20 November 2024, Preprint v1 · CC-BY 4.0 Qeios PREPRINT

Research Article

Improving Instruction-Following in Language Models through Activation Steering

Alessandro Stolfo[1], Vidhisha Balachandran[2], Safoora Yousefi[2], Eric Horvitz[2], Besmira Nushi[2]

1. ETH Zürich, Zurich, Switzerland; 2. Microsoft Research (India), Bengaluru, India

The ability to follow instructions is crucial for numerous real-world applications of language models. In pursuit of deeper insights and more powerful capabilities, we derive instruction-specific vector representations from language models and use them to steer models accordingly. These vectors are computed as the difference in activations between inputs with and without instructions, enabling a modular approach to activation steering. We demonstrate how this method can enhance model adherence to constraints such as output format, length, and word inclusion, providing inference-time control over instruction following. Our experiments across four models demonstrate how we can use the activation vectors to guide models to follow constraints even without explicit instructions and to enhance performance when instructions are present. Additionally, we explore the compositionality of activation steering, successfully applying multiple instructions simultaneously. Finally, we demonstrate that steering vectors computed on instruction-tuned models can transfer to improve base models. Our findings demonstrate that activation steering offers a practical and scalable approach for fine-grained control in language generation.

Corresponding authors: Alessandro Stolfo, stolfoa@ethz.ch; Besmira Nushi, besmira.nushi@microsoft.com

1. Introduction

Instruction-following capabilities of large language models (LLMs) have enhanced their practical applications for real-world usage. These advances are powered by instruction-tuning methods[1][2][3][4][5], which align the model's responses with user objectives, addressing the gap between pre-training and end-user needs[6]. Instruction tuning allows users to specify constraints on attributes like format, tone, or length, which direct the model's behavior and output[7][8][9]. Gaining a deeper understanding of how LLMs internally represent and follow these instructions is essential for developing more controllable and reliable models.[1]

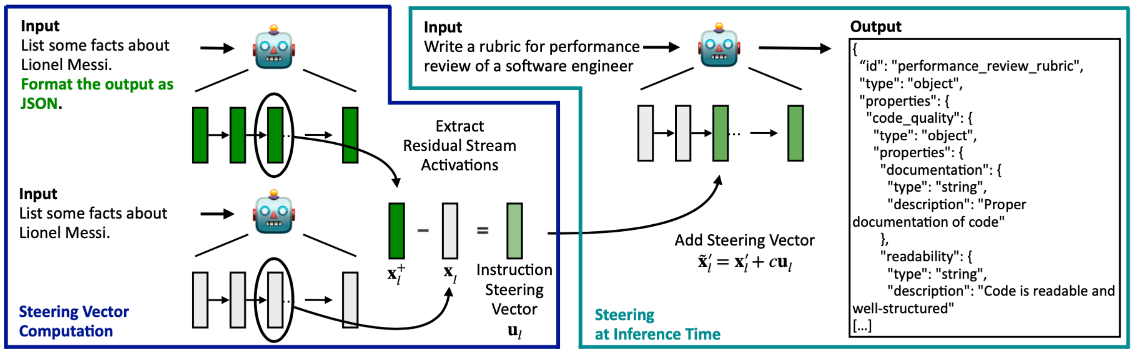
In this paper, we use a mechanistic method to investigate how language models internally represent various instructions and use these representations to influence and control the model's behavior. Prior research has shown that vector representations can be computed for tasks learned in context[10][11][12] and various stylistic and semantic input features inter alia[13][14][15][16][17]. These representations can be used for activation steering[18]: directly intervening on the model's activations to guide the generation. This approach has been successfully applied to control text attributes such as honesty[19][20][13], sentiment[21], output style and tone[22][12][23][24], harmfulness[25][26], and sycophancy[27][28].

However, user instructions in generative tasks can be more complex and involve multiple parameters that need to be attended to and satisfied during generation[29]. For example, users may ask for a response to include bullet lists, have a certain number of sentences, or include/exclude specific content. The complexity of such instructions stems from the high number of possible variations, making it impractical to generate post-training data for all scenarios. In the quest for finding more efficient methods for controlling instruction-following and guided by previous research in language model representations, we pose the question: Can we efficiently extract vector representations that encode and control specific instruction-following behavior?

We investigate this using a contrastive, additive steering method[22][27] that computes the difference in activations between inputs with and without an instruction (Figure 1, left), and applies this difference to guide the model to better follow instructions on new inputs (Figure 1, right). While previous work focused on high-level stylistic aspects, we target more diverse, verifiable instructions[30] that complement the base request's semantics (e.g., formatting, length constraints). These instructions are lower-level and more specific, with multiple possible instantiations (e.g., "Do not mention the word {keyword}"). It remains unclear whether models represent such instructions linearly and whether these representations can be used to reliably elicit specific behaviors, which would enable finer control of LLM outputs.

We conduct experiments using the Phi-3[31], Gemma 2 2B and 9B[32], and Mistral 7B[33] models, focusing on three types of instructions: output format (§3), output length (§4), and the inclusion/exclusion of specific words (§5). Our results on the IFEval dataset[30] provide evidence that vector representations can encode a wide range of instructions and enhance the model's instruction-following performance. Notably, we demonstrate that activation steering not only helps models follow constraints when no instruction is provided in the input, but it can also reinforce instruction adherence even when instructions are explicitly present, potentially countering *instruction drift*[34]. Furthermore, we show that it is possible to simultaneously steer for multiple constraints, such as controlling both format and length (§6). Finally, we present surprising evidence that cross-model steering–using vectors computed on an instruction-tuned model to steer a base model– yields better adherence than same-model steering (§7), suggesting new possibilities for transferring task-specific skills across models, similar to "task arithmetic"[35].

Our work represents an important step toward operationalizing techniques from mechanistic interpretability to achieve practical improvements in scenarios with real-world utility.


Input
List some facts about Lionel Messi.
Format the output as JSON.
Extract Residual Stream Activations
Input
List some facts about Lionel Messi.
$\mathbf{x}_l^+$
$\mathbf{x}_l$
Instruction Steering Vector $\mathbf{u}_l$
Steering Vector Computation
Input
Write a rubric for performance review of a software engineer
Add Steering Vector
$\tilde{\mathbf{x}}_l' = \mathbf{x}_l' + c\mathbf{u}_l$
Steering at Inference Time
Output

```
{
 "id": "performance_review_rubric",
 "type": "object",
 "properties": {
  "code_quality": {
   "type": "object",
   "properties": {
    "documentation": {
     "type": "string",
     "description": "Proper
documentation of code"
    },
    "readability": {
     "type": "string",
     "description": "Code is readable and
well-structured"
[...]
```

Figure 1. Instruction Steering Process. Steering vectors are computed as the difference in residual stream activations between inputs with and without the instruction. These vectors are then applied during inference to adjust the model's activations, guiding it to follow the desired instruction.

2. Steering for Instruction Adherence

In this section, we define the types of instructions we consider (§2.1), introduce the methodology used to compute the vectors and the steering procedure (§2.2), and describe the experimental setup, including the data, metrics, and evaluation details (§2.3).

2.1. Types of Instructions

The concept of instruction-following has been used to describe the broad capability of a model to answer any zero-shot query[3][4]. However, following prior work[30][7][29], we adopt a more specific definition of instruction that refers to a constraint applied to a particular aspect of the model's output. These constraints are self-contained, modular, and can be imposed on various base queries. This definition decouples the ability of the model to follow the instruction from its factual knowledge and domain-specific skills. In our work, we focus on three specific types of instructions:

1. **Format instructions**, which dictate how the output should be presented. For instance, the model may be asked to produce responses in a specific format (e.g., "Provide the answer in JSON format") or highlight parts of the response in a particular way (e.g., "Use asterisks to emphasize at least two sections of the answer").
2. **Length instructions**, which specify the desired length of the output (e.g., "Answer using at most three sentences").
3. **Word-specific instructions**, which control the inclusion or exclusion of specific words or phrases in the output (e.g., "Do not include the word AI in the answer").

2.2. Steering Procedure

Activation (or representation) engineering involves constructing vectors of activation values that cause desired changes to output text when added to the forward passes of a frozen LLM[13]. To identify a direction in the model's residual stream^2 that encodes information about a specific instruction, we use a technique called *difference-in-means*[36]. This method effectively isolates key feature directions in the model's internal activations[17][21] and has been used to control various behaviors such as refusal and sycophancy[25][27]. We adapt this method to support finer-grained instructions with multiple possible instantiations (e.g., varying length, varying words to include and exclude). The process involves pairing two versions of the same request: one with only the base query (e.g., "List some facts about Lionel Messi") and another that additionally includes the instruction we want to represent (e.g., "List some facts about Lionel Messi, ensuring the output is valid JSON").

Let us denote these two inputs by x (base query) and x^+ (base query with instruction), and consider a set of N such pairs (x_i, x_i^+), $i \in \{1, \dots, N\}$. Let $x_{i,l}, x_{i,l}^+ \in \mathbb{R}^{d_{model}}$ be the values of the residual stream vector on the two queries at the last token of the input at layer $l \in \{1, \dots, L\}$. We isolate the internal representation corresponding to the instruction by computing the difference in the residual stream vectors between the paired inputs. More formally, we compute a vector $u_l \in \mathbb{R}^{d_{model}}$ representing the steering direction at layer l for a given instruction as:

$$u_l = \frac{v_l}{\|v_l\|}, \text{ where } v_l = \frac{1}{N}\sum_{i}^{N}(x_{i,l}^{+} - x_{i,l}). \tag{1}$$

Averaging over different base queries allows us to capture the activation values most closely associated with the instruction, independent of the base query. The computation of the steering direction is carried out using the representations at the last token of the input, which effectively encapsulate the model's behavior not only for the next-token-prediction task but also for the entire generation that follows[11][23].

After identifying the steering direction, we compute the steering vector by re-scaling the unit vector u_l by a coefficient, $c \in \mathbb{R}$. For format instructions, we use a systematic scaling approach where the value of c is selected to ensure that the residual stream activations are mapped to their mean value on inputs that contain the instruction in question. In particular, during a forward pass on a new example with residual stream values $x' \in \mathbb{R}^{L \times d_{model}}$ at a given token position, we compute:

$$c = \bar{z} - x_l'^{T} u_l, \text{ where } \bar{z} = \frac{1}{N}\sum_{i}^{N} x_{i,l}^{+T} u_l. \tag{2}$$

This dynamic adjustment allows the model to effectively incorporate the constraint without over- or under-correcting its behavior. For length instructions, which have a more continuous nature, we experiment and show results with different values of c, illustrating their impact on the model's output. Finally, for word-specific constraints, we compute the weight using Eq. 2 and additionally perform a small grid search over neighboring values on a held-out set of examples to fine-tune the steering effect. The steering vector $c\mathbf{u}_l$ is then added to the corresponding residual stream layer and the forward pass is resumed with the updated residual stream value $\tilde{\mathbf{x}}_l' = \mathbf{x}_l' + c\mathbf{u}_l$. This procedure is carried out at a single layer across all token positions, motivated by previous findings that show models tend to deviate from instructions as they generate more tokens[34]. To find the optimal layer for steering, we perform a sweep across a subset of the model's layers, measuring the effect on a held-out set of queries. Layer and weight selection are further discussed in Apps. D and F.

2.3. Experimental Setup

Data. We use an augmented version of the IFEval dataset[30], which consists of 25 distinct instructions, each paired with multiple base queries and expressed in different phrasings, for a total of 541 prompts. To evaluate format instructions, we focus on a subset of 163 examples that specifically relate to the output format. This subset includes instructions such as "The entire output should be wrapped in JSON format," requests to use a particular language (e.g., "Please respond using only the Punjabi language"), formatting requirements like “Wrap your entire response with double quotation marks,” and casing instructions (e.g., “Answer using lowercase letters only”). A complete list of the instructions used is provided in Appendix A.[3] For length instructions, we generate prompts by concatenating base queries from IFEval to instructions derived from templates such as “Answer using at most $\{n\}$ sentences.” For word-specific instructions, we use a subset of IFEval containing instructions about the inclusion or exclusion of keywords. The subset has 203 examples, and each example contains a prompt with a single keyword-related instruction. For steering vector computation and layer selection, we construct a separate set of synthetically generated prompts by combining base queries from IFEval with corresponding instruction descriptions to avoid test information leakage. Additional details about the data used are provided in Apps. B and C.

Metrics. To quantify the models’ adherence to format instructions, we compute the “loose” instruction-following accuracy using the official IFEval evaluation script. For length instructions, we count the number of words or sentences in the model’s output. For word-specific constraints, we verify the presence or absence of the specified keywords in the model’s response, again using the IFEval evaluation script. We assess the statistical significance of differences in average instruction-following accuracy with and without steering using McNemar’s test[37]. Results where steering leads to a significant improvement (p-value < 0.01) are marked with an asterisk (*).

In addition to assessing the model’s instruction-following capabilities, it is important to verify that the model still effectively addresses the base query, even when generating under constraints. To measure the overall quality of the response, we set up a GPT-4o-based evaluation. For each base query x, GPT-4o generates a set of five yes/no questions aimed at assessing the quality of the response to x. For instance, given the query “Write an essay about the history of Martin Van Buren’s presidency,” GPT-4o generates questions such as “Does the essay provide context on the political, social, and economic climate during Van Buren’s presidency?” and “Is the essay written in clear, grammatically correct English, and does it follow a logical structure?”. Next, given the original query x, a model’s response to x, and the generated questions we prompt GPT-4o to answer each question, provide a rationale for each yes/no response, or reply with “N/A” if the question is not applicable. We compute the proportion of questions answered positively for a given query and average this score across all queries to obtain a *response quality score*. We repeat this experiment 3 times and report the mean and standard error. Additional details are provided in Appendix E.[4]

Evaluation. We conduct experiments using the instruction-tuned versions of Phi-3 Mini[31], Gemma 2 2B, 9B[32], and Mistral 7B v0.1[33]. In addition, we investigate the transferability of steering vectors from the instruction-tuned to the base versions of Gemma 2 2B and 9B. All models are evaluated in a zero-shot setting, with outputs decoded greedily. We assess our method's performance by steering the model under two input settings: (1) with text instructions provided in the input, and (2) without any text instructions. These settings allow us to explore two questions: Are the computed steering vectors informative enough to guide the model's behavior even without explicit instructions? And can steering further improve performance when instructions are provided in the input?

3. Format Instructions

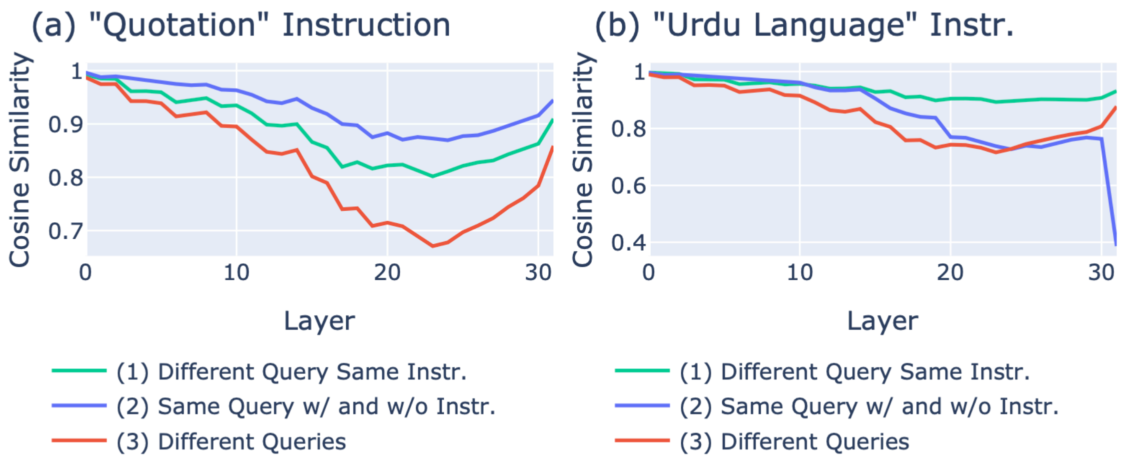


Figure 2. Residual stream similarity across layers. Phi-3's residual stream activations show higher cosine similarity between examples with the same instruction compared to those without, indicating effective capture of instruction-relevant features.

Instruction	Layer	Top Tokens
JSON Format	18	_[{, _json, _‘{, _JSON
Capitalize	28	_PRO, _TH, _ FOR, _AND
Highlight Text	26	*‘, _*, _*, (*, *\\
Lowercase	18	_lower, _lowest, _russ
Bullet List	28	*), _·, *, _*), ·, */
Quotation	26	_“", _’", _"", _",, _"
Urdu Language	24	_Islam, _Pakistan, _Pak
Hindi Language	18	_Indian, raj, _India
German Lang.	16	_die, _im, _dies, _gener

Table 1. Steering Vector Projections. The projection onto the vocabulary space of the contrastively computed vectors promotes tokens that are semantically related to the respective instruction.

Representation Analysis. To assess whether the model effectively captures instruction-related information in its internal representations, we analyze Phi-3's residual stream activations at the last token of the input by calculating cosine similarity across three sets of inputs: (1) pairs of inputs that share the same base query, one with and one without the instruction; (2) inputs with different base queries but the same instruction; and (3) inputs with different base queries and no instruction. High similarity in set (2) would suggest that the model captures a shared feature (the instruction), while we expect set (1) to show relatively high similarity due to shared base queries, and set (3) to show lower similarity due to the absence of any shared features. Figure 2 shows how these measures evolve across layers for two instructions. For the “quotation” instruction, where the model is asked to wrap the output in quotation marks, there is a clear difference between sets (2) and (3), indicating that the instruction is partially captured (green vs. red lines in Figure 2a). For the “Urdu language” instruction, set (2) shows higher similarity than set (1), suggesting a strong representation of the instruction (green vs. blue lines in Figure 2b). These results indicate that the model can effectively encode instruction-relevant features at the last input token, which supports steering model behavior based on specific constraints.

Steering Vector Computation. We compute steering vectors for each of the 12 format-related instructions in the IFEval subset and for the 19 language-based instructions specified in the dataset. During the selection of the optimal steering later, we compare the validation score with and without steering to ensure that the steering intervention leads to an improvement. If no layer shows an improvement in the validation score, no steering is applied at test time. Details about the layers selected for steering are provided in Appendix D. To ensure that the steering vectors effectively capture the information related to the instructions, we inspect them by projecting the vectors onto the model's vocabulary space[38][39]. We project the vectors using the model's unembedding matrix and examine the vocabulary tokens with the highest logit values. Table 1 presents the top tokens associated with several of the instructions we consider. We observe that the tokens promoted by the steering vectors are semantically related to the intended instruction, providing an initial validation that the vectors are capturing the desired features.

Steering Results. We first evaluate steering on inputs without explicit text instructions (Figure 3a). In this setting, the instruction-following accuracy without steering hovers around 10% as the input has no information about the instruction. This non-zero accuracy reflects cases where the models incidentally satisfy the instruction, such as when a model rephrases a sentence without using commas, thus accidentally meeting the "no comma" constraint. When steering is applied, we observe a consistent increase in instruction adherence across all models, with accuracy improving to approximately 30%. This shows that the steering vectors encode meaningful information of the instructions and can be effectively used to steer the models toward the intended behavior. Next, we evaluate on inputs with instructions provided in the text (Figure 3b). As expected, the instruction-following accuracy without steering is higher in this setting, ranging between 60% and 90%. Nevertheless, steering still results in a significant performance boost for two out of four models, demonstrating that steering can enhance instruction adherence even when the instructions are explicitly given.

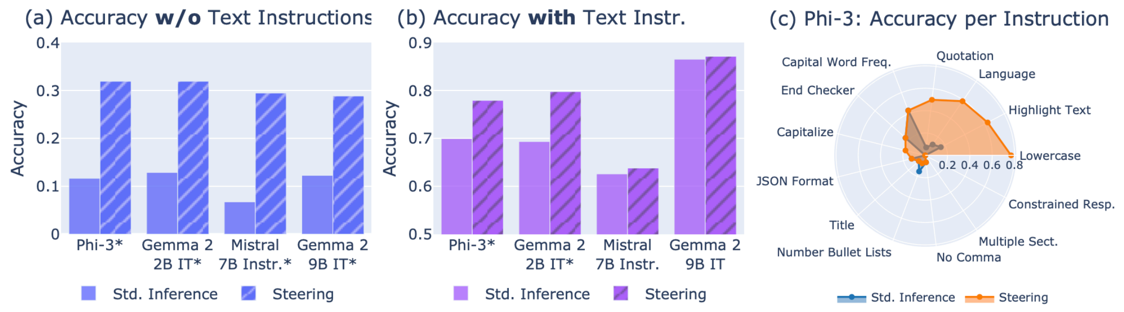


Figure 3. Format Instructions. (a) Instruction-following accuracy without explicit text instructions shows significant improvement with steering across all models. (b) Steering enhances accuracy even when text instructions are provided. (c) Per-instruction accuracy for Phi-3 without text instructions.

To further analyze which instructions benefit most from steering, we break down Phi-3's performance by instruction on input without explicit text instructions (Figure 3c). Notably, we observe significant improvements for instructions such as "Lowercase," which requires the output to be entirely in lowercase characters, and "Highlight Text," which asks the model to emphasize parts of the response using markdown syntax (e.g., "*{text}*"). However, certain instructions display variability in steering effectiveness. For instance, the "End Checker" instruction requires the model to finish the response with a specific sentence. This ending is often input-dependent; for example, the model might be asked to generate an email ending with "Hope you agree with me," or to answer a factual question and conclude with "Is there anything else I can help you with?". Since the steering vector is computed by averaging over all examples containing this instruction, it likely fails to capture the specific sentence required in each individual case, reducing its ability to steer the model toward precisely following this type of instruction.

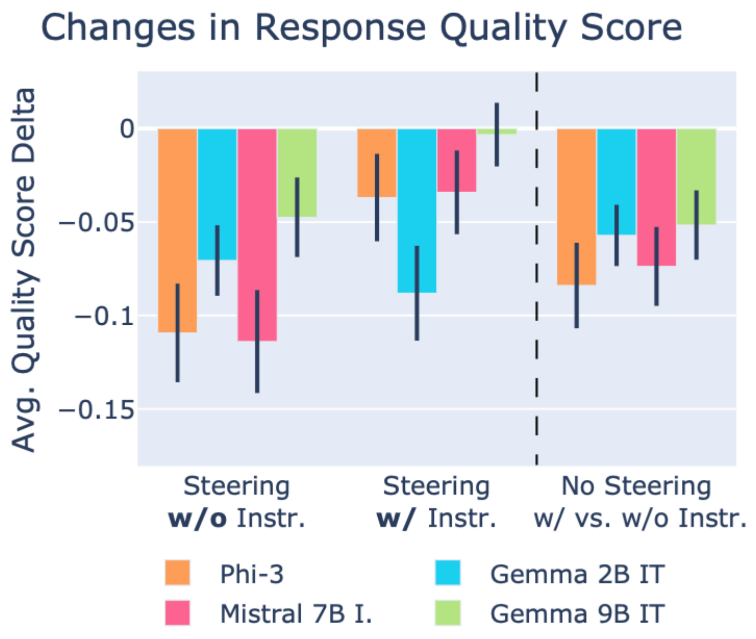


Figure 4. Quality Score. Average response quality score changes for four models, comparing the effects of steering with and without input instructions to the effect of adding instructions without steering. Steering decreases quality similarly to adding instructions.

Response Quality Evaluation. Figure 4 compares the average differences in response quality scores for the four models. The two leftmost groups of bars show the score changes due to steering, with and without input instructions (calculated only for cases where steering is applied). Satisfying specific constraints during generation may lead to less comprehensive responses and hence following instructions is expected to impact the response quality. To show this effect, we also present the differences in quality scores observed when explicit text instructions are added to the base queries without steering. We observe small decreases in quality score due to steering in the setting including instructions (with the exception of Gemma 2B) while the decreases in the setting without instructions are slightly larger but comparable to the effect caused by simply adding the instructions as text. However, we also observe instances where steering compromised the quality of generation–examples where the model generated nonsensical tokens or repeated itself. These failures likely result from suboptimal steering or partially insufficient objectives during the search for layers to intervene, as steering for specific properties can increase perplexity or reduce fluency[22][40]. We provide some examples from the few cases we observed in Appendix E.

4. Length Instructions

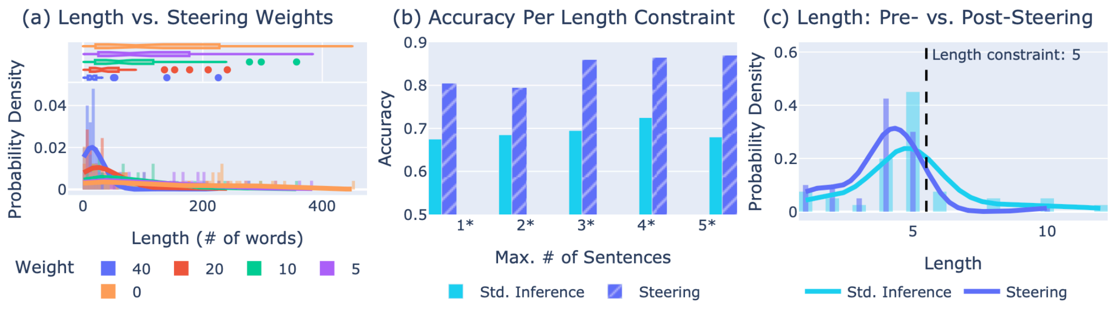


Figure 5. Length Instructions. (a) Modulating the steering weight c effectively adjusts response length, with larger values leading to more concise outputs. (b) Steering enhances adherence to the maximum length constraints when they are explicitly specified in the input text. (c) Shift in the response length distribution upon steering for outputs constrained to a maximum of 5 sentences.

Weight	Resp. to "Write a movie plot that involves dream, fist fighting, and superpower."	# of Sents.	# of Words
0	In the bustling city of New Haven, a young, introverted artist named Alex dreams of a world where his art comes to life. One night, Alex has a vivid dream where he discovers he possesses a unique superpower: the ability to bring [...]	15	324
5	In a world where dreams can be harnessed as a source of power, a young, introverted artist named Leo discovers he has the unique ability to enter and manipulate the dreams of others [...]	9	228
10	In a world where dreams can manifest into reality, a young woman named Elara discovers she possesses the rare ability to fight with her fists in her dreams [...]	10	203
20	In a world where dreams can manifest reality, a young girl named Lila discovers she can unleash her latent superpower through her dreams [...]	3	81
40	A young girl dreams of a superpower, fights a rival in a dream, and unleashes her power to save her village from destruction.	1	23

Table 2. Example of Steering for Length Control. Responses to the same prompt with increasing steering weights, resulting in progressively shorter outputs in terms of sentence and word count.

Steering Vector Computation. Length constraints can be specified in various ways, such as by the number of sentences, words, or lines. However, unlike format instructions, it seems impractical to compute a separate steering vector for each possible length constraint. Instead, we focus on capturing more general notions of *conciseness* and *verbosity*. To achieve this, we compute vector representations for instructions that prompt the model to be brief (e.g., “The answer should be brief”) or to provide more detailed responses (e.g., “Provide a long answer”). We synthetically generate these prompts by appending such instructions to base queries from IFEval. After computing these steering vectors, we evaluate the model on a separate set of IFEval base queries, this time appending length instructions specified in terms of the number of sentences (e.g., “Answer using at most 3 sentences”).

Steering Results. Unlike format instructions, steering the model for length constraints allows for continuous modulation, enabling interpolation between varying degrees of conciseness or verbosity. To explore this, we manually adjust the steering weight c and examine how this affects the response length. On a set of 50 base prompts without any explicit length instructions, we generate responses from Phi-3 using different values of c, measuring the distribution of output lengths. As shown in Figure 5a, increasing the value of c effectively shortens the model’s responses: larger steering weights produce increasingly concise outputs. We provide an example of different outputs generated on the same inputs with different steering weights in Table 2. Next, we assess whether steering for length is effective even when the model is provided with explicit length instructions. Using a set of 200 base prompts, we introduce instructions that request the model to limit its response to a maximum of $n \in \{1, \ldots, 5\}$ sentences. First, we evaluate how often the model adheres to this constraint without steering (light blue bars in Figure 5b). Then, on the same inputs, we apply the steering vector for conciseness with a weight of $c = 20$ (dark blue bars in Figure 5b). Across all five values of n, we observe a significant and consistent improvement in how often the model’s responses comply with the length constraint. Figure 5c further illustrates how steering shifts the response length distribution toward shorter outputs while still allowing for variability and avoiding overly short responses. Response quality evaluation and results on steering for verbosity are provided in Apps. F and G, respectively.

5. Word-specific Instructions

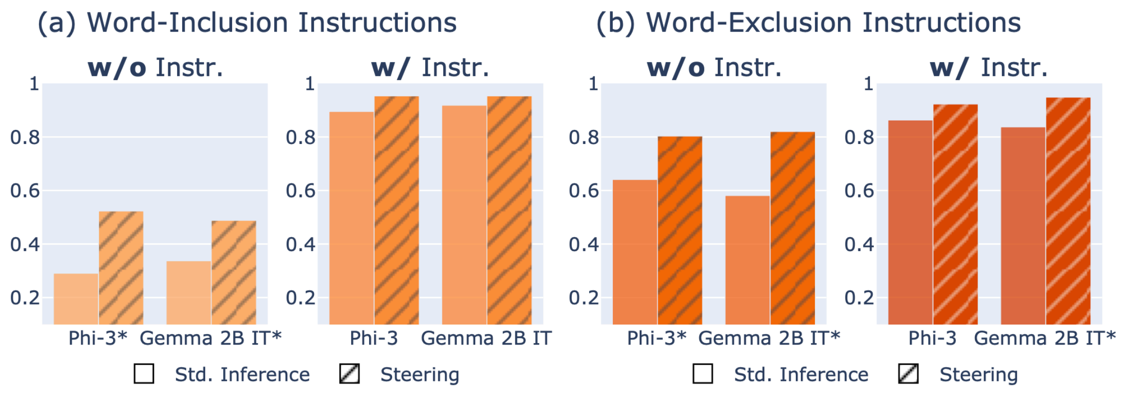


Figure 6. Word-specific Instructions. (a) Steering improves keyword inclusion accuracy for Phi-3 and Gemma 2 2B IT, with and without explicit instructions. (b) Negative steering reduces the occurrence of undesired keywords, in both settings.

Steer	**Resp. to "What are the steps to be followed for the documentation of a GM in SAP? Do not use the word *step*."**
No	Documenting a General Master in SAP involves [provides guide] By following these **steps**, you can create comprehensive [...]
Yes	## Documenting a General Master (GM) in SAP: [provides guide] By following this **comprehensive guide**, you can create a well-structured [...]

Table 3. Example of Word Exclusion. The instruction alone fails to exclude "step," but steering successfully removes it.

5. Word-Specific Instructions

Keyword Inclusion. For this set of instructions, we compute word-specific steering vectors. To generate these vectors, we append different phrasings of a request to include a specific word w (e.g., "please include the word $\{w\}$ in your response") to a set of base queries. These prompts are used to compute a vector representation for the “include word $\{w\}$” instruction. While this requires a separate steering vector for each keyword at inference time, the vectors can be generated on the fly using arbitrary base queries unrelated to the keyword itself. Furthermore, these vectors can be computed with a small number of examples; in our experiments, we use only 20 examples. The models are then evaluated on the subset of IFEval that contains keyword inclusion/exclusion constraints.[5]

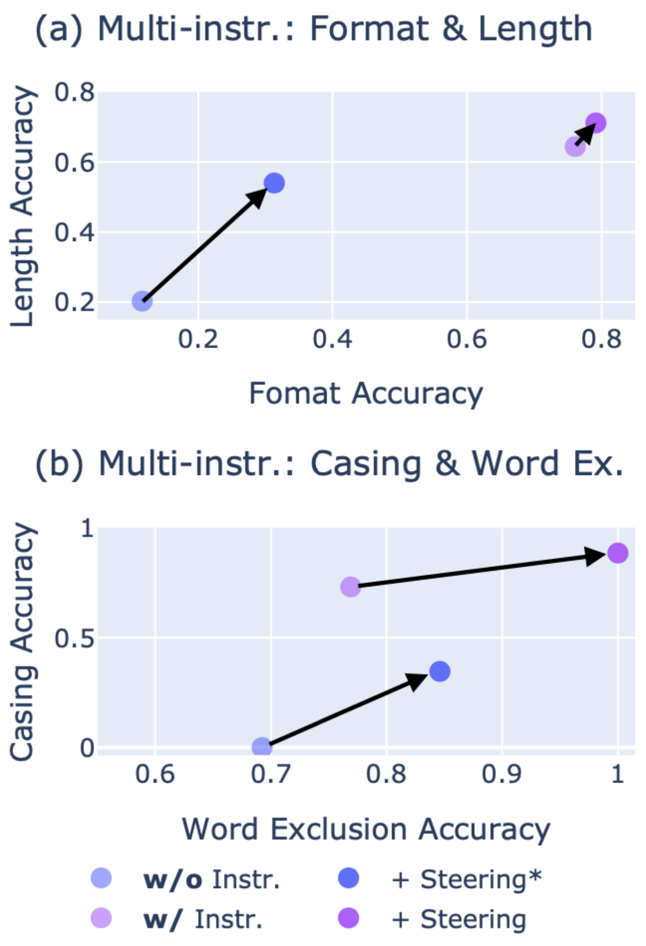


Figure 7. Multi-instruction Steering. Steering for two instructions improves adherence to both.

The results, presented in Figure 6a, demonstrate the effectiveness of steering for word inclusion. By applying the word-specific steering vectors, we observe a notable increase in the frequency with which the model successfully includes the requested keywords in its responses.

Keyword Exclusion. For keyword exclusion, we initially used the same procedure, appending instructions like "ensure the word w does not appear in the answer" to base queries and computing the corresponding steering vectors. However, upon inspecting the steering vectors, we found that projecting these vectors onto the vocabulary space resulted in high logit values for the tokens corresponding to the words meant to be excluded. In other words, adding these vectors to the model's residual stream actually increased the probability of generating the very keywords we intended to exclude. To address this issue, instead of computing exclusion vectors, we compute the vectors for inclusion and then subtract them from the model's residual stream. This technique effectively steers the model away from using the words captured by the vector. Figure 6b shows the results of steering for keyword exclusion using this approach. We observe that subtracting the inclusion vectors significantly reduces the frequency of the undesired keywords in the model's responses, both in cases where no textual instructions are present and where explicit exclusion instructions are provided in the input. Table 3 reports an example where the instruction to exclude a specific word is ineffective on its own, but applying steering successfully enforces the constraint.

6. Multi-instruction Steering

Next, we present results on using our steering approach to handle multiple instructions simultaneously. With the same method as in the previous experiments, we steer the Phi-3 model for two instructions at once, applying the steering vectors at the layers previously identified as optimal for each individual instruction. We opt for injecting multiple steering vectors simultaneously at different locations instead of combining multiple steering vectors into a single one, as previous work has shown the latter approach to be largely unsuccessful[28].

Figure 7 shows results from experiments with steering two instructions at the same time: format (all 13 instructions) and length (Figure 7a), as well as lowercase and keyword exclusion (Figure 7b). In both cases, steering leads to improvements on both axes. These findings suggest that steering for multiple instructions at once is feasible and can lead to performance gains across different constraints. However, we anticipate that issues may arise in certain cases, particularly when dealing with conflicting instructions, where refinement may be necessary to balance competing constraints.

7. Cross-model Steering

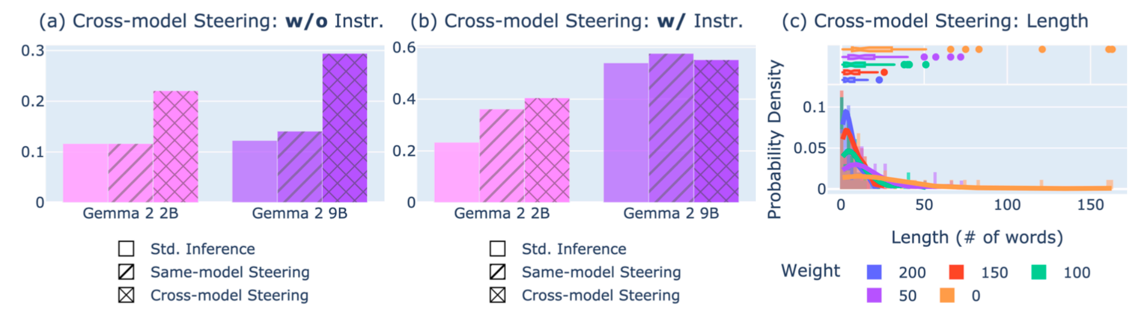


Figure 8. Cross-model Steering. (a) Without text instructions, cross-model steering improves constraint satisfaction more than same-model steering. (b) With text instructions, cross-model steering increases accuracy for Gemma 2 2B, but not for Gemma 2 9B. (c) Cross-model steering for length instructions shortens output as steering weight increases.

Instruction-tuned models show improved instruction-following abilities, suggesting that they may also be better at constructing meaningful representations of instructions. In this context, we ask: can we leverage the representations computed on instruction-tuned models to steer a base model more effectively? This idea is motivated by prior work showing that the weights of language models do not change significantly after instruction tuning[41][42]. Additionally, linear representations derived through mean activation difference and sparse autoencoders[43] have been shown to be transferable between base and chat models[27][44]. To study this in our setting, we conduct experiments with the base and instruction-tuned versions of Gemma 2 2B and 9B. We focus on format instructions, using the same data and procedures outlined in §3. The key difference is that we apply steering vectors computed on the instruction-tuned models to steer the base models, comparing this approach to same-model steering (vectors computed on the base model).

When inputs consist of base queries only (Figure 8a), steering the base Gemma models with same-model vectors results in only modest improvements. However, when using vectors from instruction-tuned counterparts, the base models satisfy the constraints much more frequently, indicating the transferability of the instruction representations. When explicit input instructions are introduced (Figure 8b), Gemma 2B sees further accuracy gains, with cross-model steering outperforming same-model steering (40.5% vs. 36.2%). For Gemma 9B, however, steering yields no significant improvement, suggesting that the performance bottleneck might not be the quality of the instruction representation. We also explore cross-model steering for length instructions. As described in §4, we compute a steering vector for conciseness from the instruction-tuned version of Gemma 9B and apply it to the base version. Figure 8c shows that increasing the steering weight shortens the outputs, demonstrating that representations learned in instruction-tuned models can still meaningfully modulate the output in the base model's space.

This is the first demonstration that cross-model steering–using vectors from instruction-tuned models–can outperform same-model steering in base models. This finding suggests that specialized representations from fine-tuned models can be leveraged to steer base models more effectively, opening new possibilities for composable transfer learning in instruction-based tasks where task vectors may originate from different models that are instruction-tuned in specialized domains.

8. Related Work

Instruction Following. Training models to follow instructions is crucial for improving LLM performance and ensuring safe deployment, with various methods developed to enhance instruction adherence[1][4][3][2][5], and datasets designed to train and evaluate instruction-following behavior[45][46][47][48][49][50][51]. Natural language instructions have demonstrated significant promise in providing fine-grained control over model outputs[7]. However, capable models still struggle with tasks that require outputs to satisfy fine-grained, hard constraints[29] and tend to drift from adhering to a constraint as the generation lengthens[34]. Motivated by these challenges, our work investigates how to improve instruction-following behavior by directly intervening on the model's activations at inference time.

Language Model Representations. Our approach is inspired by prior research that shows it is possible to obtain vector representations encoding information about tasks on which a language model has been trained[35][52] or tasks learned in context[10][11]. These studies are part of a broader body of work that examines the linear representation of features such as truthfulness[19][14][17], sentiment[21], harmlessness[13][15], sycophancy[53][27][54], factual knowledge[55], and refusal[25]. In addition, recent works have employed sparse autoencoders to identify feature directions in an unsupervised manner[56][43][16]. A shared hypothesis across these works is that LLMs represent features or concepts as linear directions in activation space[57][58][59][60][61][62]. While recent studies suggest that not all features may be linearly encoded[63][64], the linearity assumption has been effective for both concept erasure[65][66][67][68] and model steering.

Model Steering via Activation Editing. It is well-established that the generation of a language model's output can be manipulated by directly editing activation values during inference[69][18]. Recent studies have shown that this approach can effectively steer models to be more honest and truthful[19][20], sycophantic[27][28], morally aligned with human values[13][70], or to display different sentiments, output styles, and languages[22][12][21][23][24]. Steering methods have also been used to control the model's uncertainty[71], adopt different personas[72], provide alternative factual answers[73], and respond to harmful requests[25][26]. Similarly to some of these works[74][22][27][25][28], we compute steering vectors based on input pairs that differ by a specific feature—in our case, the presence or absence of instructions. However, while previous studies have focused on high-level concepts such as sentiment, style, and safety, we focus on lower-level, hard constraints defined through natural language instructions, allowing for finer-grained control of the model's output.

9. Conclusion

We demonstrated the effectiveness of activation steering for improving language models' adherence to instructions. By computing steering vectors based on activation differences between inputs with and without instructions, we guide models to follow constraints related to format, length, and word inclusion/exclusion. These vectors capture meaningful instruction representations, enabling models to meet constraints even without explicit instructions, while also enhancing performance when instructions are present. Additionally, we show that models can handle multiple constraints simultaneously, offering a flexible framework for controlled language generation. Finally, we explore cross-model steering, revealing that vectors computed on instruction-tuned models can improve the behavior of base models, often surpassing same-model steering.

Reproducibility Statement

In §2.2, we outline our methodology in detail. Information about the datasets we used, as well as the procedures for obtaining and augmenting them, is provided in §2.3, along with Apps. B and C. §2.3 also contains the details of the evaluation metrics we employed and the process for generating model outputs. E provides further details on the implementation of our quality score metric. Additional implementation and experimental details are included in F.

Acknowledgments

We would like to express our gratitude to Andy Arditi, Ahmed Awadallah, Natasha Butt, Varun Chandrasekaran, Arthur Conmy, Nico Daheim, Saibo Geng, Suriya Gunasekar, Neel Joshi, Siddharth Joshi, Mazda Moayeri, Neel Nanda, Harsha Nori, Kyle O'Brien, Vibhav Vineet, and Vilém Zouhar for their valuable feedback and insightful discussions throughout the development of this project. We also extend our thanks to the AI Frontiers group at Microsoft Research, with special recognition to Ece Kamar for practical support. Alessandro acknowledges the support of armasuisse Science and Technology through a CYD Doctoral Fellowship.

A. List of Instructions

In Table 4, we provide a list of the instructions and examples used in the subset of the IFEval dataset for our experiments about the output format (3). For the "language" instructions, the dataset includes the following languages: German, Urdu, Portuguese, Korean, Marathi, Punjabi, Kannada, Farsi, Swahili, Russian, Hindi, Arabic, Nepali, Telugu, and Gujarati.

Instruction	Example
No Comma	You are not allowed to use any commas in your response.
Lowercase	Please ensure that your response is in English, and in all lowercase letters
Language	Please respond using only the Kannada language, no other language is allowed.
JSON Format	Wrap the entire output in JSON format. You can use markdown ticks such as "'.
Quotation	Wrap your entire response with double quotation marks.
Multiple Sections	Write a 4 section [base query]. Each section should be explicitly noted as Section X.
Number of Bullet Points	Your answer must contain exactly 6 bullet point in Markdown using the following format: \n* Bullet point one. \n* Bullet point two. \n ... \n* Bullet point six
Highlighted Sections	At least 15 sections should be highlighted with markdown such as *highlighted section*.
Title	Your answer must have a title contained in double angular brackets, such as <<title>>.
Capitalize	Make sure your entire response is in English, and in all capital letters.
Capital Word Frequency	Use words in all capital letters at least 3 times to highlight key points.
End Checker	The very end of your response should read "You cannot fail with the steps listed above." No other words should follow this phrase.
Constrained Response	Answer with exactly one of the following phrases: "My answer is yes.", "My answer is no.", "My answer is maybe."

Table 4. Format-Related Instructions and Examples. Format-related instructions used in our experiments, along with examples illustrating the specific constraints the model is expected to follow.

B. Extracting IFEval Base Queries

To obtain base queries without instructions, we use GPT-4o[75]6 to strip the instructions from the base queries, as the IFEval dataset does not explicitly annotate the instructions within the prompts. We use a one-shot prompt a fixed in-context example. The prompt used is reported in Table 5. These base queries are then used to generate additional base query-instruction combinations, augmenting the original dataset. Further details on the data used for steering vector computation and evaluation are provided in Appendix C.

```
Given a question that imposes a set of constraints on the answer, make the question simpler by removing all the constraints. You will be given the original question and a set of constraints to remove from it, and should output the simplified question with the constraints removed. Nothing else should be removed other than the listed constraints. For example:

<original_question>
I am planning a trip to Japan, and I would like thee to write an itinerary for my journey in a Shakespearean style. You are not allowed to use any commas in your response. <\original_question>

<constraints>[punctuation:no_comma] <\constraints>

<output> I am planning a trip to Japan, and I would like thee to write an itinerary for my journey in a Shakespearean style. <\output>

<original_question> Write a 300+ word summary of the wikipedia page https://en.wikipedia.org/wiki/Raymond_III_Count_of_Tripoli. Do not use any commas and highlight at least 3 sections that has titles in markdown format, for example *highlighted section part 1*, *highlighted section part 2*, *highlighted section part 3*. <\original_question>

<constraints> [punctuation:no_comma, detectable_format:number_highlighted_sections, length_constraints:number_words] <\constraints>

<output>
```

Table 5. Prompt for Instruction Removal.

C. Data and Vector Computation Details

Given a simple question, we want to make the question a bit harder by adding constraints to the way it can be answered. You will be given the original question and a constraint to add to it, and should output the harder question with the constraint integrated into it. Nothing else should be added or removed from the question. Only the constraint should be added to the question. For example:

<original_question> The opposite of youth is not age, but ...? <\original_question>

<constraints>[detectable_format:number_highlighted_sections] <\constraints>

<output> The opposite of youth is not age, but ...? Highlight at least 2 sections in your answer with markdown, i.e. *highlighted section*. <\output>

<original_question> Write a 300+ word summary of the wikipedia page https://en.wikipedia.org/wiki/Raymond_III_Count_of_Tripoli. Do not use any commas and highlight at least 3 sections that has titles in markdown format, for example *highlighted section part 1*, *highlighted section part 2*, *highlighted section part 3*. <\original_question>

<constraint>detectable_format:number_highlighted_sections {'num_highlights': 2} <\constraint>

<output>

Table 6. Prompt for Instruction Addition.

Format Instructions. The steering vectors and layer selection for format instructions are computed using a separate set of synthetically generated prompts. To generate a synthetic dataset containing all base prompts with all instructions from the IFEval dataset, we start with the base prompts as described in Appendix B. We then create an augmented version of the IFEval dataset by combining each base prompt with every available instruction. This is done by prompting GPT-4o using in-context examples, as shown in Table 6. When adding a constraint to a base prompt, we include a randomly selected in-context example of the same type of constraint from the single-constraint dataset. These single-constraint examples are generated by prompting GPT-4o to remove all but one instruction from each prompt. For this, we use a fixed, manually curated in-context example. This procedure is carried for each base query and each format instruction. We then filter out any base query and instruction combinations that appear in the evaluation set to ensure that the base queries used for steering vector computation do not overlap with those in the evaluation prompts. This process yields approximately 450 examples for each instruction, which are used for both the steering vector computation and for validation. We provide examples of data generated using this procedure in Table 7.

For format instruction evaluation, we utilize the format-related subset of IFEval queries (163 in total), which includes the 13 distinct instructions detailed in Appendix A.

Length Instructions. For length instructions, the steering vectors are computed using a fixed set of 50 IFEval base queries, obtained as described in Appendix B. To this set, we append various manually annotated phrasings of instructions that request the model to generate concise responses, such as "Be concise" or "The answer should be brief."

The evaluation data used in 4 consists of a separate set of 200 base queries. Although IFEval contains some length-related instructions, it offers limited examples with significant variation in length constraints and expressed in different ways (e.g., "I don't want anything longer than 30 words," or "Answer using five paragraphs"). We opt for synthetically generating data to enable consistent evaluation across a set of constraints varying over a narrower range (1 to 5 sentences).

Word-specific Instructions. For word-specific instructions, we compute vector representations for inclusion constraints, where the model is instructed to include a specific keyword in the output (e.g., "The output should contain the word $\{w\}$" or "Ensure that the word $\{w\}$ is included in the response"). For each keyword, we use a set of 20 base queries randomly sampled from those generated as described in Appendix B, ensuring none of the queries used for vector computation are reused in evaluation. The topics and content of these base queries are often semantically unrelated to the keyword being included.

For evaluation, we use IFEval examples containing keyword inclusion and exclusion instructions. Many examples contain multiple keyword constraints in a single query, so we separate these into individual prompts, each requesting the inclusion or exclusion of a single keyword. This process yields 86 evaluation prompts for keyword inclusion and 117 for keyword exclusion.

Condition	Example
Original IFEval prompt (3 constraints)	Write a 300+ word summary of the wikipedia page https://en.wikipedia.org/wiki/Raymond_III,_Count_of_Tripoli. Do not use any commas and highlight at least 3 sections that have titles in markdown format, for example *highlighted section part 1*, *highlighted section part 2*, *highlighted section part 3*.
Synthetic single-constraint punctuation:no_comma	Write a summary of the wikipedia page https://en.wikipedia.org/wiki/Raymond_III,_Count_of_Tripoli. Do not use any commas.
Synthetic single-constraint detectable_format: number_highlighted_sections	Write a summary of the wikipedia page https://en.wikipedia.org/wiki/Raymond_III,_Count_of_Tripoli. Highlight at least 3 sections that have titles in markdown format, for example *highlighted section part 1*, *highlighted section part 2*, *highlighted section part 3*.
Synthetic single-constraint length_constraint:number_words	Write a 300+ word summary of the wikipedia page https://en.wikipedia.org/wiki/Raymond_III,_Count_of_Tripoli.
Synthetic no-constraint base query	Write a summary of the wikipedia page https://en.wikipedia.org/wiki/Raymond_III,_Count_of_Tripoli.
Synthetic augmented with detectable_format: number_bullet_lists	Write a 300+ word summary of the wikipedia page https://en.wikipedia.org/wiki/Raymond_III,_Count_of_Tripoli. Your answer should contain exactly 3 bullet points in markdown format. Use * to indicate bullets, like: * xyz * abc * opq

Table 7. Examples of Synthetically-generated Data.

Model Name	Phi-3		Gemma 2 2B IT		Mistral 7B Instr.		Gemma 2 9B IT	
Setting	w/o Instr.	w/ Instr.	w/o Instr.	w/ Instr.	w/o Instr.	w/ Instr.	w/o Instr.	w/ Instr.
Capital Word Freq.	18	20	11	-	12	24	16	8
English Capital	28	18	11	11	18	30	28	8
English Lowercase	18	30	17	7	28	6	22	-
Constrained Resp.	-	-	-	-	-	-	-	-
Json Format	16	6	-	13	16	12	-	10
Multiple Sections	-	6	-	9	-	24	-	28
Number Bullet Lists	20	16	5	-	24	12	16	12
Highlight Text	26	18	5	-	26	26	12	10
Title	-	-	-	-	-	-	-	-
No Comma	10	12	11	11	14	26	22	-
End Checker	-	18	-	5	-	24	-	-
Quotation	26	24	11	23	28	-	34	12

Table 8. Steering Layers for Format Instructions. Layer indices used for steering across different models and evaluation settings (with and without explicit instructions) for each instruction in the format subset. A dash ("-") indicates that no steering is performed.

Model Name	Phi-3		Gemma 2 2B IT		Mistral 7B Instr.		Gemma 2 9B IT	
Setting	**w/o Instr.**	**w/ Instr.**	**w/o Instr.**	**w/ Instr.**	**w/o Instr.**	**w/ Instr.**	**w/o Instr.**	**w/ Instr.**
Language Ar	18	16	-	15	22	-	20	20
Language Bg	22	-	19	-	18	6	22	-
Language Bn	18	-	19	-	18	-	20	-
Language De	16	-	15	-	16	-	20	-
Language Fa	20	-	-	-	18	20	22	-
Language Fi	18	-	15	-	16	8	22	-
Language Gu	-	-	-	-	-	-	-	-
Language Hi	18	-	15	7	26	-	22	12
Language It	16	-	15	-	16	-	20	-
Language Mr	24	-	17	15	18	18	24	-
Language Pa	-	-	-	-	22	-	-	8
Language Pt	16	-	15	7	16	-	20	-
Language Ru	16	-	15	-	16	-	22	-
Language Sw	20	-	-	-	-	14	22	20
Language Ta	18	-	-	-	18	-	22	-
Language Te	24	-	19	-	16	-	20	-
Language Th	18	-	15	-	18	-	20	-
Language Ur	24	-	-	11	-	-	28	-
Language Vi	20	-	15	-	18	-	20	-

Table 9. Steering Layers for Language. Layer indices used for steering across different models and evaluation settings (with and without explicit instructions) for each language-related instruction. A dash ("-") indicates that no steering is performed. In most cases with explicit text instructions, steering was unnecessary as the models typically followed the instruction and generated responses in the requested language.

Appendix D. Steering Layer Selection

In Table 8, we provide the layer indices used in our steering procedure for each instruction in the format subset (excluding language) across all models and evaluation settings (with and without explicit text instructions). Layer selection is based on validation instruction-following accuracy, evaluated on a set of 96 held-out examples (8 per instruction), sampled from the synthetically generated prompts described in Appendix C. We compare the validation scores for each layer against a baseline without steering to ensure that the steering intervention results in an improvement. If no layer shows an improvement, steering is not applied at test time. In the event of a tie between layers with the highest score, the earliest layer is chosen.

We observe that for two instructions, "Constrained Response" and "Title," no layer improves performance with steering. The "Constrained Response" instruction asks the model to respond using only "yes," "no," or "maybe," while the "Title" instruction requires the answer to include a title wrapped in angular brackets (e.g., "$<>$"). In these cases, the steering vectors sometimes push the model too aggressively toward outputting instruction-related tokens, resulting in poor outputs (e.g., repeating characters like "$<<<<$").

For language instructions, the selected layers are listed in Table 9. For length instructions, we intervene at layer 12 in Phi-3 and layer 16 in Gemma 2 9B. For word-specific instructions, we apply steering at layer 24 for both Phi-3 and Gemma 2 2B. These layers were chosen based on validation using a small set of manually created prompts.

Appendix E. Response Quality Evaluation

E.1. Evaluation Procedure

To evaluate the quality of the model-generated responses, we first use the base queries outlined in Appendix B and prompt GPT-4o to produce 5 questions about the query to evaluate any model-generated answer to the query (prompt reported in Table 10). For each model-generated response in our evaluation, we then prompt GPT-4o to provide a binary "yes/no" answer to each question and "Not Applicable" if the question is not applicable to the query, along with a reason (prompt in Table 11). An aggregate quality score is computed as the average % of questions that is scored as "yes". We run the experiment three times for each model and report the quality score and standard error to account for any non-determinism from the GPT-4o-based evaluator.

```
The following is a prompt that is used to evaluate the generations from a large language model. We do not know how to evaluate the quality of model answers for this prompt. Can you come up with 5 or less questions that can break down the quality to simpler evaluation tasks that we can then ask about the model answer? Each question should have a simple yes, no answer.

Prompt: {{ prompt without instruction }}
List all sub questions in the following format:
Output:
1: Question: <question>
2: Question: <question>
...
N: Question: <question>
```

Table 10. Prompt for Quality Evaluation Question Generation.

```
We need to evaluate the quality of generations from a large language model. You will be given an input prompt, the response from a language model and a set of questions assessing the quality of the response. You need to review the response against the input prompt and provide an answer to each question as either ’Yes’, ’No’ or ’Not Applicable’ if the question does not apply to the case along with a reason for your answer.

Prompt: {{ prompt without instruction }}
Response: {{response}}
Questions: {{evaluation questions}}
List your answers in the following format:
Output:
1. Question: <question>. Reason: <reason>: Answer: <answer>
2. Question: <question>. Reason: <reason>: Answer: <answer>
...
N. Question: <question>. Reason: <reason>: Answer: <answer>
```

Table 11. Prompt for Quality Score Evaluation.

Query	Output w/o Steering	Output w/ Steering
The Legend of the Sword and the Fairy is a movie in which Wan Wan is a villain. Write a story about Wan Wan's character in list format.	1. Introduction to Wan Wan:\n - Wan Wan is a cunning and ambitious antagonist in the movie "The Legend of the Sword and the Fairy."\n - She is a skilled sorceress with a deep-seated desire for power and control.\n 2. Early Life:\n - Wan Wan was born into a noble family, but her parents were killed in a tragic accident when she was young.\n - She was raised by her grandmother, who taught her the [...]	[...] 14. **WAN WAN'S REDEMPTION IS A MIRACLE,** AS HE AND THE SWORD AND THE FAIRY'S GIRL FIND HARMONY AND LOVE.\n 15. **WAN WAN'S REDEMPTION IS A TEACHING,** AS HE LEARNS THE VALUE OF LOVE, HONOR, AND SELFLESSNESS.\n 16. **WAN WAN'S REDEMPTION IS A MIRACLE,** AS HE AND THE SWORD AND THE FAIRY'S GIRL FIND HARMONY AND LOVE.\n 17. **WAN WAN'S REDEMPTION IS A TEACHING,** AS [...]
What is the difference between the 13 colonies and the other British colonies in North America?	The 13 colonies, also known as the Thirteen Colonies, were a group of British colonies on the Atlantic coast of North America founded in the 17th and 18th centuries that declared independence in 1776 and formed the United States. The other British colonies in North America, often referred to as the Middle and Southern colonies, were established later and had different characteristics. \n 1. Location: The 13 colonies were located along the Atlantic [...] \n 2. Economy: The 13 colonies [...] \n 3. Settlement Patterns: [...]	The 13 colonies, also known as the Thirteen Colonies, were a group [...] These colonies were: \1. Virginia \2. Massachusetts \3. New Hampshire \n 4. Rhode Island \5. Connecticut\n 6. New York\n 7. New Jersey\n 8. Pennsylvania\n 9. Maryland\n 10. Delaware\n 11. \n ***Kentucky*** (not originally a British colony, but later admitted as a state)\n 12. ***Missouri*** (not originally a British colony, but later admitted as a state)\n 13. ***Mississippi*** (not originally a British colony, but later admitted as a state) [...]
Write a limerick about writing a limerick.	Whose verses were both sharp and clean.\n He penned a limerick,\n With wit so quick,\n And the rhythm was quite the machine.	There once was a poet named Kent,\n Whose limericks were quite the event.\n He wrote one with glee,\n A limerick, you see,\n That was about writing a limerick, **a test**.

Table 12. Examples of Changes in Quality Score. In the first case, the model is steered for the "Uppercase" instruction, which causes the output to become repetitive, resulting in a significant quality score drop from 1 to 0.4. In the second case, the model inaccurately states that Kentucky, Missouri, and Mississippi were part of the 13 colonies, leading to a drop in quality score from 0.8 to 0. In the third case, while the output looks fairly good, the model fails to follow the correct rhyme scheme for a limerick (AABBA), and the quality score drops

from 0.8 to 0.2. These examples are taken from the top 10 outputs generated by Phi-3 with the largest drop in quality score.

E.2. Examples of Quality Score Changes

While our evaluation procedure assesses how well the model's answers address the base queries, the method used to select steering locations and weights is based on IFeval's instruction-following accuracy metric. This metric consists of checks that verify specific aspects of the model's output related to instructions but sometimes these checks may be incomplete. For example, the default metric in IFEval for section-related formatting checks that the model includes sections in the generation but it does not check whether there is content in the section or the quality of the content if present. Therefore, in such cases this sometimes results in the selection of layer indices and steering weights that over-steer the model, leading to outputs that satisfy the instruction but have poor quality (e.g., repetitive token sequences). An example of this can be seen in the first row of Table 12. These issues are more prevalent in the Phi and Mistral models, compared to the Gemma models (noticeable from the smaller deltas for these models in Figure 4). This discrepancy may be due to Gemma's longer training and a better size/performance tradeoff, potentially making them more robust to activation steering.

Another source of quality drops can be minor factual inconsistencies in the model's responses. Steering may sometimes cause slight deviations from factual accuracy, which our quality score metric captures by incorporating GPT-4-generated questions about correctness. For example, in the second row of Table 12, the model inaccurately states that Kentucky, Missouri, and Mississippi were part of the original 13 colonies. While these inconsistencies can affect quality scores, they are not widespread and represent a general challenge for activation steering. Addressing factual consistency is difficult when selecting the steering location and weight, but it remains an important aspect to consider in future work. Finally, quality score drops can also result from small and arbitrary differences in the output, as shown in the third row of Table 12, in which the output looks reasonably good, but the model fails to follow the correct rhyme scheme for a limerick (AABBA), and the quality score drops by 0.6.

In conclusion, it is well-known that direct intervention on a model's activations can sometimes compromise the quality of the output. In our case, we observe relatively few instances where this happens, but improvements could be made. Future work could mitigate these issues by conducting a more exhaustive grid search over layers and steering weights, or by better modulating the steering intensity[23] [40].

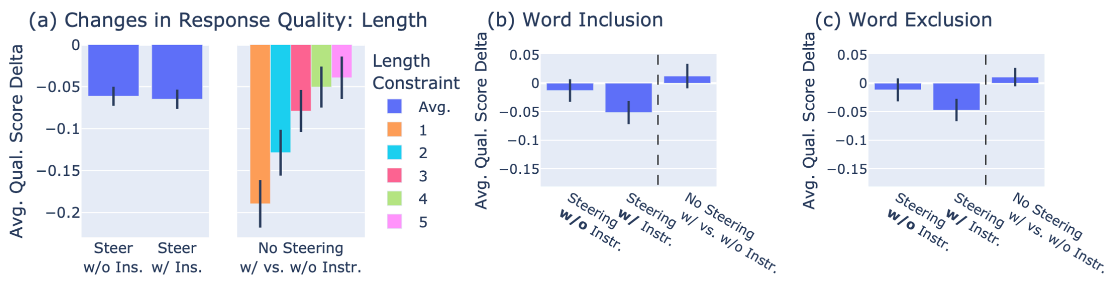


Figure 9. Changes in the Response Quality Scores. Quality score deltas for (a) length instructions, (b) word inclusion, and (c) word exclusion, across three conditions: steering without instructions, steering with instructions, and no steering with vs. without instructions.

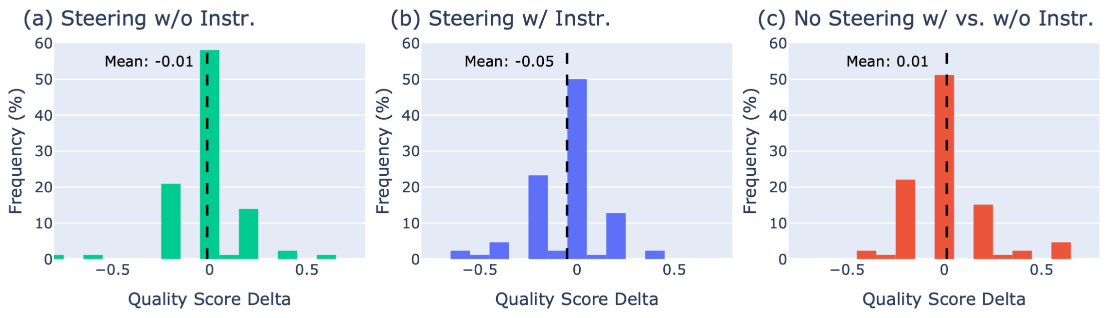


Figure 10. Distribution of Quality Score Changes for Word Inclusion. Histograms of quality score deltas under three conditions: (a) steering without explicit instructions, (b) steering with explicit instructions, and (c) no steering, comparing outputs with and without instructions. The distributions show minimal impact on quality scores across all settings, with mean values close to zero, with most shifts falling within the [-0.25, 0.25] range.

E.3. Additional Response Quality Results

We apply the response quality evaluation procedure described in Appendix F to outputs generated by steering the model for length and word-specific instructions, following the same approach used for format constraints in §3. As in the previous section, we analyze the outputs produced by Phi-3 Mini, reporting the score changes due to steering both with and without explicit text instructions, as well as the quality score differences when text instructions are added without steering.

Length Instructions. Figure 9a presents the results for length constraints. We analyze a subset of 100 outputs from the generation process used for Figure 5b in §4 (which evaluated 200 examples). The model is steered with a weight of $c = 20$ on a set of base prompts, applying different length constraints requesting the model to answer using at most n sentences ($n \in \{1, \dots, 5\}$). The two leftmost columns in Figure 9a show the changes in the quality score when steering the model on inputs with and without length instructions. These values are averaged over the 5 constraints, as the steering procedure applied is the same way for all 5 constraints (adding the *conciseness* vector with weight $c = 20$). A negative delta is expected, as steering the model to produce shorter responses generally reduces the comprehensiveness of the answer. This pattern is confirmed by the rightmost columns in Figure 9a, which represent the difference in quality score when length instructions are explicitly provided in the input text. The different colors correspond to the number of sentences (n), and as expected, shorter length constraints lead to larger drops in the quality score. Although the effect of steering is not directly comparable to any specific length constraint (since steering applies an additional reduction in length without a one-to-one mapping between steering weight and sentence count), we observe that the changes in quality scores due to steering remain within a reasonable range when compared to those induced by text instructions.

Word-specific Instructions. In Figures 9b and 9c, we report the changes in quality scores for instructions that request the inclusion or exclusion of a specific word in the answer. In this case, introducing text instructions on top of the base query does not result in a noticeable drop in quality. Similarly, steering the model toward word inclusion/exclusion without explicit text instructions does not significantly impact quality (as seen from the near-zero deltas in the leftmost columns). However, when explicit instructions are provided in the prompt, we observe a slight decrease in the quality score, around 0.05. To further investigate this, we examine the empirical distributions of the score deltas for word inclusion instructions (Figure 10). The distributions appear centered around 0, with a few outliers showing deviations larger than 0.4. We perform a paired two-sided t-test to assess whether the means of these three distributions are significantly different. The p-values for the three tests (one for each pair) are all greater than 0.05 (0.12, 0.06, and 0.30). While the lack of significance does not prove that the distributions are identical, we interpret this as evidence of the minimal impact our steering procedure has on the model's ability to address the base queries.

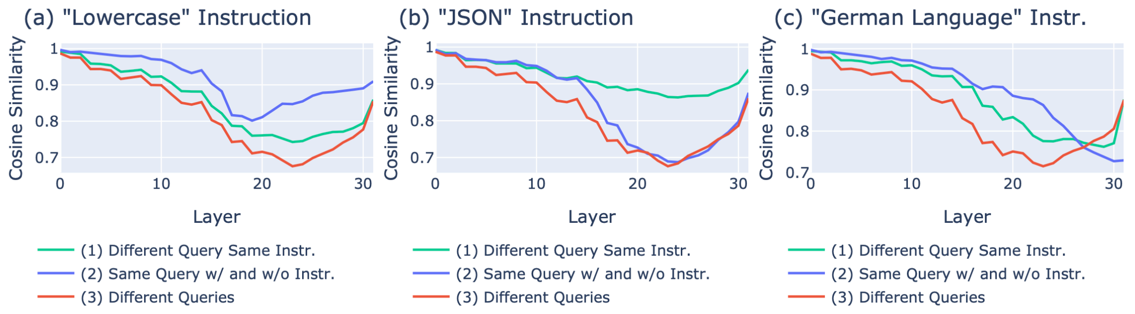


Figure 11. Residual stream similarity across layers. Cosine similarity of residual stream activations across layers for three instructions: (a) "Lowercase," (b) "JSON," and (c) "German Language." Comparisons are made between: (1) different queries with the same instruction (green), (2) same query with and without the instruction (blue), and (3) different queries without instruction (red). Results are shown for Phi-3 across all layers.

F. Implementation and Experimental Details

Tokenization and Decoding. Steering vectors are computed at the last token of the input. For instruction-tuned models, this typically corresponds to the <|assistant|> token, which marks the transition from user input to the start of the model's generation.[7] For non-instruction-tuned models, we follow previous work[76][77] and structure the prompt as "Q: {problem}\nA:." Model outputs are decoded greedily, with a maximum generation length of 2048 tokens. For efficiency, validation runs use a reduced maximum length of 384 tokens.

Steering Weight Selection. We base the selection of the steering weight on the computation in Eq. (2). For format instructions, the weight from Eq. (2) is used directly. For length and format instructions, the value computed in Eq. (2) is averaged across different inputs and is used as a reference to determine a range of suitable steering weights. For length instructions, we experiment with multiple weight values, highlighting the continuous nature of the constraint. For word inclusion instructions, we notice that in some cases (particularly with Gemma 2B), the weight is too low to impact the model's output. This could be because the steering vector is averaged across inputs regardless of whether the model satisfies the constraint. This issue tends to be more pronounced for word-specific constraints (than, e.g., for format instructions) since the strength of the instruction signal can vary greatly depending on whether the word is related to the base query (e.g., including a word related to the query is easier than including an unrelated one). To address this, we perform a small additional grid search on a set of held-out examples, refining the weight based on the initial value from Eq. (2). For word exclusion instructions, we perform the grid search around the negative of the value computed for word inclusion.

Tools and Libraries. For length-related experiments, word counts are measured as the number of space-separated sequences of characters and sentence counts are determined using NLTK[78]. To assess whether the score improvement from steering is significantly different from standard inference, we carry out McNemar's test[37]. In particular, we use the exact version of the test, which uses the binomial distribution and is more conservative. The error bars reported in Figures 4 and 9 represent the standard error of the quality score computed over three different runs of GPT-4o. The smoothed empirical distributions shown in Figures 5a, 5c, 8c, and 12 are obtained using kernel density estimation[79]. Our experiments were carried out using PyTorch[80] and the TransformersLens library[81]. We performed our data analysis using NumPy[82] and Pandas[83]. Our figures were made using Plotly[84]. The paper's bibliography was curated using Ryanize-bib[85].

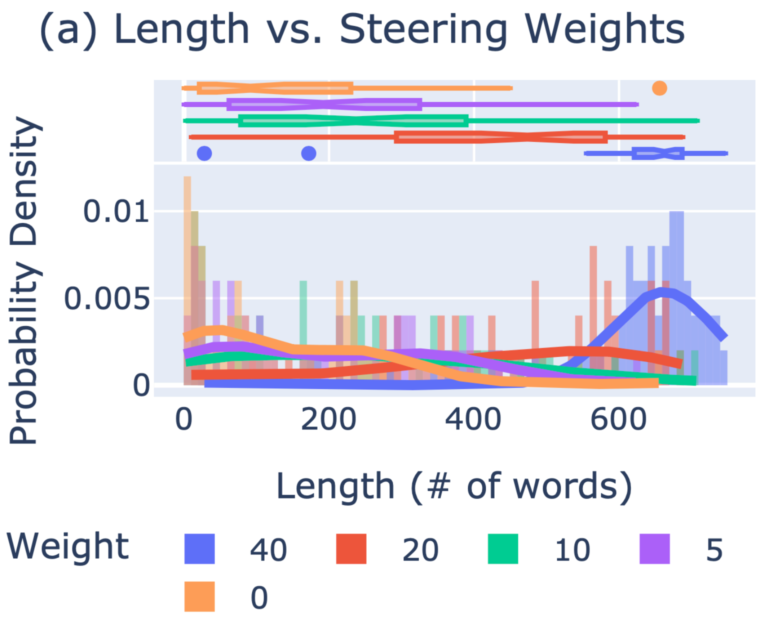


Figure 12. Output Length Distribution with Verbosity Steering. The distribution of output lengths when steering Phi-3 using a vector for verbosity, with varying steering weights.

G. Additional Results

Similarity of Activations. Figure 11 shows the cosine similarity of residual stream activations in Phi-3 across different input sets. As described in §3, we compare the similarity of representations between: (1) pairs of inputs sharing the same base query, one with and one without the instruction; (2) inputs with different base queries but the same instruction; and (3) inputs with different base queries and no instruction. The similarity scores are reported across Phi-3's layers for three instructions: "lowercase" (no uppercase characters, Figure 11a), "JSON" (response formatted as JSON, Figure 11b), and "German Language" (Figure 11c).

Length Instructions: Steering for Longer Outputs. In Figure 12, we show the distribution of output lengths when the model is steered using a verbosity vector. This vector is computed from inputs containing instructions to generate verbose responses (e.g., "Provide a long answer"). Similar to our procedure for the conciseness vector in §4, we apply different steering weights to modulate verbosity. Using the same 50 base prompts as in Figure 5a, the results show that higher steering weights result in longer outputs, demonstrating that activation steering can effectively control output length by adjusting a single scalar value.

Footnotes

1 We use *instruction* to refer to specific constraints that can be added during interactive sessions with LLMs in a modular way to modify or extend a *base* question or request. We further elaborate on this definition in §2.

2 The model's residual stream is the per-token hidden state of dimensionality d_{model} consisting of the sum of all previous component outputs[59].

3 Although they are not strictly related to format, we include language constraints (e.g., "the answer should be in German") in this group, as, like other formatting instructions, they modify the surface-level presentation of the output without affecting the underlying content.

4 Note that while the model's response may be influenced by specific instructions, these instructions are not provided to GPT-4o during the evaluation. The goal here is to focus the quality assessment solely on the comprehensiveness and relevance of the response with respect to the base query.

[5] IFEval prompts may contain instructions for the inclusion or exclusion of multiple keywords in the same example. We separate such cases into inputs with single-keyword instructions.

[6] https://openai.com/index/hello-gpt-4o/

[7] https://huggingface.co/docs/transformers/main/en/chat_templating

Notes

Alessandro Stolfo: Work done while at Microsoft Research.

References

1. a, b *Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C, Mishkin P, Zhang C, Agarwal S, Slama K, Gray A, Schulman J, Hilton J, Kelton F, Miller L, Simens M, Askell A, Welinder P, Christiano P, Leike J, Lowe R. Training language models to follow instructions with human feedback. In Oh AH, Agarwal A, Belgrave D, Cho K, editors. Advances in Neural Information Processing Systems. 2022. URL https://openreview.net/forum?id=TG8KACxEON.*
2. a, b *Bai Y, Jones A, Ndousse K, Askell A, Chen A, DasSarma N, Drain D, Fort S, Ganguli D, Henighan T, Joseph N, Kadavath S, Kernion J, Conerly T, El-Showk S, Elhage N, Hatfield-Dodds Z, Hernandez D, Hume T, Johnston S, Kravec S, Lovitt L, Nanda N, Olsson C, Amodei D, Brown T, Clark J, McCandlish S, Olah C, Mann B, Kaplan J. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. URL https://arxiv.org/abs/2204.05862.*
3. a, b, c *Wei J, Bosma M, Zhao V, Guu K, Yu AW, Lester B, Du N, Dai AM, Le QV. Finetuned language models are zero-shot learners. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.*
4. a, b, c *Sanh V, Webson A, Raffel C, Bach S, Sutawika L, Alyafeai Z, Chaffin A, Stiegler A, Raja A, Dey M, Bari MS, Xu C, Thakker U, Sharma S, Szczechla E, Kim T, Chhablani G, Nayak N, Datta D, Chang J, Jiang MT, Wang H, Manica M, Shen S, Yong ZX, Pandey H, Bawden R, Wang T, Neeraj T, Rozen J, Sharma A, Santilli A, Fevry T, Fries JA, Teehan R, Le Scao T, Biderman S, Gao L, Wolf T, Rush AM. Multitask prompted training enables zero-shot task generalization. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.*
5. a, b *Chung HW, Hou L, Longpre S, Zoph B, Tay Y, Fedus W, Li Y, Wang X, Dehghani M, Brahma S, et al. Scaling instruction-finetuned language models. Journal of Machine Learning Research. 2024;25(70):1-53. URL http*

s://www.jmlr.org/papers/v25/23-0870.html.

6. ^*Askel A, Bai Y, Chen A, Drain D, Ganguli D, Henighan T, Jones A, Joseph N, Mann B, DasSarma N, Elhage N, Hatfield-Dodds Z, Hernandez D, Kernion J, Ndousse K, Olsson C, Amodei D, Brown T, Clark J, McCandlish S, Olah C, Kaplan J. A general language assistant as a laboratory for alignment, 2021. URL https://arxiv.org/abs/2112.00861.*
7. a, b, c*Zhou W, Jiang YE, Wilcox E, Cotterell R, Sachan M. Controlled text generation with natural language instructions. In Krause A, Brunskill E, Cho K, Engelhardt B, Sabato S, Scarlett J, editors. Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pp. 42602-42613. PMLR, 23-29 Jul 2023. URL https://proceedings.mlr.press/v202/zhou23g.html.*
8. ^*Zhang H, Song H, Li S, Zhou M, Song D. A survey of controllable text generation using transformer-based pre-trained language models. ACM Comput. Surv.. 2023;56(3). doi:10.1145/3617680. URL https://doi.org/10.1145/3617680.*
9. ^*Lou R, Zhang K, Yin W. Large language model instruction following: A survey of progresses and challenges. Computational Linguistics. 2024;1-43. DOI: 10.1162/coli_a_00523. URL: https://doi.org/10.1162/coli_a_00523.*
10. a, b*Hendel R, Geva M, Globerson A. In-context learning creates task vectors. In Bouamor H, Pino J, Bali K, editors. Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 9318-9333, Singapore, dec 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.624. URL https://aclanthology.org/2023.findings-emnlp.624.*
11. a, b, c*Todd E, Li M, Sen Sharma A, Mueller A, Wallace BC, Bau D. Function vectors in large language models. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG.*
12. a, b, c*Liu S, Ye H, Xing L, Zou JY. In-context vectors: Making in context learning more effective and controllable through latent space steering. In Forty-first International Conference on Machine Learning, 2024. URL https://openreview.net/forum?id=dJTChKgv3a.*
13. a, b, c, d, e*Zou A, Phan L, Chen S, Campbell J, Guo P, Ren R, Pan A, Yin X, Mazeika M, Dombrowski AK, Goel S, Li N, Byun MJ, Wang Z, Mallen A, Basart S, Koyejo S, Song D, Fredrikson M, Kolter JZ, Hendrycks D. Representation engineering: A top-down approach to AI transparency, 2023. URL https://arxiv.org/abs/2310.01405.*
14. a, b*Azaria A, Mitchell T. The internal state of an LLM knows when it's lying. In Bouamor H, Pino J, Bali K, editors. Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 967-976, Singapore, dec 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.68. URL: https://aclanthology.org/2023.findings-emnlp.68.*

15. a, b*Zheng C, Yin F, Zhou H, Meng F, Zhou J, Chang KW, Huang M, Peng N. On prompt-driven safeguarding for large language models. In ICLR 2024 Workshop on Secure and Trustworthy Large Language Models, 2024. URL https://openreview.net/forum?id=lFwf7bnpUs.*

16. a, b*Templeton A, Conerly T, Marcus J, Lindsey J, Bricken T, Chen B, Pearce A, Citro C, Ameisen E, Jones A, Cunn ingham H, Turner NL, McDougall C, MacDiarmid M, Freeman CD, Sumers TR, Rees E, Batson J, Jermyn A, Ca rter S, Olah C, Henighan T. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. Transformer Circuits Thread, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/in dex.html.*

17. a, b, c*Marks S, Tegmark M. The geometry of truth: Emergent linear structure in large language model repres entations of true/false datasets. In First Conference on Language Modeling, 2024. URL https://openreview.ne t/forum?id=aajyHYjjsk.*

18. a, b*Subramani N, Suresh N, Peters M. Extracting latent steering vectors from pretrained language models. In Muresan S, Nakov P, Villavicencio A, editors. Findings of the Association for Computational Linguistics: ACL 2022, pp. 566-581, Dublin, Ireland, May 2022. Association for Computational Linguistics. DOI: 10.18653/v1/20 22.findings-acl.48. URL: https://aclanthology.org/2022.findings-acl.48.*

19. a, b, c*Li K, Patel O, Viégas F, Pfister H, Wattenberg M. Inference-time intervention: Eliciting truthful answers f rom a language model. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=aLLuYpn83y.*

20. a, b*Qiu Y, Zhao Z, Ziser Y, Korhonen A, Ponti EM, Cohen SB. Spectral editing of activations for large language model alignment, 2024. URL https://arxiv.org/abs/2405.09719.*

21. a, b, c, d*Tigges C, Hollinsworth OJ, Nanda N, Geiger A. Language models linearly represent sentiment, 2024. U RL https://openreview.net/forum?id=iGDWZFc7Ya.*

22. a, b, c, d, e*Turner AM, Thiergart L, Leech G, Udell D, Vazquez JJ, Mini U, MacDiarmid M. Activation addition: St eering language models without optimization. arXiv preprint arXiv:2308.10248, 2023. URL https://arxiv.org/a bs/2308.10248.*

23. a, b, c, d*Scalena D, Sarti G, Nissim M. Multi-property steering of large language models with dynamic activati on composition, 2024. URL https://arxiv.org/abs/2406.17563.*

24. a, b*von Rütte D, Anagnostidis S, Bachmann G, Hofmann T. A language model's guide through latent space, 2 024. URL https://arxiv.org/abs/2402.14433.*

25. a, b, c, d, e*Arditi A, Obeso O, Syed A, Paleka D, Panickssery N, Gurnee W, Nanda N. Refusal in language models is mediated by a single direction. arXiv preprint arXiv:2406.11717, 2024. URL https://arxiv.org/abs/2406.11717.*

26. a, b*Wang H, Shu K. Trojan activation attack: Red-Teaming large language models using activation steering f or safety-alignment, 2024. URL https://arxiv.org/abs/2311.09433.*

27. a, b, c, d, e, f, g*Panickssery N, Gabrieli N, Schulz J, Tong M, Hubinger E, Turner A. Steering llama 2 via contrasti ve activation addition. In Ku LW, Martins A, Srikumar V, editors. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 15504-15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.828.*

28. a, b, c, d*van der Weij T, Poesio M, Schoots N. Extending activation steering to broad skills and multiple behavi ours, 2024. URL https://arxiv.org/abs/2403.05767.*

29. a, b, c*Sun J, Tian Y, Zhou W, Xu N, Hu Q, Gupta R, Wieting J, Peng N, Ma X. Evaluating large language models o n controlled generation tasks. In Bouamor H, Pino J, Bali K, editors. Proceedings of the 2023 Conference on E mpirical Methods in Natural Language Processing, pp. 3155-3168, Singapore, dec 2023. Association for Comp utational Linguistics. DOI: 10.18653/v1/2023.emnlp-main.190. URL: https://aclanthology.org/2023.emnlp-mai n.190.*

30. a, b, c, d*Zhou J, Lu T, Mishra S, Brahma S, Basu S, Luan Y, Zhou D, Hou L. Instruction-following evaluation for l arge language models, 2023. URL https://arxiv.org/abs/2311.07911.*

31. a, b*Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Che n, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfen g Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenk o, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyu an Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Lilia ng Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Sali m, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shu ohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yan*

g, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

32. a, b*Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.*

33. a, b*Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.*

34. a, b, c*Kenneth Li, Tianle Liu, Naomi Bashkansky, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Measuring and controlling instruction (in)stability in language model dialogs. In First Conference on Language Modeling, 2024. URL https://openreview.net/forum?id=60a1SAtH4e.*

35. a, b*Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.*

36. ^*Nora Belrose. Diff-in-means concept editing is worst-case optimal: Explaining a result by Sam Marks and Max Tegmark, 2023. URL https://blog.eleuther.ai/diff-in-means/*

37. a, b*McNemar Q. Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika. 12(2): 153--157, 1947. URL https://link.springer.com/article/10.1007/bf02295996.*

38. ^*nostalgebraist. Interpreting GPT: The logit lens. LessWrong, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.*

39. ^*Geva M, Caciularu A, Wang K, Goldberg Y. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In: Goldberg Y, Kozareva Z, Zhang Y, editors. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing; 2022 Dec; Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. p. 30--45. doi:10.18653/v1/2022.emnlp-main.3. URL: https://aclanthology.org/2022.emnlp-main.3.*

40. a, b*Asa Cooper Stickland, Alexander Lyzhov, Jacob Pfau, Salsabila Mahdi, and Samuel R. Bowman. Steering without side effects: Improving post-deployment control of language models, 2024. URL https://arxiv.org/abs/2406.15518.*

41. ^*Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K. Kummerfeld, and Rada Mihalcea. A mechanistic understanding of alignment algorithms: A case study on DPO and toxicity. In Forty-first Inter*

national Conference on Machine Learning, 2024. URL https://openreview.net/forum?id=dBqHGZPGZI.

42. ^*Samyak Jain, Ekdeep Singh Lubana, Kemal Oksuz, Tom Joy, Philip Torr, Amartya Sanyal, and Puneet K. Do kania. What makes and breaks safety fine-tuning? a mechanistic study. In ICML 2024 Workshop on Mechan istic Interpretability, 2024. URL https://openreview.net/forum?id=BS2CbUkJpy.*

43. a, b*Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoenco ders find highly interpretable features in language models. In The Twelfth International Conference on Lear ning Representations, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.*

44. ^*Connor Kissane, Robert Krzyzanowski, Arthur Conmy, and Neel Nanda. SAEs (usually) transfer between ba se and chat models. Alignment Forum, 2024. URL https://www.alignmentforum.org/posts/fmwk6qxrpW8d4j vbd/saes-usually-transfer-between-base-and-chat-models.*

45. ^*Ye Q, Lin BY, Ren X. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In: Moens MF, Huang X, Specia L, Yih SW, editors. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing; 2021 Nov; Online and Punta Cana, Dominican Republic. Association for Computation al Linguistics. p. 7163--7189. doi:10.18653/v1/2021.emnlp-main.572. URL: https://aclanthology.org/2021.emnlp -main.572.*

46. ^*Wang Y, Mishra S, Alipoormolabashi P, Kordi Y, Mirzaei A, Naik A, Ashok A, Dhanasekaran AS, Arunkumar A, Stap D, Pathak E, Karamanolakis G, Lai H, Purohit I, Mondal I, Anderson J, Kuznia K, Doshi K, Pal KK, Patel M, Moradshahi M, Parmar M, Purohit M, Varshney N, Kaza PR, Verma P, Puri RS, Karia R, Doshi S, Sampat S K, Mishra S, Reddy SA, Patro S, Dixit T, Shen X. Super-NaturalInstructions: Generalization via declarative inst ructions on 1600+ NLP tasks. In: Goldberg Y, Kozareva Z, Zhang Y, editors. Proceedings of the 2022 Conferen ce on Empirical Methods in Natural Language Processing; 2022 Dec; Abu Dhabi, United Arab Emirates. Asso ciation for Computational Linguistics. p. 5085--5109. doi:10.18653/v1/2022.emnlp-main.340. URL: https://acl anthology.org/2022.emnlp-main.340.*

47. ^*Gupta P, Jiao C, Yeh YT, Mehri S, Eskenazi M, Bigham J. InstructDial: Improving zero and few-shot generaliz ation in dialogue through instruction tuning. In: Goldberg Y, Kozareva Z, Zhang Y, editors. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing; 2022 Dec; Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. p. 505--525. doi:10.18653/v1/2022.emnlp-main.33. URL: https://aclanthology.org/2022.emnlp-main.33.*

48. ^*Finlayson M, Richardson K, Sabharwal A, Clark P. What makes instruction learning hard? an investigation and a new challenge in a synthetic environment. In: Goldberg Y, Kozareva Z, Zhang Y, editors. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing; 2022 Dec; Abu Dhabi, United A*

rab Emirates. Association for Computational Linguistics. p. 414--426. doi:10.18653/v1/2022.emnlp-main.27. URL: https://aclanthology.org/2022.emnlp-main.27.

49. ^*Mishra S, Khashabi D, Baral C, Hajishirzi H. Cross-task generalization via natural language crowdsourcing instructions. In: Muresan S, Nakov P, Villavicencio A, editors. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); 2022 May; Dublin, Ireland. Association for Computational Linguistics. p. 3470--3487. doi:10.18653/v1/2022.acl-long.244. URL: https://aclanthology.org/2022.acl-long.244.*
50. ^*Longpre S, Hou L, Vu T, Webson A, Chung HW, Tay Y, Zhou D, Le QV, Zoph B, Wei J, Roberts A. The flan collection: Designing data and methods for effective instruction tuning. In: Krause A, Brunskill E, Cho K, Engelhardt B, Sabato S, Scarlett J, editors. Proceedings of the 40th International Conference on Machine Learning; 2023 Jul; PMLR. p. 22631--22648. URL https://proceedings.mlr.press/v202/longpre23a.html.*
51. ^*Köpf A, Kilcher Y, von Rütte D, Anagnostidis S, Tam ZR, Stevens K, Barhoum A, Nguyen D, Stanley O, Nagyfi R, ES S, Suri S, Glushkov D, Dantuluri A, Maguire A, Schuhmann C, Nguyen H, Mattick A. Openassistant conversations - democratizing large language model alignment. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Advances in Neural Information Processing Systems; 2023. Curran Associates, Inc. p. 47669--47681. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/949f0f8f32267d297c2d4e3ee10a2e7e-Paper-Datasets_and_Benchmarks.pdf.*
52. ^*Huang SC, Li PZ, Hsu Y, Chen KM, Lin YT, Hsiao SK, Tsai R, Lee H. Chat vector: A simple approach to equip LLMs with instruction following and model alignment in new languages. In: Ku LW, Martins A, Srikumar V, editors. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); 2024 Aug; Bangkok, Thailand. Association for Computational Linguistics. URL: https://aclanthology.org/2024.acl-long.590.*
53. ^*Perez E, Ringer S, Lukosiute K, Nguyen K, Chen E, Heiner S, Pettit C, Olsson C, Kundu S, Kadavath S, Jones A, Chen A, Mann B, Israel B, Seethor B, McKinnon C, Olah C, Yan D, Amodei D, Amodei D, Drain D, Li D, Tran-Johnson E, Khundadze G, Kernion J, Landis J, Kerr J, Mueller J, Hyun J, Landau J, Ndousse K, Goldberg L, Lovitt L, Lucas M, Sellitto M, Zhang M, Kingsland N, Elhage N, Joseph N, Mercado N, DasSarma N, Rausch O, Larson R, McCandlish S, Johnston S, Kravec S, Showk S, Lanham T, Telleen-Lawton T, Brown T, Henighan T, Hume T, Bai Y, Hatfield-Dodds Z, Clark J, Bowman SR, Askell A, Grosse R, Hernandez D, Ganguli D, Hubinger E, Schiefer N, Kaplan J. Discovering language model behaviors with model-written evaluations. In: Rogers A, Boyd-Graber J, Okazaki N, editors. Findings of the Association for Computational Linguistics: ACL 2023; 2023 Jul; Toro*

nto, Canada. Association for Computational Linguistics. p. 13387--13434. doi:10.18653/v1/2023.findings-acl.847. URL: https://aclanthology.org/2023.findings-acl.847.

54. ^*Sharma M, Tong M, Korbak T, Duvenaud D, Askell A, Bowman SR, DURMUS E, Hatfield-Dodds Z, Johnston SR, Kravec SM, Maxwell T, McCandlish S, Ndousse K, Rausch O, Schiefer N, Yan D, Zhang M, Perez E. Towards understanding sycophancy in language models. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=tvhaxkMKAn.*
55. ^*Gurnee W, Tegmark M. Language models represent space and time. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=jE8xbmvFin.*
56. ^*Bricken T, Templeton A, Batson J, Chen B, Jermyn A, Conerly T, Turner N, Anil C, Denison C, Askell A, Lasenby R, Wu Y, Kravec S, Schiefer N, Maxwell T, Joseph N, Hatfield-Dodds Z, Tamkin A, Nguyen K, McLean B, Burke JE, Hume T, Carter S, Henighan T, Olah C. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.*
57. ^*Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: Burges CJ, Bottou L, Welling M, Ghahramani Z, Weinberger KQ, editors. Advances in Neural Information Processing Systems; 2013. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.*
58. ^*Bolukbasi T, Chang KW, Zou JY, Saligrama V, Kalai AT. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In: Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R, editors. Advances in Neural Information Processing Systems; 2016. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf.*
59. a, b*Elhage N, Nanda N, Olsson C, Henighan T, Joseph N, Mann B, Askell A, Bai Y, Chen A, Conerly T, DasSarma N, Drain D, Ganguli D, Hatfield-Dodds Z, Hernandez D, Jones A, Kernion J, Lovitt L, Ndousse K, Amodei D, Brown T, Clark J, Kaplan J, McCandlish S, Olah C. A mathematical framework for transformer circuits. Transformer Circuits Thread, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.*
60. ^*Nanda N, Lee A, Wattenberg M. Emergent linear representations in world models of self-supervised sequence models. In: Belinkov Y, Hao S, Jumelet J, Kim N, McCarthy A, Mohebbi H, editors. Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP; 2023 Dec; Singapore. Association for Computational Linguistics. p. 16--30. doi:10.18653/v1/2023.blackboxnlp-1.2. URL https://aclanthology.org/2023.blackboxnlp-1.2.*

61. ^*Kiho Park, Yo Joong Choe, Victor Veitch. The linear representation hypothesis and the geometry of large lan guage models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan S carlett, Felix Berkenkamp (eds.), Proceedings of the 41st International Conference on Machine Learning, volu me 235 of Proceedings of Machine Learning Research, pp. 39643--39666. PMLR, 21--27 Jul 2024. URL https:// proceedings.mlr.press/v235/park24c.html.*

62. ^*Christopher Olah. What is a linear representation? what is a multidimensional feature? Transformer Circui ts Thread, 2024. URL https://transformer-circuits.pub/2024/july-update/.*

63. ^*Joshua Engels, Isaac Liao, Eric J. Michaud, Wes Gurnee, Max Tegmark. Not all language model features are l inear, 2024. URL https://arxiv.org/abs/2405.14860.*

64. ^*Róbert Csordás, Christopher Potts, Christopher D. Manning, Atticus Geiger. Recurrent neural networks lear n to store and generate sequences using non-linear representations, 2024. URL https://arxiv.org/abs/2408.10 920.*

65. ^*Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In Dan Jurafsky, Joyce Chai, Natalie Schluter, Joel Tetreault (eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 7237--7256, O nline, July 2020. Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.647. URL: https://a clanthology.org/2020.acl-main.647.*

66. ^*Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, Stella Biderman. LEAC E: Perfect linear concept erasure in closed form. In Thirty-seventh Conference on Neural Information Process ing Systems, 2023. URL https://openreview.net/forum?id=awIpKpwTwF.*

67. ^*Shun Shao, Yftah Ziser, Shay B. Cohen. Gold doesn't always glitter: Spectral removal of linear and nonlinear guarded attribute information. In Andreas Vlachos, Isabelle Augenstein (eds.), Proceedings of the 17th Confer ence of the European Chapter of the Association for Computational Linguistics, pp. 1611–1622, Dubrovnik, Cr oatia, May 2023. Association for Computational Linguistics. doi:10.18653/v1/2023.eacl-main.118. URL: https:// aclanthology.org/2023.eacl-main.118.*

68. ^*Clément Guerner, Anej Svete, Tianyu Liu, Alexander Warstadt, Ryan Cotterell. A geometric notion of causal probing, 2024. URL https://arxiv.org/abs/2307.15054.*

69. ^*Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, Rosa nne Liu. Plug and play language models: A simple approach to controlled text generation. In International C onference on Learning Representations, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.*

70. ^Dawn Lu, Nina Panickssery. Investigating bias representations in llama 2 chat via activation steering, 2024. URL https://arxiv.org/abs/2402.00402.

71. ^Nate Rahn, Pierluca D'Oro, Marc G. Bellemare. Controlling large language model agents with entropic activation steering, 2024. URL https://arxiv.org/abs/2406.00244.

72. ^Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, Jinghui Chen. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization, 2024. URL https://arxiv.org/abs/2406.00045.

73. ^Evan Hernandez, Belinda Z. Li, Jacob Andreas. Inspecting and editing knowledge representations in language models. In First Conference on Language Modeling, 2024. URL https://openreview.net/forum?id=ADtL6fgNRv.

74. ^Collin Burns, Haotian Ye, Dan Klein, Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

75. ^OpenAI. GPT-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

76. ^Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, Yusuke Iwasawa. Large language models are zero-shot reasoners. In ICML 2022 Workshop on Knowledge Retrieval and Language Models, 2022. URL https://openreview.net/forum?id=6p3AuaHAFiN.

77. ^Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, Xinyun Chen. Large language models as optimizers. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

78. ^Steven Bird, Edward Loper. NLTK: The natural language toolkit. In Proceedings of the ACL Interactive Poster and Demonstration Sessions, pp. 214--217, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/P04-3031.

79. ^Emanuel Parzen. On estimation of a probability density function and mode. The Annals of Mathematical Statistics, 33(3): 1065--1076, 1962. ISSN 00034851. URL http://www.jstor.org/stable/2237880.

80. ^Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (eds.), Advances in Neural Information Processing Systems, volum

e 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288f ee7f92f2bfa9f7012727740-Paper.pdf.

81. ^*Neel Nanda, Joseph Bloom. Transformerlens, 2022. URL https://github.com/TransformerLensOrg/Transfor merLens.*
82. ^*Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournape au, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, M arten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlk e, Travis E. Oliphant. Array programming with NumPy. Nature, 585(7825): 357--362, sep 2020. doi:10.1038/s4 1586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.*
83. ^*Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt, Jarrod Millman (eds.), Proceedings of the 9th Python in Science Conference, pp. 56 -- 61, 2010. doi:10.25080/Majora-92bf192 2-00a. URL https://scipy.org.*
84. ^*Plotly Technologies Inc. Collaborative data science, 2015. URL https://plot.ly.*
85. ^*Vilém Zouhar. Ryanize bib, 2023. URL https://github.com/zouharvi/ryanize-bib.*

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.