Bearly Prompting: Extracting Prompts by Inverting LLM Outputs

A Tensorflow reimplementation of *output2prompt*: a memory-efficient black-box approach for recovering user and system prompts from LLM outputs

Logan Rabe, Rahul Mani, Xinyi Wang, Korgan Atillasoy

CSCI 1470 Spring 2025

Introduction

The paper *Extracting Prompts by Inverting LLM Outputs* inverts the LLM process through its method output2prompt. Instead of predicting an output from a user's prompt like most LLMs, output2prompt takes LLM outputs and predicts the prompt used to generate that output. Specifically, this is a structured prediction problem where the model outputs tokens given an LLM output. The paper considers both user and system prompts. Previous inversion methods used the LLM's logits (logit2prompt) which are both hard to access and computationally expensive when training, and adversarial/jailbreaking queries when predicting system prompts. Output2prompt eliminates the need for any of these while achieving higher accuracy and transferability to other datasets. This paper uses pytorch and we will be implementing it in tensorflow.

Preprocessing

We are using several datasets from the original paper with prompt or prompt-answer pairs to train the inversion model. For user prompts, we use、

Instructions-2M (with prompts only), ShareGPT(with both prompts and responses), and Unnatural Instructions. For system prompts we use Synthetic GPTs, Real GPTs and Awesome-ChatGPT-Prompts. There will be around 85k prompts and responses in total, and they can be used directly without preprocessing.

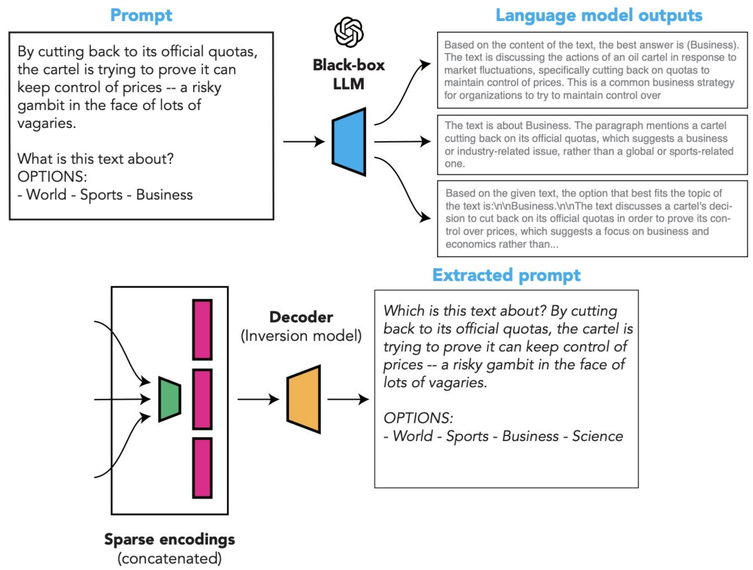
	Type	Total	Training	Finetuning	Testing
Instructions-2M	User	2M	30K		1K
ShareGPT	User	54K		100	400
Unnatural Instructions	User	240K		100	400
Synthetic GPTs	System	26K	25K		1K
Real GPTs	System	79		50	29
Awesome-ChatGPT-Prompts	System	153		50	103

Model Architecture

The model uses a pre-trained transformer encoder-decoder architecture. The default hyperparameters include a batch size of 8 per device and 1 epoch of training.

To predict prompts from outputs, the model first uses a sparse attention encoder. With the sparse encoder, each output only attends to itself, drastically reducing compute time. Using full attention takes about 4 times as long to run while only improving cosine similarity by 0.3 (96.7→97)

To prevent longer than desired prompts, the inversion model (decoder) limits sequence length to 64 tokens for user prompts and 256 for system prompts. It uses 222 million parameters.



Challenges

- We had issues running the code locally. Even though the paper is less than a year old, we ran into issues with the accelerate library changing since the release of the paper. Additionally, the default batch_size was too much for our computers' memory and the large dataset is too much to train on locally.
  - To solve these problems, we downgraded accelerate to version 0.21.0, changed the batch_size from 8 to 1, and trained on a small subset (usually 1%) of the data when running locally.
- This model uses the GPT-3.5 API to generate outputs for the prompt dataset and the GPT-4o API when calculating "LLM Eval" and cosine similarity accuracy. We do not have access to these API keys.
  - We do not train on any new data or use the LLM Eval/CS to evaluate our accuracy.

Results

User dataset	type	CS	BLEU	Token F1	LLM Eval
Instructions-2M	original	96.7	56.9	79.5	82.9
ShareGPT	original	84.2	2.5	29.5	43.8
Unnatural Instructions	original	83.3	4.4	36.0	63.0
Instructions-2M	implemented		51.2	73.2	
ShareGPT	implemented		11.0	40.3	
Unnatural Instructions	implemented		13.6	45.5	

System dataset	type	CS	BLEU	Token F1	LLM Eval
Synthetic GPTs	original	98.1	36.8	73.8	99.0
Real GPTs	original	89.9	6.4	37.6	65.5
Awesome-ChatGPT-Prompts	original	83.9	2.1	28.8	77.2
Synthetic GPTs	implemented		36.2	70.6	
Real GPTs	implemented		0.8	37.9	
Awesome-ChatGPT-Prompts	implemented		3.8	29.1	

Conclusion

- Discussion
  - Our user prompts overperformed on the test data despite less training– maybe there was overtraining in the paper?
  - We weren't quite able to get the results we wanted due to training on a smaller dataset and only 1 epoch instead of 3
  - Lack of CS / LLM Eval means we can't really tell how accurate our models are especially on other datasets
- Future Work
  - Continue creating and training on new datasets
  - Aim to have models that work well on multiple datasets
  - Continue innovating new loss-functions (e.g. non-OpenAI embeddings)