement or refine existing knowledge.

In conclusion, it could be interesting to test the LLM by providing it with more comprehensive information beyond just the official course programs. This is especially relevant in dynamic online learning systems, where course offerings change dynamically. Furthermore, this approach could support future research to understand whether the courses offered within a study program are sufficient to cover all the required prerequisites. In this work, the focus has been on identifying dependencies between the courses while ensuring that the LLM does not generate new course names. However, an extension of this study would be to assess whether the set of courses available in a program adequately covers the fundamental knowledge required for all the courses in the program. This means that the methodology could also play a crucial role in designing an entire educational curriculum for schools and universities other than optimizing the structure of an already existing study program, ensuring that it provides a coherent and comprehensive learning path.

References

1. Ban, T., Chen, L., Lyu, D., Wang, X., Zhu, Q., Chen, H.: Llm-driven causal discovery via harmonized prior. IEEE Transactions on Knowledge and Data Engineering (2025)
2. Ban, T., Chen, L., Wang, X., Chen, H.: From query tools to causal architects: Harnessing large language models for advanced causal discovery from data. arXiv preprint arXiv:2306.16902 (2023)

3. Berti, A., Kourani, H., van der Aalst, W.M.: Pm-llm-benchmark: Evaluating large language models on process mining tasks. arXiv preprint arXiv:2407.13244 (2024)
4. Berti, A., Kourani, H., Häfke, H., Li, C.Y., Schuster, D.: Evaluating large language models in process mining: Capabilities, benchmarks, and evaluation strategies. In: International Conference on Business Process Modeling, Development and Support. pp. 13–21. Springer (2024)
5. Berti, A., Qafari, M.S.: Leveraging large language models (llms) for process mining (technical report). arXiv preprint arXiv:2307.12701 (2023)
6. Berti, A., Schuster, D., van der Aalst, W.M.: Abstractions, scenarios, and prompt definitions for process mining with llms: A case study. In: International Conference on Business Process Management. pp. 427–439. Springer (2023)
7. Buss, A., Kratsch, W., Schmid, S.J., Wang, H.: Processllm: A large language model specialized in the interpretation, analysis, and optimization of business processes. In: International Conference on Business Process Management. pp. 221–232. Springer (2024)
8. Chen, L., Ban, T., Wang, X., Lyu, D., Chen, H.: Mitigating prior errors in causal structure learning: Towards llm driven prior knowledge. arXiv preprint arXiv:2306.07032 (2023)
9. Feng, T., Qu, L., Tandon, N., Li, Z., Kang, X., Haffari, G.: From pre-training corpora to large language models: What factors influence llm performance in causal discovery tasks? arXiv preprint arXiv:2407.19638 (2024)
10. Hadi, M.U., Qureshi, R., Shah, A., Irfan, M., Zafar, A., Shaikh, M.B., Akhtar, N., Wu, J., Mirjalili, S., et al.: A survey on large language models: Applications, challenges, limitations, and practical usage. Authorea Preprints **3** (2023)
11. Kampani, S., Hidary, D., van der Poel, C., Ganahl, M., Miao, B.: Llm-initialized differentiable causal discovery. arXiv preprint arXiv:2410.21141 (2024)
12. Khatibi, E., Abbasian, M., Yang, Z., Azimi, I., Rahmani, A.M.: Alcm: Autonomous llm-augmented causal discovery framework. arXiv preprint arXiv:2405.01744 (2024)
13. Kiciman, E., Ness, R., Sharma, A., Tan, C.: Causal reasoning and large language models: Opening a new frontier for causality. Transactions on Machine Learning Research (2023)
14. Long, S., Schuster, T., Piché, A.: Can large language models build causal graphs? arXiv preprint arXiv:2303.05279 (2023)
15. Redis, A.C., Sani, M.F., Zarrin, B., Burattin, A.: Processtbench: An llm plan generation dataset for process mining. arXiv preprint arXiv:2409.09191 (2024)
16. Vashishtha, A., Reddy, A.G., Kumar, A., Bachu, S., Balasubramanian, V.N., Sharma, A.: Causal inference using llm-guided discovery. arXiv preprint arXiv:2310.15117 (2023)
17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
18. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems **35**, 24824–24837 (2022)
19. Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al.: A survey of large language models. arXiv preprint arXiv:2303.18223 **1**(2) (2023)
20. Zheng, H.S., Mishra, S., Chen, X., Cheng, H.T., Chi, E.H., Le, Q.V., Zhou, D.: Take a step back: Evoking reasoning via abstraction in large language models. arXiv preprint arXiv:2310.06117 (2023)