APIO: Automatic Prompt Induction and Optimization for Grammatical Error Correction and Text Simplification

Artem Chernodub[*ς] **Aman Saini**[γ] **Yejin Huh**[γ] **Vivek Kulkarni**[γ] **Vipul Raheja**[γ]
[ς]Zendesk [γ]Grammarly

Correspondence: a.chernodub@gmail.com

Abstract

Recent advancements in large language models (LLMs) have enabled a wide range of natural language processing (NLP) tasks to be performed through simple prompt-based interactions. Consequently, several approaches have been proposed to engineer prompts that most effectively enable LLMs to perform a given task (e.g., chain-of-thought prompting). In settings with a well-defined metric to optimize model performance, automatic prompt optimization (APO) methods have been developed to refine a seed prompt. Advancing this line of research, we propose APIO, a simple but effective prompt induction and optimization approach for the tasks of Grammatical Error Correction (GEC) and Text Simplification, without relying on manually specified seed prompts. APIO achieves a new state-of-the-art performance for purely LLM-based prompting methods on these tasks. We make our data, code, prompts, and outputs publicly available.[1]

1 Introduction

Prompt engineering has become a popular and crucial technique for steering large language models (LLMs) toward desired outputs, but finding effective prompts remains challenging. Prompting methods like chain-of-thought (CoT) prompting, best-of-n sampling, etc. are general strategies that have been shown to be effective. However, even when these advanced prompting strategies are used, recent studies show that LLMs are highly sensitive to seemingly minor variations in prompts (e.g. phrasing (Li et al., 2023), ordering of information (Liu et al., 2024), or formatting (Sclar et al., 2024), which can lead to significant performance variation. Consequently, in practice for many tasks, prompts are tuned by prompt engineers to maximize gains in task performance. Since manual prompt tuning can be tedious, there has been some research on automatic prompt optimization (APO) methods that tune a base prompt based on performance on training and validation sets – the most relevant work being that of (Pryzant et al., 2023).

However, APO, in general, mainly focuses on text classification tasks such as Jailbreak Detection, Math Reasoning, and BIG-bench Hard tasks (Zhou et al., 2022; Pryzant et al., 2023; Ye et al., 2024; Ma et al., 2024) and has been underexplored for text revision tasks such as Grammatical Error Correction (GEC) and Text Simplification. In this paper, we address this gap and propose a novel prompt induction and optimization method called APIO . In contrast to existing prompt optimization methods that require a seed prompt, APIO does not rely on a manually specified prompt. Instead, it induces a reasonable list of instructions and subsequently optimizes them. In short, APIO performs both automatic prompt induction and optimization. We evaluate APIO against strong baselines on standard GEC and Text Simplification benchmarks and show that APIO sets a state-of-the-art performance on these benchmarks.

Our main contributions are:

- We introduce a novel method **A**utomatic **P**rompt **I**nduction and **O**ptimization (APIO) for text revision tasks (specifically, GEC and Text Simplification).
- We set the new state-of-the-art for LLM-based prompting methods on these tasks. For the GEC task, we achieve a score of 59.40 on the BEA-2019 test dataset, ahead of the previous state-of-the-art (57.41) (Loem et al., 2023). For the Text Simplification task, we achieve a SARI score of 49.47 on the ASSET-Test dataset, ahead of the previous state-of-the-art (47.94) (Vadlamannati and Şahin, 2023).

[*]The work was done while working at Grammarly.
[1]https://github.com/achernodub/apio

Proceedings of Recent Advances in Natural Language Processing, pages 234–239
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-028

2 APIO

APIO has two main steps:

1. **Prompt Induction**. We first induce a prompt given gold-standard examples of task-specific input and output pairs.

2. **Prompt Optimization**. We then optimize the induced prompt to maximize training and validation performance.

Prompt Induction Unlike other APO methods which start from an initial, manually crafted seed prompt, APIO requires only a few input–output examples that demonstrate the task — typically available as training data. Given these examples, we use a state-of-the-art LLM to infer a prompt to solve the task. A key feature of our prompt induction approach is to induce structure to the inferred prompt. In particular, the LLM generates a prompt that consists of a markdown-style list of single-sentence instructions between the prompt's header and footer, which are not optimized.

Structuring the prompt as a list of independent instructions allows for instruction-level tuning, and enables more fine-grained control as opposed to tuning a flat text blob. Formally, the output of this step will be a prompt $\mathcal{P}$, consisting of an ordered list of instructions $\mathcal{L}$. Each instruction in the list is derived by the LLM from a single "training" input-output pair.

Prompt Optimization In this step, we optimize the induced prompt $\mathcal{P}$ that consists of a list of instructions $\mathcal{L}$ iteratively as follows:

1. We consider the instructions in the current pool of size $\mathcal{M}$, which is initialized to $\mathcal{L}$—the set of instructions inferred during the Prompt Induction step.

2. We then seek to expand the above pool of instructions through a beam search with a beam size B. In particular, we expand the pool through three prompting operations:

   - **Improve**: Here, we generate beam candidates by prompting an LLM to improve the given pool of instructions to reduce the error rate on the given input-output examples as much as possible. In our experiments, we use word-level Levenshtein edit distance as a metric for optimization for all domains.
   - **Rephrase**: Next, we expand the current pool by prompting the LLM to rephrase each instruction without changing the underlying meaning.
   - **Permute**: Finally, we take $N_{permute}$ instructions and randomly change their order in the current list of instructions.

3. After expanding the pool using the above three operations, we obtain three candidate sets – each set being a list of instructions. We rank them by their performance on the validation set and add the best B to the pool. To control for divergence from prior iterations, we additionally introduce a word-level Levenshtein edit distance penalty on the prompts.

3 Experimental Setup

3.1 Tasks, Datasets and Metrics

We conduct our experiments on two prominent text revision tasks: GEC and Text Simplification. We use the current standard evaluation sets and evaluation metrics for each task.

Grammatical Error Correction is the task of correcting text for spelling and grammatical errors. We report results on the Test split of the W&I+LOCNESS Corpus from the BEA-2019 GEC Shared Task (Bryant et al., 2019). We refer to this dataset as BEA-2019-Test. We evaluate results using $F_{0.5}$ score measured using ERRANT tool[2] launched at CodaLab platform[3]. Train and dev datasets are sampled from the BEA-2019-Dev dataset (4384 samples).

Text Simplification is the task of rewriting text in a simpler form without altering its original meaning (Saggion, 2017). We report results on the ASSET-Test dataset (359 samples) (Alva-Manchego et al., 2020) as the main evaluation set. We evaluate results using the SARI score (Xu et al., 2016) measured using the EASSE package[4] (Alva-Manchego et al., 2019). Train and dev datasets are sampled from the ASSET-Dev dataset (2000 samples).

3.2 Baselines

Copy We consider a simple baseline that copies the input text to the output.

[2] https://github.com/chrisjbryant/errant

[3] https://codalab.lisn.upsaclay.fr/competitions/4057

[4] https://github.com/feralvam/easse

Best reference As a best-case baseline, we provide the scores obtained by the best-performing reference if available.

SFT We consider state-of-the-art Supervised Fine-Tuning (SFT) methods as an alternative to prompt-based learning.

Zero Shot We consider a simple 0-shot prompt, which describes the task as an instruction.

Few Shot We augment the prompt used in the 0-shot setting with a few randomly selected examples demonstrating the task.

3.3 APIO Setup

In addition to evaluating our full proposed method, we also perform an ablation where we only perform the first step of APIO – namely automatic prompt induction. We denote that in our experiments with APIO -INDUCTION-ONLY.

Induced prompts: The induced prompts are derived by extracting three instructions from three randomly selected input-output pairs in the training dataset. To identify the best induced prompt, we perform 10 trials on the validation dataset.

Optimized prompts: We optimize the prompts induced in the previous step by continuously adding new instructions using the *Improve* meta-prompt, rephrasing them using the *Rephrase* meta-prompt, and adjusting their order using the *Permute* operation. In our experiments, number of epochs N_{epochs} = 15, $N_{permute}$ = 2, beam size B = 32.

The above parameters were an expedient choice and we did not extensively tune them. With regards to the choice of LLMs used in prompting based approaches, we experiment with two very popular LLMs, namely `GPT-4o-mini`[5] and `GPT-4o`[6]. We use different generation parameter settings for prompt induction and optimization versus test-time inference. For prompt induction and optimization, we set the temperature $t = 1.0$ and nucleus sampling top-p = 1.0 for better creativity. For inference, we set temperature $t = 0.0$ and top-p = 0.1 to decrease randomness in outputs, as instability in outputs leads to worse convergence during optimization.

4 Results

GEC APIO shows substantial gains over zero-shot, few-shot, and induction-only approaches on GEC (Table 1). With GPT-4o, APIO achieves an $F_{0.5}$ score of 59.40 (using 10 instructions), which is comparable to the state-of-the-art performance among prompt-based LLMs (which was 57.41 by GPT-3). However, we also note that APIO performance, still falls significantly short of non-prompting SFT ensemble techniques (which scored 72.80), highlighting limitations of solely prompting-based approaches on this task.

Text Simplification APIO shows significant improvements over baseline methods with both LLMs for the task (Table 1). Notably, APIO using GPT-4o achieves a SARI score of 49.47, surpassing the previous state-of-the-art score (47.94) for prompt-based methods on the ASSET-Test dataset.

Overall, we observe that APIO is a highly effective method for automating prompt engineering in text revision tasks. Its strength lies in significantly boosting performance over standard prompting techniques and achieving state-of-the-art for text revision tasks among prompting-based methods—without the need for manual prompt design. The prompt optimization step was shown to be particularly crucial, yielding substantial performance gains, especially in GEC (compare APIO with APIO -INDUCTION-ONLY). While limitations exist compared to non-prompting methods in GEC, APIO represents a valuable advancement in making prompt engineering easier and accessible.

5 Related Work

5.1 LLM Prompting for Text Revision

Fang et al. (2023) was the first work to evaluate zero-shot performance using LLMs (ChatGPT in their case) for GEC at both sentence and document levels, finding that ChatGPT exhibited high fluency and produced corrections that enhanced the original text beyond the provided references. However, ChatGPT faced challenges in adhering to specific step-by-step formats when given simple prompt instructions. More recently, numerous works (Coyne et al., 2023; Loem et al., 2023; Davis et al., 2024; Kaneko and Okazaki, 2024; Katinskaia and Yangarber, 2024) have evaluated both open-source and commercial LLMs on multiple GEC benchmarks, finding that LLMs do not consistently outperform supervised models, especially on minimal edit tasks, and often struggle to balance fluency improvements and preservation of the original meaning. Similarly, many recent works (Kew et al., 2023; Qiang et al., 2025; Farajidizaji et al., 2024)

[5] `gpt-4o-mini-2024-07-18`
[6] `gpt-4o-2024-05-13`

Task	Approach	LLM	Test Score
GEC *	Copy	–	0.00
	SFT (Omelianchuk et al., 2024)	Multiple	**72.80**
	Zero-shot (Loem et al., 2023)	GPT-3	53.07
	Few-shot (16 examples) (Loem et al., 2023)	GPT-3	**57.41**
	Few-shot (4 examples) (Tang et al., 2024)	GPT-3.5-Turbo	53.20
	Zero-shot (adapted from (Loem et al., 2023))	GPT-4o-mini	49.90
	Few-shot (3 randomly sampled examples)	GPT-4o-mini	53.01
	APIO-INDUCTION-ONLY (3 instructions)	GPT-4o-mini	38.72
	APIO (7 instructions)	GPT-4o-mini	**57.07**
	Zero-shot (adapted from (Loem et al., 2023))	GPT-4o	54.66
	Few-shot (3 examples, randomly sampled)	GPT-4o	44.50
	APIO-INDUCTION-ONLY (3 instructions)	GPT-4o	43.37
	APIO (10 instructions)	GPT-4o	**59.40**
Text Simplification	Copy	–	20.70
	SFT (Sheang and Saggion, 2021)	T5-base	45.04
	Best reference (ref-0)	–	**52.62**
	Few-shot (15 SARI-selected examples, random ordering) (Vadlamannati and Şahin, 2023)	GPT-3-175B	47.94
	Zero-shot (adapted from (Raheja et al., 2023))	GPT-4o-mini	48.03
	Few-shot (3 randomly sampled examples)	GPT-4o-mini	47.16
	APIO -INDUCTION-ONLY (3 instructions)	GPT-4o-mini	48.79
	APIO (6 instructions)	GPT-4o-mini	**49.27**
	Zero-shot (adapted from (Raheja et al., 2023))	GPT-4o	47.73
	Few-shot (3 examples, randomly sampled)	GPT-4o	47.87
	APIO -INDUCTION-ONLY (3 instructions)	GPT-4o	48.93
	APIO (10 instructions)	GPT-4o	**49.47**

Table 1: GEC (BEA-2019-Test | $F_{0.5}$) and Text Simplification results (ASSET-Test | SARI). Results are grouped by baselines (Copy, Best-reference, and SFT), and by other prompt-based methods from different models. *Best reference baseline is unavailable for the GEC task because the BEA-2019-Test dataset has not been published.

have explored and demonstrated the effectiveness of prompt-based methods for text simplification.

5.2 LLM-based Automatic Prompt Optimization (APO)

Prior work show that LLMs are highly sensitive to seemingly minor prompt variations, such as task specification, information ordering, or stylistic formatting, which can lead to significant performance differences, making prompt engineering a tedious trial-and-error process (Li et al., 2025).

Several methods have been proposed to automatically identify better-performing prompts, using both continuous and discrete prompt optimization methods (Li and Liang, 2021; Prasad et al., 2023; Deng et al., 2022; Zhang et al., 2023).

Recent work has focused on incorporating LLMs into the optimization process, leveraging their ability to generate natural text. By providing example data to the LLM, Honovich et al. (2023) generated task instructions directly without an initial prompt. LLMs have also been used to conduct Monte Carlo search (Zhou et al., 2023) generating additional prompt candidates. Various iterative workflows have been designed to prompt LLMs to self-reflect, analyzing errors and improving upon a previous prompt (Pryzant et al., 2023; Ye et al., 2024). Evolutionary algorithms (Guo et al., 2024) suggest systematically refining prompt candidates.

Our work extends this literature by adapting APO specifically for text revision, combining advances in APO with the unique requirements of text editing tasks.

6 Conclusion

We present APIO, a new technique for automatic prompt induction and optimization for the tasks of Grammatical Error Correction and Text Simplification. Our method achieves state-of-the-art performance when compared to other prompting-based baselines on these tasks. APIO represents a significant step forward in automating and simplifying the process of advanced prompt engineering techniques, while making them more accessible and achieving high quality.

References

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020. ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4668–4679, Online. Association for Computational Linguistics.

Fernando Alva-Manchego, Louis Martin, Carolina Scarton, and Lucia Specia. 2019. EASSE: Easier automatic sentence simplification evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 49–54, Hong Kong, China. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Steven Coyne, Keisuke Sakaguchi, Diana Galvan-Sosa, Michael Zock, and Kentaro Inui. 2023. Analyzing the performance of gpt-3.5 and gpt-4 in grammatical error correction. *Preprint*, arXiv:2303.14342.

Christopher Davis, Andrew Caines, Øistein E. Andersen, Shiva Taslimipoor, Helen Yannakoudakis, Zheng Yuan, Christopher Bryant, Marek Rei, and Paula Buttery. 2024. Prompting open-source and commercial language models for grammatical error correction of English learner text. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11952–11967, Bangkok, Thailand. Association for Computational Linguistics.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tao Fang, Shu Yang, Kaixin Lan, Derek F Wong, Jinpeng Hu, Lidia S Chao, and Yue Zhang. 2023. Is chatgpt a highly fluent grammatical error correction system? a comprehensive evaluation. *arXiv preprint arXiv:2304.01746*.

Asma Farajidizaji, Vatsal Raina, and Mark Gales. 2024. Is it possible to modify text to a target readability level? an initial investigation using zero-shot large language models. *Preprint*, arXiv:2309.12551.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*.

Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. 2023. Instruction induction: From few examples to natural language task descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1935–1952, Toronto, Canada. Association for Computational Linguistics.

Masahiro Kaneko and Naoaki Okazaki. 2024. Controlled generation with prompt insertion for natural language explanations in grammatical error correction. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3955–3961, Torino, Italia. ELRA and ICCL.

Anisia Katinskaia and Roman Yangarber. 2024. GPT-3.5 for grammatical error correction. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 7831–7843, Torino, Italia. ELRA and ICCL.

Tannon Kew, Alison Chi, Laura Vásquez-Rodríguez, Sweta Agrawal, Dennis Aumiller, Fernando Alva-Manchego, and Matthew Shardlow. 2023. BLESS: Benchmarking large language models on sentence simplification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13291–13309, Singapore. Association for Computational Linguistics.

Cheng Li, Jindong Wang, Yixuan Zhang, Kaijie Zhu, Wenxin Hou, Jianxun Lian, Fang Luo, Qiang Yang, and Xing Xie. 2023. Large language models understand and can be enhanced by emotional stimuli. *arXiv preprint arXiv:2307.11760*.

Wenwu Li, Xiangfeng Wang, Wenhao Li, and Bo Jin. 2025. A survey of automatic prompt engineering: An optimization perspective. *Preprint*, arXiv:2502.11560.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Mengsay Loem, Masahiro Kaneko, Sho Takase, and Naoaki Okazaki. 2023. Exploring effectiveness of GPT-3 in grammatical error correction: A study on performance and controllability in prompt-based methods. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 205–219, Toronto, Canada. Association for Computational Linguistics.

Ruotian Ma, Xiaolei Wang, Xin Zhou, Jian Li, Nan Du, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Are large language models good prompt optimizers? *arXiv preprint arXiv:2402.02101*.

Kostiantyn Omelianchuk, Andrii Liubonko, Oleksandr Skurzhanskyi, Artem Chernodub, Oleksandr Korniienko, and Igor Samokhin. 2024. Pillars of grammatical error correction: Comprehensive inspection of contemporary approaches in the era of large language models. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, pages 17–33, Mexico City, Mexico. Association for Computational Linguistics.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3845–3864, Dubrovnik, Croatia. Association for Computational Linguistics.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Jipeng Qiang, Minjiang Huang, Yi Zhu, Yunhao Yuan, Chaowei Zhang, and Kui Yu. 2025. Redefining simplicity: Benchmarking large language models from lexical to document simplification. *Preprint*, arXiv:2502.08281.

Vipul Raheja, Dhruv Kumar, Ryan Koo, and Dongyeop Kang. 2023. CoEdIT: Text editing by task-specific instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5274–5291, Singapore. Association for Computational Linguistics.

Horacio Saggion. 2017. *Automatic text simplification*, volume 32. Springer.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *ICLR*.

Kim Cheng Sheang and Horacio Saggion. 2021. Controllable sentence simplification with a unified text-to-text transfer transformer. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 341–352, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Chenming Tang, Fanyi Qu, and Yunfang Wu. 2024. Ungrammatical-syntax-based in-context example selection for grammatical error correction. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1758–1770, Mexico City, Mexico. Association for Computational Linguistics.

Subhadra Vadlamannati and Gözde Şahin. 2023. Metric-based in-context learning: A case study in text simplification. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 253–268, Prague, Czechia. Association for Computational Linguistics.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

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. Association for Computational Linguistics.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. 2023. TEMPERA: test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.