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

No Author Given

No Institute Given

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. LLMs hyperparameter values.

Parameter	Value
temperature	0.7
max_token	100∼400
top_p	1
frequency_penalty	0
presence_penalty	0

B MLM (`roberta-125m`) hyperparameters settings

In all evaluation experiments, batch sizes of 8 were used for training in all datasets. 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. 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

Table 2. `roberta-125m` hyperparameter values.

Parameter	Value
max_sequence_length	128, 256
epoch	10
optimizer	Adam
lr	2e-5
eps	1e-08
linear_warmup_proportion	0.1
dropout	0.1
$hidden\_layer(FC, GRU)$	1024
seed	1234

C CLM (`bloomz-560m`) hyperparameters settings

In all evaluation experiments, batch sizes of 4 were used for training in all datasets. We used the `bloomz-560m`[4] model from the `huggingface` models library for all datasets. Table 3 summarizes the hyperparameter values for fine-tuning the `bloomz-560m` model experiments.

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

Parameter	Value
max_sequence_length	128, 256
epoch	10
optimizer	Adam
lr	2e-5
eps	1e-08
linear_warmup_proportion	0.1
dropout	0.1
$hidden\_layer(FC, GRU)$	1024
seed	1234

D Seq2SeqLM (`t5-base`) hyperparameters settings

In all evaluation experiments, batch sizes of 8 were used for training in all datasets. We used the `t5-base`[5] model from the `huggingface` models library for all datasets. Table 4 summarizes the hyperparameter values for fine-tuning the `t5-base` model experiments.

[4] https://huggingface.co/bigscience/bloomz-560m
[5] https://huggingface.co/google-t5/t5-base

Table 4. `t5-base` hyperparameter values.

Parameter	Value
max_sequence_length	128, 256
epoch	10
optimizer	Adam
lr	2e-5
eps	1e-08
linear_warmup_proportion	0.1
dropout	0.1
$hidden\_layer(FC, GRU)$	1024
seed	1234

All models are implemented in Python with `Pytorch`[6] and `Transfomer`[7] library. The random seed of 1234 is set for all experiments.

E Querying the KGs

We access the **Hetionet** KG through its official public Neo4j API[8], with `neo4j.v1` Python library. Neo4j[9] 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[10].

As for the **Wikidata**, we access the KG through its official public SPARQL endpoint[11], 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.

F Prompt example

(to-be-added)

[6] https://pytorch.org/
[7] https://huggingface.co/docs/transformers/en/index
[8] bolt://neo4j.het.io
[9] https://neo4j.com/
[10] https://neo4j.het.io/browser/
[11] https://query.wikidata.org/sparql

References