Local Prompt Optimization

Yash Jain **Vishal Chowdhary**

Microsoft

Abstract

In recent years, the use of prompts to guide the output of Large Language Models have increased dramatically. However, even the best of experts struggle to choose the correct words to stitch up a prompt for the desired task. To solve this, LLM driven prompt optimization emerged as an important problem. Existing prompt optimization methods optimize a prompt globally, where in all the prompt tokens have to be optimized over a large vocabulary while solving a complex task. The large optimization space (tokens) leads to insufficient guidance for a better prompt. In this work, we introduce Local Prompt Optimization (LPO) that integrates with any general automatic prompt engineering method. We identify the optimization tokens in a prompt and nudge the LLM to focus only on those tokens in its optimization step. We observe remarkable performance improvements on Math Reasoning (GSM8k and MultiArith) and BIG-bench Hard benchmarks across various automatic prompt engineering methods. Further, we show that LPO converges to the optimal prompt faster than global methods.

1 Introduction

Large Language Models (LLMs) are everywhere. LLMs are automating all the tasks that required specialized models a few years ago (Dubey et al., 2024; OpenAI, 2023). The easiest and cheapest way to control an LLM's output is to do prompt engineering (Zhou et al., 2023a; Zhao et al., 2021; Yang et al., 2023; Lu et al., 2022). Unfortunately, writing a prompt is extremely tricky (Pryzant et al., 2022). Although the prompts are in English, the choice of words that effectively have the same meaning makes a huge difference in the prompt's performance on a task (Kojima et al., 2022; Wei et al., 2022; Amatriain, 2024). Furthermore, an LLM is inherently biased towards its own vocabulary, making the task even more challenging. Thus, LLMs

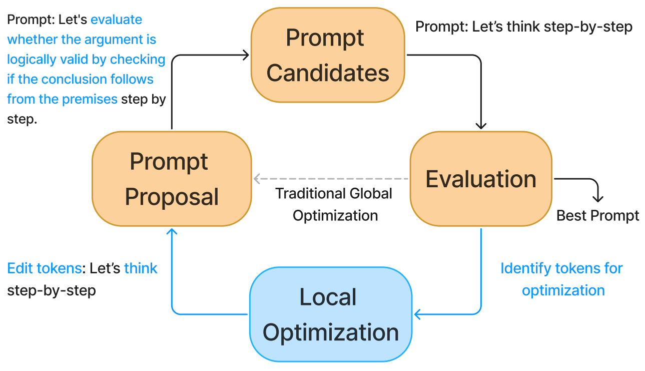

Prompt: Let's evaluate whether the argument is logically valid by checking if the conclusion follows from the premises step by step.
Prompt Candidates
Prompt: Let's think step-by-step
Prompt Proposal
Traditional Global Optimization
Evaluation
Best Prompt
Edit tokens: Let's think step-by-step
Local Optimization
Identify tokens for optimization

Figure 1: Local Prompt Optimization integrated in a general automatic prompt engineering framework.

are used to modify prompts in a process called Prompt Optimization (Zhou et al., 2023a).

Prompt optimization techniques follow a two-step process as shown in Fig. 1. First, the prompt is validated against a training set where the incorrect predictions are identified. Optionally, a feedback step is added where a natural language feedback, termed 'textual gradients', is obtained by querying the LLM (Ye et al., 2024; Tang et al., 2024). Finally, the prompt is optimized using the textual gradients (incorrect examples or natural language feedback) to obtain an optimized prompt. The cycle is repeated for a fixed number of steps.

Traditional prompt optimization techniques (Pryzant et al., 2023; Zhou et al., 2023b; Ye et al., 2024; Tang et al., 2024) optimize prompts globally *,i.e.,* mutate all tokens in the prompt. However, optimizing all the tokens in a prompt while searching over the vocabulary to solve a complex problem, makes the prompt optimization very challenging. Further, for many production applications, it is desirable to optimize only subsections of the prompt while keeping the other parts static. Doing so requires us to limit the scope of the 'prompt proposal' on subsection of the prompt, hence, the need of Local Prompt Optimization (LPO). Thus, we reduce the optimization space (tokens) for the LLM to simplify the problem and control the edit direction of a prompt.

In this work, we evaluate the efficacy and pitfalls of doing local prompt optimization compared to global prompt optimization. We incorporate local optimization in three automatic prompt optimization algorithms and evaluate on GSM8k (Cobbe et al., 2021), MultiArith (Roy and Roth, 2015), and BIG-bench hard (Suzgun et al., 2023) benchmarks. We highlight that local optimization leads to faster convergence of optimal prompt while improving prompt performance. Finally, we test local optimization on a real-world application by evaluating it on a production prompt.

2 Background and Method

In this section, we will describe a general framework of automatic prompt engineering (Zhou et al., 2023a) and highlight the gap in the framework. Building on this, we will introduce local prompt optimization.

2.1 Automatic Prompt Engineering

Given a dataset $D = (x, y)$, a prompt engineering system aims to find a prompt p^* that maximizes the score on an evaluator function f. Specifically,

$$p^* = \arg\max_{p} \sum_{(x,y)\in D} f(\mathcal{M}_{task}(x;p), y) \quad (1)$$

where $\mathcal{M}_{task}(x;p)$ is the output generated by the task model $\mathcal{M}_{task}$ when conditioning on the prompt p.

A general automatic prompt engineering system has three parts: Prompt Initialization, Prompt Proposal, and Search Procedure.

(1) Prompt Initialization: An initial prompt is provided to an automatic prompt system that needs to be optimized. Prompt Initialization can be done by a manual human-written instruction or it can be few shot examples from the dataset D (Zhao et al., 2021).

(2) Prompt Proposal: In this step new prompt generation takes place. At any timestep t, a new set of prompts $p^{(t+1)}$ are generated from a set of candidate prompts p^t. A proposal LLM $\mathcal{M}_{proposal}$ is used to propose new prompts, grounded on 'textual gradients' g^t obtained on the current prompt p^t. These 'textual gradients' consists of a meta prompt along with additional information which vary between automatic prompt engineering techniques. These include incorrect examples (Zhou et al., 2023b), or a natural language LLM feedback of the incorrect examples (Pryzant et al., 2023) to a combination of both along with previous prompts $p^{(t-1)}$ and their scores (Ye et al., 2024).

$$p^{(t+1)} = \mathcal{M}_{proposal}(p^t, g^t). \quad (2)$$

However, the edits in prompt $p^{(t)}$ can take place anywhere inside the prompt including complete rewriting the prompt at every timestep causing slow update towards the optimal prompt. Further, it does not provide any control required in a typical production prompt engineering where a professional would want prompt edits to take place within a specific scope of the prompt. Thus, the global optimization leads to slow prompt convergence and provides no control over direction of prompt optimization.

(3) Search: Finally, among the candidate prompts across all timesteps $p^0 \cup p^1 \cup ... \cup p^t$, a subset of the best performing prompts are retained and the process is repeated.

2.2 Local Prompt Optimization

The basic function of 'textual gradients' g^t is to inform how the optimization process (gradient values) should adjust according to model's performance (Tang et al., 2024). However, it does not specify where the optimization should take place or analogously in deep learning on which parameters should the gradient descent should take place. We incorporate this intuition of parameter selection to reduce the optimization space through local prompt optimization.

Following the intuition of Chain-of-Thought logic (Wei et al., 2022), we first identify the potential tokens in the prompts which are responsible for incorrect predictions by adding an instruction in the meta-prompt before the Prompt Proposal step as depicted in Fig. 1. We use `<edit>` tags to highlight the edit tokens, the meta-instruction is shown in Fig. 2. The goal is to identify tokens within the prompt that the proposal LLM $\mathcal{M}_{proposal}$ should optimize.

Once the prompt edit tokens are identified, we proceed with the Prompt Proposal step. The instruction '`Reply with the new instruction without the <edit>, </edit> tags.`' is provided to $\mathcal{M}_{proposal}$ to output the updated prompt

$p^{(t+1)}$. Tab 1 shows the complete prompt evolution with local and global optimization.

```
First, identify the scope of tokens within the
prompt where edits should take place.
Prompt edits include adding, deleting or
modifying tokens.
Mark the scope of the prompt that needs editing
by putting <edit>, </edit> tags.
You can have multiple <edit> tags and each <edit>
tag should not entail more than 5 words.
Do not cover the whole sentence with multiple
<edit> tags.
Reply with the prompt with <edit>, </edit> tags.
Do not include any other text.
```

Figure 2: Illustration of the Prompt for identifying potential optimization tokens.

3 Experiments

The goal of this section is to highlight the efficacy of local optimization over existing global optimization across different automatic prompt engineering methods.

3.1 Datasets

Following PE2 (Ye et al., 2024) closely, we perform evaluation on three set of tasks varying in their objectives and domain. We use the same train-dev-test split as provided by (Ye et al., 2024).

(1) BIG-bench Hard or BBH (Suzgun et al., 2023) is a set of 23 tasks (27 subtasks) which can be categorized as algorithmic, natural language understanding, world knowledge, and multlingual reasoning tasks.

(2) Math Reasoning consists of two datasets MultiArith (Roy and Roth, 2015) and GSM8K (Cobbe et al., 2021). Both contains grade school math problems requiring 2 to 8 steps of algebraic reasoning to reach the final answer.

(3) Production Prompt is an internal classification prompt, developed to orchestrate the correct tool for further LLM calls. The prompt would take in a user query and would identify the 'intent' of the query. It would then output a function call with appropriate arguments. It has been developed by in-domain experts and is 8k tokens long. The prompt contains sections of skill definitions, specific classification instruction, safety instructions and so on, making it an ideal candidate for evaluation.

Initial Prompt	Let's think step by step.
Global Optimization	
Optimum	Ensure all given initial values and specific contexts (e.g., rounding rules, phrase interpretation) are considered, and explain the arithmetic operations logically and clearly, step-by-step.
Local Optimization	
Identifying edit	Let's `<edit>` think `</edit>` `<edit>` step by step `</edit>`.
Optimum	Let's carefully read and clearly understand the problem. Next, let's think through each step and verify each calculation carefully.

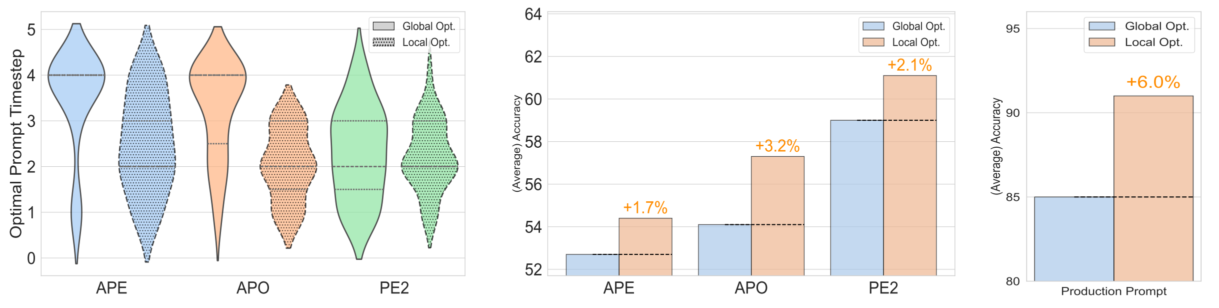
Table 1: MultiArith prompts found by comparing traditional global optimization approach against our proposed local optimization.

3.2 Prompt Optimization methods

For fair comparison, we select three representative prompt optimization techniques and modify their global optimization step with our local optimization step as explained in Sec. 2 and Fig. 1. (1) **APE** (Zhou et al., 2023b) leverages LLMs to come up with variants of the input prompt, given few examples and then select the best performing prompt. An improved variant of APE called **Iterative APE**, repeats this process a few times to get a better optimized prompt. We use Iterative APE for comparison in the paper. (2) **APO** (Pryzant et al., 2023) is builds over Iterative APE and adds an incorrect prediction feedback in its prompt optimization process. This feedback is often termed as 'textual gradients' and is used to make edits in correct direction on the candidate prompt. **APO** is named as **ProTeGi** in their recent draft. (3) **PE2** (Ye et al., 2024) further innovates in the 'textual gradients' and make them rich by adding old prompt and their feedback history to guide the edit process. They also limit the number of edits in the prompt.

3.3 Implementation Details

Across all experiments, we consistently use `gpt-3.5-turbo` as the task solving model and `gpt-4o` as the prompt optimizer. The remaining design details follow those of PE2 (Ye et al., 2024). We limit the search budget to 3 optimization steps, using a beam size of 4 and generating 4 prompts at each step. Further, we initialize the prompts for BBH and Math Reasoning datasets with the standard prompt "Let's think step by step" (Kojima et al., 2022; Wei et al., 2022). We keep the hyperparameters for all the prompt optimization methods same across global and local optimization.



(a) Optimal Prompt Timestep in the 27 subtasks of BBH benchmark. Local Opt. achieves faster convergence.

(b) Average Accuracy on BBH. Local Opt. consistently outperforms global opt. across various methods.

(c) Production Prompt performance after employing local opt.

Figure 3: Experiments on BBH and Production Prompt, showcasing LPO benefits in both performance and efficiency.

Method	LPO	GSM8k (↑)	MultiArith (↑)	# steps (↓)
APE	-	77.7	93.2	**2.5**
	✓	**78.0**	**96.2**	4
APO	-	77.7	96.0	4
	✓	**79.7**	**97.5**	**2**
PE2	-	78.7	97.0	2.5
	✓	**80.6**	**97.5**	**2**

Table 2: Results of Local Prompt Optimization (LPO) on Math Reasoning benchmark.

4 Results and Analysis

Local Prompt Optimization improves existing automatic prompting techniques. We evaluate APE, APO and PE2 algorithms with and without Local Optimization on GSM8K and MultiArith datasets as depicted in Tab. 2. We observe that Local Prompt Optimization is able to improve prompts for Math Reasoning tasks by an average of 1.5% while decreasing the number of optimization steps required. Additionally, we demonstrate the wide applicability of Local optimization on BIG-bench Hard benchmark (27 subtasks). In Fig. 3b, we show that local optimization supports various automatic prompting techniques over a large variety of tasks. We outperform traditional global optimization approach by an average of 2.3% across methods. We hypothesize that since Local Optimization reduces the optimization tokens for the proposal LLM $\mathcal{M}_{proposal}$ and introduces a Chain-of-Thought approach in the optimization step, $\mathcal{M}_{proposal}$ is able to more efficiently solve the task and provide better prompt outputs.

Local Prompt Optimization results in faster convergence. We estimate the timestep where the optimal prompt is produced over the 27 subtasks in BIG-bench Hard benchmark. The number of iterations were kept to 3 and we assign a timestep of 4 when the initialization prompt is found to be the best performing prompt. Fig. 3a depicts the violin curves of optimal prompt timestep. Notably, we observe majority of tasks reaching earlier convergence than global optimization approaches, saving a lot of LLM compute and time. Global optimization often leads to rewriting the complete prompt from scratch for the LLM, making the task more challenging and complex. On the other hand, we believe reducing the optimization space through local optimization keeps the gradient updates aligned towards the minima.

Local Prompt Optimization can allow control over prompt editing. Perhaps, the biggest benefit of LPO is to control the scope of editing. In the production prompt written by domain expert, the prompt has specific sections where the different tools are defined followed by instructions on individual tools and their use. Using LPO, we can specify which tool's instruction needs to be updated without affecting the other tools. Further, it ensures that there is no regression in performance of the prompt in other classes due to the optimization process. In our evaluation, we gained a significant jump of 6% on the production prompt as shown in Fig. 3c.

5 Conclusion

In this work, we identify the gap in the optimization step of the existing automatic prompt engineering techniques. Traditionally, prompts are mutated globally in each step. However, this global optimization increases the task complexity as the optimizer (LLM) has to work on a larger number of parameters (tokens) to find the optimal update. Furthermore, many production prompts require optimizing only a section of the prompt and not rewriting the complete prompt from scratch. As a fix, we introduce Local Prompt Optimization (LPO) where we identify the optimization tokens and nudge the optimizer to focus only on those tokens. We observe consistent performance improvements over Math Reasoning and BIG-bench Hard benchmark. Notably, we observe that local optimization searches the optimal prompt significantly quicker than the traditional approach. Further, LPO can be integrated well with long prompts, which are more common in practical settings, further showcasing the ubiquity of our method. Looking ahead, we are optimistic about prompt optimization techniques built from the perspective of local optimization to benefit from the gains in performance and efficiency.

6 Limitations

We believe our study has three limitations which we believe can be overcome in future works. (1) Multilinguality: We primarily focused on English language as the base in this work, from prompts to datasets to LLMs. However, we believe the ideas introduced in the paper are extendable to other languages as well and implore the community to build over our work. (2) Local Optimization sometimes leads to overfitting the prompt with dev score reaching close to 99%. We believe that a better search strategy can solve this problem and hope to see future works addressing it. (3) Closed-source models: We have used `GPT-4o` as the optimizer to benchmark large datasets in this work. This poses a challenge to the reproducibility of this work. However, we believe that showcasing local optimization capabilities on proprietary models is a good signal for both academic and industry to incorporate the ideas in their prompt engineering methods.

References

Xavier Amatriain. 2024. Prompt design and engineering: Introduction and advanced methods. *arXiv preprint arXiv:2401.14423.*

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168.*

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei

Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.

Reid Pryzant, Ziyi Yang, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2022. Automatic rule induction for efficient semi-supervised learning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 28–44, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. 2024. Unleashing the potential of large language models as prompt optimizers: An analogical analysis with gradient-based model optimizers. *arXiv preprint arXiv:2402.17564*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Kexin Yang, Dayiheng Liu, Wenqiang Lei, Baosong Yang, Xiangpeng Wei, Zhengyuan Liu, and Jun Xie. 2023. Fantastic expressions and where to find them: Chinese simile generation with multiple constraints. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 468–486, Toronto, Canada. Association for Computational Linguistics.

Qinyuan Ye, Mohamed Ahmed, Reid Pryzant, and Fereshte Khani. 2024. Prompt engineering a prompt engineer. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 355–385, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023a. Controlled text generation with natural language instructions. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 42602–42613. PMLR.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023b. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.