Bachelor Project

Czech Technical University in Prague

F3
Faculty of Electrical Engineering
Artificial Intelligence Center

Meta-prompts for LLM Prompt Optimization

abcd

Vojtěch Klouda

Supervisor: Ing. Jan Drchal PhD.
Field of study: Artificial Intelligence
Subfield: Natural Language Processing
May 2025

Acknowledgements

=)))

Declaration

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 10. May 2025

Abstract

TODO

Keywords: language model, optimization

Supervisor: Ing. Jan Drchal PhD.
Resslova 307/9 Praha, E-322

Abstrakt

TODO

Klíčová slova: jazykový model, optimalizace

Překlad názvu: abcd — abcd

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Background

In recent years large language models (LLMs) emerged as a revolutionary force not only in the field natural language processing (NLP) but in everyday life of the general population **cite smth**. The fundamental way to interact with these models is via text instructions or **prompts**. These prompts can be standalone or act as templates into which additional information, like a specific task or context, like from a database search, is inserted. This introduces the problem of **prompt engineering**, or finding the optimal prompt for a given situation. Prompt engineering emerges as a largely empirical field, with even experts experienced in the inner workings of LLMs relying on extensive trial-and-error bouts before finding a suitable prompt. With even experts struggling, it is significantly harder for non-experts to design an effective prompt. This motivates the recent research into automated prompt engineering. Several research directions have emerged, such as soft prompt tuning and discrete prompt optimization. Soft prompt tuning is effective but relies on access to inner states of the LLMs with are unavailable when using current state-of-the-art APIs such as the OpenAI API. Furthermore, prompts resulting from soft prompt tuning are often unreadable. In this work we focus on methods of discrete prompt optimization. This problem is very challenging because of the high dimesionality of the language space in which we conduct the search and non-differentiability as there are no gradients in the text space.

1.2 Objectives

The goal of this thesis is to compare several prompt optimization methods utilizing LLMs as the optimizing actors. That means, given a prompt p_t, where t denotes a step in the optimization process, the next prompt can be formulated as $p_{t+1} = \mathcal{L}_{iter}(p_t)$, where $\mathcal{L}_{iter}$ denotes an LLM initialized with specific instructions to facilitate the optimization mapping. The function $\mathcal{L}_{iter}$ is an instance of a general LLM $\mathcal{L}$ and

$$\mathcal{L}_{iter}(p_t) = \mathcal{L}(p_t|m), \tag{1.1}$$

where m denotes the set of instructions, or a prompt, for optimizing p_t. In context of prompt optimization when we are talking about a prompt for optimizing another prompts, we call it a **meta-prompt**. In this case, we can imagine m as

$$m \approx \text{Improve the following prompt \{insert\_prompt\}.}$$

By changing the metaprompt, one can easily change the nature of the optimization operator. In the following chapters, we will examine options for designing such operators and compare their performance and effectivity on industry-standard datasets, as well as on a brand new custom dataset.

Chapter 2

Literature

2.1 Large Language Models

2.1.1 Brief history of NLP approaches

The goal of this section is to familiarize the reader with the progress in the Natural Language Processing (NLP) field in the recent decade.

Pre-transformer era

Statistical NLP. Data-driven methods such as Hidden Markov models, Conditional Random Fields and Max Entropy models are being used for part-of-speech tagging, named entity recognition, machine translation and speech recognition.

Word embeddings. Algorithms that encode meaning of words in high-dimensional vectors allow models to understand words and relationships between them.

lstm, seq2seq, attention. Attention allows models to connect key parts of input.

Transformer era

Attention is all you need. Discovers that simplifying the architecture and focusing on the attention mechanisms allows for much better efficiency in training and paves way for a new era in NLP.

Pre-training+fine-tuning. New paradigm where the language model is first pre-trained on an enormous corpus of data and later fine-tuned for specific tasks, like instruction-tuning for assistants.

multimodality. Visual embedders allow LLMs to understand images. Embeddings are conserved under different modalities ("dog" and a picture of a dog have the same embedding).

Mixture-of-experts (MoE). More efficient architecture that allows for delegating of work to several "expert" submodels, resulting in sparse computation as only a fraction of the model parameters is activated.

Reinforcement learning. Supervised Fine-tuning (SFT) has been combined or sometimes replaced altogether by various forms of Reinforcement learning (RL). Proximal policy optimization based on human or AI feedback is the basis for assistant chat bots such as ChatGPT. Novel RL approaches (GRPO) used to promote reasoning.

Inference-time compute. Development of reasoning models is converging on the idea that letting the model spend more compute time on each answer leads to better results. Using SFT and/or RL the model is taught to "think" or show its "inner monologue" as a part of the answer inside a designated "<think>" tag. When leaving the thought chain, the "</think>" tag can be substituted by an introspective question like "Wait, did I forget something?" resulting in a prolonged thinking chain. potentially catching errors.

Overview of best current models. Current models, with sizes around 1 trillion parameters, match or surpass average human performance on many benchmarks including math and coding.

◼ 2.2 Prompting techniques

Prompting techniques encode human priors, making it difficult to assess a language model's intrinsic reasoning abilities [1]

◼ 2.2.1 Prompt engineering

Motivation for prompt engineering is to improve the model's capabilities not by changing the underlying weights with training on data but by crafting an optimal instruction string, or a prompt. This can be done by providing examples of the task as a part of the prompt or by instructing the model how to solve the task. In its essence, the model is a left-to-right text completion engine. We can make the analogy with human thinking modes, where it is said that humans have a fast automatic "System 1" mode and a slow and deliberate "System 2" mode [2]. With a good prompt we can shift the model from "System 1" to "System 2".

◼ In-context learning

Prompts are distinguished based on the number of included examples. Research[3]

Prompting Type	Description
Zero-shot Prompting	Prompt has no examples, model relies on its pre-trained knowledge.
One-shot Prompting	Prompt has one example to guide the model.
Few-shot Prompting	Prompt includes a few examples (typically 2 to 5).

Table 2.1: Comparison of Zero-shot, One-shot, and Few-shot Prompting

has shown that with growing model size the knowledge-generalizing ability of the model increases. Instead of expensive fine-tuning models can reuse knowledge from pre-training and solve many tasks when provided just by a few examples.

2.2.2 Prompting techniques

Chain-of-thought (CoT)

Principles

Chain of thought = coherent series of intermediate reasoning steps that lead to the final answer for a problem [4]

CoT allows models to allocate additional computation to problems with more reasoning steps (inference-time scaling) [4]

Examples of CoT reasoning in the prompt in a one/few-shot setting to facilitate a reasoning chain response. [4]

CoT prompting is an emergent ability of model scale, does not positively impact performance for small models [4]

CoT can been elicited by prompting techniques - few-shot with steps demonstrations or zero-shot with specific instructions [1]

Few-shot-CoT requires human engineering of multi-step reasoning prompts and their performance deteriorates if prompt example and task question types are unmatched, suggesting high sensitivity to prompt design. [5]

LLMs are decent zero-shot reasoners by adding a simple "Let's think step-by-step" prompt, which is versatile and task-agnostic. Similar prompts that encourage reasoning also improve performance, which leaves the question of how to design better Zero-shot-CoT templates. [5]

Existing work suggest LLMs falter in direct-QA scenarios. [1] Greedy decoding in a direct-QA scenario often does not contain a CoT path, which may stem from the model's skewed perception of problem difficulty as it might have been trained on simpler questions [1]

Direct prediction is inaccurate for some inferences because the relevant variables are rarely seen together in training. [6]

Chain-of-thought reasoning improves estimation by incrementally chaining local statistical dependencies that are observed frequently in training. [6]

CoT without prompting

CoT can been elicited by model training or tuning with significant amount of reasoning data [1]

Exploring top-k, k>0 tokens at the first decoding step and continuing with greedy reveals natural CoT reasoning in many cases with the resulting answer having much higher confidence and the decoding path with the highest answer confidence among the top-10 decoding paths contains a CoT path in 88% of the GSM8k samples. CoT-decoding can be easily combined with CoT-prompting or weighted aggregation similar to self-consistency, yielding even larger reasoning gains over multiple language models [1]

Applicability

More performance gains for more complicated problems. [4]

Zero-shot-CoT outperforms simple Zero-shot on arithmetic tasks but does not provide performance gains on commonsense reasoning tasks. [5]

CoT can impact performance on tasks where verbal thinking and deliberation hurts performance in humans but for some it is less clear if they should generalize to LLMs [7]

Tasks where human limitations generalize to LLMs: Implicit statistical learining - 36% performance decrease from gpt-4o to o1-preview Facial recognition, classifying data with patterns and exceptions - performance drops [7]

Tasks where human limitations do not generalize a significant performance decrease is not seen or performance increases: explaining a logical inconsistency: assumes a reasonable baserate in recognizing logical inconsistencies spatial intuition: consequence of information encoded in visual and motor representations that are likely not available to models aggregating features for a decision, complex, multi-dimensional tasks that exceed human working memory capacity but can easily fit in the context window of an LLM [7]

More advanced inference-time reasoning techniques like ToT alleviate performance a little but still underperform zero-shot. [7]

Self-consistency

Self-consistency aggregates answers from diverse reasoning chains and selects the best one based on majority voting. [8]

It significantly improves accuracy in a range of arithmetic and commonsense reasoning tasks and is useful for collecting rationales and providing uncertainty estimates. [8]

Works with both few-shot and zero-shot CoT. [8]

One limitation of self-consistency is that it incurs more computation cost. Small number of paths is enough, as in most cases the performance saturates quickly. [8]

■ Reflexion

Reflexion converts binary or scalar feedback from the environment into verbal feedback in the form of a textual summary, which is then added as additional context for the LLM agent, e.g. CoT or ReAct module, in the next episode. [9]

Reflections go into a long-term memory context limited to a sliding window with maximum capacity. [9]

Improves performance over strong baselines on sequential decision making, reasoning and programming tasks. [9]

■ ReAct

Multi-turn prompting technique that forms the basis of agentic LLMs. The model is given a set of tools, such as a Wikipedia search function or a math

expression evaluator. The model can go through several steps of using the tools, which generates "observation". The model uses these observations to generate a final answer and leaves the ReAct chain when ready using a "finish" function.

▪ Tree-of-thought

Similar to CoT with self-consistency but the reasoning chain is split up into steps creating a tree. This reasoning tree can then be explored using a graph search algorithm such as DFS.

■ 2.3 Optimization methods

■ 2.3.1 Basics

Optimization is the search for the optimum (maximum or minimum) of an arbitrary function. We can divide optimization into two categories based on the nature of the decision variables:

1. continuous

2. discrete.

■ 2.3.2 Continuous optimization methods

Gradient descent, Newton's method, EAs

■ 2.3.3 Discrete optimization methods

Hill-climber, search methods, GAs

■ 2.4 Prompt optimization

■ 2.4.1 Soft prompt tuning

Prompts for models which allow access to gradients, which is not the case for proprietary models accessed via APIs, can be optimized in the high-dimensional embedding space.

This makes the optimization problem continous. Soft prompts however pose the problem of interpretability and are non-transferable across different LLMs [10].

Continuous prompt-optimization techniques, although effective, require parameters of LLMs inaccessible to black-box APIs and often fall short of interpretability. [11]

■ 2.4.2 Discrete prompt tuning

The area of optimizing prompts discretely while utilizing language models as optimization operators has attracted significant research interest in recent years.

Natural language prompt engineering is particularly interesting because it is a natural interface for humans to communicate with machines, but plain language prompts do not always produce the desired result. [12]

Natural language program synthesis search space is infinitely large. [12]

Meta-prompts are flexible but studies lack principled guidelines about their design. [13]

Reproduces key model parameter learning factors - update direction and update method - in LLMs to seek theoretical foundations. [13]

OPRO[14] and APO[15] introduced analogical "gradient" forms. [13]

Analogical momentum forms inspired by the momentum method involve including the optimization trajectory in the meta-prompt. To fit into the context limit and reduce noise, trajectory can be summarized or k most recent/relevant/important gradients can be retrieved. [13]

To mimic effects or learning rate, prompt variation can be limited by edit distance (maximum words to be changed). Warm-up and decay strategies can be applied to this constraint. [13]

New prompt can be created by editing a previous prompt or generate a new one by following a demonstration. [13]

In an experiment on BBH, authors found that optimization without reflection performs better and the best momentum method being relevance. For prompt variation control, the best combination was cosine decay and no warm-up. [13]

Summarization-based trajectory is less helpful because it tends to only capture common elements. [13]

Task input-output examples are beneficial in the meta-prompt to provide additional context to the LLM to understand the task. [13]

GPT-4 can consistently find better task prompts than GPT-3.5-turbo, which suggests the need for a capable model as the prompt optimizer [13]

Trajectory-based methods perform very well possible because trajectory helps the prompt optimizer pay more attention to the important information instead of the noise in the current step. [13]

APE LLMs are used to construct a good set of candidate solutions by inferring the most likely instructions from input/output demonstrations. [12]

Local search around the best candidates by resampling - asking the LLM to paraphrase the candidate prompt - this however only provides marginal improvements over just choosing the best-performing prompt from instruction induction. [12]

APE was used to improve on Zero-Shot-CoT [5] universal "Let's think by step" prompt"on GSM8k.[12]

Prompt to the LLM optimizer is called the meta-prompt and includes previous prompts with their training accuracies sorted in ascending order along with the task description and training set samples. [14]

The main advantage of LLMs for optimization is their ability of understanding natural language, which allows people to describe their optimization tasks without formal specifications. [14]

Motivated by linear regression and TSP and on small-scale traveling salesman problems, OPRO performs on par with some hand-crafted heuristic algorithms. [14]

Optimization stability can be improved by generating multiple solutions when relying on random ICL samples. [14]

To balance between exploration and exploitation, LLM sampling temperature can be tuned. Lower temperature encourages exploitation in the local solution space and higher temperature allows more aggressive exploration of different solutions. [14]

Only the top instructions are kept in the meta-prompt to fit in the LLM context limit. [14]

New outstanding solution is usually found only all the prompts are of similar quality: first all the worse prompts are purged and substituted by a prompt similar to the current best. [14]

Semantically similar instructions have vastly different performance on GSM8k: "Let's think step by step." achieves accuracy 71.8, "Let's solve the problem together." has accuracy 60.5, while the accuracy of "Let's work together to solve this problem step by step." is only 49.4. [14]

▪ Textual gradients

Naturally there are no gradients in the text space but some researchers try to emulate them using reflection-based operators.

APO mirrors the steps of gradient descent within a text-based Socratic

dialogue substituting differentiation with LLM feedback and backpropagation with LLM editing [15]

Beam search is an iterative optimization process where in current prompt is expanded into many more candidates in each iteration and a selection process decides which will be used in the next iteration. [15]

Expansion first uses gradients to edit the current prompt and then explores the local monte-carlo search space by paraphrasing the editions [15]

To limit the computation used on evaluating prompts, an approach inspired by best arm identification in bandit optimization is utilized. [15]

Applying previous iterative prompt optimization methods, based on prompt+score pairs, to text generation tasks is challenging due to the lack of effective optimization signals. [16]

Critiques and suggestions, written in natural language, are more helpful for prompt improvement than a single score.[16]

CriSPO uses prompt+score+critique triples for next candidate generation. [16] Unlike APE [15] prompt generation is decoupled from suggestions and a history of critiques and suggestions as packed into the optimizer for a more stable optimization. [16]

CoT is applied in optimization by first asking to compare high-score prompts to low-score ones and draft general ideas. [16]

Critique-based optimization explores a larger space, which is indicated by lower similarity of the prompts in lexicons and semantics.[16]

CriSPO outperforms OPRO [14] both on summarization and QA tasks and metaprompt allows for creating ICL and RAG template prompts. [16]

DSPy optimizers

Most prompt optimizer approaches do not apply to multi-stage LLM programs where we lack gold labels or evaluation metrics for individual LLM calls. [17]

Proposing a few high-quality instructions is essential due to the intractably large search space. [17]

Uses a surrogate Bayesian optimization model, which is updated periodically by evaluating the program on batches, to sample instructions and demonstrations for each stage of the LLM program [17]

Optimizing demonstrations alone usually yields better performance than just optimizing instructions, but optimizing both yield the best performance. [17]

Optimizing instructions is most valuable for tasks with subtle conditional rules not expressible by a few examples. [17]

For LLM programs, it is beneficial to alternate between optimizing weights (fine-tuning) and optimizing prompts. [18]

∎ **Evolutionary optimization**

Building upon the inherent ability of LLMs to paraphrase (mutation) and combine (crossover) text, an interesting intersection of traditional evolutionary algorithms and modern LLMs has formed.

Sequences of phrases can be regarded as gene sequences in typical Evolutionary algorithms. [11]

Considers two widely used EAs: Genetic Algorithm and Differential Evolution with DE outperforming GA on most tasks [11]

Initial population consists of manually-written prompts to leverage human knowledge as well as some prompts generated by LLMs to reflect the fact that EAs start from random solutions to avoid local optima. [11]

DE-inspired approached builds on the idea that the common elements of the current best prompts need to be preserved [11]

Evoprompt performs best with roulette selection when compared with

tournament and random selection. [11]

Similar results are achieved when population is initialized with the best and with random prompts, hinting that the crafted design of initial prompts is not essential. [11]

Previous research optimized zero-shot instructions and examples separately, overlooking their interplay and resulting in sub-optimal performance. [19]

There is a prevailing notion that prompt engineering sacrifices efficiency for performance due to the lengthening of prompts, but PhaseEvo actively shortens the prompts [19]

Current EA applications to prompt optimization suffer from extremely high computational cost and slow convergence speed due to the complexity of the high-dimensional search space. [19]

PhaseEvo alternates between two phases: exploration with evolution operators and exploitation using a feedback "gradient". [19]

TABLE 1 **recreate** compares all 5 operators. [19]

4 phases: initialization - lamarck or manual, local feedback mutation, global evolution with EDA and CR operators, local semantic mutation (paraphrasing) [19]

Candidates for evolution operators are selected based on a "performance vector", combining prompts that do not make the same mistakes. [19]

When the performance improvement with an operator stagnates up to some operator-specific tolerance, the current phase is terminated. [19]

Evolution in phases outperforms random operator selection. [19]

PhaseEvo is the most cost-effective but still needs around 12 iterations and 4000 API calls. [19]

APE [12] ran into problems with diminishing returns and abandoning the

iterative approach entirely, Promptbreeder aims to solve this with a diversity-maintaining evolutionary algorithm for self-referential self-improvement of prompts [20]

Prompt optimization techniques utilize the fact that LLMs are effective at generating mutations from examples and can encode human notions of interestingness and can be used to quantify novelty. [20]

Self-referential system should improve the way it is improving, thus Promptbreeder used a "hyper-prompt" to optimize its meta-prompt [20]

Uses a binary tournament genetic algorithm. [20]

Uses a random uniformly sampled mutation operators out of 9 total from 5 broad categories for each replication event. [20]

Zero-order mutation (creating a prompt from task description) generates new task prompts more aligned with the task description in the event the evolution diverges. [20]

LLMs tend to be biased to examples found later in EDA mutation lists. Lying to the LLM and telling it that the prompts are sorted by performance in a descending order improves diversity. [20]

Removing any self-referential operator in ablation is harmful under nearly all circumstances [20]

◾ Metaprompting

Metaprompting or "prompting to create prompts". Research shows that meta-prompting will always be superior to prompting through category theory[21].

Chapter 3

Methodology

■ 3.1 Datasets

Datasets were chosen according to the following requirements:

1. The task is challenging for modern LLMs using a standard CoT prompt but has non-zero accuracy

2. Complex output (no multiple-choice answers)

3. Easy to check programmatically to avoid human/AI judges

■ 3.1.1 External dataset

We found the Livebench datasets to meet our requirements.

■ 3.1.2 Custom dataset design

Sequences dataset challenges the pattern recognition and algebraic capabilities of the model

■ 3.2 Optimization methods

■ 3.2.1 Optimization operators

Metaprompts that define the transition between optimizer generations.

3.3 Experimental setup

Language model used for solving is gpt-4o-mini. Prompts for the solver model are optimized by the optimized model, for which we use the gpt-4o. To encourage diversity and exploration in the optimization process, a temperature of 0.7 is used for the optimizer model. The solver model uses temperature 0.0 to keep the outputs deterministic.

Chapter 4

Experiments

ctuthesis t1606152353

■ 4.1 Comparative analysis of optimization operators

Appendix **A**

Bibliography

[1] X. Wang and D. Zhou, "Chain-of-thought reasoning without prompting," 2024.

[2] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," 2023.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[4] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023.

[5] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 22199–22213, Curran Associates, Inc., 2022.

[6] B. Prystawski, M. Y. Li, and N. D. Goodman, "Why think step by step? reasoning emerges from the locality of experience," 2023.

[7] R. Liu, J. Geng, A. J. Wu, I. Sucholutsky, T. Lombrozo, and T. L. Griffiths, "Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse," 2024.

[8] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," 2023.

[9] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," 2023.

[10] M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. P. Xing, and Z. Hu, "Rlprompt: Optimizing discrete text prompts with reinforcement learning," 2022.

[11] Q. Guo, R. Wang, J. Guo, B. Li, K. Song, X. Tan, G. Liu, J. Bian, and Y. Yang, "Connecting large language models with evolutionary algorithms yields powerful prompt optimizers," 2024.

[12] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," 2023.

[13] X. Tang, X. Wang, W. X. Zhao, S. Lu, Y. Li, and J.-R. Wen, "Unleashing the potential of large language models as prompt optimizers: An analogical analysis with gradient-based model optimizers," 2024.

[14] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, "Large language models as optimizers," 2024.

[15] R. Pryzant, D. Iter, J. Li, Y. T. Lee, C. Zhu, and M. Zeng, "Automatic prompt optimization with "gradient descent" and beam search," 2023.

[16] H. He, Q. Liu, L. Xu, C. Shivade, Y. Zhang, S. Srinivasan, and K. Kirchhoff, "Crispo: Multi-aspect critique-suggestion-guided automatic prompt optimization for text generation," 2024.

[17] K. Opsahl-Ong, M. J. Ryan, J. Purtell, D. Broman, C. Potts, M. Zaharia, and O. Khattab, "Optimizing instructions and demonstrations for multi-stage language model programs," 2024.

[18] D. Soylu, C. Potts, and O. Khattab, "Fine-tuning and prompt optimization: Two great steps that work better together," 2024.

[19] W. Cui, J. Zhang, Z. Li, H. Sun, D. Lopez, K. Das, B. Malin, and S. Kumar, "Phaseevo: Towards unified in-context prompt optimization for large language models," 2024.

[20] C. Fernando, D. Banarse, H. Michalewski, S. Osindero, and T. Rocktäschel, "Promptbreeder: Self-referential self-improvement via prompt evolution," 2023.

[21] A. de Wynter, X. Wang, Q. Gu, and S.-Q. Chen, "On meta-prompting," 2024.