(APPENDIX)
Knowledge Graph Structure as Prompt: Improving Small Language Models Capabilities for Knowledge-based Causal Discovery

No Author Given

No Institute Given

The subsequent sections constitutes the appendix.

A Baseline (`GPT-3.5-turbo-instruct`) hyperparameters

In this work, we used OpenAI[1] API with `gpt-3.5-turbo-instruct` engine. In general we assume only query access to the LLMs (i.e., no gradients, no log probabilities). Table 1 summarizes the hyperparameter values for the ICL model experiment with `gpt-3.5-turbo-instruct`.

Table 1. Hyperparameter values.

hyperparameter	value
`temperature`	0.7
`max token`	$100 \sim 400$
`top p`	1
`frequency penalty`	0
`presence penalty`	0

B MLM hyperparameters settings

We used the `biomed_roberta_base`[2] for all biomedical datasets (COMAGC, GENEC, and DDI), and `roberta-base`[3] for the SEMEVAL dataset. We used both models from the `huggingface` models library. All models are implemented in Python with `Pytorch`[4] and `Transfomer`[5] library. The random seed of 203 is set for all experiments. Table 2 summarizes the hyperparameter values for fine-tuning the model experiments.

[1] https://platform.openai.com/
[2] https://huggingface.co/allenai/biomed_roberta_base
[3] https://huggingface.co/FacebookAI/roberta-base
[4] https://pytorch.org/
[5] https://huggingface.co/docs/transformers/en/index

Table 2. MLM hyperparameter values.

hyperparameter	COMAGC	GENEC	DDI	SEMEVAL
batch size	8	8	8	8
max length	256	256	256	256
optimizer	Adam	Adam	Adam	Adam
lr	3e-5	3e-5	3e-5	3e-5
gradient acc. steps	4	4	4	4
adam eps.	1e-06	1e-06	1e-06	1e-06
warmup proportion	0.06	0.06	0.06	0.06
weight decay	0.01	0.01	0.01	0.01
max grad norm	1	1	1	1
epoch	30	30	30	20

C CLM/decoder-only hyperparameters settings

In all evaluation experiments, the `bloomz-560m`[6] model from the `huggingface` models library were used for all datasets. All models are implemented in Python with `Pytorch`[7] and `Transfomer`[8] library. The random seed of 203 is set for all experiments. Table 3 summarizes the hyperparameter values for fine-tuning the `bloomz-560m` model experiments.

Table 3. `bloomz-560m` hyperparameter values.

hyperparameter	COMAGC	GENEC	DDI	SEMEVAL
batch size	4	4	4	4
max length	256	256	256	256
optimizer	AdamW	AdamW	AdamW	AdamW
lr	3e-5	3e-5	3e-5	3e-5
gradient acc. steps	4	4	4	4
warmup proportion	0.06	0.06	0.06	0.06
max grad norm	1	1	1	1
epoch	30	10	15	20

D Seq2SeqLM hyperparameters settings

In all evaluation experiments, the `t5-base`[9] model from the `huggingface` models library were used for training all datasets. All models are implemented in Python

[6] https://huggingface.co/bigscience/bloomz-560m
[7] https://pytorch.org/
[8] https://huggingface.co/docs/transformers/en/index
[9] https://huggingface.co/google-t5/t5-base

with `Pytorch`[10] and `Transfomer`[11] library. The random seed of 203 is set for all experiments. Table 4 summarizes the hyperparameter values for fine-tuning the `t5-base` model experiments.

Table 4. `t5-base` hyperparameter values.

hyperparameter	COMAGC	GENEC	DDI	SEMEVAL
`batch size`	8	4	8	8
`max length`	256	256	256	256
`optimizer`	`AdamW`	`AdamW`	`AdamW`	`AdamW`
`lr`	3e-3	3e-4	3e-4	3e-4
`gradient acc. steps`	4	4	4	4
`warmup proportion`	0.06	0.06	0.06	0.06
`max grad norm`	1	1	1	1
`epoch`	30	20	20	20

E Querying the KGs

We access the **Hetionet** KG through its official public Neo4j API[12], with `neo4j.v1` Python library. Neo4j[13] is a third-party graph database that supports the Cypher language for querying and visualizing a knowledge graph. Hetionet also provides a public Neo4j browser app[14].

As for the **Wikidata**, we access the KG through its official public SPARQL endpoint[15], with `SPARQLWrapper` Python library. We employ the official wikidata API (e.g., `wbsearchentities` and `wbgetentities` functions) for extracting the Wikidata IDs for all variable pairs.

On Hetionet, we query up to 4 hops for extracting the KGs structures. However, for Wikidata, we query up to one hops to extract the KG structures, constrained by its huge sizes. To train a robust models that generalize well given any KG structures, we opted to not optimize the content from the KG structures to be included in the prompt. For instance, when there are more than m metapaths for a pair, we randomly select m of them, m being a *hyperparameter* of the number of metapaths to be included as prompt.

In all experiments, we include up to the following: 4 neighbors nodes, 5 common neighbors nodes, and 1 metapath, to be included in the prompt.

[10] https://pytorch.org/
[11] https://huggingface.co/docs/transformers/en/index
[12] bolt://neo4j.het.io
[13] https://neo4j.com/
[14] https://neo4j.het.io/browser/
[15] https://query.wikidata.org/sparql

F Prompt hyperparameters and example

(to-be-added)

References