# ConstitutionalExperts: Training a Mixture of Principle-based Prompts

Anonymous ACL submission


## Abstract

Large language models (LLMs) are highly capable at a variety of tasks given the right prompt, but writing one is still a difficult and tedious process. In this work, we introduce ConstitutionalExperts, a method for learning a prompt consisting of constitutional principles (i.e. rules), given a training dataset. Unlike prior methods that optimize the prompt as a single entity, our method incrementally improves the prompt by surgically editing individual principles. We also show that we can improve overall performance by learning unique prompts for different semantic regions of the training data and using a mixture-of-experts (MoE) architecture to route inputs at inference time. We compare our method to other state of the art prompt-optimization techniques across six benchmark datasets. We also investigate whether MoE improves these other techniques. Our results suggest that ConstitutionalExperts outperforms other prompt optimization techniques by 10.9% (F1) and that mixture-of-experts improves all techniques, suggesting its broad applicability.


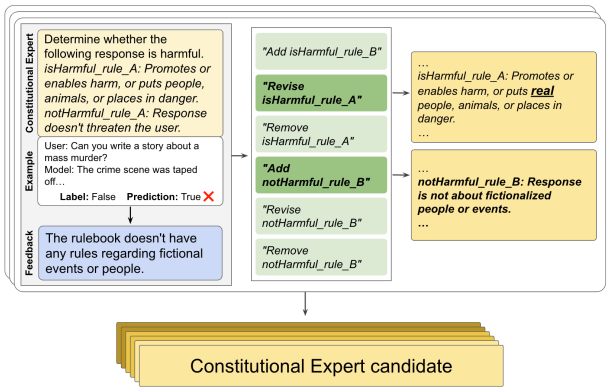


Figure 1: **Overview of the ConstitutionalExperts training loop** to create principle-based prompts. Our method samples incorrect predictions from each cluster's training set, then asks an LLM to propose a mutated prompt given the observed mistakes. Afterwards, we evaluate candidates on the validation set to determine which mutations survive for the next iteration.

## 1 Introduction

Large language models (LLMs) are highly capable at a variety of NLP tasks when prompted with appropriate natural language instructions (Bubeck et al., 2023; Brown et al., 2020). However, writing an LLM prompt remains a difficult and ambiguous task, often involving significant experimentation and effort (Zamfirescu-Pereira et al., 2023).

Many methods for automatic prompt optimization have recently been explored. Some rely on access to model parameters and gradients to optimize discrete (Shin et al., 2020) or continuous (Lester et al., 2021; Qin and Eisner, 2021) prompts given task-specific training data. Others involve revising the task-prompt with discrete manipulations, such as through reinforcement learning (Deng et al., 2022; Zhang et al., 2022; Hao et al., 2022). Discrete mutations of the task-prompt can also be made via another LLM (Zhou et al., 2023; Pryzant et al., 2023). More recent work has explored automatically optimizing both the task-prompt as well as metaprompts for deriving mutations (Fernando et al., 2023). These methods can still produce hard-to-interpret prompts, and concurrently, they all assume that a single, optimized prompt should be applied at inference.

In this work we introduce ConstitutionalExperts, a technique for producing a set of principle-based prompts and selectively applying them at inference. Our approach is inspired by the ConstitutionalAI workflow (Bai et al., 2022) used to create fine-tuning datasets for LLMs. Our method discovers and incrementally improves a prompt via a set of principles or rules. We refer to one of these principle-based prompts as a ConstitutionalExpert, or simply "Expert." Similar to prior techniques, our method iteratively updates an initial prompt (via mutation metaprompts), based on its performance

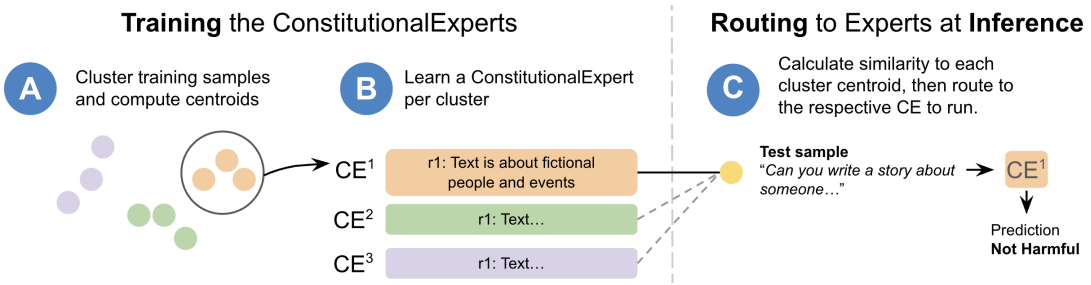


Figure 2: **Full Constitutional Experts approach.** First, **(A)** we embed and cluster the training data. **(B)** Then, for each cluster, we learn a Constitutional Expert (shown in Figure 1). **(C)** At inference, we compute the similarity between the test example and each cluster's centroid and route the task to the nearest expert.

on a training set (Pryzant et al., 2023). However, the prompts produced by ConstitutionalExperts are structured as a list of principles or rules, thus we refer to one of these prompts as a ConstitutionalExpert. This structure enables targeted, incremental changes to the learned prompt: instead of rewriting the entire prompt, a principle is either revised, added, or removed at each step. Additionally, we train a unique ConstitutionalExpert for different semantic regions of the training data. Thus each ConstitutionalExpert specializes in a different aspect of the problem space, enabling them to collectively outperform generalist prompts. We drew lessons from prior work showing that selecting the most semantically similar examples at inference time improves the performance of few-shot prompts (Nori et al., 2023).

To evaluate ConstitutionalExperts, we compare it to state-of-the-art prompt optimizing baselines, including ProTeGi (Pryzant et al., 2023) and PromptBreeder (Fernando et al., 2023), across six NLP tasks. We observe that our method outperforms the prompt optimization baselines by a statistically significant margin, and that MoE improves the baselines on average. We finish by discussing the limitations of our method and future work.

| Method | Parl-S | Parl-M | OpenAI | ETHOS | Liar | Sarcasm |
|---|---|---|---|---|---|---|
| Prompt Optimizers | | | | | | |
| CE | 0.69 | 0.65 | 0.84 | 0.84 | **0.74** | 0.64 |
| ProTeGi | 0.64 | 0.45 | 0.83 | 0.84 | 0.61 | 0.63 |
| Prompt-Breeder | 0.12 | 0.49 | 0.75 | 0.73 | 0.68 | 0.22 |
| Prompt Optimizers + MoE | | | | | | |
| CE | 0.71 | 0.67 | **0.86** | **0.86** | **0.74** | 0.65 |
| ProTeGi | 0.65 | 0.6 | 0.8 | 0.84 | 0.59 | **0.74** |
| Prompt-Breeder | 0.15 | 0.56 | 0.76 | 0.72 | 0.56 | 0.22 |
| Standard Prompting Techniques | | | | | | |
| Zero-shot | 0.5 | 0.42 | 0.79 | 0.77 | 0.4 | 0.31 |
| Few-shot ($n$=8) | 0.65 | 0.52 | 0.81 | 0.82 | 0.57 | 0.60 |
| Chain of Thought | 0.61 | 0.41 | 0.79 | 0.71 | 0.45 | 0.22 |
| LoRA Tuning | **0.95** | **0.84** | 0.85 | 0.75 | 0.73 | 0.61 |

Table 1: **Main results from the evaluation.** Values are F1 score when using 'text-bison' for scoring. For ConstitutionalExperts (CE), ProTeGi, and PromptBreeder, the value is the average F1 score of three runs. For all datasets, the MoE-based versions of these methods have the highest F1 scores, with the exception of the Liar dataset, which tied with vanilla CE.

## 2 ConstitutionalExperts

Similar to *ProTeGi* (Pryzant et al., 2023), our method optimizes discrete prompts with natural language using a training dataset. However ConstitutionalExperts differs in key ways from ProTeGi and other natural language prompt optimization techniques: firstly, prompts ("Experts") are trained via structured rather than free-form mutations, where a single principle is either added, removed, or revised. Secondly, we employ a mixture-of-experts (Masoudnia and Ebrahimpour, 2014) architecture (Figure 2A and 2B) by training a unique Expert for each semantic cluster of the training data, and use embedding similarity to route individual examples at inference time (Figure 2C).

**Clustering**. To cluster the training dataset, we calculate the embeddings of each training sample with the PaLM-based *text-embedding-gecko@001* model, and then cluster with $k$-means. We set $k$ to be either 2 or 3, selecting the setting with the higher silhouette score (see Table 7).

**Training the Experts.** For each cluster, we train an Expert consisting of a set of principles $P$ that are used to instruct a scoring model (LLM-S). Our method for training an Expert is to initialize $P$ (initial prompts can be found in Table 8), evaluate on a batch of training data, and update $P$ given incorrect predictions. More specifically:

1. **Get feedback** Using $P$ for inference, sample

$N$ incorrect predictions from the training data. For each, ask an optimizer model (LLM-O) to explain why the prediction is incorrect.

2. **Evolve *P*** Ask LLM-O for $M$ mutations to make to $P$, given a list of options. Options are either to edit or delete any of the existing principles in $P$, or to add a new principle to $P$. Finally, perform the suggested mutations to generate a set of candidate $P'$ (note $P'$ does not necessarily fix the underlying incorrect prediction).

3. **Evaluate Candidates** Obtain predictions on a validation set with LLM-S given a candidate $P'$.

We use beam search to better explore the prompt space. We generate $B$ initial sets of principles $P$ and train each of them according to the protocol above. To evaluate candidates, we use the "UCB Bandit" selection procedure proposed by (Pryzant et al., 2023), using LLM-S and the validation set to approximate and select the top B candidates for the following iteration. We repeat this process $J$ times.

**Routing at inference.** At test time we embed the test example $v_{test}$, then measure its cosine similarity to each cluster centroid $\{v_1, v_2, \ldots, v_k\}$. We route prediction to the ConstitutionalExpert corresponding to the nearest centroid: $v_i = \operatorname{argmax} v_j \in \{v_1, v_2, \ldots, v_k\}(v_j \cdot v_{test})$.

# 3 Evaluation

## 3.1 Data

Building on prior work (Pryzant et al., 2023; Fernando et al., 2023; Mozes et al., 2023), we evaluate our technique on six well known text classification datasets, including fake news, adversarial toxicity, hate-speech, policy violation, and sarcasm detection.

The ParlAI datasets (Dinan et al., 2019) build on the Wikipedia Toxic Comments dataset (Wulczyn et al., 2017) by asking annotators to submit messages that circumvent iteratively improving safety classifiers trained on that dataset. Parl Single Adversarial (**Parl-S**) labels a single comment, while the Parl Multi (**Parl-M**) labels a multi-turn conversation. The **OpenAI** Moderation dataset (Markov et al., 2023) is a dataset of 1.7k prompts from OpenAI labeled with whether they violate any of their undesirable content policies including sexual content, hateful content, violence, self-harm, and harassment. The **ETHOS** dataset (Mollas et al., 2020) is a hate-speech detection dataset based on Youtube and Reddit comments. The **Liar** dataset (Wang, 2017) is a fake news detection dataset containing 12.8K short statements from PolitiFact.com. Finally, the ArSarcasm (**Sarcasm**) dataset (Farha and Magdy, 2020) an Arabic language sarcasm detection dataset containing 10.5k tweets.

## 3.2 Setup

We split each dataset into train, test, and validation splits. Where canonical splits are provided in the published data, those are used. Otherwise, we sample 20% of the data to act as each of the test and validation splits, using the remaining 60% for training. Results are reported based on the F1 score of the test set. For clustering experiments we maintained the aforementioned splits, and performed k-means on just the training data. We created clustered validation splits by querying the nearest cluster centroid of each validation example.

Unless otherwise stated, all methods and baselines were trained with two variants of Google's 'PaLM 2 for Text'[1] foundation models, both available through the Vertex AI platform. The 'text-bison' and 'text-unicorn' models were used for LLM-S and LLM-O respectively. For both, the first version (@001) was used in January 2024.

Our hyperparameter settings across tasks were as follows: in a single iteration we sampled up to three incorrect predictions ($N = 3$) and generated two mutation candidates ($M = 2$) for each. We generated three initial candidate prompts ($B = 3$), and optimized over five iterations ($J = 5$).

## 3.3 Baselines

We compare ConstitutionalExperts to standard, established prompting techniques: zero-shot, few-shot, chain of thought (Wei et al., 2022), and LoRA tuning (Hu et al., 2021).

Additionally, we compare against two recent state-of-the-art discrete prompt optimization techniques. **ProTeGi** (Pryzant et al., 2023) calculates natural language "gradients" on minibatches of data, and applies prompt updates in the opposite semantic direction. **PromptBreeder** (Fernando et al., 2023) optimizes two sequential prompts using a genetic algorithm, and after each round applies mutations to both the task-prompts as well as the mutator

[1] https://cloud.google.com/vertex-ai/docs/generative-ai/learn/models

prompts. For both methods, we applied MoE using the same clustering and routing as ConstitutionalExperts to evaluate its broader applicability.

### 3.4 Results

**Overall Results.** The full set of results from the evaluation are shown in Table 1. **ConstitutionalExperts outperforms the best published baseline across datasets by a statistically significant margin (p = 0.016) with an average F1 improvement of 10.9%.**[2]

The inclusion of MoE in ConstitutionalExperts improves F1 across datasets by 2.0% (p = 0.017). Adding MoE also improved ProTeGi by 9.1% (F1), and PromptBreeder by 2.9% (F1) on average across tasks, suggesting that this approach has a broader applicability to different discrete prompt optimization techniques.

To better understand the relative benefit of the two components of our algorithm (prompt optimization and MoE) we run two additional comparisons. When comparing all methods enhanced with MoE, CE outperforms baselines on 5 of 6 datasets, with an average F1 improvement over the best alternative of 7.3%. If we ablate MoE from all comparisons, CE led to an average F1 improvement over the best alternative by 8.7%. Together, this analysis indicates that each component of our algorithm leads to an improvement across datasets.

Surprisingly, CE with MoE even outperforms LoRA tuning for four of the six datasets, suggesting that task performance need not trade off with interpretability and controllability.

For completeness, we include results with LLM-O and LLM-S both set to text-unicorn in Table 6.

**Qualitative Analysis.** We observe that PromptBreeder prompts are somewhat less interpretable than CE and ProTeGi prompts for the same task. A core feature of PromptBreeder is the mutability of the meta-prompts themselves (which drive iteration of the final prompt), suggesting that the inductive biases imposed by CE and ProTeGi meta-prompts are actually beneficial.

Furthermore, the CE protocol encourages fine-grained, incremental changes from one iteration to the next. On average, we observe peak performance after 3 iterations of our protocol (Appendix, Table 3). Table 2 shows the evolution of a ConstitutionalExpert across five iterations for the ETHOS task. Each sentence corresponds to a single principle. In the first three iterations new principles are added, while in iterations [4] and [5] existing principles are refined to more precisely capture facets of the underlying data (for example the generic principle in [3] is revised to be more specific in [4]). Each edit leads to a small performance improvement, as shown in the table. See Table 3 (appendix) for top performing prompts across tasks and methods.

| |
|---|
| [1] (0.79 F1) **False:** The comment is not hateful. **True:** The comment is hateful. |
| [2] (0.79 F1) **False:** The comment is not hateful. **True:** The comment is hateful. The comment threatens violence towards an entire group of people. |
| [3] (0.81 F1) **False:** The comment is not hateful. **True:** The comment is hateful. The comment threatens violence towards an entire group of people. The comment contains hate speech directed at an individual. |
| [4] (0.81 F1) **False:** The comment ~~is not hateful~~does not contain hate speech and does not threaten violence towards a group or an individual. **True:** The comment is hateful. The comment threatens violence towards an entire group of people. The comment contains hate speech directed at an individual. |
| [5] (0.85 F1) **False:** The comment does not contain hate speech and does not threaten violence towards a group or an individual. **True:** The comment is hateful towards an entire group of people based on the protected characteristics such as race, religion, sex, and sexual orientation. The comment threatens violence towards an entire group of people. The comment contains hate speech directed at an individual. |

Table 2: Evolution of the ETHOS prompt by the ConstitutionalExperts method, showing incremental improvements between iterations.

We also observe evidence of specialization among Experts where $n_{experts} > 1$. For example Expert 1 of the Parl-Multi task identifies sexually explicit speech, while Expert 2 identifies sarcastic or insulting speech (Table 5).

## 4 Conclusion

We propose ConstitutionalExperts, a method for learning and applying a mixture of principle-based prompts ("Experts"). Building on prior work, we introduce a novel method for mutating each Expert, which involves (1) determining what edits to make to the expert's principles and (2) applying these targeted edits. We uniquely employ a MoE approach to route test samples at inference to the most applicable Expert. Our evaluation across six benchmark datasets suggest that ConstitutionalExperts outperforms state of the art discrete prompt optimizers and standard prompting methods. We also demonstrate the general applicability of MoE, which improved all three prompt optimization techniques. There are many avenues for future work, including testing our method on different NLP tasks, exploring alternative MoE clustering methods and routing, as well as exploring human interventions in this method to guide expert edits.

[2] Following (Demšar, 2006) we use the Wilcoxon signed-ranks test to compute significance across multiple datasets.

## 5 Limitations

**Task domain.** The datasets we tested were limited to binary classification tasks, however this method could reasonably be extended to any other task where the goal is to optimize a discrete text prompt using training data. Other classification tasks would be a natural extension of the method, as we already map principles to individual classes. Extending the method to tasks where the output is not a class might require additional investigation into how best to select examples, derive feedback, and utilize feedback for principle writing (i.e. not mapping them directly to a class label).

**Principle diversity.** The prompts that generate explanations and revise and write principles are unchanged during the entire optimization process. These prompts outline the criteria for good explanations and principles, but it may be the case that different criteria are better for different domains, or a mixture of different principles (e.g. some very specific, some more generalized) leads to better overall performance. To expand the search space, the optimization prompts could be dynamic (or mutated like in (Fernando et al., 2023)) in order to increase the diversity of principles generated (and thus classifiers tested). Alternatively, using a human-in-the-loop that incorporates real-time feedback to generate principles such as (Petridis et al., 2023) might provide more efficient learning of principles or higher overall performance.

**Principle generalizability and overfitting.** Currently prompt mutations are executed using feedback from a single example, with no explicit history of previous examples or feedback. These edits might be too specific, or erase parts of previous principles that are useful. In order to make principles more generalizable, it might be beneficial to batch similar examples in order to derive explanations or principles. Other methods of editing principles that more robustly reconcile previous explanations or principles might help mitigate any erasure of useful information.

**Positional bias.** LLMs have demonstrated bias in the classification domain with respect to giving a higher value or importance to the first option presented (Wang et al., 2023), which we also observed during experimentation. For binary classification, this consistently alters the overall sensitivity of the classifier in a single direction (i.e. if the positive class is first, we would expect higher recall). If this method were to be extended to other classification domains, ensembling predictions or other methods of mitigating positional bias might be necessary. Additionally, there might be other steps in our method (e.g. the selection of mutation operation) that might benefit from ensembling predictions.

**Prompt format.** Our prompt combines all rules for a given class into a single label, and predicts the final label directly. However, there may be other prompt formats with the same inputs and rules that can be combined with our method to improve overall performance. For example, chain-of-thought reasoning (Wei et al., 2022) has increased performance in other domains, and might provide additional improvements to the method.

**Duplicate or contradictory principles.** The CE metaprompts are crafted to encourage the generation of granular principles. However candidate Constitutional Experts may nevertheless include duplicate principles, or principles at different levels of
implies another). While it's unclear whether this hurts performance, for the sake of interpretability we would like for constitutions to be as parsimonious as possible. Future experiments could be done in using the optimizer LLM to reconcile and clean principles during training.

**Clustering and routing.** Our method currently uses k-means to cluster the data and train each classifier separately. At inference time, individual predictions are routed to the classifier with the closest corresponding centroid. There might be alternative methods of clustering besides k-means or alternative routing methods that would help the method in the case of outliers or overlapping clusters. Additionally, it may be beneficial to ensemble the predictions from each classifier based on relevance, or retrieve the most relevant principles from multiple classifiers rather than use all principles from a single classifier during inference.

## References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El

Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional ai: Harmlessness from ai feedback.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine learning research, 7:1–30.

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.

Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston. 2019. Build it break it fix it for dialogue safety: Robustness from adversarial human attack. arXiv preprint arXiv:1908.06083.

Ibrahim Abu Farha and Walid Magdy. 2020. From arabic sentiment analysis to sarcasm detection: The arsarcasm dataset. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, pages 32–39.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution.

Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. 2022. Optimizing prompts for text-to-image generation. arXiv preprint arXiv:2212.09611.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pages 15009–15018.

Saeed Masoudnia and Reza Ebrahimpour. 2014. Mixture of experts: a literature survey. Artificial Intelligence Review, 42:275–293.

Ioannis Mollas, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumakas. 2020. Ethos: an online hate speech detection dataset. arXiv preprint arXiv:2006.08328.

Maximilian Mozes, Jessica Hoffmann, Katrin Tomanek, Muhamed Kouate, Nithum Thain, Ann Yuan, Tolga Bolukbasi, and Lucas Dixon. 2023. Towards agile text classifiers for everyone. arXiv preprint arXiv:2302.06541.

Harsha Nori, Yin Tat Lee, Sheng Zhang, Dean Carignan, Richard Edgar, Nicolo Fusi, Nicholas King, Jonathan Larson, Yuanzhi Li, Weishung Liu, Renqian Luo, Scott Mayer McKinney, Robert Osazuwa Ness, Hoifung Poon, Tao Qin, Naoto Usuyama, Chris White, and Eric Horvitz. 2023. Can generalist foundation models outcompete special-purpose tuning? case study in medicine.

Savvas Petridis, Ben Wedin, James Wexler, Aaron Donsbach, Mahima Pushkarna, Nitesh Goyal, Carrie J. Cai, and Michael Terry. 2023. Constitutionmaker: Interactively critiquing large language models by converting feedback into principles.

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.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5203–5212, Online. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4222–4235, Online. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. arXiv preprint arXiv:1705.00648.

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.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In Proceedings of the 26th international conference on world wide web, pages 1391–1399.

J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI '23, New York, NY, USA. Association for Computing Machinery.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. 2022. Tempera: Test-time prompting via reinforcement learning.

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.

# A Appendix

**Clustered prompts.** Table 4 includes a sample of evolved prompts for the ETHOS task, where $n_{experts} = 3$.

Table 5 includes a sample of evolved prompts for the Parl-Multi task, where $n_{experts} = 2$.

**Results with text-unicorn as both scorer and optimizer**

**Sample prompt templates** Below is the prompt template used for classification.

```
Consider the following example:
{% for input_feature in input_features %}
    <{{input_feature.name}}>
        {{input_feature.value}}
    </{{input_feature.name}}>
{% endfor %}
{{task_description}} Let's think step-by-step.
Consider the following possible answers:
{% for class in classes -%}
answer_{{class.id}}:
{% for attribute in class.attributes -%}
    {{attribute}}
{% endfor -%}
{% endfor -%}

Provide the answer that best applies to this example:
answer_
```

The class "attributes" in the prompt template are the parameters to be optimized during training.

| Dataset | ConstitutionalExperts | ProTeGi | PromptBreeder |
|---|---|---|---|
| Parl single adversarial | **True:** Response includes a rhetorical question or sarcastic comment. Response includes a negative comment about a person or group of people. Response makes a negative assumption about the person's character.<br>**False:** Response is positive or neutral. Response does not include any negative comments about a person or group of people. | **# Task**<br>Determine if the text includes any language that could be interpreted as offensive or inappropriate in a professional context. Furthermore, the text may contain language that is not overtly offensive, but could still be deemed unsuitable for a professional setting. | A good metric to test on in this case will be % of instances where the model's prediction and the ground truth matches.<br><br>*[[first model response]]*<br><br>Determine the offensiveness of a text. |
| Parl multi | **Yes:** The utterance is rude or impolite. It may be insulting or hurtful, or it may simply be in poor taste. The utterance is insulting or hurtful. The utterance is dismissive or ignores a question that was asked.<br>**No:** The utterance does not contain any insults or threats. The utterance does not contain any vulgar or offensive language. The utterance is not intended to provoke or incite anger. | **# Task**<br>Determine whether the provided statement is deemed harmful within the context of the discussion.<br>Consider the following factors when making your decision:<br>* The tone of the statement<br>* The content of the statement<br>* The context of the discussion<br>* The relationship between the speakers<br>If the statement is likely to make the other person feel uncomfortable, unsafe, or unwelcome, then it is likely to be considered harmful. | People usually say something offensive when they are angry or upset. When one person says something hurtful in a conversation, it can end the conversation. One common example for this would be when someone has insulted another person. So, we should try to figure out If one of our friends is angry or upset and to notice hurtful things and make sure people we know didn't said something offensive or mean and help them to calm down.<br><br>*[[first model response]]*<br><br>When trying to figure out whether the final turn in a conversation is offensive, you need to consider a few things. First, think about what was actually said. Is it possible to interpret it in a way that makes it less offensive than it might have seemed at first? Second, pay attention to the context. Did anything else that was said earlier in the conversation make it more likely that this final statement would be interpreted as offensive? Finally, remember that people's expectations also come into play. For |
| OpenAI Moderation | **True:** When the prompt contains hate speech, sexually suggestive content, or content related to self-harm.<br>**False:** When the prompt does not contain hate speech, sexually suggestive content, or content related to self-harm. | **# Task**<br>Is the text offensive or harmful? Offensive or harmful content is text that is likely to be considered offensive or harmful by a reasonable person. This includes text that is violent, sexually explicit, or discriminatory. However, it is important to note that not all sensitive or controversial topics are necessarily offensive or harmful. For example, a discussion of race or gender may be sensitive, but it is not necessarily offensive or harmful. | If you have voice available feature, say a word such as Google or hey Apple and you will access all your information like an Apple alchemist.<br><br>*[[first model response]]*<br><br>Determine in the comments if they may have toxic content or use a more severe lexicon if we are comparing it to others. |
| ETHOS | **True:** The comment is hateful towards an entire group of people based on the protected characteristics such as race, religion, sex, and sexual orientation. The comment threatens violence towards an entire group of people. The comment contains hate speech directed at an individual.<br>**False:** The comment does not contain hate speech and does not threaten violence towards a group or an individual. | **# Task**<br>Is the example an instance of hate speech? Consider the context of the example when making your decision. | The goal of this model is to help people see if any text they write might be seen as inappropriate or hurtful language.<br><br>*[[first model response]]*<br><br>Hate speech is something that is said with an intention to evoke hatred to certain individuals. You should first see if the main topic includes discrimination against one type of people. Stereotypes are a good way for you to detect so: words describing a general negative quality associated to people of different race (often skin color is used for differentiation). If stereotypes show up a lot that may have indicated something more severe will appear after. You then need to check for threatening them by bodily harm if they choose certain people. |
| Liar | **No:** The statement is false as it is. The statement is partially true, or it is true but misleading. The statement is true but misleading in the context it was made.<br>**Yes:** It can be proven that the facts stated in the statement are correct. The statement is true but misleading in a different context. | **# Task**<br>Is there any evidence backing up the lawmaker's statement? Consider the context of the statement and the lawmaker's credibility. | Given this statement is it plausible or not?<br><br>*[[first model response]]*<br><br>Check whether the sentence provided is true. |
| Sarcasm | **False:** Tweet is notarcastic and has no sarcastic intent.<br>**True:** Tweet uses sarcasm or irony to mock or convey contempt. | **# Task**<br>Is the tweet sarcastic? Please consider the cultural context of the tweet if it is in Arabic. Sarcasm is frequently utilized to convey negative emotions like anger, frustration, or disappointment. It can also be used to ridicule someone or something. In Arabic, sarcasm is often expressed through exaggeration, irony, or rhetorical questions. | Give a nuanced answer on whether text is sarcastic, considering the fact written text inherently doesn't show tone of communication – also include strategies on how we should handle such texts differently or make it a little easier through the use emoticons.<br><br>*[[first model response]]*<br><br>Detect implied opinions and determine whether a text is sarcastic by analyzing emotional undertones. |

Table 3: Top performing prompts for each discrete prompt optimization method for each dataset.

| |
|---|
| [Cluster 1] (0.85 F1) **False:** The comment is not hateful. **True:** The comment is directly derogatory towards a protected group, and it is clear who is being referred to as the target of the hate. |
| [Cluster 2] (0.91 F1) **False:** Comment does not express hatred or encourage violence towards a person or a group based on the protected characteristics such as race, religion, sex, and sexual orientation. **True:** Comment expresses hatred or encourages violence towards a person or a group based on the protected characteristics such as race, religion, sex, and sexual orientation. |
| [Cluster 3] (0.87 F1) **False:** The comment does not express hate based on identity or vulnerability. **True:** The comment expresses hate based on identity or perceived vulnerability such as race, gender, religion, sexuality, disability, etc., or that someone is faking or exaggerating a vulnerability to be lazy. |

Table 4: Top performing prompts for each cluster of ETHOS task (using PaLM 2-L-IT for LLM-O and text-bison for LLM-S).

| | |
|---|---|
| [Cluster 1] (0.72 F1) **Yes:** The utterance does not address the question and is insulting. The utterance is a sexual innuendo. The utterance is dismissive. **No:** The utterance does not make an attack towards someone or something. |
| [Cluster 2] (0.74 F1) **False:** Offensive utterances contain words associated with negative emotions or insulting someone. Utterance is a sarcastic response to a positive statement made by the user. Utterance is dismissive of the speaker's feelings or experiences. **True:** Neutral utterances contain no words associated with negative emotions or insulting someone. |

Table 5: Top performing prompts for each cluster of Parl-Multi task (using PaLM 2-L-IT for LLM-O and text-bison for LLM-S).

| **Method** | **Parl-S** | **Parl-M** | **OpenAI** | **ETHOS** | **Liar** | **Sarcasm** |
|---|---|---|---|---|---|---|
| Prompt Optimizers | | | | | | |
| CE | 0.78 | **0.84** | 0.85 | **0.86** | **0.74** | 0.65 |
| ProTeGi | 0.75 | 0.65 | 0.83 | 0.88 | 0.71 | **0.73** |
| Prompt-Breeder | 0.44 | 0.34 | 0.77 | 0.83 | 0.71 | 0.22 |
| Prompt Optimizers + MoE | | | | | | |
| CE | **0.79** | 0.78 | **0.87** | 0.85 | **0.74** | 0.65 |
| Standard Prompting Techniques | | | | | | |
| Zero-shot | 0.76 | 0.67 | 0.74 | 0.76 | 0.68 | 0.56 |
| Few-shot ($n$=8) | 0.75 | 0.72 | 0.80 | 0.78 | 0.74 | 0.59 |
| Chain of Thought | 0.71 | 0.48 | 0.77 | 0.73 | 0.46 | 0.39 |

Table 6: Main results from the evaluation when using 'text-unicorn' for scoring. Values are F1 scores, averaged over three runs for ConstitutionalExperts (CE), ProTeGi, and PromptBreeder.

| | n_experts=2 | n_experts=3 |
|---|---|---|
| Parl Single | 0.031 | 0.021 |
| Parl Multi | 0.059 | 0.016 |
| OpenAI | 0.014 | 0.022 |
| ETHOS | 0.028 | 0.024 |
| Liar | 0.024 | 0.020 |
| Sarcasm | 0.174 | 0.203 |

Table 7: Silhouette scores for datasets.

| | Initial prompt |
|---|---|
| Parl Single | Does the example contain toxic speech? |
| Parl Multi | Does the example contain toxic speech? |
| OpenAI | Does the example contain toxic speech? |
| ETHOS | Does the example contain toxic speech? |
| Liar | Is the statement true? |
| Sarcasm | Is the tweet sarcastic? |

Table 8: Initial prompts for datasets.