Reducing the Scope of Language Models

David Yunis [1,2] *, **Siyu Huo** [1], **Chulaka Gunasekara**[1], **Danish Contractor**[1]

[1] IBM Research AI
[2] Toyota Technological Institute at Chicago
dyunis@ttic.edu, siyu.huo@ibm.com, chulaka.gunasekara@ibm.com, danish.contractor@ibm.com

Abstract

Large language models (LLMs) are deployed in a wide variety of user-facing applications. Typically, these deployments have some specific purpose, like answering questions grounded on documentation or acting as coding assistants, but they require general language understanding. In such deployments, LLMs should respond only to queries that align with the intended purpose and reject all other requests, such as generating poetry or answering questions about physics, a task we refer to as 'scoping'. We conduct a comprehensive empirical evaluation of various methods, ranging from prompting, fine-tuning to preference learning and the recently proposed general alignment technique known as Circuit Breakers (CB). Across three families of language models and a broad variety of tasks, we show that it is possible to scope language models. We examine scoping for multiple topics, and fine-grained topics. We ablate diversity of irrelevant queries, layer different techniques, conduct adversarial evaluations and more. Among other results, we find that when diverse examples of irrelevant queries are available, simple supervised fine-tuning produces the best results, but when such diversity is low, Circuit Breakers perform quite well. One can often get the benefits of both methods by layering them in succession. We intend our study to serve as a practitioner's guide to scoping LLMs.

Code — https://github.com/IBM/llm-scoping
Extended version — https://arxiv.org/abs/2410.21597

1 Introduction

In recent years, large language models have surged into public awareness. One major recent addition is the "alignment" process through Reinforcement Learning with Human Feedback (RLHF) (Christiano et al. 2017; Ouyang et al. 2022), which has made the current generation of language models much less likely to emit toxic content than previous generations (Wolf, Miller, and Grodzinsky 2017), and thus much more acceptable for general use. As a result, many businesses and individuals now feel more comfortable using these technologies than they did in the past.

Although we have generally capable language models that can refuse to answer toxic or dangerous queries, deploying them still remains challenging. Even if they avoid producing harmful content, they often respond to any question, relevant or not, without discernment. This becomes a problem when we wish to use them in specific contexts: e.g. shopping bots currently give coding advice[1] or answer other questions,[2] while assistive copilots can be taken off course by prompt injections.[3] Thus, there is still a need to scope language models for these specific uses.

We define LLM *scoping* as a conditional generation task in which a language model must: (i) identify whether an input query falls within a relevant domain, (ii) reject irrelevant queries, and (iii) maintain high-quality generation for relevant queries. This contrasts with traditional text classification, which maps inputs to discrete labels without requiring natural language generation.

Let $\mathcal{Q}$ denote the set of all possible natural language queries. Let $\mathcal{D}_{\text{rel}} \subset \mathcal{Q}$ denote the subset of *relevant* or *in-domain* (should be *accepted*) queries, and let $\mathcal{D}_{\text{irr}} = \mathcal{Q} \setminus \mathcal{D}_{\text{rel}}$ denote the set of *irrelevant* or *out-of-domain* (should be *rejected*) queries. Let $\mathcal{R}$ be the space of valid natural language responses, and let $\perp \notin \mathcal{R}$ be a special token representing rejection.

The scoping is a function:

$$f_\theta : \mathcal{Q} \to \mathcal{R} \cup \{\perp\}$$

such that:

$$f_\theta(q) \in \mathcal{R}, \quad \text{if } q \in \mathcal{D}_{\text{rel}},$$
$$f_\theta(q) = \perp, \quad \text{if } q \in \mathcal{D}_{\text{irr}}.$$

In contrast, text classification models are not required to preserve generation quality, whereas in LLM scoping, degrading the model's performance on $\mathcal{D}_{\text{rel}}$ constitutes a failure of the task. Further, unlike the traditional LLM refusal task in the context of safety and alignment, which typically corresponds to a much smaller reject set than accept set, the scoping tasks consider the opposite.

*David Yunis is a PhD student at the Toyota Technological Institute at Chicago. Work was performed during an internship at IBM.

[1] https://shorturl.at/qf3FA

[2] https://www.forbes.com/sites/lesliekatz/2024/07/13/amazon-ai-shopping-assistant-rufus-answers-non-shopping-questions-too/

[3] https://oecd.ai/en/incidents/2025-10-08-0fbf

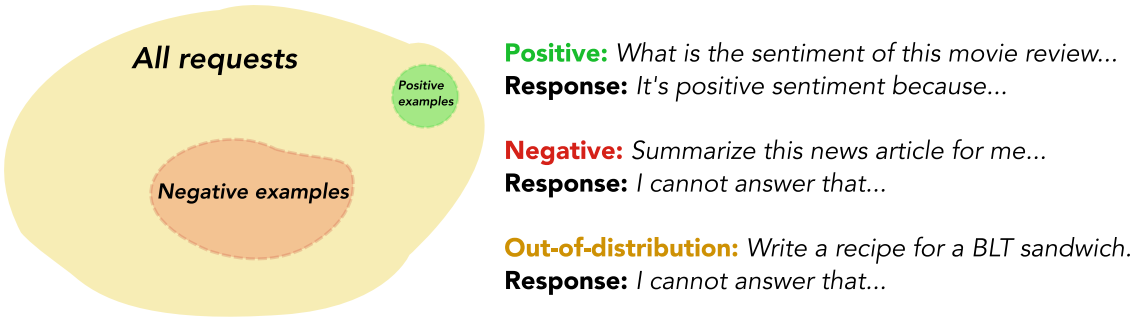

Figure 1: We study the ability to scope language models to specific topics. We assume access to a set of relevant (accept) queries and irrelevant (reject) queries, where the accept queries correspond to a relatively narrow domain. We examine how well different methods cause the language model to accept only the relevant examples, while rejecting all other examples, including out-of-distribution requests that weren't seen during training.

Currently, LLMs can be scoped through two-stage approaches like relevance classifiers, or system prompting and then text generation, but these options are brittle (Chao et al. 2023; Mehrotra et al. 2023; Zeng et al. 2024; Wei, Haghtalab, and Steinhardt 2023) and easy to circumvent. We shed further light on this problem, conducting a comprehensive empirical study on scoping language models to specific tasks. We apply existing methods to this problem, including system prompting, supervised fine-tuning, preference learning (Rafailov et al. 2024), probing, and a recently-introduced method called Circuit Breakers (Zou et al. 2024). We scope language models more broadly for multiple tasks, and more finely for specific niche tasks. We ablate over diversity of training sets, language model size, adversarial prompting and more. Our specific contributions include:

- We introduce the task of scoping LLMs
- We conduct a broad experimental exploration of existing methods for this task
- We show that it is possible to scope language models, even for multiple and fine-grained tasks
- We find that when training data exhibits high diversity, supervised fine-tuning yields the best performance
- Conversely, in settings with low data diversity, the Circuit Breakers method (Zou et al. 2024) provides superior results
- Finally, we show it is possible to layer these two, often preserving the best performance

2 Related Work

Aligning Language Models: The advent of the current era of language models has been marked by a process of aligning language models so that generations are more helpful, and safer for deployment (Ouyang et al. 2022; Bai et al. 2022a). The primary way this is accomplished is through reinforcement learning with human feedback (RLHF) (Christiano et al. 2017) which was first proposed in robotic simulation tasks. RLHF proceeds by collecting preference pairs of completions, and training a reward model from human judgments on those preference pairs, then performing reinforcement learning with the language model against that reward model. From tasks in simulation, it was developed in language (Stiennon et al. 2020), until it reached its current state. Other works have removed the human aspect of human feedback, allowing for synthetic feedback from models (Bai et al. 2022b; Sudalairaj et al. 2024). Lately, Rafailov et al. (2024) have removed the need for a reward model, making for a stabler and simpler objective function without many of the complexities of RL training. A budding line of work also explores aligning not just to a single reward model, but preferences of many different individual users (Chakraborty et al. 2024; Lee et al. 2024). All of these methods focus on some general notion of alignment, without considering the specific task, unlike our work.

Adapting for Specific Purposes: Typically after pretraining, language models go through an instruction finetuning stage, where they gain the ability to follow instructions (Mishra et al. 2022; Ouyang et al. 2022; Wei et al. 2022). After this, they proceed through an alignment phase as discussed above, usually to avoid harmful behavior (Bai et al. 2022a). It is possible to adapt language models for specific purposes simply with a system message (Touvron et al. 2023), but many examples of black-box adversarial attacks (Chao et al. 2023; Anil et al. 2024; Wei, Haghtalab, and Steinhardt 2023; Zeng et al. 2024) demonstrate it is difficult only to rely on the system prompt for such control. Wallace et al. (2024) propose finetuning with different levels of priority, similar to Zhang et al. (2023b), but these works focus primarily on general safety and not the task. These ideas are based on the fact that current language models can often be distracted by irrelevant context (Shi et al. 2023; Yoran et al. 2024). Thus, it seems important to finetune the language model if we want it to be deployed to a particular domain. For domains where there is sufficient data, we may also pretrain and fix the language model's purpose ahead of time (Beltagy, Lo, and Cohan 2019; Wu et al. 2023; Li et al. 2023) or continue pretraining from a base language model (Gururangan et al. 2020). It is an open question however whether finetuning retains the robustness capabilities, or if it is similarly as brittle as system prompting for out-of-distribution questions.

Refusal in Language Models: As our work deals with scoping models to refuse irrelevant queries, we review refusal.

Category	Example task	# Datasets	# Tasks	# Instances
Sentiment Analysis (SA)	Predicing if a movie review is relevant or not	8	10	31248
Toxic Language Detection (TLD)	Detecting if a comment contains cursing	5	9	33849
Summarization (S)	Condensing a news article	4	4	13096
Text Completion (TC)	Filling in the blanks in a transcript	3	3	10515
Story Composition (SC)	Writing a new ending for a story	4	4	15556
Dialogue Generation (DG)	Continuing a dialogue between parties	3	4	12744
Program Execution (PE)	Computing the result of a described function	26	26	94001
Question Answering (QA)	Answering biology multiple-choice questions	19	30	84065
GSM8k (Cobbe et al. 2021)	Answering simple math word problems	1	1	5978
Alpaca (Taori et al. 2023)	General requests like writing a recipe for lunch	1	-	18793

Table 1: Breakdown of data. We reserve at least 20% of the data from each dataset for validation. We will use at most 2048 instances from each category for training, though this is sampled from a much larger number. PE is so large as the data is synthetically generated. All categories above the divider will be used for training and evaluation, while categories below the divider are only used for out of distribution evaluation.

More detail is available in a comprehensive survey by Wen et al. (2024). One common case to train for refusal is when the answer is unknown or the model is unconfident (Zhang et al. 2023a; Cao 2023; Xu et al. 2024). Another is for unsafe inputs (Varshney et al. 2023; Zhang et al. 2023b; Wallace et al. 2024). Supervised fine-tuning (SFT) to reject unsafe prompts can still lead to unsafe behavior, though parameter efficient methods like LoRA (Hu et al. 2022) have better tradeoffs (Brahman et al. 2024). Both Brahman et al. (2024) and Cheng et al. (2024) take an approach to refusal using SFT and DPO, which we will adapt to our case. Other methods to induce refusal may be prompt-based (Xie et al. 2023; Zhang et al. 2024) or based on probing model representations (Kadavath et al. 2022; Slobodkin et al. 2023). Zou et al. (2024) design a method that conditionally rejects unsafe inputs based on orthogonalizing internal representations that we will adapt for our study. Though these methods lay out a set of techniques to explore for our task, all of them are oriented toward general alignment qualities like safety, as opposed to specific tasks that we will explore.

3 Experimental Setup

We would like to scope language models to provide completions to relevant tasks, and reject queries corresponding to irrelevant tasks. In particular, we assume we are given a set of "relevant" or "accept" queries $\{q_{rel}|q_{rel} \sim \mathcal{D}_{rel}\}$, where $\{\mathcal{D}_{rel}\}$ is a set of *accepted* tasks, and a set of "irrelevant" queries $\{q_{irr}|q_{irr} \sim \mathcal{D}_{irr}\}$ where $\{\mathcal{D}_{irr}\}$ is a set of *rejected* tasks. We are given a language model $f_\theta : q \mapsto y$ which predicts completion y from input q, with parameters θ; a classifier $g : y \mapsto c \in \{0, 1\}$ which decides whether a LLM completion is accepted (0) or rejected (1). We would like to compute an update Δ such that we minimize $\mathbb{E}_{q_{rel}}\, g(f_{\theta+\Delta}(q_{rel}))$ and maximize $\mathbb{E}_{q_{irr}}\, g(f_{\theta+\Delta}(q_{irr}))$. Thus we want 'relevant' queries to be accepted and 'irrelevant' queries to be rejected.

As an additional goal of scoping, we would like performance on the accept tasks not to degrade. Given a scoring function $h : (q, y) \mapsto s \in [0, 1]$ which scores the completion on task performance where 1 is best, we would also like to maximize $\mathbb{E}_{q_{rel}}\, h(q_{rel}, f_{\theta+\Delta}(q_{rel}))$.

Datasets & Metrics

We conduct many experiments with different mixtures of accept and reject queries. In order to standardize the format, we draw prompts from Super-NaturalInstructions (SNI) (Wang et al. 2022). SNI is a meta-dataset composed of many different "tasks", sometimes with multiple tasks per dataset, for example generating questions from passages for a reading comprehension dataset, or generating answers to provided questions from the same reading comprehension dataset. Each task, specified by a task instruction, comes with a collection of examples. We use SNI as it is publicly available, and contains a broad range of complex tasks which current language models should be able to perform. To get our training datasets, we first manually select a set of tasks that are straightforward to automatically evaluate, leaving out many more subjective tasks that may require a human reader. We then group those tasks that we select by category provided from SNI. Details and statistics on categories are provided in Table 1.

Each of these categories contains multiple datasets, so the distribution for each task is quite broad. We will also combine multiple tasks in the accept or reject set. For all experiments, we always evenly split the training data for accept/reject set between all tasks. We reserve at least 20% of the prompts as a validation set that are not seen during training. Where not specified, we use 2048 prompts for the accept set, and 2048 prompts for the reject set. We evaluate Sentiment Analysis and Toxic Language Detection with accuracy (the classes are mostly balanced), while for other tasks we use a standard metric for generation, Rouge-L (Lin 2004), between the generation and ground truth completion as a proxy for performance (Accept Score). Our goal is mostly to study rejection behavior, so a rough performance proxy is all we need.

Irrespective of the specific accept and reject sets used during training, we evaluate performance across all categories listed in Table 1. We then report the average rejection rates separately for the accept set (Accept), all in-distribution reject sets (ID Reject), and all out-of-distribution reject sets (OOD Reject). Note if the training set consists of SA in Accept, and S in Reject, OOD Reject will contain the 8 other categories. Tasks above the divider in Table 1 will be used in different ex-

periments for both training and evaluation, while tasks below the divider will only be used for OOD evaluation. As stated previously, ideally we would like to have 0% rejection on Accept, and 100% rejection on ID Reject and OOD Reject.

Methods

For all methods that require training the language model, we use LoRA (Hu et al. 2022) training with rank 16, $\alpha = 16$ and dropout of 0.05. We use the Adam (Kingma 2014) optimizer without any regularization and tune learning rates (see Appendix B).

System Prompting (Sys.): The simplest method to scope language models is simply to instruct them to refuse irrelevant prompts. For example, for SA the system prompt is: *You are an assistant who only answers requests related to Sentiment Analysis. For all other requests you respond "I cannot answer that."* With multiple accept categories, we comma separate the category names (e.g. *"...related to Sentiment Analysis, Text Completion and Summarization..."*). This system prompt is prepended to all instructions at evaluation time. In addition, all other methods also use the system prompt both at training and evaluation time. This is similar to methods proposed by Xie et al. (2023); Zhang et al. (2024).

Supervised Fine-Tuning (SFT): Supervised Fine-Tuning (SFT) consists of tuning the language model to produce particular outputs. For the accept tasks the completions y_{rel} are the groundtruth completions provided by the dataset. For the reject tasks, the completions y_{irr} are always *"I cannot answer that."*. We tune learning rate and step budget for SFT. This is a similar approach to Brahman et al. (2024); Cheng et al. (2024). In experiments, the $\mathcal{L}_{gen}$ is about generation (task completion) loss between y_{rel} and $f_\theta(q), q \in \mathcal{D}_{\text{rel}}$, and the $\mathcal{L}_{rej}$ is about irrelevant task rejection loss between y_{irr} and $f_\theta(q), q \in \mathcal{D}_{\text{irr}}$. The total scoping loss is $\mathcal{L}_{scope} = \mathcal{L}_{gen} + \lambda\mathcal{L}_{rej}$. We use balanced scoping loss where $\lambda = 1$ for loss computing. As the finetuning dataset can be quite small, loss is only computed on the completions so as to avoid overfitting to the small set of instructions, agreeing with common practice (Mishra et al. 2022; Ouyang et al. 2022; Wei et al. 2022).

Direct Preference Optimization (DPO): Given its role in post-training we explore a preference learning method. We choose to experiment on Direct Preference Optimization (DPO) (Rafailov et al. 2024), as it does not require an additional reward model. DPO requires pairs of preference data, so for accept queries we provide the dataset completion as preferred, and the completion *"I cannot answer that."* as rejected. For reject queries we do the reverse, preferring *"I cannot answer that."* over the ground truth completion. For DPO we tune learning rate, step budget, and the loss weighting term regularizing the KL divergence from the base model predictions. This is similar to Brahman et al. (2024); Cheng et al. (2024).

Two-stage on Probing Classifier (Probe): Probes of representations are a common method to accomplish tasks as they base predictions on the internal state of the language model (Conneau et al. 2018; Tenney et al. 2019; Zou et al. 2023a). Previous work on Circuit Breakers (Zou et al. 2024) showed that probing representations was competitive for detecting dangerous language. However, they only designed probes on a single layer of a language model. Here we design a stronger probe. Once an instruction is fed to the frozen language model, we first remove the first position as it is quite anomalous due to large magnitude (Xiao et al. 2024), then we average all positions per layer and normalize the average vector to norm 1 so as to match norms between layers. Finally, we concatenate the mean-pooled representations from each layer to form a single feature vector, which is then passed through a two-layer multilayer perceptron (MLP) with a hidden dimension of 256 to perform binary classification, determining whether to accept or reject the input. Only the MLP layers are trained, and we tune the learning rate and step budget. This is justified by prior choices in Kadavath et al. (2022); Slobodkin et al. (2023); Zou et al. (2024). We use this probe in a two-stage setup where the probe first classifies the request and if it is determined to be an 'accept' task, a response generated via an LLM.

Circuit Breakers (CB): (Zou et al. 2024) first introduce a method they call Circuit Breakers (CB) for accepting normal queries while rejecting dangerous ones. We repurpose their method for this task. Essentially, given a function which extracts the representations of a language model at particular layers, they design an optimization objective with two components: $\mathcal{L}_{gen, q\in\mathcal{D}_{\text{rel}}}(q, \Delta) = \|\text{rep}(f_\theta(q)) - \text{rep}(f_{\theta+\Delta}(q))\|_2^2$ and $\mathcal{L}_{rej, q\in\mathcal{D}_{\text{irr}}}(q, \Delta) = \max\{0, \cos(\text{rep}(f_\theta(q)), \text{rep}(f_{\theta+\Delta}(q))\}$. The total loss is $\mathcal{L}_{scope} = \alpha(t)\mathcal{L}_{gen} + \beta(t)\mathcal{L}_{rej}$ where the two components of the loss are scheduled over time.

This loss function keeps the representations of accept tasks from drifting, while making the representations of reject tasks orthogonal from their original position. This orthogonalization breaks the language model generation on bad inputs. For CB we tune learning rate and step budget and more, and show later that CB is particularly sensitive to hyperparameters.

SFT $\rightarrow$ CB: As we will see, SFT and CB tend to be the best methods for scoping in slightly different circumstances. In order to improve accept task performance and preserve the benefits of both, we propose to layer CB on top of SFT. We first run SFT, then run CB training afterwards. We keep hyperparameters from the SFT and CB tuning respectively.

Detecting rejection

Though a strong language model judge (Zheng et al. 2023) may seem like a good choice for judging rejection, we followed prior work in first experimenting with simpler heuristics (Zou et al. 2023b, 2024) based on string matching and the rejection behavior of Circuit Breakers. We found these heuristics to be very strong, and much less expensive than running a frontier judge for the many experiments in this paper. We also experimented with a smaller hosted language model judge, but found its performance much poorer than the heuristics. More details on evaluation are available in Appendix B.

4 Experiments

In this section we explore a number of empirical questions: can we scope language models, how does scoping behave

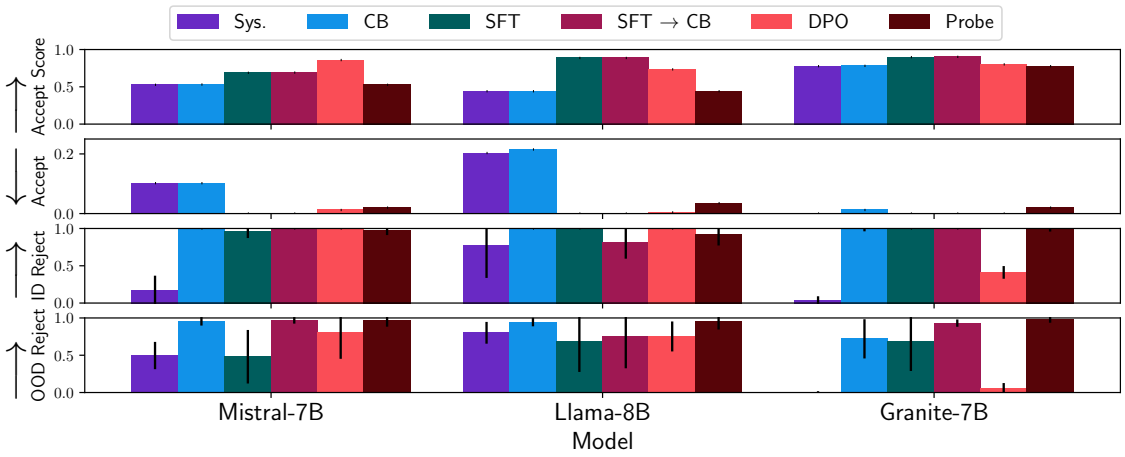

Figure 2: Scoping across different language models. We see that system prompting is insufficient, and different methods have different success rates for different models. Clearly it is possible to scope language models to particular distributions.

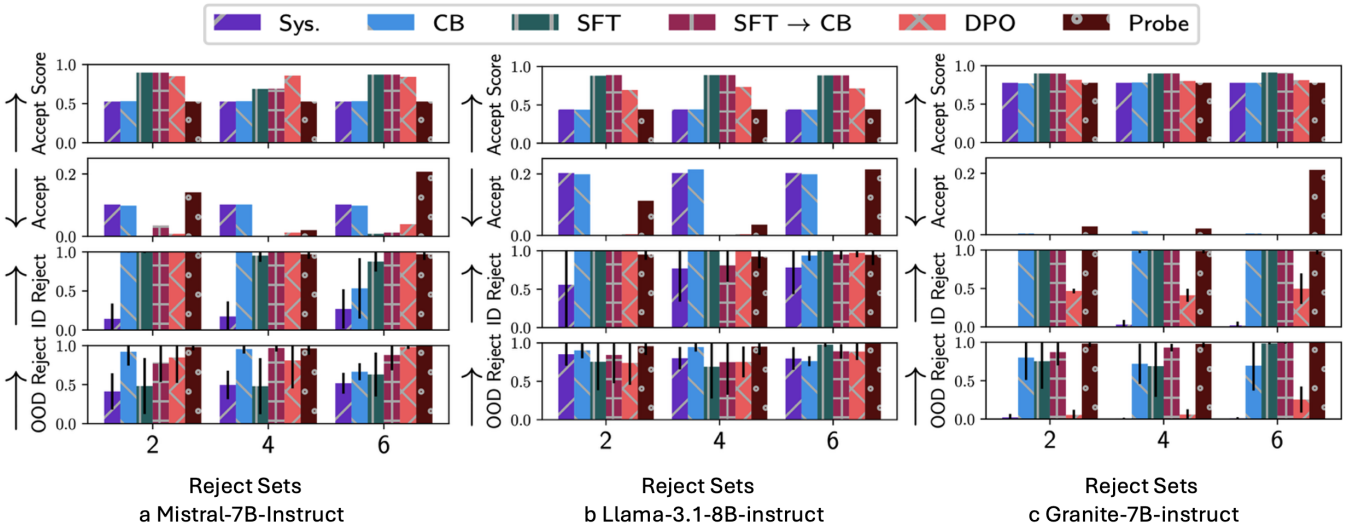

Figure 3: Results for increasing diversity of rejection set. We see across models that CB performs relatively better than SFT when data diversity is low, but SFT is much stronger with more rejections sets. Probing appears strong across the board, though sometimes leads to overrejection on the Accept set.

across scale, how much diversity is needed for scoping, or whether scoping is possible for multiple tasks simultaneously. We aim to be comprehensive, thus demonstrate results across 2-3 different categories per dataset. Where not detailed, our accept sets will be Sentiment Analysis (SA), Summarization (S) and Program Execution (PE).

All experiments contain evaluations of task performance (Accept Score) on the accept set (which should be high), rejection rate on the in-distribution accept (Accept) set (which should be low) as well as rejection rate on the in-distribution reject set (ID Reject) and out of distribution data (OOD Reject) (which should be high). We describe experiments in broad strokes, and defer precise details on hyperparameters to Appendix B. We begin by presenting results across a variety of language models (`Mistral-7B-Instruct-v0.2`, `granite-7b-instruct`, `Llama-3.1-8B-Instruct`) (Jiang et al. 2023; Sudalairaj et al. 2024; Grattafiori et al. 2024). We explore scale on the Llama-3.2-instruct family, and for other results choose Mistral-7b-instruct as we do not have the compute or space to run every experiment on every model.

Scoping language models

We start with the basic question: is it possible to scope language models? We explore each method across a variety of models using Sentiment Analysis (SA) as our accept task in Figure 2, where we see that although system prompting is insufficient, a broad variety of different methods are successful to different extents with different models.

	Classification and Generation				Math and Programming			
Method	Accept Score	Accept	ID Reject	OOD Reject	Accept Score	Accept	ID Reject	OOD Reject
Sys.	0.25 ± 0.18	0.10 ± 0.11	0.70 ± 0.04	0.42 ± 0.11	0.15 ± 0.14	0.16 ± 0.22	0.33 ± 0.36	0.28 ± 0.22
CB	0.25 ± 0.18	0.10 ± 0.12	$\mathbf{1.0 \pm 0.0}$	0.79 ± 0.23	0.14 ± 0.15	0.16 ± 0.22	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.96 \pm 0.08}$
SFT	$\mathbf{0.46 \pm 0.24}$	$\mathbf{0.01 \pm 0.02}$	0.95 ± 0.03	0.28 ± 0.16	0.26 ± 0.04	$\mathbf{0.0 \pm 0.0}$	0.99 ± 0.01	0.52 ± 0.25
SFT $\rightarrow$ CB	$\mathbf{0.46 \pm 0.24}$	0.07 ± 0.11	$\mathbf{1.0 \pm 0.0}$	0.54 ± 0.22	$\mathbf{0.27 \pm 0.04}$	0.26 ± 0.11	$\mathbf{1.0 \pm 0.0}$	0.64 ± 0.22
DPO	0.21 ± 0.11	$\mathbf{0.01 \pm 0.02}$	$\mathbf{1.0 \pm 0.0}$	0.54 ± 0.06	0.23 ± 0.20	0.01 ± 0.01	$\mathbf{1.0 \pm 0.0}$	0.78 ± 0.28
Probe	–	0.19 ± 0.21	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.91 \pm 0.13}$	–	0.04 ± 0.05	$\mathbf{1.0 \pm 0.0}$	0.93 ± 0.11

Table 2: Evaluation when accepting multiple categories. We show it is quite possible to do so. In general SFT-based methods are best for in-domain performance, and CB or Probe are strong choices for OOD rejection.

Scoping over different scales

Here we ask: how does scoping vary with model scale? We use the Llama-3.1 family (Grattafiori et al. 2024) as an example. We fix the accept task as Sentiment Analysis again and check how different methods behave with 1B, 3B and 8B parameter instruct-models. We see in Figure 4 broadly that larger models lead to improved results. The only other major surprise is that probing performs poorly with smaller language models. This may be due to the fact that it is difficult for a small probe to disentangle the representations of smaller models, as they are lower dimension and the information is more compressed.

Rejection set diversity

One of the most critical questions when attempting to restrict the generations of language models is what data might be necessary to do so. If models overfit to a particular data distribution, then it may be difficult to reject requests that were not specified in the training distribution. Thus, here we ask: how much data diversity is necessary in the rejection set to robustly scope models? If very little diversity is needed, and rejection extends to OOD requests, then adapting models to new deployments becomes quite inexpensive. In Figure 3 we study Sentiment Analysis across different models with more and more diverse rejections sets consisting of an increasing number of tasks, and defer results on additional tasks with Mistral-7b-instruct to Appendix A.1.

In general we see that CB performs relatively better than SFT when data diversity is low, but SFT is much stronger with more rejection sets. Probing appears strong across the board, though it sometimes leads to overrejection on the Accept set, which is undesirable. In general it may be quite straightforward to collect diverse data, so unless the practitioner is quite constrained, it's worthwhile to always start with Probe and SFT.

Accepting multiple tasks

Here we ask: is it possible to still reject tasks when there are multiple tasks in the accept set? Such a setting is natural as most language models will have a few different specific uses, like a programming bot that can write code and also answer questions about documentation. We demonstrate results on `Mistral-7b-Instruct-v0.2` in Table 2 with two choices of accepts sets: Classification and Generation (SA,S,TLD,SC,TC,DG) and Math and Programming (PE,GSM8k). We use the opposite set as the ID reject set in this case.

Classification and Generation: We see strong scores for SFT-based methods here. On the accept set, Probe is worst, while Sys. is poor leading CB and SFT-CB to suffer. In distribution all methods work well except Sys. and SFT. Out of distribution, CB and Probe perform well, while SFT-CB and DPO are even.

Math and Program Execution: SFT-based methods perform best on the task. Surprisingly, SFT-CB has a very high rejection rate on the accept task. In-distribution every method but Sys. works well. Out-of-distribution there is a similar story to the previous case, where CB works quite well, and Probe is also strong, but the rest less so.

Takeaways: We see that it is possible to support multiple accept tasks. In particular, CB and Probe work best for out-of-distribution evaluation, but as its performance on the accept task is tied to the system prompt, any issues with the base model will carry over.

Additional analysis

Here we briefly discuss some additional results based on `Mistral-7b-Instruct-v0.2`, deferring full treatment to the Appendix.

Adversarial Evaluation: We examine the behavior of different methods under adversarial prompts in Appendix A.4, where we find that CB-based methods are more robust than others, echoing results of Zou et al. (2024).

Representation Analysis: To get a sense of how different methods operate, we study how the representations of tokens change before and after training. In summary, we found that DPO and SFT only change the representations of the tail of the context, whereas for CB all representations have changed and make CB more robust under attack. See detailed analysis in Appendix A.3.

Precise Scoping: We find that one can scope precisely, (e.g. only News summarization instead of all summarization). In general many methods are appropriate for this, though consistent with prior results SFT suffers with a low diversity of training examples. For more details, see Appendix A.5.

Tuning of CB: In general, it was much more difficult to tune CB than SFT. Depending on the model, the optimal hyperparameters in terms of the target layers, and the choice

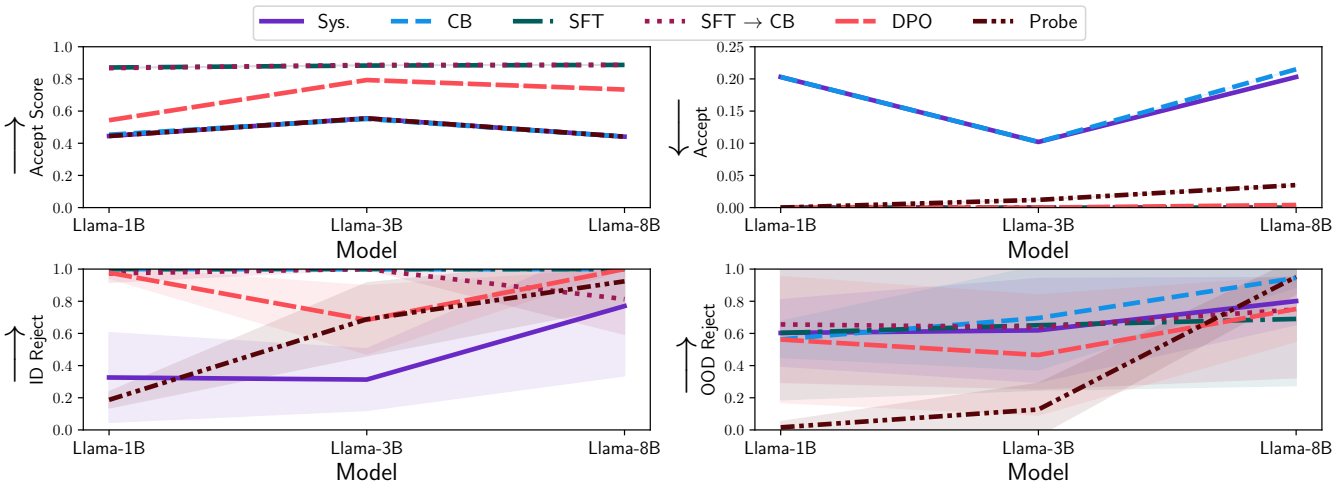

Figure 4: Scoping across different model scales. Larger models lead to improved results. In particular probing performs poorly at small scales, possibly due to the inability of a small probe to disentangle the representations well.

of α made a large difference. This is important as CB often had very strong performance where appropriately tuned, but was less stable than other methods. We explore this more in Appendix A.6.

The effect is particularly clear on the in-distribution rejection set, but preceding sections demonstrate that most methods are fairly comparable in distribution. Out of distribution, the effect of CB is much less, though still there is a much more substantial difference from the original model than SFT or DPO which make only small changes to the tail of context in deeper layers. With SFT-CB, we can clearly see the layering of the tail edit as well as the orthogonalization across the entire context.

Effect of Data Quantity: We find that most methods work quite well with very little data (as little as 128 instances). DPO in particular benefits monotonically, while CB has issues as the raw number of training examples in the dataset scales, perhaps due to the difficulty of simultaneous orthogonalization of many different reject instances, see Appendix A.7 for more details.

Effect of LoRA Rank: Overall, it does appear that rank can have a substantial effect on the performance of methods. While DPO seems to scale monotonically with LoRA rank, CB-based methods have a sweet spot for performance, above which it seems optimization becomes difficult.

5 Discussion

Though current language models are generally applicable, there is still a need at deployment time to specify the kinds of queries they should and should not be able to answer. Otherwise, agents deployed in the wild may be easy to distract, and if used as a part of a pipeline may lead to cascading errors. Hence scoping is crucial. In this work, we conducted a comprehensive empirical study of scoping language models using three model families and multiple tasks.

Our findings reveal several key insights. First, system prompting alone is generally inadequate across a range of models and datasets. While performance varies depending on the specific method, model, and dataset, certain trends emerge. Supervised fine-tuning (SFT) tends to perform well when the training data is diverse, whereas Circuit Breakers (CB) is more effective in low-data regimes with less diversity, likely because the orthogonalization objective it employs is easier to optimize in such settings. Probing methods can also yield strong results, provided that the probe is sufficiently expressive to disentangle the model's internal representations; however, this approach incurs additional inference overhead due to the use of an auxiliary model. Combining SFT and CB typically results in performance that reflects the strengths of both approaches, but this layered method is sensitive to the failure of either component, which can degrade overall performance.

We saw that it was possible to scope across language model scales, for multiple tasks at a time and for very fine-grained tasks. We demonstrated that different methods have different effects on the internal representations of the models: SFT and DPO only modify the tail of the language model context, unlike CB which modifies representations across the context. Such different behavior may explain why CB is stronger under adversarial attacks (Appendix A.4), the original setting it was proposed for (Zou et al. 2024).

While these results indicate that CB can be a promising approach, it presents considerable challenges in practice. Specifically, its performance can fluctuate wildly based on very small step-count difference. Additionally, optimal target layers may need to be selected on a per-model basis, further complicating its application. Given these sensitivities, we recommend defaulting to simpler methods such as supervised fine-tuning (SFT) or probing, particularly in settings where data diversity is not a limiting factor.

References

Anil, C.; Durmus, E.; Sharma, M.; Benton, J.; Kundu, S.; Batson, J.; Rimsky, N.; Tong, M.; Mu, J.; Ford, D.; et al.

2024. Many-shot jailbreaking. *Anthropic, April*.

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Bai, Y.; Kadavath, S.; Kundu, S.; Askell, A.; Kernion, J.; Jones, A.; Chen, A.; Goldie, A.; Mirhoseini, A.; McKinnon, C.; et al. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Beltagy, I.; Lo, K.; and Cohan, A. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

Brahman, F.; Kumar, S.; Balachandran, V.; Dasigi, P.; Pyatkin, V.; Ravichander, A.; Wiegreffe, S.; Dziri, N.; Chandu, K.; Hessel, J.; et al. 2024. The art of saying no: Contextual noncompliance in language models. *arXiv preprint arXiv:2407.12043*.

Cao, L. 2023. Learn to Refuse: Making Large Language Models More Controllable and Reliable through Knowledge Scope Limitation and Refusal Mechanism. *arXiv preprint arXiv:2311.01041*.

Chakraborty, S.; Qiu, J.; Yuan, H.; Koppel, A.; Huang, F.; Manocha, D.; Bedi, A. S.; and Wang, M. 2024. MaxMin-RLHF: Towards equitable alignment of large language models with diverse human preferences. *arXiv preprint arXiv:2402.08925*.

Chao, P.; Robey, A.; Dobriban, E.; Hassani, H.; Pappas, G. J.; and Wong, E. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.

Cheng, Q.; Sun, T.; Liu, X.; Zhang, W.; Yin, Z.; Li, S.; Li, L.; He, Z.; Chen, K.; and Qiu, X. 2024. Can AI Assistants Know What They Don't Know? In *Forty-first International Conference on Machine Learning*.

Christiano, P. F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Conneau, A.; Kruszewski, G.; Lample, G.; Barrault, L.; and Baroni, M. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In Gurevych, I.; and Miyao, Y., eds., *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2126–2136. Melbourne, Australia: Association for Computational Linguistics.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Gururangan, S.; Marasović, A.; Swayamdipta, S.; Lo, K.; Beltagy, I.; Downey, D.; and Smith, N. A. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8342–8360.

Hu, E. J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Kadavath, S.; Conerly, T.; Askell, A.; Henighan, T.; Drain, D.; Perez, E.; Schiefer, N.; Hatfield-Dodds, Z.; DasSarma, N.; Tran-Johnson, E.; et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Kingma, D. P. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lee, S.; Park, S. H.; Kim, S.; and Seo, M. 2024. Aligning to thousands of preferences via system message generalization. *arXiv preprint arXiv:2405.17977*.

Li, R.; Zi, Y.; Muennighoff, N.; Kocetkov, D.; Mou, C.; Marone, M.; Akiki, C.; Jia, L.; Chim, J.; Liu, Q.; et al. 2023. StarCoder: may the source be with you! *Transactions on Machine Learning Research*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

Mehrotra, A.; Zampetakis, M.; Kassianik, P.; Nelson, B.; Anderson, H.; Singer, Y.; and Karbasi, A. 2023. Tree of attacks: Jailbreaking black-box llms automatically. *arXiv preprint arXiv:2312.02119*.

Mishra, S.; Khashabi, D.; Baral, C.; and Hajishirzi, H. 2022. Cross-Task Generalization via Natural Language Crowdsourcing Instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3470–3487.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Shi, F.; Chen, X.; Misra, K.; Scales, N.; Dohan, D.; Chi, E. H.; Schärli, N.; and Zhou, D. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, 31210–31227. PMLR.

Slobodkin, A.; Goldman, O.; Caciularu, A.; Dagan, I.; and Ravfogel, S. 2023. The Curious Case of Hallucinatory (Un)answerability: Finding Truths in the Hidden States of Over-Confident Large Language Models. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 3607–3625. Singapore: Association for Computational Linguistics.

Stiennon, N.; Ouyang, L.; Wu, J.; Ziegler, D.; Lowe, R.; Voss, C.; Radford, A.; Amodei, D.; and Christiano, P. F. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021.

Sudalairaj, S.; Bhandwaldar, A.; Pareja, A.; Xu, K.; Cox, D. D.; and Srivastava, A. 2024. Lab: Large-scale alignment for chatbots. *arXiv preprint arXiv:2403.01081*.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford Alpaca: An Instruction-following LLaMA model.

Tenney, I.; Xia, P.; Chen, B.; Wang, A.; Poliak, A.; McCoy, R. T.; Kim, N.; Van Durme, B.; Bowman, S. R.; Das, D.; et al. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Varshney, N.; Dolin, P.; Seth, A.; and Baral, C. 2023. The art of defending: A systematic evaluation and analysis of llm defense strategies on safety and over-defensiveness. *arXiv preprint arXiv:2401.00287*.

Wallace, E.; Xiao, K.; Leike, R.; Weng, L.; Heidecke, J.; and Beutel, A. 2024. The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions. *arXiv preprint arXiv:2404.13208*.

Wang, Y.; Mishra, S.; Alipoormolabashi, P.; Kordi, Y.; Mirzaei, A.; Arunkumar, A.; Ashok, A.; Dhanasekaran, A. S.; Naik, A.; Stap, D.; et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36.

Wei, J.; Bosma, M.; Zhao, V.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

Wen, B.; Yao, J.; Feng, S.; Xu, C.; Tsvetkov, Y.; Howe, B.; and Wang, L. L. 2024. The art of refusal: A survey of abstention in large language models. *arXiv preprint arXiv:2407.18418*.

Wolf, M. J.; Miller, K.; and Grodzinsky, F. S. 2017. Why we should have seen that coming: comments on Microsoft's tay" experiment," and wider implications. *Acm Sigcas Computers and Society*, 47(3): 54–64.

Wu, S.; Irsoy, O.; Lu, S.; Dabravolski, V.; Dredze, M.; Gehrmann, S.; Kambadur, P.; Rosenberg, D.; and Mann, G. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Xiao, G.; Tian, Y.; Chen, B.; Han, S.; and Lewis, M. 2024. Efficient Streaming Language Models with Attention Sinks. In *The Twelfth International Conference on Learning Representations*.

Xie, Y.; Yi, J.; Shao, J.; Curl, J.; Lyu, L.; Chen, Q.; Xie, X.; and Wu, F. 2023. Defending chatgpt against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5(12): 1486–1496.

Xu, H.; Zhu, Z.; Ma, D.; Zhang, S.; Fan, S.; Chen, L.; and Yu, K. 2024. Rejection Improves Reliability: Training LLMs to Refuse Unknown Questions Using RL from Knowledge Feedback. *arXiv preprint arXiv:2403.18349*.

Yoran, O.; Wolfson, T.; Ram, O.; and Berant, J. 2024. Making Retrieval-Augmented Language Models Robust to Irrelevant Context. In *The Twelfth International Conference on Learning Representations*.

Zeng, Y.; Lin, H.; Zhang, J.; Yang, D.; Jia, R.; and Shi, W. 2024. How johnny can persuade llms to jailbreak them: Rethinking persuasion to challenge ai safety by humanizing llms. *arXiv preprint arXiv:2401.06373*.

Zhang, H.; Diao, S.; Lin, Y.; Fung, Y. R.; Lian, Q.; Wang, X.; Chen, Y.; Ji, H.; and Zhang, T. 2023a. R-tuning: Teaching large language models to refuse unknown questions. *arXiv preprint arXiv:2311.09677*.

Zhang, Y.; Ding, L.; Zhang, L.; and Tao, D. 2024. Intention analysis prompting makes large language models a good jailbreak defender. *arXiv preprint arXiv:2401.06561*.

Zhang, Z.; Yang, J.; Ke, P.; and Huang, M. 2023b. Defending large language models against jailbreaking attacks through goal prioritization. *arXiv preprint arXiv:2311.09096*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36: 46595–46623.

Zou, A.; Phan, L.; Chen, S.; Campbell, J.; Guo, P.; Ren, R.; Pan, A.; Yin, X.; Mazeika, M.; Dombrowski, A.-K.; et al. 2023a. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

Zou, A.; Phan, L.; Wang, J.; Duenas, D.; Lin, M.; Andriushchenko, M.; Wang, R.; Kolter, Z.; Fredrikson, M.; and Hendrycks, D. 2024. Improving Alignment and Robustness with Short Circuiting. *arXiv preprint arXiv:2406.04313*.

Zou, A.; Wang, Z.; Carlini, N.; Nasr, M.; Kolter, J. Z.; and Fredrikson, M. 2023b. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.