Compositional Steering of Large Language Models with Steering Tokens

Gorjan Radevski[2*], **Kiril Gashteovski**[1,5], **Giwon Hong**[1,3], **Carolin Lawrence**[1], **Goran Glavaš**[4]
[1]NEC Laboratories Europe, Germany [2]Independent
[3]University of Edinburgh, United Kingdom
[4]Center for Artificial Intelligence and Data Science, University of Würzburg, Germany
[5]CAIR, Ss. Cyril and Methodius University of Skopje, North Macedonia

Abstract

Deploying LLMs in real-world applications requires controllable output that satisfies multiple desiderata at the same time. While existing work extensively addresses LLM steering for a single behavior, *compositional steering*—i.e., steering LLMs simultaneously towards multiple behaviors—remains an underexplored problem. In this work, we propose *compositional steering tokens* for multi-behavior steering. We first embed individual behaviors, expressed as natural language instructions, into dedicated tokens via self-distillation. Contrary to most prior work, which operates in the activation space, our behavior steers live in the space of input tokens, enabling more effective zero-shot composition. We then train a dedicated *composition token* on pairs of behaviors and show that it successfully captures the notion of composition: it generalizes well to *unseen* compositions, including those with unseen behaviors as well as those with an unseen *number* of behaviors. Our experiments across different LLM architectures show that steering tokens lead to superior multi-behavior steering of verifiable constraints (e.g., length, format, structure, language) compared to competing approaches (instructions, activation steering, and LoRA merging). Moreover, we show that steering tokens complement natural language instructions, with their combination resulting in further gains.

1 Introduction

The ubiquitous use of Large Language Models (LLMs) necessitates that their behavior can be controlled in an ad-hoc manner (Khamassi et al., 2024). In most real-world domains, due to intrinsic task complexities, the need for *compositional steering*, that is, controlling LLMs for multiple behaviors simultaneously (as opposed to just a single behavior[1]), arises naturally (Sorensen et al., 2024). While fine-tuning LLMs for desired behaviors (Tian et al., 2024; Tan et al., 2024) is the most straightforward solution to achieve such control, it comes with major limitations. Entailing updates to the LLM's parameters, it (i) is computationally intensive and (ii) jeopardizes the general-purpose utility of the LLM (i.e., there is a risk of negative interference and forgetting of pretrained knowledge); though both can be (somewhat) mitigated via parameter-efficient fine-tuning (Hu et al., 2022). The need to update the model renders fine-tuning impractical for ad-hoc steering, especially for compositional behavior: N behaviors that can be arbitrarily composed imply 2^N (independent) fine-tuning procedures.

The goal of general-purpose instruction-tuning with diverse task instructions (Ouyang et al., 2022; Zhang et al., 2025), in contrast, is exactly to enable flexible and generalizable control via natural language prompts. Behavior steering via prompting, however, is very brittle (Ngweta et al., 2025; Errica et al., 2025): it yields inconsistent LLM behavior for semantically equivalent prompts.

Various recent prompt compression approaches (Li et al., 2025) have been proposed as a middle ground between the inflexibility (and cost) of fine-tuning and the brittleness of prompting, including activation steering (Stolfo et al., 2025; Cao et al., 2024), gist tokens (Mu et al., 2023; Dong et al., 2025), and persona vectors (Lim et al., 2025; Pai et al., 2025). While these methods efficiently compress prompts and improve LLMs' conformance for individual behaviors, they largely fail to demonstrate effectiveness and generalization in compositional steering. Put differently, while good solutions for compressing individual behaviors into embeddings are abundant, we still lack an effective representation for the concept of *composition* that

*Correspondence to: gorjan.radevski@gmail.com or kiril.gashteovski@neclab.eu

[1]We define behavior as any controllable aspect of the LLM's output (e.g., length, language, reasoning style).

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 31087–31104

would generalize across arbitrary behavior combinations (and *number* of composed behaviors).

In this work, we address this gap by introducing *compositional steering tokens* for multi-behavior steering (Fig. 1). We first compress individual behaviors, formulated as natural language instructions, into specialized tokens (vectors) via self-distillation: unlike in most related work, where behavior representations interact with model internals, our steering tokens reside in the model's input space, which leads to better zero-shot composition. As a central novelty of our work, we then train a dedicated *composition token* to capture the general concept of behavior composition. After training on a fixed set of behavior pairs, we show that our composition token truly encapsulates the notion of composition, exhibiting strong generalization to: (i) *unseen* compositions of *seen* behaviors, (ii) compositions with *unseen* behaviors, and (iii) compositions of *an unseen number* of behaviors.

We carry out an extensive evaluation on *verifiable* behavioral constraints (response length, language, formatting, etc.), enabling strict automatic assessment of compositional generalization at scale. Specifically, we find that: ① Steering tokens achieve superior compositional generalization compared to existing methods, particularly on zero-shot compositions of behavior combinations (§5.1); ② Compositional gains generalize across diverse model architectures with hybrid methods providing universal benefits (§5.2); ③ Both compositional accuracy and robustness scale with model size (up to the 14B models) (§5.3); ④ A learned <and> composition operator is essential for compositional generalization, and orthogonality regularization proves critical for zero-shot fusion (§5.4); ⑤ Steering tokens and text instructions exhibit complementary strengths, with hybrid methods consistently achieving superior accuracy-robustness tradeoffs across all behaviors (§5.5).

2 Related Work

Behavior steering of LLMs is an active area of NLP research (Li et al., 2025; Xie et al., 2025). We focus on the lines of work we deem most relevant: activation steering, gist tokens, and the few preliminary attempts at compositional steering. Additionally, in Appendix A we discuss how the proposed steering tokens are connected to soft prompt tuning.

Activation Steering. These methods embed the behavior into vectors that are combined with models' activations (i.e., token representations) via simple arithmetic operations. Most activation steering methods are *contrastive*: steering vectors are obtained by comparing activations under exhibited behavior with some type of baseline activations, either in the original activation space (Rimsky et al., 2024; Stolfo et al., 2025; Im and Li, 2025; Sterz et al., 2025; Wu et al., 2025) or a derived one (e.g., PCA or sparse decomposition) (Siddique et al., 2025; Bayat et al., 2025). The most relevant for our work is that of Stolfo et al. (2025), who obtain the activation steers by contrasting LLM's activations for two different prompts: *with* vs. *without* the behavior instruction. We adopt the same two-prompt approach, but distill the instruction into the steering token (appended to the prompt without the behavior instruction) by contrasting the distributions of the two outputs (see §3 and Figure 1). Critically, while the steering vectors of Stolfo et al. (2025) improve instruction following *on top of* prompt steering (i.e., behavior instruction in the prompt), *alone* they are dramatically worse than prompt-based steering. In stark contrast, our steering tokens for individual behaviors are, alone, as effective as instruction steering (see Table 9 in Appendix D).

Gist Tokens. Concerned with test-time compute efficiency, this line of work compresses complex behaviors (i.e., behaviors described by long instructions) into a few input tokens, via various objectives (Mu et al., 2023; Han et al., 2024; Petrov et al., 2025). Mu et al. (2023), for example, train them by modifying the self-attention patterns (gist tokens attend over the behavior description, and answer tokens over gist tokens), whereas the concurrent work of Dong et al. (2025) adopts a self-distillation approach, which is conceptually similar to ours, albeit more complex (see §3). Gist tokens have been used in various use-cases, including for better selection of in-context examples (Gupta et al., 2024), conditioned decoding (Li et al., 2024), hierarchical compression for longer contexts (by gisting the gist tokens) (Petrov et al., 2025; Tarasov et al., 2025), and in task-specific gisting (Phang, 2024; Jiang et al., 2024). Gist approaches, however, tackle only single-instruction steering, whereas we are primarily interested in behavior compositions.

Compositional Steering. While compositional steering remains much less studied than steering for individual behaviors, a few efforts in multi-behavior steering do exist. Most attempts merely interpolate between individual steering vectors to

obtain compositional behavior (Han et al., 2024; Scalena et al., 2024; Cao et al., 2024). Directly composing independently trained modules on top of the same LLM parameters or activations, however, is known to be destructive; this favors sparse changes to the underlying LLM (Ansell et al., 2024; van der Weij et al., 2024). Stolfo et al. (2025) insert different steers in different LLM layers. Nguyen et al. (2025) sparsify individual steering vectors and force their orthogonality, thereby mitigating the negative interference in interpolation-based composition. The orthogonality constraint, however, demands that all behaviors are known in advance; addition of a new behavior requires recomputation of all steering vectors. In contrast, our approach yields compositional generalization: we train a dedicated composition token that learns the very function of composing behaviors and can thus be applied to behaviors unseen in its training.

Critically, these efforts lack rigorous evaluation of multi-behavior steering. Most focus on single-behavior steering and address composition merely as a side experiment, providing only anecdotal evidence that some behavior compositions work (Stolfo et al., 2025; Cao et al., 2024; Han et al., 2024). Scalena et al. (2024) provide a quantitative evaluation but no baseline performance, whereas Nguyen et al. (2025) compare only against in-context learning and fine-tuning approaches (SFT and DPO); both fail to include the simplest, yet very competitive baseline (cf. §5): compositional steering in the prompt (i.e., via instruction). In contrast, we perform a rigorous and extensive empirical evaluation of multi-behavior steering with automatically verifiable behaviors (§4).

3 Compositional Steering with Tokens

We tackle multi-behavior steering, where LLMs need to generate output that simultaneously conforms to multiple constraints. Formally, given a set of behaviors $\mathcal{B} = \{b_1, b_2, \ldots, b_k\}$ (e.g., b_i =*"Answer in one sentence."*; b_j =*"Answer in French."*) and an input prompt x (e.g., x = *"Why penguins can't fly?"*), expect the LLM to produce a high-quality answer y for the prompt x, while satisfying all constraints from $\mathcal{B}$. Standard instruction-based steering would simply concatenate behavior descriptions (e.g., "Answer in French. Use one sentence."), but is known to be brittle (Jaroslawicz et al., 2025). We propose *compositional steering tokens*: input embeddings for compositional steering that do not change models' internals.

Steering Tokens: Input-Level Control. Our steering tokens operate in the input embedding space; we keep the base LM frozen. For each behavior $b \in \mathcal{B}$, we introduce a trainable *steering token* <b>, that is, an embedding $\mathbf{e}_b \in \mathbb{R}^d$, with d as the model's hidden size. Additionally, we add a trainable *composition token* <and>, i.e., a vector $\mathbf{e}_{\texttt{<and>}} \in \mathbb{R}^d$ in which we embed the notion of composition itself. For a prompt x constrained with two behaviors $\{b_i, b_j\}$, we feed the following concatenation of tokens to the LLM: $[\mathbf{E}_x, \mathbf{e}_{b_i}, \mathbf{e}_{\texttt{<and>}}, \mathbf{e}_{b_j}]$, where $\mathbf{E}_x$ denotes the sequence of (frozen) embeddings of x's tokens. This design (i) prevents model collapse, as keeps all LLMs' parameters frozen, (ii) facilitates composition, as we combine behaviors through learned interactions in the input space; and (iii) is computationally efficient: we only need to learn $|\mathcal{B}| + 1$ d-dimensional vectors: one for each behavior and one for the composition token.

Compositional Self-Distillation. A key challenge in multi-behavior control is enabling *compositional generalization*: the ability to combine behaviors at inference time without training on all possible behavior combinations. To this end, our proposal is the compositional operator (i.e., token) <and>, meant to mediate interactions between behavior embeddings. We thus train in two stages: ① we independently train individual behavior tokens <b>, keeping the LLM frozen (left side of Fig. 1), and then ② train the composition token <and> (right side of Fig. 1) on different two-behavior combinations b_i and b_j (unless specified otherwise), keeping both the LLM and the behavior embeddings <b$_\text{i}$> and <b$_\text{j}$> *frozen*. Keeping the behavior tokens frozen in the second step is key for *compositional generalization* as it ensures that <and> learns the behavior-independent concept of *composition*, instead of just modifying individual behavior representations.

We train both the individual behavior tokens <b> and the composition token <and> via self-distillation. For individual behaviors b, the input to the teacher is the prompt x with a behavior's natural language instruction I_b, whereas the student receives x appended with a steering token <b>. The teacher and student are the very same frozen LLM, and they both need to output the same answer y. Our distillation objective minimizes the KL-divergence between the teacher's and student's

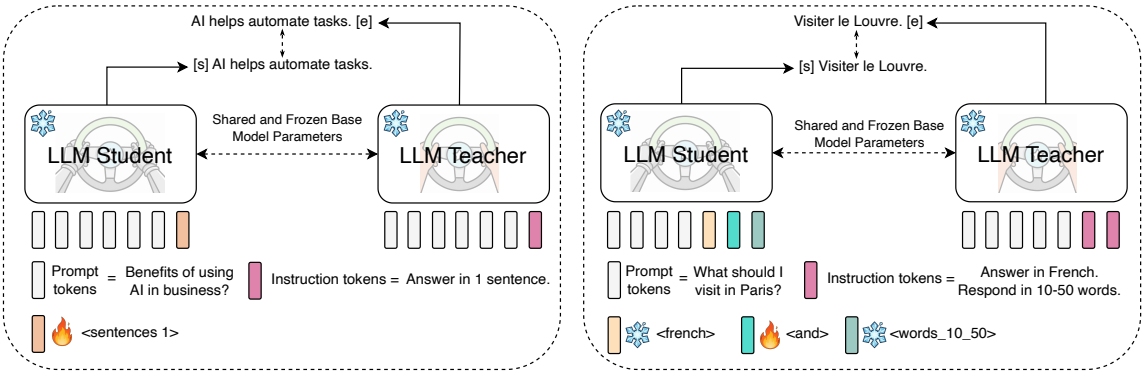


Figure 1: Illustration of our compositional self-distillation. **Left:** ① Training individual behavior steering tokens; **Right:** ② Training the composition token <and>. The LLM remains fully frozen (including its subword embeddings). We *self-distill* from the instruction-prompted LLM to train the respective steering token: *single* behavior tokens in ①, and the <and> token in ②.

output distributions over answer tokens:

$$\mathcal{L}_{\text{dist}} = \text{KL}(P_{\text{teacher}}(y|x, I_b) \| P_{\text{student}}(y|x, \texttt{<b>}))$$

where both distributions are temperature-scaled with T: $P(y) = \text{softmax}(\text{logits}(y)/T)$. We scale the final loss by T^2 to account for temperature normalization (Radevski et al., 2023). In our experiments, we use high temperature ($T = 10.0$) to encourage the student to match the full probability distribution rather than just the mode. To prevent overfitting of behavior tokens to instruction formulations, we define 10 different instruction paraphrases for each behavior (e.g., *"{Answer, Respond, Reply} in Spanish"*) and randomly sample one per training example.

When training the composition token <and>, the teacher receives x concatenated with instructions of two behaviors, $x \oplus I_{b_i} \oplus I_{b_j}$ (e.g., *"Why can't penguins fly? Answer in Spanish. Use 10 to 50 words."*) and the student receives $x \oplus$ <b$_i$><and><b$_j$> (e.g., *"Why can't penguins fly?* <ES><and><words_10_50>*"*.

Initialization and Orthogonality. We initialize embeddings of behavior tokens *"semantically"*: as the mean of the LLM's (frozen) embeddings of tokens in the behavior's instruction. We use zero initialization for the embedding of the composition token (we ablate this choice in §5.4): $\mathbf{e}_{\texttt{<and>}} = \mathbf{0}$, to avoid biasing it toward any one behavior and allowing it to learn "to compose" purely from the training data.

To prevent the <and> token from collapsing into representations similar to existing behavior tokens, we introduce *orthogonality regularization* in composition training (we ablate its impact in §5.4): it forces the trainable <and> embedding to remain orthogonal to *all* frozen behavior embeddings:

$$\mathcal{L}_{\text{orth}} = \sum_{b \in \mathcal{B}_{\text{seen}}} (\mathbf{e}_{\texttt{<and>}} \cdot \mathbf{e}_b / (\|\mathbf{e}_{\texttt{<and>}}\| \cdot \|\mathbf{e}_b\|))^2$$

where $\mathcal{B}_{\text{seen}}$ is the set of all (frozen) behavior tokens *seen* in composition training. The final loss for the <and> token training is then: $\mathcal{L} = \mathcal{L}_{\text{dist}} + \lambda \cdot \mathcal{L}_{\text{orth}}$. We initially set $\lambda = 0.5$ and found it to work well.

4 Experimental Setup

Unlike work on steering for subjective behaviors such as sycophancy or myopic reward (Nguyen et al., 2025; Bayat et al., 2025; Scalena et al., 2024; Pai et al., 2025), we follow (Stolfo et al., 2025) and focus on *verifiable* constraints where satisfaction can be automatically assessed, allowing for a large-scale compositional evaluation while avoiding both human evaluation and costly and error-prone use of LLMs as judges (Marioriyad et al., 2025).

Models. We experiment with seven instruction-tuned LLMs from four model families: Qwen3-4B, Qwen3-8B, Qwen3-14B, Llama-3.2-3B, Llama-3.2-8B, SmolLM3-3B, OLMo-7B, and Gemma3. We use Qwen-8B as the primary model for comparing steering tokens with baselines (§5.1) and the Qwen family models for the scaling analysis (§5.3). For efficiency, we use Qwen3-4B in ablations (§5.4).

Behaviors. We experiment with 15 behaviors from four categories: *languages*: Spanish, French, Italian, Portuguese, and German; *length*: 10-50, 50-70, 70-90, 90-120 words; *formatting*: lowercase, uppercase, title case; and *structure*: 1-5 sentences. To test compositional generalization, we partition

properties into *seen* (11 used during <and> token training) and *unseen* (4 held out: German, title case, 70-90 words, 3 sentences). This design ensures that each category has held-out properties to validate zero-shot composition. We provide instructions for all behaviors in the Appendix I.

Compositions. We evaluate two types of behavior combinations: (i) **Seen compositions**: both behaviors are from the seen set, i.e., their combination was part of the <and> token training (e.g., <ES> <and> <words_10_50>); here we test if the composition operator successfully learned to compose seen behaviors; (ii) **Unseen compositions**: one or both behaviors are from the *unseen* behavior set (e.g., <DE> <and> <words_10_50>, <ES> <and> <title_case> or <DE> <and> <title_case>: this setup tests the generalization ability of our composition token operator.[2]

In both cases, we test on 2-behavior and 3-behavior compositions. Crucially, 3-behavior compositions are *all unseen*, since we train the <and> only on 2-behavior combinations, providing a different test of generalization—to compositions with different *number* of behaviors. For the Qwen family, we contrast our default training of the composition token <and> exclusively on 2-behavior compositions (**2-token-only**), against training on both 2-behavior and 3-behavior compositions (**2+3-token**): this way, we evaluate whether explicit training on 3-behavior compositions improves generalization or causes overfitting.

Data. We source prompts from the Smoltalk dataset[3] (Allal et al., 2025) which contains instruction-labeled dialogs. We use Qwen3-30B-A3B-Instruct to separate the core question for each prompt from constraints (e.g., *"respond in 5 sentences,"*). For each of our 15 behaviors, we randomly sample 50k prompts and generate answers with Qwen3-30B-A3B-Instruct (for each example, we randomly sample from 10 behavior paraphrases). For the <and> token training, we generate responses for all cross-category 2-behavior combinations (e.g., *language* + *length*, *language* + *format*). We exclude all unseen behaviors (German, title case, 70-90 words, 3 sentences) from the training data creation and save them for evaluation of *unseen* compositions (see App. B for further details). For testing, we use 1,000 held-out prompts for each 2- and 3-behavior combination, resulting in >1M evaluations per model across all compositions and orderings of behavior tokens.

Metrics. We measure the following: (1) *Mean accuracy*, as the percentage of LLM generations that satisfy all $k \in 2, 3$ behaviors, averaged across all $k!$ token orders: this way, we remove any potential model bias w.r.t. the order of behavior tokens; (2) *Order variance* is the largest absolute difference in accuracy across any two token orders of a composition, averaged across all compositions. Lower values indicate more robust, order-invariant behavior; and (3) *Response quality*: for *accurate* responses (i.e., satisfying all behaviors) constraints, we evaluate semantic correctness and coherence of the answers on a Likert (1-5) scale, using an LLM judge (Qwen3-30B-A3B-Instruct). We measure whether steering accuracy comes at the expense of degraded content (see details in Appendix. F).

Baselines and Variants. We compare our steering tokens against four baselines that instantiate different types of steering. **(1)** In practice, *instruction steering* is the default paradigm for multi-behavior steering of LLMs; despite that, existing work on compositional steering (Cao et al., 2024; Han et al., 2024; Nguyen et al., 2025), with the exception of (Stolfo et al., 2025), fails to evaluate this competitive baseline. We simply append the behavior instructions I_b to the prompt (e.g., *"Answer in German. Use 70-90 words. Apply title case."*). To ensure fair comparison, we randomly sample from 10 behavior paraphrases. **(2)** Standard activation steering as per Rimsky et al. (2024)[4] **(3)** Fine-tuning for behavior alignment is an expensive but effective steering approach. We first train behavior-specific low-rank adapters with the same self-distillation objective, then merge individual adapters using an interference-reducing approach of Yu et al. (2024) (*LoRA DARE*); this is a meaningful parameter-space alternative to our input-space steering. **(4)** LM-Steer (Han et al., 2024) is an established approach with steering vectors obtained as linear projections of LLMs' word embeddings: as such, it is essentially a type of token-based steering; moreover, the authors claim (but do not quan-

[2]The seen/unseen split is defined relative to the <and> token's training data; instruction steering inherits it for comparability. Any gap there reflects genuine difficulty variation: e.g., German paired with `title_case` conflicts with German's capitalization of all nouns.

[3]The dataset is released under Apache 2.0 License.

[4]Stolfo et al. (2025) empirically shows that the steering vectors by Rimsky et al. (2024), when applied in a standalone fashion clearly and substantially trail instruction-based steering for *verifiable* behaviors.

Method	2-Behavior Composition				3-Behavior Composition			
	Seen ↑	Unseen ↑	Ord. Var. ↓	Resp. Qual. ↑	Seen ↑	Unseen ↑	Ord. Var. ↓	Resp. Qual. ↑
Baselines								
CAA (Rimsky et al., 2024)	1.6 ±0.1	0.5 ±0.1	–	1.1	0.6 ±0.1	0.1 ±0.1	–	1.1
LM-Steer (Han et al., 2024)	18.1 ±0.1	13.4 ±0.3	–	1.3	2.2 ±0.1	2.1 ±0.2	–	1.2
LoRA DARE (Yu et al., 2024)	81.5 ±0.1	44.8 ±0.2	–	4.7	58.4 ±0.1	17.6 ±0.1	–	4.6
Instruction Steering (Stolfo et al., 2025)	90.7 ±0.1	71.8 ±0.2	7.8	4.9	83.7 ±0.1	54.0 ±0.1	18.1	4.9
Steering Tokens								
Concatenation	81.3 ±0.1	62.1 ±0.2	25.3	4.8	59.6 ±0.1	33.2 ±0.1	55.8	4.8
Composition (native "and" token)	82.9 ±0.1	66.4 ±0.2	12.9	4.8	68.8 ±0.1	47.1 ±0.2	40.7	4.8
Composition (<and>)	90.9 ±0.1	**76.9** ±0.2	5.3	4.9	83.1 ±0.1	59.5 ±0.1	25.5	4.9
Hybrid: Composition (<and>) + Instr.	**92.2** ±0.1	76.3 ±0.2	**4.4**	4.9	**87.9** ±0.1	**62.9** ±0.1	**15.2**	4.9

Table 1: Results for compositional steering with Qwen3-8B for *seen* and *unseen* 2- and 3-behavior combinations. The learned <and> composition operator outperforms text instructions on unseen compositions; Combining our token steering with instructions (Hybrid) yields the best performance; Concatenation without explicit composition learning fails on complex compositions; LoRA DARE (Yu et al., 2024) shows poor compositional generalization, whereas LM-Steer (Han et al., 2024) and and CAA (Rimsky et al., 2024) fails to compose altogether.

tify) LM-Steer's compositional steering abilities. Check Appendix H for implementation details of the baselines.

We evaluate four variants of the steering token approach: **(1)** a simple *concatenation* of behavior tokens (i.e., no composition token <*and*>), shedding light on the necessity of explicit composition learning; **(2)** using the native "and" LLM token embedding as a composition token, in order to understand whether the trained <*and*> operator encodes something qualitatively different from the word "and"; **(3)** the default <*and*> composition; and **(4)** we combine our token steering with instructions (*Hybrid*), testing the complementarity of learned vectors and natural language guidance.

5 Results and Analyses

5.1 Steering tokens are superior compositional generalizers

Table 1 compares our compositional steering against the baselines—instructions, merging LoRA adapters (Da Silva et al., 2025) and output steering (Han et al., 2024)—for 2- and 3-behavior compositions, with Qwen3-8B as the LLM. We report performance for *seen* (during <and> training) and *unseen* behavior compositions, with the latter indicating compositional generalization.

We find that training an explicit composition operator (<and> token) is crucial: simple concatenation of behavior tokens or using the native "and" token as a compositional operator collapses on unseen 3-behavior composition (33.2% and 47.1% vs. 59.6%) and exhibits larger order variance. Our full compositional steering outperforms text instructions on *unseen* compositions (+5.1% for 2-behavior compositions, +5.5% for 3-behavior compositions), while performing comparably on *seen* compositions. This implies that our learned composition operator generalizes better than natural language composition. Text instructions are more order-robust, exhibiting lower order variance, despite lower accuracy. Importantly, Hybrid steering (tokens + instructions) achieves the best overall results (62.9% accuracy, 15.2% variance for the most difficult generalization case: *unseen* 3-behavior compositions. This encouraging result suggests complementarity between our compositional token steering and natural language guidance. LoRA DARE fails to generalize compositionally, performing much worse on *unseen* compositions, while CAA and LM-Steer completely collapse.[5] Response quality remains solid and is largely comparable across steering methods; except for CAA and LM-Steer, which exhibit broken model outputs.

5.2 Robustness across model families

Table 2 summarizes the compositional steering performance for steering tokens, instructions, and their Hybrid across seven models (Qwen3-4B/8B, Llama3-3B/8B, SmolLM3-3B, OLMo-7B, Gemma3-4B) and five model families. The results render our findings from §5.1 to be robust across different LLM architectures: (1) our steering tokens largely outperform instruction-steering (and when not, they offer on par performance); (2) the Hybrid combination of tokens and instructions offers further gains. There is, however, some family-

[5] LM-Steer applies a linear transformation to output word embeddings and is designed for soft distributional preferences (e.g., sentiment); it is a poor fit for our hard constraints (exact word counts, strict language, character-level formatting).

Model	Method	Unseen (2|3) ↑	Order Var. (2|3) ↓
Qwen3-4B	Instruction	68.9 | 55.6	5.3 | **15.1**
	Steering	69.1 | 60.7	6.2 | 22.7
	Hybrid	**69.2** (+0.3) | 58.0 (+2.4)	**5.3** (+0.0) | 18.6 (+3.5)
Qwen3-8B	Instruction	71.8 | 54.0	5.1 | 15.1
	Steering	**76.9** | 59.5	**4.1** | 21.2
	Hybrid	76.3 (+4.5) | **62.9** (+8.9)	4.4 (−0.7) | **12.4** (−2.7)
Llama3-3B	Instruction	66.7 | 33.8	3.1 | **8.7**
	Steering	69.3 | 33.9	3.9 | 18.7
	Hybrid	**74.9** (+8.2) | **43.4** (+9.6)	**2.8** (−0.3) | 10.6 (+1.9)
Llama3-8B	Instruction	67.8 | 40.2	3.6 | 13.6
	Steering	67.0 | 39.5	5.9 | 19.2
	Hybrid	**76.3** (+8.5) | **52.9** (+12.7)	**3.2** (−0.4) | **11.0** (−2.6)
Smol3-3B	Instruction	53.2 | 32.5	**5.1** | **14.5**
	Steering	53.2 | 35.5	11.8 | 35.9
	Hybrid	**53.5** (+0.3) | **37.2** (+4.7)	7.3 (+2.2) | 17.1 (+2.6)
Olmo-7B	Instruction	56.8 | 30.9	**2.9** | 7.3
	Steering	56.9 | 28.4	3.6 | 12.3
	Hybrid	**60.9** (+4.1) | **37.5** (+6.6)	3.7 (+0.8) | **6.4** (−0.9)
Gemma3-4B	Instruction	39.9 | 21.1	**3.4** | **6.1**
	Steering	37.4 | 18.8	4.9 | 14.0
	Hybrid	**49.5** (+9.6) | **26.7** (+5.6)	6.5 (+3.1) | 8.2 (+2.2)

Table 2: Cross-architecture generalization of steering tokens, text instructions, and hybrid methods across seven models spanning four architectural families. Hybrid methods consistently achieve the best performance across all architectures, with strong benefits on weaker compositional models (Llama, OLMo). The compositional advantage holds universally but shows architecture-dependent magnitudes. Numbers in brackets represent improvement on top "Instruction" baseline. See App. D for performance on seen categories.

based variance: Qwen models are more steerable, and favor compositional steering tokens over instruction steering (76.9% for unseen 2-behavior on Qwen3-8B); Llama models, in contrast, are much less steerable, regardless of the steering approach (Llama3-8B: 39.5% token steering, 40.2% instructions). Importantly, the Hybrid combination of token-based compositional steering and natural language instructions rescues weak models: we observe a +12.7% gain for Llama3-8B for 3-behavior compositions. This again points to strong complementarity between our compositional token steering and natural language instructions.

5.3 Robustness across model sizes

We next investigate whether compositional accuracy and robustness of our compositional steering tokens scale with model size, as well as whether training on both 2- and 3-behavior compositions improves generalization (compared to our default training on 2-behavior compositions only). Table 3 summarizes the results. We observe that both compositional token steering and instruction-based steering benefit from scale. Performance on 3-behavior compositions for compositional steering tokens (training on 2-behavior compositions only) jumps from 59.5% at 8B to 68.0% at 14B (+8.5%), while instructions improve from 52.1% to 61.4% (+9.3%). The Hybrid approach also scales effectively, achieving 69.2% at 14B and becomes much more robust to behavior ordering, exhibiting only 6.2% order variance (compared to 18.6% at 4B). Somewhat surprisingly, explicitly training our composition token also for 3-behavior compositions degrades performance at 14B: (2-behavior +3-behavior training yields only 63.9% accuracy, compared to 68.0% we get when training only on 2-behavior compositions; and it also exhibits higher order variance (17.3% vs. 13.9%). This again suggests that we successfully learned a general composition operator (i.e., already from the 2-behavior compositions). At 8B, 2- + 3-behavior training provides marginal variance reduction (compared to 4B; from 21.2% to 15.5%) with comparable accuracy, indicating that training data efficiency is likely scale-dependent. Note that, in Appendix C we perform inference using Qwen3-14B on 4-behavior compositions where the 4-th behavior is *json wrap* of the LLM outputs. We find that at 4-behavior compositions, pure steering becomes order-unstable and hybrid is the regime that scales.

5.4 Ablation studies: Effect of <and> token initialization and λ weight

In Table 4, we ablate initialization strategies and orthogonality regularization, to isolate what mechanisms enable compositional learning in the <and> token. We report results using Qwen3-4B.

We first observe that <and> is essential: without it, *seen* performance degrades to 73.6% (vs. 93-95% with <and>), and *unseen* to 49.7% (vs. 64-71%), and order variance increases to 27.0% (vs. 5-11%). All <and> initialization strategies achieve comparable *seen* performance (93-95%), suggesting that behavior learning does not depend on initialization. However, *unseen* performance varies widely (55-71%), isolating compositional generalization as the key differentiator. Orthogonality regularization is critical here, especially for semantic behavior initialization: "and" embedding without orthogonality achieves only 55.2% *unseen*, but with the orthogonality constraint it improves by 15.6%. For zero initialization, orthogonality primarily reduces variance from 11.2% to 5.3% while only modestly improving accuracy. Token-average initialization benefits least from orthogonality, suggesting that averaging of behavior em-

Model	Method	2-Behavior Composition				3-Behavior Composition			
		Seen ↑	Unseen ↑	Ord. Var. ↓	Resp. Qual. ↑	Seen ↑	Unseen ↑	Ord. Var. ↓	Resp. Qual. ↑
Qwen3-4B	Instruction	93.3	68.9	**5.3** (+0.0)	4.8	88.2	55.6	**15.1** (+0.0)	4.8
	Steering (2-only)	93.7	69.1	6.2	4.8	89.3	**60.7** (+5.1)	22.7	4.8
	Hybrid	**93.7** (+0.4)	**69.2** (+0.3)	**5.3** (+0.0)	4.8	**90.7** (+2.5)	58.0	18.6	4.8
Qwen3-8B	Instruction	91.0	71.6	5.1	4.9	82.9	52.1	15.1	4.9
	Steering (2-only)	90.9	**76.9** (+5.3)	**4.1** (−1.0)	4.9	83.1	59.5	21.2	4.9
	Steering (2+3)	91.2	77.0	4.2	4.9	85.9	59.7	15.5	4.9
	Hybrid	**92.2** (+1.2)	76.3	4.4	4.9	**87.9** (+5.0)	**62.9** (+10.8)	**12.4** (−2.7)	4.9
Qwen3-14B	Instruction	92.1	72.2	5.2	4.9	88.2	61.4	11.2	4.9
	Steering (2-only)	92.9	75.2	4.6	4.9	90.4	68.0	13.9	4.9
	Steering (2+3)	93.0	73.8	5.3	4.9	89.7	63.9	17.3	4.9
	Hybrid	**93.9** (+1.8)	**78.3** (+6.1)	**2.7** (−2.5)	4.9	**91.7** (+3.5)	**69.2** (+7.8)	**6.2** (−5.0)	4.9

Table 3: Scaling analysis comparing steering tokens, text instructions, and their hybrid combination. Both steering and instruction methods benefit from scale, while training on 2-behavior combinations proves sufficient: explicit 3-behavior supervision (2+3) helps variance at 8B but degrades performance at 14B, suggesting larger models learn compositional patterns from simpler examples. Hybrid methods achieve the best accuracy-variance tradeoff at all scales. Numbers in brackets—improvement on top "Instruction" baseline. See App. D for extended analysis.

<and> init.	$\mathcal{L}_{\text{orth}}$	Seen ↑	Unseen ↑	Ord. Var. ↓
No <and> token	–	73.6	49.7	27.0
Zero vector	✗	**94.5**	66.9	11.2
Zero vector	✓	93.7 (−0.8)	**69.1** (+2.2)	**6.2** (−5.0)
"and" embedding	✗	**94.2**	55.2	9.4
"and" embedding	✓	**94.2** (+0.0)	**70.8** (+15.6)	**7.1** (−2.3)
Avg. steering tokens	✗	94.3	58.4	8.6
Avg. steering tokens	✓	**93.5** (−0.8)	**64.4** (+6.0)	**6.2** (−2.4)

Table 4: Ablation study of <and> token initialization and orthogonality regularization (Qwen3-4B; 2-behavior composition). The initializations effect matters only on unseen combinations, isolating compositional generalization as the key differentiator. Orthogonality loss proves critical, while without the <and> token, performance collapses. Zero initialization with orthogonality provides the best accuracy-variance tradeoff.

λ	Unseen(2|3) ↑	Ord. Var. (2|3) ↓
0.0	67.6 | 59.8	5.3 | 21.7
0.25	65.0 | 53.4	5.1 | **21.1**
0.5 (default)	**69.1** | **60.7**	6.2 | 22.7
1.0	63.9 | 51.3	**5.0** | 22.3

Table 5: Ablation study of the orthogonality loss weight (λ) using Qwen3-4B on 2|3 unseen behavior composition. $\lambda = 0.5$ performs best and $\lambda = 1.0$ worst. Order variance shows no clear trend. Overall, the method does not seem to be overly sensitive to the choice of λ.

beddings provides a poor starting point for composition. These results stress the importance of (i) an explicit learned operator (the <and> token), and (ii) enforcing the orthogonality of its embedding to behavior token representations.

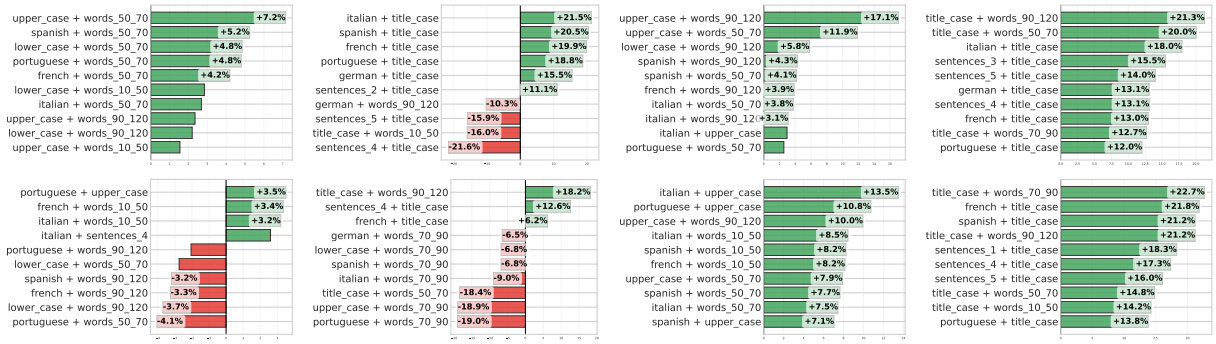
We further conduct a λ ablation on Qwen3-4B (zero initialization, 2-behavior training). We report results in Table 5. We observe that $\lambda = 0.5$ performs best (69.1% 2-tok, 60.7% 3-tok) and $\lambda = 1.0$ worst (63.9%, 51.3%). Order variance shows no clear trend across values of λ. We conclude that the method does not seem to be overly sensitive to the choice of this hyperparameter. Finally, see Appendix E for an interpretability analysis of the <and> token.

5.5 Per-behavior breakdown

Figure 2 presents a granular per-composition analysis for Qwen3-14B and Llama3-8B, revealing which behaviors benefit most from compositional token steering vs. text instructions. We find that unseen behavior combinations, and in particular `title_case` paired with language or length constraints, drive the gains of token steering. This indicates that unseen *cross-category* compositions benefit the most from learned <and> operators. However, LLM-dependent failure modes emerge: while Qwen3-14B shows better performance with token steering for most unseen combinations, Llama3-8B exhibits stark category-specific divergence, with token steering excelling on compositions with formatting behaviors but failing on compositions with the *unseen* length behavior (`words_70_90`), where the instruction steering seems superior. Critically, the hybrid combination seems to eliminate these failure modes, reaffirming that complementarity between steering with learned embeddings and text instructions is key for robust compositional control, especially for weaker LLMs.

6 Conclusion

In this work, we presented *steering tokens*, an effective approach for compositional control of LLMs,



(a) Seen – Steering vs Text (b) Unseen – Steering vs Text (c) Seen – Hybrid vs Text (d) Unseen – Hybrid vs Text

Figure 2: Average relative performance gains (%) for Qwen3-14B (top row) and Llama3-8B (bottom row). Top 20 behavior combinations (2- and 3-behavior) yielding the largest differences between methods. Green bars indicate improvements over the text baseline; red bars indicate degradations.

that does not require modification of the models' internals. By means of self-distillation, in the first step we train input embeddings for individual (automatically verifiable) model behaviors, which we call steering tokens. In the second step, we freeze the behavior-specific steering tokens and train only the explicit composition token <and>. Our extensive experimentation renders steering tokens (1) superior to instruction-based steering as well as to other steering methods, while maintaining comparable response quality. Further, we show that the steering tokens are (2) complementary to instructions, obtaining further gains from combination with natural language-based guidance. Crucially, (3) we demonstrate that steering tokens successfully generalize to behaviors and compositions *unseen* during the training of our composition operator (i.e., the <*and*> token).

Finally, we show that our compositional steering tokens generalize well across different model families and sizes, provide even larger steering gains for larger models. Importantly, training exclusively on 2-behavior compositions proves sufficient for larger models. This work introduces steering tokens as a solid mechanism for compositional control in LLMs, offering an efficient alternative to instruction-based steering while complementing it.

Limitations

While steering tokens demonstrate strong compositional capabilities, several limitations warrant attention in future work:

Constraint verifiability. Our evaluation focuses exclusively on verifiable behavioral properties (response length, formatting conventions, etc.) where ground-truth satisfaction is automatically assessed. Steering tokens represent a general control mechanism that applies to broader semantic constraints (tone, style, etc.), but evaluating such properties requires human annotation or LLM-based judges. Future work should extend compositional evaluation to subjective and nuanced behavioral dimensions.

Compositional complexity. We evaluate compositions of up to three properties, demonstrating zero-shot generalization from 2-property training to 3-property inference. Real-world applications may require simultaneous control over many more constraints (e.g., language, length, tone, domain, etc.). Therefore, scaling steering tokens to higher-order compositions remains an open question, particularly regarding whether compositional accuracy degrades gracefully or rapidly as the property count increases.

Model scale. Our experiments span 3B to 14B parameter models, observing compositional improvements with scale. However, frontier models now exceed those sizes, and it remains unclear whether steering tokens continue to benefit from scale or encounter diminishing returns. Extension to larger models would clarify whether compositional reasoning improves further and whether hybrid methods remain necessary at scale.

Acknowledgements

We thank Andreas Ripke for his support with hosting the LLMs used in our experiments. The work of Goran Glavaš was supported by the Alcatel-Lucent Stiftung and Deutsches Stiftungszentrum through the grant "Equitably Fair and Trustworthy Language Technology" (EQUIFAIR, Grant Nr. T0067/43110/23).

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, and 1 others. 2025. SmolLM2: When Smol Goes Big–Data-Centric Training of a Small Language Model. In *Conference on Language Modeling (COLM)*.

Alan Ansell, Ivan Vulić, Hannah Sterz, Anna Korhonen, and Edoardo M Ponti. 2024. Scaling Sparse Fine-Tuning to Large Language Models. *arXiv preprint arXiv:2401.16405*.

Reza Bayat, Ali Rahimi-Kalahroudi, Mohammad Pezeshki, Sarath Chandar, and Pascal Vincent. 2025. Steering Large Language Model Activations in Sparse Spaces. In *Conference on Language Modeling (COLM)*.

Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. 2024. Personalized Steering of Large Language Models: Versatile Steering Vectors Through Bi-Directional Preference Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:49519–49551.

Patrick Queiroz Da Silva, Hari Sethuraman, Dheeraj Rajagopal, Hannaneh Hajishirzi, and Sachin Kumar. 2025. Steering off Course: Reliability Challenges in Steering Language Models. In *Association for Computational Linguistics (ACL)*, page 19856–19882.

Jiancheng Dong, Pengyue Jia, Jingyu Peng, Maolin Wang, Yuhao Wang, Lixin Su, Xin Sun, Shuaiqiang Wang, Dawei Yin, and Xiangyu Zhao. 2025. Behavior-Equivalent Token: Single-Token Replacement for Long Prompts in LLMs. *arXiv preprint arXiv:2511.23271*.

Federico Errica, Davide Sanvito, Giuseppe Siracusano, and Roberto Bifulco. 2025. What Did i Do Wrong? Quantifying LLMs' Sensitivity and Consistency to Prompt Engineering. In *Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1543–1558.

Shivanshu Gupta, Clemens Rosenbaum, and Ethan R Elenberg. 2024. GistScore: Learning Better Representations for In-Context Example Selection with Gist Bottlenecks. In *Workshop on Mathematical and Empirical Understanding of Foundation Models (ME-FoMo@ICLR)*.

Chi Han, Jialiang Xu, Manling Li, Yi Fung, Chenkai Sun, Nan Jiang, Tarek Abdelzaher, and Heng Ji. 2024. Word embeddings are steers for language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16410–16430.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*.

Shawn Im and Yixuan Li. 2025. A Unified Understanding and Evaluation of Steering Methods. *arXiv preprint arXiv:2502.02716*.

Daniel Jaroslawicz, Brendan Whiting, Parth Shah, and Karime Maamari. 2025. How Many Instructions Can LLMs Follow at Once? In *Workshop: Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling@NeurIPS*.

Yichen Jiang, Marco Vecchio, Mohit Bansal, and Anders Johannsen. 2024. Hierarchical and Dynamic Prompt Compression for Efficient Zero-Shot API Usage. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2162–2174.

Mehdi Khamassi, Marceau Nahon, and Raja Chatila. 2024. Strong and Weak Alignment of Large Language Models with Human Values. *Scientific Reports*, 14(1):19399.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 3045–3059.

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *arXiv preprint arXiv:2101.00190*.

Xinze Li, Zhenghao Liu, Chenyan Xiong, Shi Yu, Yukun Yan, Shuo Wang, and Ge Yu. 2024. Say More with Less: Understanding Prompt Learning Behaviors through Gist Compression. *arXiv preprint arXiv:2402.16058*.

Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. 2025. Prompt Compression for Large Language Models: A Survey. In *Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 7182–7195.

Seungwon Lim, Seungbeen Lee, Dongjun Min, and Youngjae Yu. 2025. Persona Dynamics: Unveiling the Impact of Personality Traits on Agents in Text-Based Games. In *Association for Computational Linguistics (ACL)*, page 31360–31394.

Arash Marioriyad, Mohammad Hossein Rohban, and Mahdieh Soleymani Baghshah. 2025. The Silent Judge: Unacknowledged Shortcut Bias in LLM-as-a-Judge. In *Workshop on Reliable ML from Unreliable Data@NeurIPS*.

Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to Compress Prompts with Gist Tokens. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:19327–19352.

Duy Nguyen, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2025. Multi-Attribute Steering of Language Models via Targeted Intervention. In *Association for Computational Linguistics (ACL)*, page 20619–20634.

Lilian Ngweta, Kiran Kate, Jason Tsay, and Yara Rizk. 2025. Towards LLMs Robustness to Changes in Prompt Format Styles. In *Student Research Workshop @ Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 529–537.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training Language Models to Follow Instructions with Human Feedback. *Neural Information Processing Systems (NeurIPS)*, 35:27730–27744.

Tsung-Min Pai, Jui-I Wang, Li-Chun Lu, Shao-Hua Sun, Hung-Yi Lee, and Kai-Wei Chang. 2025. BILLY: Steering Large Language Models via Merging Persona Vectors. *arXiv preprint arXiv:2510.10157*.

Aleksandar Petrov, Mark Sandler, Andrey Zhmoginov, Nolan Miller, and Max Vladymyrov. 2025. Long Context In-Context Compression by Getting to the Gist of Gisting. *arXiv preprint arXiv:2504.08934*.

Jason Phang. 2024. Investigating the Effectiveness of Hypertuning via Gisting. *arXiv preprint arXiv:2402.16817*.

Gorjan Radevski, Dusan Grujicic, Matthew Blaschko, Marie-Francine Moens, and Tinne Tuytelaars. 2023. Multimodal Distillation for Egocentric Action Recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5213–5224.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering Llama 2 via Contrastive Activation Addition. In *Association for Computational Linguistics (ACL)*, pages 15504–15522.

Daniel Scalena, Gabriele Sarti, and Malvina Nissim. 2024. Multi-Property Steering of Large Language Models with Dynamic Activation Composition. In *BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP @ ACL*, pages 577–603.

Zara Siddique, Irtaza Khalid, Liam D Turner, and Luis Espinosa-Anke. 2025. Shifting Perspectives: Steering Vector Ensembles for Robust Bias Mitigation in LLMs. *arXiv preprint arXiv:2503.05371*.

Taylor Sorensen, Jared Moore, Jillian Fisher, Mitchell Gordon, Niloofar Mireshghallah, Christopher Michael Rytting, Andre Ye, Liwei Jiang, Ximing Lu, Nouha Dziri, and 1 others. 2024. A Roadmap to Pluralistic Alignment. *arXiv preprint arXiv:2402.05070*.

Hannah Sterz, Fabian David Schmidt, Goran Glavaš, and Ivan Vulić. 2025. ReCoVeR the Target Language: Language Steering without Sacrificing Task Performance. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 19390–19405.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2025. Improving Instruction-Following in Language Models through Activation Steering. In *International Conference on Learning Representations (ICLR)*.

Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. 2024. Democratizing Large Language Models via Personalized Parameter-Efficient Fine-Tuning. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 6476–6491.

Dmitrii Tarasov, Elizaveta Goncharova, and Kuznetsov Andrey. 2025. Sentence-Anchored Gist Compression for Long-Context LLMs. *arXiv preprint arXiv:2511.08128*.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. 2024. Fine-Tuning Language Models for Factuality. In *International Conference on Learning Representations (ICLR)*.

Teun van der Weij, Massimo Poesio, and Nandi Schoots. 2024. Extending Activation Steering to Broad Skills and Multiple Behaviours. *arXiv preprint arXiv:2403.05767*.

Zhengxuan Wu, Qinan Yu, Aryaman Arora, Christopher D Manning, and Christopher Potts. 2025. Improved representation steering for language models. *arXiv preprint arXiv:2505.20809*.

Zhouhang Xie, Junda Wu, Yiran Shen, Yu Xia, Xintong Li, Aaron Chang, Ryan Rossi, Sachin Kumar, Bodhisattwa Prasad Majumder, Jingbo Shang, and 1 others. 2025. A Survey on Personalized and Pluralistic Preference Alignment in Large Language Models. In *Conference on Language Modeling (COLM)*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. In *International Conference on Machine Learning (ICML)*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Guoyin Wang, and Fei Wu. 2025. Instruction Tuning for Large Language Models: A Survey. *ACM Computing Surveys*.

Supplementary Matrial

We organize the supplementary material as follows:

- §A: Discussion how our method is related to soft prompt tuning (Li and Liang, 2021).
- §B: All implementation details in order to ensure reproducibility of our work.
- §C: Experiments featuring 4-behavior composition.
- §E: Interpretability analysis of the steering tokens.
- §D: Additional experiments that we did not include in the main paper.
- §F: The LLM-as-judge prompts we use to measure the response quality of the outputs.
- §G: Additional details about the evaluation metrics.
- §H: Implementation details for the baselines.
- §I: The instructions we use for each steering behavior.
- §J: Model outputs for different steering combinations.

A "Behavior Steering" vs. Soft/Postfix Tuning

Our method is related to soft prompt tuning (Lester et al., 2021; Li and Liang, 2021), and since our tokens are appended after the prompt, "postfix tuning" is an accurate description of the mechanism. There is, however, one crucial difference w.r.t. standard soft prompting: our interventions are strictly limited to the input (i.e., the addition of steering tokens), whereas soft prompting (prefix or postfix tuning), when used as a parameter-efficient fine-tuning approach (PEFT), typically introduces trainable parameters in each transformer layer – and usually significantly more than a single trainable embedding vector per fine-tuning task. It is precisely the fact that we manipulate only the input space that allows for compositional generalization, i.e., to steer the model ad-hoc at inference for behavior combinations not seen in training; this is not possible for PEFT approaches that modify model internals (as shown in Table 1 for the LoRA-based baseline).

However, the key distinction is not the training mechanism, but rather the compositional generalization objective. Soft prompt tuning optimizes a fixed embedding for a single task. In contrast, we introduce a dedicated composition operator (the <and> token) trained on behavior pairs, which then generalizes zero-shot to: (i) unseen behavior combinations, (ii) combinations involving behaviors never seen during <and> training, and (iii) compositions with some behaviors never seen at training (3-behavior and 4-behavior compositions). This generalization across behavior combinations is neither demonstrated by nor the goal of prior soft prompt tuning methods.

We adopt the term "steering" in the sense established by the behavior steering literature (controlling specific output properties), which is orthogonal to the (input) placement of the steering tokens. Our key novelty is not the training mechanism (input-space optimization with a frozen LLM) but the compositional generalization enabled by the learned <and> operator—a property not demonstrated by prior work. Finally, we note that our method requires LLM gradients during training (similar to other steering methods, e.g., LM-Steer (Han et al., 2024)), unlike activation-steering methods that compute steering vectors as aggregates of activations.

B Implementation Details

We train all steering tokens using AdamW with learning rate $\alpha = 1 \times 10^{-4}$, weight decay $\lambda = 1 \times 10^{-3}$, batch size as large as we can fit on a GPU, and 2 epochs over 50,000 training examples. Learning rate follows linear warmup (10% of steps) then linear decay to 0. Gradients are clipped to a maximum norm of 1.0. We set the temperature to 10.0. We use bfloat16 mixed precision training. Training data is preprocessed to remove the top 0.1% longest examples.

C 4-way Behavior Composition

We train a new 4th behavior steering token, in order to test for 4-way compositions of behaviours via the <and> token, which remains trained on 2-behavior compositon only. We denote the behavior as *json*, where we steer the model towards having the answer in JSON format, where the key "answer" is mandatory. We evaluate for this by checking if (i) the JSON can be parsed via `json.loads` in Python; (ii) the JSON object contains the "answer" as key;

Method	Acc. (3|4 Behavior) ↑	Ord. Var. (3|4) ↓
Instr. steering (Stolfo et al., 2025)	70.6 | 64.4	13.7 | **19.5**
Composition (<and>)	76.3 | 53.5	14.1 | 60.0
Hybrid	**78.0** | **67.5**	**7.0** | 25.5

Table 6: 4-behavior composition using <and> steering token trained with only 2-behavior data.

and (iii) has only one key. We train this behavior for Qwen3-14B, and pair it with the *languages*, *formatting*, *length*, and *structure*, and report the results in Table 6.

We observe that the <and> token does generalize to 4-behavior compositions – non-trivially, since it has never seen $2 \rightarrow 4$ in training. Pure steering's order variance increases substantially at 4-behaviors (60.0%), while the hybrid method keeps it more contained (25.5%), though instruction steering remains the most order-robust (19.5%). We consider the fact that the <and> token generalizes across behaviors ($2 \rightarrow 3 \rightarrow 4$) with a monotonic but contained drop is strong evidence of a genuine compositional operator rather than a single $2 \rightarrow 3$ jump.

D Additional Results

We report supplementary results to the ones in the main paper in Table 7, Table 8, and Table 9.

E Interpretability of the Steering Tokens

We have conducted two interpretability analyses to analyze the meaning of the steering tokens.

Nearest vocabulary neighbors. We computed the top-5 nearest vocabulary tokens by cosine similarity for each learned steering token (Qwen3-8B and Llama3-8B) and observed the following: **Language tokens:** nearest neighbor is the corresponding language name (e.g., embedding of "French" is the closest to the embedding of the steering token <french> at 0.44–0.52 cosine sim). **Length tokens:** nearest neighbors are digit tokens matching the numeric boundaries (e.g., "50", "70"). **<and> token:** nearest neighbors have cosine similarity $\sim 0.09 - 0.10$ on both models—effectively near-random. The learned composition embedding does not align with any interpretable vocabulary region. It is placed in a new, distinct region of the model's embedding space. This is expected, given our design: behavior tokens are initialized as the mean of the word embeddings of their corresponding instruction, so they retain proximity to semantically related vocabulary. The <and> token, by contrast, is initialized to the zero vector and explicitly trained to be orthogonal to all seen behavior tokens—so its substantial distance from interpretable vocabulary is expected, not incidental.

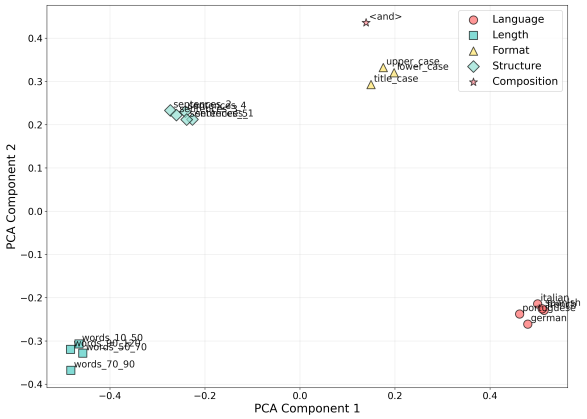


Figure 3: 2D PCA projection of the Qwen3-14B steering token embeddings.

PCA clustering. We project all learned steering tokens of Qwen3-14B into 2D, behavior tokens cluster tightly by category (language, length, format, structure). We observe that the <and> token is a clear outlier, isolated from all behavior token clusters. See Figure 3.

Together, these results support the claim that <and> learns a structurally distinct function of composing (arbitrary) behavior, whereas the individual behavior tokens essentially compress the semantics of the corresponding instructions. <and> token's embedding is not a rephrasing of any single behavior or of the vocabulary token "and."

F LLM-judge for Response Quality Evaluation

System Prompt

You are an expert evaluator of AI-generated responses. Your task is to assess the quality of a model's answer to a question, taking into account the instruction that was provided when generating the answer.

Do not evaluate whether the instruction was followed—assume it was. Instead, use the instruction to set realistic expectations for the scope, depth, style, and language of the answer.

Examples

- If the instruction says to answer in 10–50

Model	Method	2-Tok Seen Mean / Best (↑)	2-Tok Unseen Mean / Best (↑)	2-Tok Var Both (↓)	3-Tok Seen Mean / Best (↑)	3-Tok Unseen Mean / Best (↑)	3-Tok Var Both (↓)
Qwen3-4B	Instruction	93.3 / 94.7	68.9 / 73.2	**5.3**	88.2 / 91.7	55.6 / 64.8	**15.1**
	Steering token	**93.7** / 94.7	69.1 / **75.0**	6.2	89.3 / 93.7	**60.7 / 75.5**	22.7
	Steering token + Instruction	**93.7 / 94.9**	**69.2** / 73.9	**5.3**	**90.7 / 93.5**	58.0 / 71.2	18.6
Qwen3-8B	Instruction	90.7 / 92.2	71.8 / 75.7	5.1	83.7 / 88.7	54.0 / 63.9	15.1
	Steering token	90.9 / 92.5	**76.9 / 79.6**	4.1	83.1 / 89.8	59.5 / 70.7	21.2
	Hybrid	**92.2 / 93.7**	76.3 / 79.6	**4.4**	**87.9 / 91.8**	**62.9 / 70.3**	**12.4**
Llama3-3B	Instruction	86.6 / 87.7	66.7 / 68.9	3.1	68.5 / 72.0	33.8 / 38.7	**8.7**
	Steering token	87.7 / 89.1	69.3 / 72.0	3.9	69.8 / 74.5	33.9 / 45.8	18.7
	Hybrid	**91.7 / 92.4**	**74.9 / 77.3**	**2.8**	**76.9 / 79.9**	**43.4 / 50.4**	10.6
Llama3-8B	Instruction	88.1 / 89.3	67.8 / 70.3	3.6	79.4 / 83.8	40.2 / 48.2	13.6
	Steering token	88.1 / 89.3	67.0 / 72.4	5.9	80.7 / 84.2	39.5 / 54.1	19.2
	Hybrid	**92.3 / 93.2**	**76.3 / 78.8**	**3.2**	**86.9 / 89.0**	**52.9 / 61.3**	**11.0**
Smol3-3B	Instruction	74.2 / 76.9	53.2 / 55.6	**5.1**	62.3 / 68.1	32.5 / 40.4	**14.5**
	Steering token	68.0 / 73.0	53.2 / **60.4**	11.8	50.3 / 65.9	35.5 / 53.0	35.9
	Hybrid	**75.7 / 79.1**	**53.5** / 57.5	7.3	**66.2 / 73.9**	**37.2 / 46.5**	17.1
Olmo-7B	Instruction	78.0 / 79.0	56.8 / 58.7	**2.9**	55.5 / 58.6	30.9 / 34.8	7.3
	Steering token	77.5 / 78.5	56.9 / 59.7	3.6	53.9 / 56.8	28.4 / 36.6	12.3
	Hybrid	**78.3 / 80.2**	**60.9 / 62.7**	3.7	**57.8 / 60.8**	**37.5 / 40.6**	**6.4**

Table 7: Extension using *best accuracy* of Table 2.

Model	Method	2-Tok Seen Mean / Best (↑)	2-Tok Unseen Mean / Best (↑)	2-Tok Var Both (↓)	3-Tok Seen Mean / Best (↑)	3-Tok Unseen Mean / Best (↑)	3-Tok Var Both (↓)	Response Quality (↑)
Qwen3-4B	Instruction	93.3 / 94.7	68.9 / 73.2	**5.3**	88.2 / 91.7	55.6 / 64.8	**15.1**	4.8
	Steering token (2-only)	93.7 / 94.7	69.1 / **75.0**	6.2	89.3 / 93.7	**60.7 / 75.5**	22.7	4.8
	Steering token + Instruction	**93.7 / 94.9**	**69.2** / 73.9	**5.3**	**90.7 / 93.5**	58.0 / 71.2	18.6	4.8
Qwen3-8B	Instruction	91.0 / 92.5	71.6 / 75.2	5.1	82.9 / 88.2	52.1 / 61.5	15.1	4.9
	Steering token (2-only)	90.9 / 92.5	76.9 / 79.6	**4.1**	83.1 / 89.8	59.5 / **70.7**	21.2	4.9
	Steering token (2+3)	91.2 / 92.5	**77.0 / 80.1**	4.2	85.9 / 90.2	59.7 / 68.7	15.5	4.9
	Steering token + Instruction	**92.2 / 93.7**	76.3 / 79.6	4.4	**87.9 / 91.8**	**62.9** / 70.3	**12.4**	4.9
Qwen3-14B	Steering token (2-only)	92.9 / 94.5	75.2 / 78.4	4.6	90.4 / **94.3**	68.0 / 76.0	13.9	4.9
	Steering token (2+3)	93.0 / 94.6	73.8 / 77.9	5.3	89.7 / 93.1	63.9 / 73.7	17.3	4.9
	Instruction	92.1 / 93.7	72.2 / 76.2	5.2	88.2 / 91.6	61.4 / 67.6	11.2	4.9
	Steering token + Instruction	**93.9 / 94.8**	**78.3 / 80.2**	**2.7**	**91.7** / 93.6	**69.2 / 72.9**	**6.2**	4.9

Table 8: Extension using *best accuracy* of Table 3.

Model	Method	Seen Tokens ↑	Unseen Tokens ↑	All Tokens ↑
Llama3-3B	Text	$93.6 \pm 0.2\%$	$86.7 \pm 0.5\%$	$92.0 \pm 0.2\%$
	Steering token	$94.5 \pm 0.2\%$	$89.6 \pm 0.4\%$	$93.4 \pm 0.2\%$
	Hybrid	$\mathbf{96.3 \pm 0.2\%}$	$\mathbf{92.6 \pm 0.4\%}$	$\mathbf{95.5 \pm 0.1\%}$
Llama3-8B	Text	$93.0 \pm 0.2\%$	$81.2 \pm 0.5\%$	$90.4 \pm 0.2\%$
	Steering token	$93.4 \pm 0.2\%$	$83.2 \pm 0.5\%$	$91.1 \pm 0.2\%$
	Hybrid	$\mathbf{95.7 \pm 0.2\%}$	$\mathbf{88.7 \pm 0.5\%}$	$\mathbf{94.1 \pm 0.2\%}$
Qwen3-4B	Text	$\mathbf{96.6 \pm 0.2\%}$	$82.1 \pm 0.5\%$	$93.2 \pm 0.2\%$
	Steering token	$96.1 \pm 0.2\%$	$80.6 \pm 0.5\%$	$92.7 \pm 0.2\%$
	Hybrid	$\mathbf{96.6 \pm 0.1\%}$	$\mathbf{84.3 \pm 0.5\%}$	$\mathbf{93.8 \pm 0.2\%}$
Qwen3-8B	Text	$\mathbf{94.8 \pm 0.2\%}$	$81.6 \pm 0.5\%$	$91.7 \pm 0.2\%$
	Steering token	$94.7 \pm 0.2\%$	$\mathbf{83.2 \pm 0.5\%}$	$\mathbf{92.0 \pm 0.2\%}$
	Hybrid	$94.1 \pm 0.2\%$	$82.6 \pm 0.5\%$	$91.5 \pm 0.2\%$

Table 9: Single-behavior performance on seen, unseen, and all tokens across models.

words, do not penalize brevity.

- If the instruction says to answer in a different language (e.g., Spanish), do not evaluate the answer based on English fluency.

Rating Scale

- 1 = Very poor
- 2 = Poor
- 3 = Acceptable
- 4 = Good
- 5 = Excellent

Output only the number. Do not include any explanation.

Evaluation Template

Instruction: `{instruction}`

Question: `{question}`

Model's Answer: `{answer}`

Evaluation:

G Metrics

Mean and best accuracy: For a k-property combination, accuracy measures the percentage of responses satisfying all k constraints simultaneously (evaluated on $n = 1000$ samples). Since token orderings can affect performance, we evaluate all $k!$ possible orderings and report the mean accuracy across orderings as our primary metric. We also report **best accuracy** (maximum across orderings) to characterize the upper bound achievable with optimal ordering.

Order variance: For each token combination tested with multiple orderings, we compute the maximum pairwise absolute difference in accuracy: $\Delta_{\text{max}} = \max_{i,j} |a_i - a_j|$, where a_i and a_j are accuracies for different orderings of the same token set. We report two statistics: (1) the mean of Δ_{max} across all token combinations with multiple orderings (Avg), and (2) the maximum Δ_{max} observed across all combinations (Max). Lower variance indicates more robust, order-invariant compositional behavior. This metric is critical for practical deployment scenarios, where users cannot manually optimize token ordering for each task. Unlike standard deviation, which measures spread around the mean, the maximum pairwise difference captures the worst-case sensitivity to ordering—directly quantifying the risk that an unfavorable token arrangement could substantially degrade performance. For a truly order-invariant composition mechanism, Δ_{max} should approach zero regardless of token arrangement.

H Baseline Implementation Details

We document the three parameter-based baselines compared against steering tokens in Table 1: **LoRA DARE**, **LM-Steer** output steering baseline, and **CAA**. We train LoRA DARE and LM-Steer using the same self-distillation objective, training data, and training schedule as steering tokens (see §B), so any difference in performance reflects the steering method rather than the supervision signal. The base LLM is frozen in both cases; only the baseline-specific parameters are updated. For the **Contrastive Action Addition** baseline (Rimsky et al., 2024) we use their official codebase: `https://github.com/nrimsky/CAA`

H.1 LoRA DARE Baseline

Per-behavior adapter. For each behavior b, we train a separate LoRA adapter (Hu et al., 2022) with rank $r = 8$ and scaling factor $\alpha = 16$, dropout 0.1, and no additive bias (`bias="none"`). The adapter is inserted *only into the MLP sublayer projections* of every transformer block—specifically the modules `gate_proj`, `up_proj`, and `down_proj`—-while attention projections and embedding/LM-head matrices remain unmodified and frozen.

Composition via DARE Linear. For multi-behavior steering we merge the per-behavior adapters into a single adapter using **DARE Linear** (Yu et al., 2024): adapter weights are randomly dropped with probability p_{drop} and the surviving entries are rescaled by $1/(1 - p_{\text{drop}})$, then combined by a linear (weighted-average) aggregation of the sparsified updates. The merged adapter is applied as a single PEFT module at inference. This is a merging procedure used to produce the 2- and 3-behavior numbers in Table 1. No separate adapter is trained per combination; composition is purely post-hoc.

Inference. Inputs are prepared with the model's native chat template. Generation uses vLLM with the same greedy-decoding configuration as our main experiments (temperature 0, up to 256 tokens), with `max_loras` set to accommodate all evaluated combinations.

Order-variance reporting. Because DARE Linear merging is invariant to the order of behaviors, we leave the Ord. Var. column in Table 1 blank for this baseline.

H.2 LM-Steer (Output Steering) Baseline

Intervention. Following Han et al. (2024), the LM-Steer baseline inserts a learned linear transformation into the *final hidden state* of the transformer, immediately before the LM head:

$$\mathbf{h}' \;=\; \mathbf{h} + \varepsilon\, \mathbf{W}\, \mathbf{h}, \qquad \varepsilon = 10^{-3}.$$

The weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$ (with d the model hidden size) is parameterised via a rank-r factorisation

$$\varepsilon \mathbf{W} \;\approx\; \mathbf{P}_1 \mathbf{P}_2, \qquad \mathbf{P}_1 \in \mathbb{R}^{d \times r},\ \mathbf{P}_2 \in \mathbb{R}^{r \times d},$$

with $r = 64$. Both factors are initialised as $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.01$ (the paper explicitly warns against zero-initialisation, which we respect). Only $\mathbf{P}_1$ and $\mathbf{P}_2$ are trainable; all other model parameters are frozen. Concretely, the forward pass runs the transformer backbone in the usual way, takes the final `last_hidden_state`, applies

$\mathbf{h}' = \mathbf{h} + \mathbf{h}\,\mathbf{P}_1\,\mathbf{P}_2$, and then feeds $\mathbf{h}'$ into the original (frozen) `lm_head` to obtain logits. The loss is the standard causal-LM loss on $\mathbf{h}'$-derived logits, which backpropagates only into $\mathbf{P}_1$ and $\mathbf{P}_2$.

Per-behavior matrices. We train one pair $(\mathbf{P}_1^{(p)}, \mathbf{P}_2^{(p)})$ per behavior b, using the same 50k-sample self-distillation data, optimiser, and schedule as our steering tokens. The trained matrices cover all 15 properties reported in the main experiments (*languages*, *format*, *length*, *structure*).

Additive composition. For multi-behavior inference, the loaded per-behavior factor pairs are *added* in parameter space before application—matching the additive composition recipe of Han et al. (2024):

$$\mathbf{h}' \;=\; \mathbf{h} + \mathbf{h}\left(\sum_{p\in S}\mathbf{P}_1^{(p)}\right)\left(\sum_{p\in S}\mathbf{P}_2^{(p)}\right),$$

where S is the set of active behaviors.

Inference. As for LoRA, the prompt contains no steering tokens. Because LM-Steer has no ordering at the prompt level and sum-of-factors composition is symmetric (commutative under addition), there is again no ordering axis to average over, so the Ord. Var. column is left blank in Table 1. Inference uses greedy decoding with up to 256 tokens

H.3 CAA Baseline

Steering vector construction. Following Rimsky et al. (2024), we construct one steering vector per behavior from hidden-state differences. For each behavior b, we collect paired positive and negative examples D_b, extract the hidden state at the last token position from layer l, compute the activation difference for each pair, and define the behavior steering vector as the mean of these differences:

$$\mathbf{v}^{(b)} \;=\; \frac{1}{|D_b|}\sum_{(x^+,x^-)\in D_b}\left(\mathbf{h}^{(l)}_{\text{last}}(x^+) - \mathbf{h}^{(l)}_{\text{last}}(x^-)\right)$$

Composition. For multi-behavior steering, we compose the per-behavior vectors additively. Given a set of active behaviors S, we form a single steering vector as a weighted sum

$$\mathbf{v}_S \;=\; \sum_{b\in S}\alpha_b\mathbf{v}^{(b)},$$

where in our experiments all active behaviors use the same multiplier $\alpha_b = 3.0$.

Inference. During generation, we add the composed steering vector to the model hidden state at the same transformer layer used for extraction: $\mathbf{h}^{(l)} \leftarrow \mathbf{h}^{(l)} + \mathbf{v}_S$. Thus, unlike LoRA DARE and LM-Steer, CAA performs activation-space intervention directly at inference time rather than learning additional trainable parameters.

Hyperparameters. We extract and apply steering vectors at layer $l = 10$. We set the steering multiplier to 3.0, selected based on single-behavior performance on a small validation set. Generation uses greedy decoding with temperature 0 and a maximum of 256 new tokens.

Order-variance reporting. Because CAA composes behaviors by summing steering vectors, the composition rule is invariant to behavior order. We therefore leave the Ord. Var. column blank for this baseline in Table 1.

I Instruction Template Variants

Language Instruction Variants ({language})

- Answer in {language}.
- Respond in {language}.
- Provide your answer in {language}.
- Reply in {language}.
- Give your response in {language}.
- Write your answer in {language}.
- Use {language} for your response.
- Provide a {language} response.
- Respond using {language}.
- Answer using the {language} language.

Word Range Instruction Variants ({min}, {max})

- Answer in {min} to {max} words.
- Respond in {min} to {max} words.
- Keep your response between {min} and {max} words.
- Use {min}-{max} words in your answer.
- Provide an answer of {min} to {max} words.
- Your response should contain {min} to {max} words.
- Limit your answer to {min}-{max} words.
- Give a response that is {min} to {max} words long.
- Answer using between {min} and {max} words.
- Provide a {min} to {max} word response.

Sentence Count Instruction Variants ({num})

- Provide a response in {num} sentences.
- Answer in {num} sentences.
- Give a {num}-sentence response.
- Respond using exactly {num} sentences.
- Your answer should be exactly {num} sentences.
- Limit your response to {num} sentences.
- Provide an answer consisting of {num} sentences.
- Reply with {num} sentences.
- Use {num} sentences for your answer.
- Answer in {num} sentences only.

Case Formatting Instruction Variants ({case_type})

- Each word in the answer should be in {case_type}.
- Write your answer with each word in {case_type}.
- Use {case_type} for every word in your response.
- Use only {case_type} letters in your answer.
- Format your answer in {case_type}.
- All words should be in {case_type}.
- Apply {case_type} formatting to your answer.
- Your response should use {case_type} for all words.
- Make all letters {case_type}.
- Use {case_type} throughout your answer.

Json Wrap Instruction Variants ({json_format})

- Wrap your answer in JSON with the key 'answer'.
- Return your response as a JSON object with an 'answer' field.
- Format your reply as JSON: {änswer̈: ...ÿ}.
- Respond with a JSON object containing an 'answer' key.
- Output your answer as JSON with the key 'answer'.
- Provide your response in JSON format using the key 'answer'.
- Give your answer inside a JSON object with an 'answer' field.
- Structure your response as {änswer̈: ÿour response}.
- Reply in JSON format with your answer under the 'answer' key.
- Encode your response as a JSON object with an 'answer' property.

J Model Generations

Table 10 presents 19 representative examples of **compositional control** using steering tokens across three model sizes (Qwen-4B, Qwen-8B, Qwen-14B). All examples shown are success cases where all specified constraints are satisfied simultaneously.

Model	Steering Tokens	Prompt	Generation
Qwen-8B	spanish words_10_50	What are the most effective ways to reduce energy consumption in a small office setting as part of an environmental...	Para reducir el consumo energético en una oficina pequeña, utiliza luces LED, apaga dispositivos no en uso, usa computadoras en modo de suspensión, y realiza mantenimiento regular...
Qwen-8B	french words_10_50	In what ways does the Mediterranean diet compare to the American diet in terms of reducing the risk of heart attacks?	La diète méditerranéenne réduit davantage le risque d'infarctus en privilégiant les fruits, légumes, oléagineux et poissons, tandis que la diète américaine, riche en graisses...
Qwen-8B	lower_case words_10_50	How should I structure a follow-up email after accepting a job offer to confirm the details?	thank you for the offer. i confirm my acceptance and appreciate the opportunity. please confirm the start date, role, and any additional details. looking forward to joining the...
Qwen-8B	upper_case french	What are the advantages of using cloud-based [application] over on-premise solutions for [company]?	LES AVANTAGES DE L'UTILISATION DE SOLUTIONS EN CLOUD POUR [COMPANY] PAR RAPPORT AUX SOLUTIONS HÉBERGÉES EN INTERNE SONT LES SUIVANTS : PLUS GRANDE FLEXIBILITÉ, COÛTS...
Qwen-8B	words_10_50 portuguese	What does the theme of love versus infatuation reveal about Romeo and Juliet's relationship in the play?	O tema do amor versus a infatuação revela que o relacionamento de Romeo e Julieta é baseado em paixão intensa, mas imatura, que se transforma em tragédia devido à falta de...
Qwen-8B	upper_case words_10_50	In what ways does wind power contribute to lowering greenhouse gas emissions compared to traditional fossil fuel-based...	WIND POWER GENERATES CLEAN ENERGY WITHOUT BURNING FUELS, REDUCING COAL, OIL, AND GAS USE. IT CUTS CARBON DIOXIDE AND OTHER GREENHOUSE GASES, HELPING MITIGATE CLIMATE CHANGE. NO...
Qwen-8B	lower_case spanish sentences_4	What are the key steps to ensure a successful nonprofit organization launch focused on environmental conservation?	primero, define un propósito claro y una misión específica relacionada con la conservación ambiental. segundo, desarrolla un plan de acción detallado que incluya estrategias de...
Qwen-8B	sentences_5 lower_case spanish	Can you provide historical examples where perseverance was key to achieving major breakthroughs?	en la historia, la perseverancia ha sido clave para lograr avances importantes. maria curie, a pesar de las dificultades y la falta de apoyo, logró descubrir el radio y el...
Qwen-4B	portuguese words_10_50	What are the main advantages of using a project management tool for team collaboration?	Ferramentas de gerenciamento de projetos facilitam a comunicação, centralizam tarefas, melhoram a transparência e aumentam a produtividade, permitindo que equipes trabalhem de...
Qwen-14B	german words_10_50	Should a first-aid kit be considered one of the top 5 essential items for a beach vacation, and what should it include?	Ja, ein Erste-Hilf-Kasten gehört zu den Top 5 für einen Badeurlaub. Er sollte Verbände, Desinfektionsmittel, Schmerztabletten, Pflaster und eine Pinzette enthalten, um...
Qwen-14B	italian words_10_50	How can I effectively practice behavioral interview questions for a role in the engineering industry?	Per prepararti efficacemente alle domande comportamentali per un ruolo ingegneristico, ripassa esperienze passate, utilizza il metodo STAR (Situazione, Compito, Azione, Risultato)...
Qwen-14B	spanish lower_case	How do bullet points contribute to the accessibility of written content for individuals with reading difficulties?	los puntos de viñeta ayudan a hacer más accesible el contenido escrito para personas con dificultades de lectura al dividir la información en partes más pequeñas y manejables....
Qwen-14B	words_10_50 italian	How should I structure a follow-up email after accepting a job offer to confirm the details?	Gentile [Nome], confermo l'accettazione dell'offerta di lavoro. Posso confermare i dettagli dell'incarico, della retribuzione e della data di inizio? Resto a disposizione per...
Qwen-14B	french words_10_50 lower_case	What strategies can a manager use to enhance workplace efficiency, boost employee productivity, and foster better...	utiliser des outils de gestion du temps, encourager la communication ouverte, organiser des formations régulières, promouvoir l'autonomie et reconnaître les performances pour...
Qwen-14B	words_10_50 lower_case spanish	In what ways can a project management tool help a school administrator organize and track the progress of various...	una herramienta de gestión de proyectos puede ayudar a un administrador escolar a organizar tareas, asignar responsables, establecer plazos y monitorear el progreso de proyectos...
Qwen-14B	sentences_1 lower_case italian	I'm planning a trip to Bangkok. What are the best places to experience local culture, and what should I be cautious...	i migliori posti per esperire la cultura locale a bangkok sono il mercato fluttuante di damnoen saduak, il tempio di wat pho e il quartiere di chinatown, mentre si deve essere...
Qwen-14B	sentences_1 lower_case french	Are there specific types of exercise that are most effective for improving mental health in teenagers?	les exercices aérobiques réguliers, comme la course à pied, la natation ou le vélo, sont particulièrement efficaces pour améliorer la santé mentale des adolescents.
Qwen-14B	lower_case italian words_10_50	What role does Mr. Darcy play in the development of the main plot in *Pride and Prejudice*?	mr. darcy è centrale nel plot principale di "orgoglio e pregiudizio". il suo rapporto con elizabeth evoluisce da ostilità a amore, influenzando gli eventi chiave, come il...
Qwen-14B	lower_case sentences_1 italian	What is the capital of France, and what is the name of the famous department store that is both a shopping destination...	la capitale della francia è parigi e il famoso centro commerciale che è anche un attrazione turistica si chiama le bon marché.

Table 10: Example generations demonstrating compositional control with steering tokens.