AIDE: Automated Influence Diagram Extraction with Large Language Models

Yifan Hong[1]*, **Sizhong Qin** [2]*,
[1] 2023310542, Department of Industrial Engineering, Tsinghua University
[2] 2023310083, Department of Civil Engineering, Tsinghua University

Abstract

We propose to carry out a **research project**. Our work will focus on structured knowledge extraction from unstructured text. Specifically, we try to construct an influence diagram (ID) from the text, including a graphical structure and related conditional probabilities. Our work will lead to more efficient and trustworthy structured knowledge representation for downstream risky decision-making.

Keywords: structured knowledge extraction, LLMs, risk communication, decision support.

1 Introduction

Decision-making in the real world involves uncertainty and trade-offs and requires expertise from different fields. For example, resource allocation decisions during natural disasters are influenced by uncertain events like secondary disasters, and trade-offs between life-saving and the security of rescue personnel.

Influence diagram (ID, Howard and Matheson (2005)) is one of the systematic methods that are proposed to support decision-making. ID uses directed acyclic graphs (DAGs) for problem representation and is compatible with automated evaluation programs. It provides a principled framework not only for individuals' decision-making under uncertainty (Edwards et al., 2007) but also for risk sharing (Pennock and Wellman, 2005) and team collaboration (Detwarasiti and Shachter, 2005).

One of the challenges for ID is the construction process. Traditionally, IDs are constructed manually, with probabilities and reward structures assigned by experts. The manual construction process can be slow and costly. Besides, subjective assessments of uncertainty are often subject to mistrust. A more efficient way of constructing IDs will help spread this useful tool to more scenarios.

Large language models (LLMs) have the potential to ease the pain of the manual construction process. They have been successfully applied to tasks like knowledge graph construction (Ye et al., 2022), material knowledge extraction (Polak and Morgan, 2024), and general relation extraction (Wadhwa et al., 2023) tasks. However. it remains unexplored whether LLMs can capture conflicting objectives and the context-dependent state of uncertainty, which are crucial for successful decision-making.

In this work, we explore the power of LLMs in extracting IDs from unstructured text. The model needs to identify stakeholders and goals, related factors, and valid actions, along with quantitative relations between them. We aim to propose a general method for the structured knowledge extraction task. We manually annotated influence diagrams from texts in codes for structural design. Our method is able to achieve reliable performances on these texts of high expertise. We also use the case study to showcase the effectiveness of ID for better decision-making.

Contributions Our contributions are three-fold:

(1) We propose a new information extraction task, influence diagram extraction, for language models.

(2) We investigate whether LLM can effectively identify key variables and qualitative influence relations, and handle expressions of value and uncertainty.

(3) We further study the effect of different prompts on the model performance.

2 Background

2.1 Information extraction

Natural language expressions can be highly unstructured and include ambiguity. For example, there may be pronouns in a sentence, the meaning of which has to be inferred from the context. Information extraction aims to from an exhaustive

*Yifan Hong and Sizhong Qin contributed equally.

set of structured knowledge based on unstructured text data. Information extraction can be formulated as a generative task and solved with end-to-end auto-regressive models (Josifoski et al., 2022; Ye et al., 2022). The generative approach continues to gain more attention in the era of LLMs (Wadhwa et al., 2023).

LLMs for structured information extraction LLMs are powerful in-context learners with amazing zero-shot abilities (Brown et al., 2020; Kojima et al., 2022). They have been applied to structured information extraction from unstructured text with the development of the generative information extraction paradigm. For example, Wu et al. (2024) leverages the zero-shot capability of LLMs to construct a Chinese medical knowledge graph. To maximize the power of LLMs, prompting techniques like few-shot prompting or chain-of-thought prompting (Wei et al., 2022) have been proposed. These methods do not make changes to the model parameters and thus are cost-effective and convenient for use.

Prompt-engineering techniques Prompt-based methods have been successful in extracting information from text (Wei et al., 2023; Polak and Morgan, 2024). Typically, they design schemas or question-answering workflows so that existing LLMs can better handle the extraction of complex relations, and use self-reflection to correct imprecise information. For example, Wei et al. (2023) propose a two-stage multi-turn question-answering framework with chained prompt templates to overcome the shortcomings of one-time prediction when multiple dependent elements are present, achieving impressive performance on information extraction tasks.

Overall, LLMs have great potential in extracting structured information. However, for risk analysis, it remains to be seen whether LLMs can handle concerns of different stakeholders and uncertainty properly.

2.2 Influence Diagram

Influence diagram (ID) is a graphical model for decision analysis (Howard and Matheson, 2005). It provides both a formal description of the decision problem that can be handled by computers and an understandable representation for people of different technical proficiency. fig. 1 provides a minimal example; two manually constructed examples are shown in the Appendix.

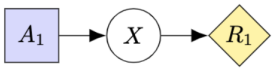

Figure 1: An example of an influence diagram.

Formally, an influence diagram $G = (N, A)$ is a directed acyclic graph that consists of nodes N and arcs A. Nodes correspond to variables and are divided into three types. Chance nodes C represent aleatory variables, decision nodes D represent variables that can be intervened, and utility nodes U reflect the goal of the decision-maker. Arcs represent information, statistical relevance or functional relationships between nodes. An arc $(x, y) \in A$ means x, y can be dependent, i.e. given current state of knowledge S over all other variables,

$$p(x, y|S) = p(x|S)p(y|x, S).$$

Notice that this expression is generally true, and the decomposition can be changed, which means that the arrow can be reversed as long as no loops are formed. Meanwhile, absence of an arc between two nodes suggests conditional independence. In this way, ID specifies a decomposition of the joint probability over the chance nodes C. Given the current state of knowledge S,

$$P(C|S) = \prod_{x \in C} p(x|\text{PA}_x, S), \tag{1}$$

where $\text{PA}_x = \{y \in N : (y, x) \in A\}$ is the set of direct predecessors (parents) of x.

IDs can be manipulated and transformed to decision trees, for which an optimal solution an be derived. Combined with convenient graph manipulation methods (e.g., reverse arcs), ID provides valuable insights for decision-making.

Although IDs have a strong representation power, certain disadvantages hinder their widespread application. Construction of IDs requires assignments of conditional probabilities. Traditionally, these distributions are assigned by human experts, making the process slow and costly. Also, subjective evaluations can be subject to human biases.

3 Methodology

3.1 Extraction framework

We propose to extract an ID $G = (N, A)$ from unstructured text T. The language model accomplishes the extraction task by performing next-

token prediction. The probability of G being implied by the text can be decomposed as

$$\begin{aligned} p_M(G|T) &= p_M(N|T)p_M(A|N,T) \\ &= p_M(N|T)\prod_{x\in N} p_M(\text{PA}_x|N,T) \end{aligned} \quad (2)$$

where the subscript M suggests the probability resides in the model. The second equation decomposes the set of arcs according to the nodes.

Beyond the graphical structure, a key component of IDs is the conditional probability, which, as shown in eq. (2), can be a complex multi-to-one quantitative relation. We propose using mean-field approximations to these conditional probabilities. Therefore, the language model predicts

$$p_M(G|T) = p_M(N|T)\prod_{x\in B}\prod_{y\in \text{PA}_x} p_M((y,x)|N,T) \quad (3)$$

We design the extraction process in two stages according to the decomposition eq. (2). In the first stage, we identify key variables and extract the nodes N from the text. In the second stage, (probabilistic) dependence relations are identified as arcs between the nodes extracted in stage 1.

Schema A schema must be defined for the extraction of structured components of the ID. We use JSON format and treat the nodes and arcs separately. We record a node's name, type (information, decision, or utility node), and valid values. Arcs are represented with condition, variable, and the conditional probability. eq. (3) implies the condition is a single node, since the distribution is fully factorized. For probability, we use a nested dictionary to record the value of the condition, the value of the variable, and the probability of the variable value. Verbal expressions of probability are allowed, which can be transformed into numerical values according to certain rules.

3.2 Extraction methods

Prompting method In each prompt, we pass in the instructions and format instructions along with other input. For this challenging extraction task, different prompting techniques beyond zero-shot instruction is considered.

For node extraction, we consider few-shot prompting and chain-of-thought (CoT) prompting. Few-shot prompts includes a handful of examples of nodes extracted from a short text. We deliberately manipulate the types of nodes in the few-shot examples. CoT prompting can be implemented in a zero-shot manner, by adding "let's think step by step". We further add explicit instructions of extraction process, asking the model to extract utility nodes first, then decision nodes, and chance nodes at last.

Post processing LangChain is utilized for output parsing (Chase, 2022). When parsing error happens, we give the original completion with invalid format and the format instruction to a LLM, and ask it to fix the invalid format.

Extracted nodes can be overlapping. We use LLM to merge nodes that correspond to identical variables.

4 Experiments

4.1 Dataset

We manually annotated the 500 text chunks from codes for structural design, among which over 100 contain meaningful nodes. For each node, the annotation follows the format instruction, and includes variable names, variable types and their values.

4.2 Node extraction

4.2.1 Evaluation

Evaluation faces the challenge of synonymous expressions. A variable can have many potential names, and values may be expressed in many ways. A valid evaluator should have a good understanding of synonymy. Therefore, we resort to human evaluation for reliability in the presence of synonymous expressions of node names. However, human evaluation can be very time-consuming. As an alternative, we also consider using another language model to evaluate the performance of node extraction. We display results of the two evaluation methods, which turn out to be similar.

Human evaluation We consider an extracted variable (name) as *positive*, and all other potential variables that are not present as *negative*. Successfully extracted nodes, which appear in the annotation, correspond to true positive (TP). Missing nodes, which appear in the annotation but not in the extraction, correspond to false negative (FN). Wrong nodes, which appear in the extraction but not in the annotation, correspond to false positive (FP). We can then calculate *precision* and *recall* from these metrics. For successfully extracted nodes, the *accuracy* of node type is evaluated. For values of successfully extracted nodes, we similarly

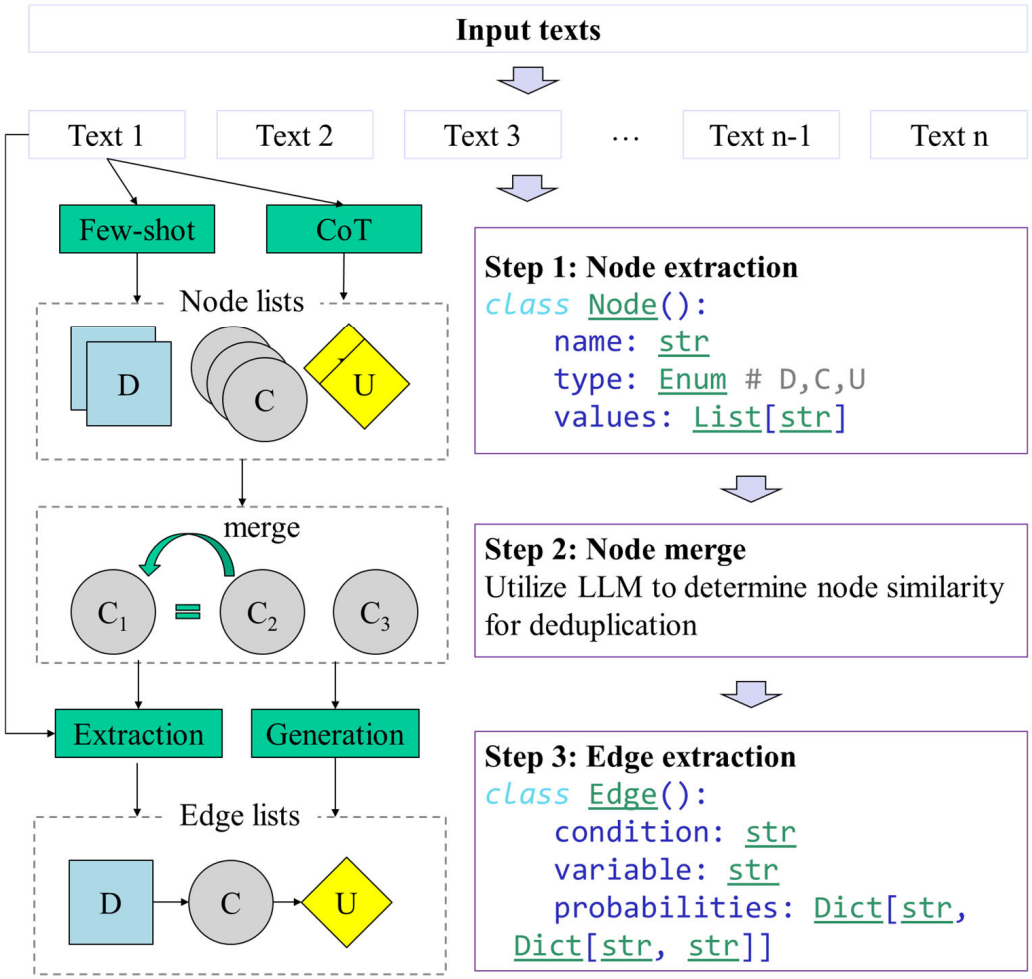

Figure 2: Flow chart of the extraction process and illustration of the schema.

evaluate by recording: (1) TP, (2) FN, (3) FP, and calculate precision and recall.

LLM evaluation To enhance evaluation efficiency, an LLM evaluator was constructed. The primary task of the LLM is to compare two lists consisting of different text elements. If an element in one list has the same meaning as an element in the other list, the LLM outputs a tuple of these two elements. Based on this capability, the ground truth and extracted node lists can first be compared by their names. If two nodes are deemed identical, the LLM evaluator can further compare the values of these nodes. Additionally, node types, which have only three choices, can be compared directly.

4.2.2 Experiment design

We designed six different types of prompts: 0-shot, 1-shot, 3-shot (single), 3-shot (complete), Zero-shot CoT, and Instruction CoT. Here, "shot" indicates the number of examples provided; "single" means the examples contain only one type of node; "complete" means the examples contain all types of nodes; "CoT" stands for Chain of Thought, and "instruction" refers to providing a specific CoT reasoning method.

For both human evaluation and LLM evaluation methods, we record the mean metrics along with the 95% confidence intervals. For detailed results, see table 1 and table 2, where the best possible methods are highlighted in bold.

Finally, we compared the differences between human evaluation and LLM evaluation by using correlation coefficients and Spearman's rank correlation to assess each metric, exploring the disparities between human and LLM evaluations. The result is shown in table 3.

4.2.3 Results

The 3-shot prompt with all kinds of nodes generally exhibits the best performance, particularly in terms of name recall, type accuracy and values recall as demonstrated in table 1 and table 2. The

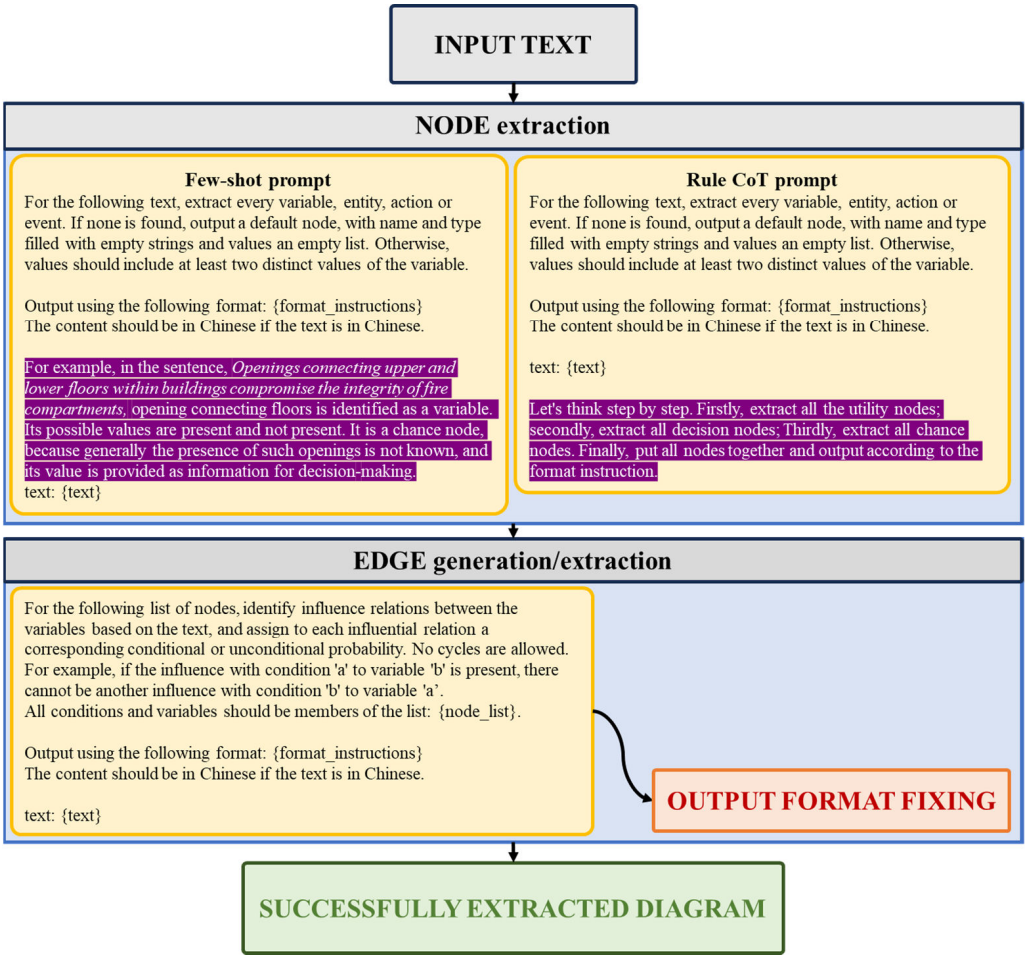

Figure 3: A flow chart of prompting method of extracting influence diagram from text.

more cases provided, the better the results obtained. Providing cases that cover a wider range of node types also demonstrates better performance. This phenomenon is particularly evident in human evaluations.

The Zero-shot CoT prompt method exhibits poor recall in both tables, indicating its ineffectiveness in capturing more information. It is likely due to the extraction task lacking a clear sequential logic, necessitating the identification of various nodes within the text and analyzing their roles. However, after adding the specific instruction, the CoT prompt performs significantly better, approaching the performance of the 3-shot prompt with all types of nodes.

When using LLM as evaluator, the differences in metrics are not as pronounced as with human annotations. However, it is still generally observed that few-shot methods have some impact. Adding specific instructions in CoT also enhances the final results to a certain extent.

Further discussing the differences between the two methods, we observe from table 3 that the correlation coefficients and Spearman's rank correlation for name recall, type accuracy, and values recall are all very close to 1. This indicates that the two methods yield similar results for these three metrics. For precision, which inherently has a smaller variance, there is some discrepancy between the two evaluation methods, but they are still positively correlated.

4.3 Edge extraction

4.3.1 Evaluation

Graph edit distance The evaluation of the topology structure of the extracted diagram is performed using graph edit distance (Sanfeliu and Fu, 1983). Graph edit distance quantifies the minimum number of edit operations required to transform one graph into another, providing a measure of the dis-

Table 1: Node extraction performance evaluated by human

	name		type	values	
prompt	precision	recall	accuracy	precision	recall
0-shot	**0.7872±0.0494**	0.7276±0.0456	**0.7559±0.0542**	0.8675±0.0476	0.8388±0.0485
1-shot	**0.7871±0.0476**	0.7409±0.0422	0.7234±0.0573	0.8884±0.0388	0.8714±0.0400
3-shot (single)	**0.7413±0.0459**	**0.8048±0.0380**	0.6862±0.0483	**0.9253±0.0243**	**0.9102±0.0256**
3-shot (complete)	**0.7519±0.0436**	**0.8362±0.0349**	**0.7818±0.0467**	**0.9400±0.0223**	**0.9285±0.0232**
Zero-shot CoT	**0.7503±0.0539**	0.5347±0.0502	0.6000 ±0.0761	0.7658±0.0597	0.7359±0.0598
Instruction CoT	**0.7766±0.0457**	**0.8384±0.0446**	**0.7435±0.0448**	**0.9384±0.0299**	0.8695±0.0466

Table 2: Node extraction performance evaluated by LLM

	name		type	values	
prompt	precision	recall	accuracy	precision	recall
0-shot	**0.7926±0.0412**	0.7337±0.0391	**0.7156±0.0558**	**0.8916±0.0427**	**0.8681±0.0452**
1-shot	**0.8128±0.0403**	0.7730±0.0378	**0.6921±0.0572**	**0.9207±0.0374**	**0.9110±0.0405**
3-shot (single)	**0.7653±0.0445**	**0.8335±0.0371**	**0.6710±0.0491**	**0.8948±0.0413**	**0.8948±0.0413**
3-shot (complete)	**0.7455±0.0422**	**0.8499±0.0356**	**0.7432±0.0538**	**0.9175±0.0395**	**0.9223±0.0385**
Zero-shot CoT	**0.7973±0.0501**	0.5994±0.0540	0.5554±0.0760	**0.8738±0.0536**	**0.8576±0.0549**
Instruction CoT	**0.7348±0.0486**	**0.8025±0.0462**	**0.6668±0.0588**	**0.8803±0.0545**	**0.8576±0.0558**

similarity between the two graphs. A smaller graph edit distance indicates a higher degree of similarity between the graphs.

The basic graph edit operations include: (1) **node insertion:** to introduce a single new labeled node. (2) **node deletion:** to remove a single node from a graph. (3) **node substitution:** to change the label of a given node. (4) **edge insertion:** to introduce a new edge between a pair of nodes. (5) **edge deletion:** to remove a single edge between a pair of nodes. (6) **edge substitution:** to change the label of a given edge.

Different weights for edit operations affect the graph edit distance. For our graph, only the name is initially considered as the label of the given node, with no labels assigned to edges. All operations has the same cost of 1.

In future, more labels of nodes and edges can be introduced to achieve a more accurate evaluation.

Human evaluation For more fine-grained evaluation of edges, we can similarly use *precision* and *recall* to evaluate extraction quality. Meanwhile, a critical ingredient of influence diagrams is conditional probabilities. For each successfully extracted edge, human evaluators (1) move a slider to locate the extracted expression on a pre-determined scale, and (2) judge whether the extracted conditional probabilities match the annotated probabilities.

4.3.2 Experiment design

After generating the nodes, there are two methods to generate edges: "generate," which infers causal relationships directly from the nodes, and "extract," which involves inputting both the original text and the nodes to allow the LLM to infer their causal relationships. Additionally, we considered four prompt designs: 0-shot, 1-shot, 3-shot, and zero-shot CoT to explore the impact of different prompt designs on the results. The final results were evaluated using graph edit distance, as detailed in table 4

4.3.3 Results

From the results in table 4, it can be seen that the extraction method, which simultaneously extracts from the text and nodes, yields better results. This is because there are many potential causal relationships in the text that facilitate LLM learning. Additionally, the 3-shot method performs better, further validating the effectiveness of few-shot learning. Due to time constraints, we have not yet completed the human evaluation, nor have we further assessed more details in edges. These tasks can be addressed in future work.

5 Case study: risky decision-making

In this section, we present the text description of a toxic-chemical usage problem adapted from Howard and Matheson (2005). The case is shown

Table 3: Comparison of two evaluation methods

	name		type	values	
	precision	recall	accuracy	precision	recall
Human variance	0.0004	0.0130	0.0042	0.0044	0.0047
LLM variance	0.0010	0.0084	0.0042	0.0004	0.0008
Correlation	0.3331	0.9715	0.9512	0.4879	0.7136
Spearman correlation	0.2000	0.8286	0.8286	0.4286	0.8510

Table 4: Edge extraction/generation performance

	graph edit distance	
prompt	generated	extracted
0-shot	3.3398	2.6765
1-shot	3.2330	2.6373
3-shot	3.2524	**2.4356**
Zero-shot CoT	3.5146	2.5882

in table 5, with names of "ground-truth" nodes in bold and underlined. There are six nodes in total, where "usage" is the only decision node.

Input: text for extraction
Let us suppose that a chemical having some benefits is also suspected of possible carcinogenicity. We wish to determine whether to ban, restrict, or permit its use. The **economic value** of the product and the **cancer cost** attributed to it both depend on the decision regarding **usage** of the chemical. The economic value given the usage decision is independent of the **human exposure**, **carcinogenic activity**, and the cancer cost. However, the cancer cost is dependent on the usage decision as well as on both the carcinogenic activity and human exposure levels of the chemical. The **net value** of the chemical given the economic value and the cancer cost is independent of the other variables. Also, human exposure and carcinogenic activity are independent.

Table 5: Illustration of the case study.

Using `moonshot-8k-v1`, the proposed method extract all six nodes successfully, with slightly different names. It identified three nodes as utility nodes, but considering only net value can be enough. This shows that the model is capable of identifying key variables, but still lack the capability of choosing utilities properly. This can be a result of the current task formulation, i.e. extraction, and may be resolved when the target of decision-making is emphasized more.

For edges, we compare extraction and generation. The final result of extracted influence diagrams are shown in fig. 4 and fig. 5, respectively. For this simple case, extraction method performs well, extracting all mentioned relations without adding unnecessary arcs between nodes. LLMs can be reliable in extracting qualitative relations from text.

On the other hand, generation method makes some mistakes in the relation between identified utility nodes. However, it successfully generated arcs between the chance nodes and the node *cancer cost*. It is amazing that since only information of nodes is available, the model is completing the graph with its inherent knowledge, and it's only making justifiable mistakes. This implies the potential of LLMs in completing graphical models when some relations are not mentioned in the text.

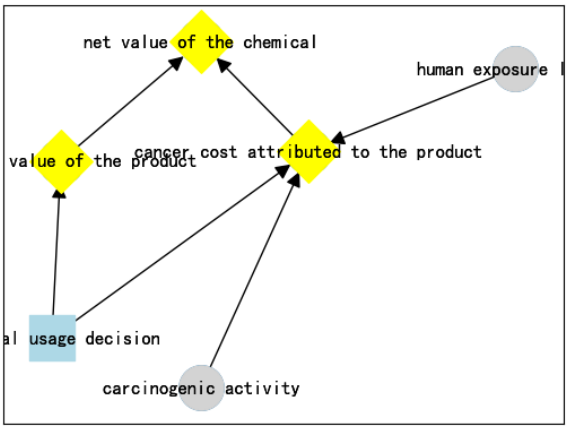

Figure 4: Influence diagram with nodes and edges extracted from text.

6 Discussion

Decision-making in the real world involves two levels of capabilities. The lower level, which is perception, requires effective information process-

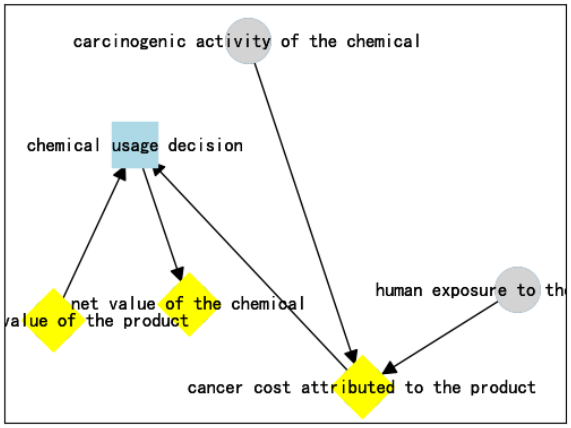

Figure 5: Influence diagram with nodes extracted from text and generated edges.

ing, a task that is well-handled by state-of-the-art foundation models with impressive multi-modal abilities. The higher level, deliberation, involves analyzing highly abstract information for optimal decision-making. Extensive research in optimization theory and algorithms is aimed at solving this problem. In recent years, lines are blurring between research in optimization, especially decision-making (e.g., reinforcement learning), and foundation models (Yang et al., 2023). It will be intriguing to investigate how to enable large language models to solve real-world problems autonomously. Our work, with a focus on risky decision modeling, makes a meaningful first step.

With an ID at hand, LLMs are only one step from autonomous decision-making. By plugging a graph manipulation module, the model can find the optimal choice. However, one remaining challenge is model debugging, since extracted graphical models will contain certain errors. For example, a loop may be present and lead to failed parsing. Introducing Socratic method QA by including extra rounds of questions in the prompt may solve this problem. Taking a step further, using LLM to empower agents to design effective modeling workflows makes a promising research direction.

Reliability and explainability are important issues if LLM-based decision models or agents are to be used in the real world. Decision-making copilot has been envisioned to democratize mathematical optimization in business decision-making (Wasserkrug et al., 2024). However, things are different in risky decision-making, where high risk and huge losses may be involved. An important question to investigate is how to make LLM-based decision support systems reliable, trustworthy, and transparent.

6.1 Limitations

Currently, the current extraction workflow is manually designed, and considers only one kind of problem representation. However, the current schemas may not be *optimal*. For example, LLMs do not handle the nested dictionary structure well, so using a table format may be a better choice. Besides, the method tends to confuse value with tendency. For example, a valid value for probability is "high", while the model may depend too much on the text and extract "increase". This may be solved by incorporating clear instructions for distinguishing values.

6.2 Future work

In the future, we aim to develop a specialized agent in the structural domain with knowledge memory and decision-making capabilities, as illustrated in fig. 6 . Using this architecture, various tasks in the structural field can be decomposed into several decision-making problems. Some decisions can be directly made based on standard regulations, while others can be made using influence diagrams. The toolkit can include our current intelligent design algorithms and structural calculation software, thereby enabling decision-making for structural design problems.

7 Conclusion

In this study, we proposed and evaluated a methodology for automated influence diagram extraction using large language models (LLMs). Our goal was to streamline the process of creating influence diagrams (IDs) from unstructured text, a task traditionally performed manually and fraught with inefficiencies and potential biases. By leveraging the capabilities of LLMs, we aimed to improve the efficiency and accuracy of ID construction, thereby facilitating better decision-making under uncertainty.

Our approach involved developing a structured extraction framework that operates in two stages: identifying key variables and extracting nodes from the text, followed by identifying probabilistic dependence relations as edges between these nodes. We implemented various prompting techniques, including few-shot prompting and CoT prompting, to enhance the performance of the LLMs.

Key findings

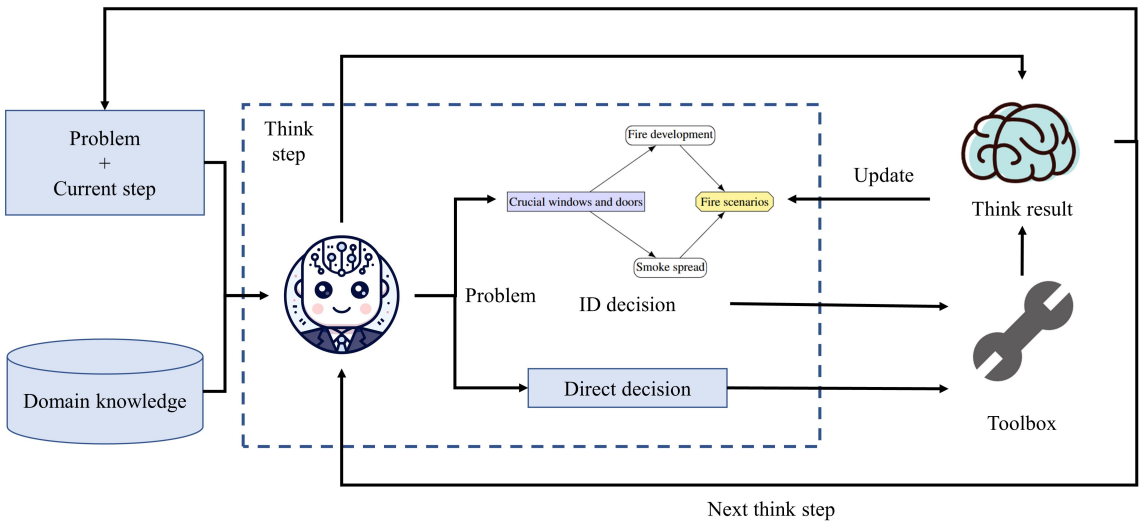

Figure 6: An engineer agent for decision making

- **Prompting Techniques**: The study demonstrated that few-shot and CoT prompting techniques significantly improve the extraction performance of LLMs. Specifically, the 3-shot (complete) prompting method exhibited the best performance in terms of name recall, type accuracy, and values recall. This finding underscores the importance of providing diverse and comprehensive examples to the model.

- **Evaluation Methods**: We conducted a comparative analysis of human and LLM-based evaluations. The results indicated a high correlation between human and LLM evaluations for most metrics, suggesting that LLMs can reliably evaluate the performance of node extraction. However, some discrepancies were observed in precision metrics, highlighting areas for further refinement in evaluation techniques.

- **Generation or Extraction**: The study showed that the extraction method, which integrates text and node information, yields superior results compared to methods that infer causal relationships solely from nodes. This approach leverages the contextual information within the text, facilitating better identification of relevant relationships.

In conclusion, the automated extraction of influence diagrams using LLMs represents a promising advancement in the field of structured knowledge extraction. Our study provides a foundation for future research and development, with the potential to significantly impact decision-making processes across various domains.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Harrison Chase. 2022. Langchain.

Apiruk Detwarasiti and Ross D Shachter. 2005. Influence diagrams for team decision analysis. *Decision Analysis*, 2(4):207–228.

Ward Edwards, Ralph F Miles, and Detlof Von Winterfeldt. 2007. Advances in decision analysis. *Cambridge, New York*.

Ronald A Howard and James E Matheson. 2005. Influence diagrams. *Decision Analysis*, 2(3):127–143.

Martin Josifoski, Nicola De Cao, Maxime Peyrard, Fabio Petroni, and Robert West. 2022. Genie: Generative information extraction. In *Proceedings of*

the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4626–4643.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

David M Pennock and Michael P Wellman. 2005. Graphical models for groups: Belief aggregation and risk sharing. *Decision Analysis*, 2(3):148–164.

Maciej P Polak and Dane Morgan. 2024. Extracting accurate materials data from research papers with conversational language models and prompt engineering. *Nature Communications*, 15(1):1569.

Alberto Sanfeliu and King-Sun Fu. 1983. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3):353–362.

Somin Wadhwa, Silvio Amir, and Byron C Wallace. 2023. Revisiting relation extraction in the era of large language models. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2023, page 15566. NIH Public Access.

Segev Wasserkrug, Leonard Boussioux, Dick den Hertog, Farzaneh Mirzazadeh, Ilker Birbil, Jannis Kurtz, and Donato Maragno. 2024. From large language models and optimization to decision optimization copilot: A research manifesto. *arXiv preprint arXiv:2402.16269*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. 2023. Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*.

Ling-I Wu, Yuxin Su, and Guoqiang Li. 2024. Zero-shot construction of chinese medical knowledge graph with gpt-3.5-turbo and gpt-4. *ACM Transactions on Management Information Systems*.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. 2023. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*.

Hongbin Ye, Ningyu Zhang, Hui Chen, and Huajun Chen. 2022. Generative knowledge graph construction: A review. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1–17.

A Example Appendix

Example 1: "The unauthorized opening of crucial doors and windows by untrained personnel or temporary visitors can result in the rapid development of fires and the spread of smoke. These actions can create new potential fire scenarios."

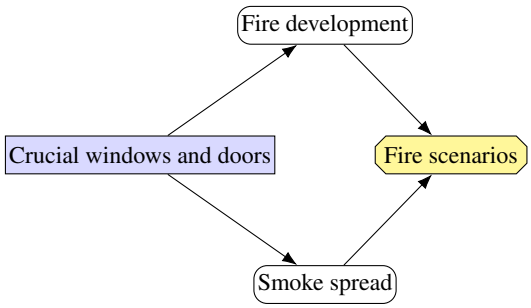

Figure 7: Influence diagram for example 1.

Example 2: "Openings connecting upper and lower floors within buildings compromise the integrity of fire compartments, potentially leading to the spread of fire across multiple areas and floors. Therefore, reliable fire separation measures should be implemented in these connected spaces to prevent the rapid upward spread of fire."

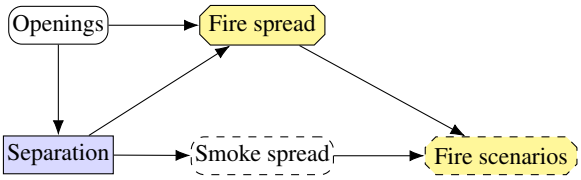

Figure 8: Influence diagram for example 2.