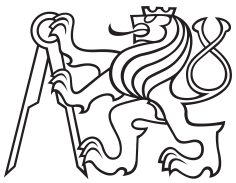
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Artificial Intelligence Center

Methods of Evolutionary Optimization of Prompts for Large Language Models

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, evolutionary algorithm

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

Abstrakt

TODO

Klíčová slova: jazykový model, evoluční algoritmus

Překlad názvu: Metody evoluční optimalizace vstupních řetězců pro velké jazykové modely

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 $p_{t+1} = \mathfrak{L}_{iter}(p_t)$, where $\mathfrak{L}_{iter}$ denotes an LLM initialized with specific instructions to facilitate the optimization mapping. The function $\mathfrak{L}_{iter}$ is an instance of a general LLM $\mathfrak{L}$ and

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

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

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

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

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 [1]. 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[2]

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)

Instructing the model to "think step-by-step" results significant improvements on many benchmarks. Some research has shown that this approach hurts

performance on some tasks where humans perform better without thinking. Apart from more compute time being allocated at inference, the model also benefits from having its whole reasoning chain as a part of the context when writing the final answer to the task. Several variants exist, such as CoT with self-consistency, where the final answer is acquired by majority-voting from several thinking chains.

Reflexion

Model reflects on its response and improves it.

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 [3].

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.

Textual gradients

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

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.

Metaprompting

Metaprompting or "prompting to create prompts". Research shows that meta-prompting will always be superior to prompting through category theory[4].

Chapter 3

Methodology

Chapter 4

Experiments

Appendix A

Bibliography

[1] 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.

[2] 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.

[3] 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.

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