MIND YOUR OWN ROLE: REDUCING ROLE OVERSTEPPING FOR EFFECTIVE AND EFFICIENT MULTI-AGENT COLLABORATION VIA DYNAMIC ROLE ACTIVATION STEERING

Anonymous authors
Paper under double-blind review

ABSTRACT

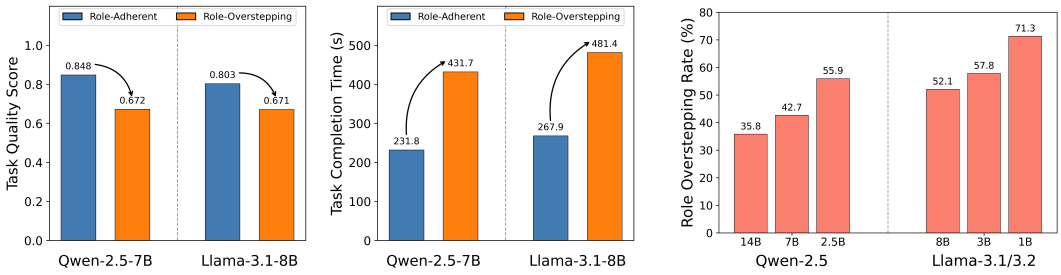
Multi-Agent Systems (MAS) have shown strong capability in solving complex tasks by assigning different agents specialized roles and corresponding subtasks. However, agents may overstep their assigned responsibilities and perform subtasks belonging to other roles, which can disrupt workflow, reduce system efficiency, and degrade task performance. To address this issue, we investigate role-related patterns in the model's internal activation space and uncover two key observations: (1) role instructions induce consistent directional shifts in activation space, and (2) role-adherent and role-overstepping trajectories exhibit distinct activation patterns. Based on these observations, we propose Dynamic Role Activation Steering (DRAS), a lightweight and effective inference-time activation intervention approach to reduce role overstepping in multi-agent collaboration. Specifically, DRAS constructs a role steering vector and a lightweight role-overstepping detector for each role. During LLM generation, the detector estimates the probability of role overstepping from the current activation state, and DRAS dynamically adjusts the steering strength accordingly to guide the agent toward its assigned responsibilities. Experimental results show that DRAS effectively reduces the number of role-overstepping trajectories by up to 86.6% compared with Vanilla MAS, thereby improving task quality score by 10.0%–11.1%, achieving a 1.50×–1.61× task-completion speedup, and reducing token consumption by 26.8%–38.7%.

1 INTRODUCTION

In recent years, LLM-driven multi-agent systems have made significant progress in solving complex tasks (Ye et al., 2025; Hu et al., 2025; Chen et al., 2024), particularly for highly collaborative applications like software development. By equipping agents with distinct role profiles and defining interaction protocols, these systems decompose the long-horizon task into coordinated subtasks, allowing different agents to handle different subtasks while jointly completing the overall task. Most well-known multi-agent systems (e.g., ChatDev (Qian et al., 2024) and MetaGPT (Hong et al., 2024) and AutoGen (Wu et al., 2023)) employ such structured collaboration to improve reliability and end-to-end task performance.

However, recent studies (Kim et al., 2026; Cemri et al., 2026; Rath, 2026) have shown that in practice it does not always hold that each agent always follows its assigned role. Agents may exceed their predefined responsibilities and perform subtasks that belong to other roles, resulting in **role overstepping** (Zhou et al., 2026a). For example, an agent responsible for code review may go beyond providing review comments and directly modify the code. Unlike an isolated generation error, the content generated by role overstepping may propagate along the collaboration chain, and cause redundant execution, responsibility conflicts, and contamination of the shared context. Consequently, role overstepping may degrade the overall task performance and increase the task completion time (Kim et al., 2026; Rath, 2026). As shown in Fig. 1a, for ChatDev with Qwen-2.5-7B (Yang et al., 2024), tasks with role overstepping exhibit a 20.75% decrease in task quality scores and an 86.24% increase in task completion time compared with role-adherent tasks. Moreover, our empirical results in Fig. 1b show that role overstepping is prevalent across various model sizes and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Task quality score and task completion time.

(b) Role overstepping rates.

Figure 1: Prevalence and impact of role overstepping in ChatDev. (a) Comparison of task performance and task completion time between role-adherent and role-overstepping tasks. (b) Role overstepping rates across model sizes.

becomes more pronounced as the model size decreases. For example, even for Qwen-2.5-14B, the role-overstepping rate reaches 35.8%, and further increases to 55.9% for the 2.5B model. With ever growing interest of deploying multi-agent systems on resource-constrained mobile devices that typically run medium or small sized models (Fu et al., 2025; Yi et al., 2026; Zhou et al., 2026b), such a high role-overstepping rate and its consequent task performance degradation and low system efficiency become an essential problem.

To address this issue, Wang et al. (2026) examine agent outputs after generation, using rule-based detectors or an additional LLM-as-a-Judge to determine whether a response violates the assigned role, and triggers regeneration when overstepping is detected. However, such post-hoc methods can only identify overstepping after a response has already been generated, wasting the computation spent on the overstepping response. The detection and regeneration process also introduces substantial computational cost and task completion time. Alternatively, Zhou et al. (2026a) leverage role-adherent trajectories to perform LoRA-based LLM fine-tuning (Hu et al., 2021) to improve role adherence. However, fine-tuning the LLM for each role incurs substantial training cost and hardware requirements, limiting its practicality and deployment flexibility. Therefore, how to promptly detect role overstepping and efficiently steer agent behavior remains an important challenge.

Activation engineering has recently emerged as an effective approach to steering LLM behavior at inference time (Rimsky et al., 2024; Wehner et al., 2025; Valentino et al., 2026). Its key idea is to identify directions in the model's activation space associated with specific behavioral attributes, such as honesty (Qiu et al., 2024), truthfulness (Li et al., 2023), and humor (Von Rütte et al., 2024), and to derive steering vectors from these directions. Instead of modifying model parameters, these methods directly intervene in internal activations using the extracted steering vectors, enabling cost-efficient behavioral control with adjustable steering strength. Inspired by this, we propose to study **if it is possible to analyze and intervene in the model's activation space during generation, so that we can promptly detect role-overstepping and efficiently fix it via activation steering.** To answer this question, we conduct exploratory experiments and reveal two key observations. (i) Role instructions induce consistent directional shifts in activation space, and steering along these directions can be viewed as reinforcing the behavioral constraints specified by the role instructions. (ii) Role-adherent and role-overstepping behaviors exhibit distinct activation patterns. These observations indicate that the activation space contains both the directional information needed for role steering and the state information needed for role-overstepping detection.

Based on those observations above, in this paper, we propose Dynamic Role Activation Steering (DRAS), an inference-time activation intervention approach for reducing role overstepping in multi-agent collaboration. For each role, we (i) compare the internal activations produced by the same subtask context with and without the corresponding role instruction and derive a role steering vector from their activation differences, and (ii) collect activations from role-adherent and role-overstepping trajectories to train a lightweight role-overstepping detector. During LLM generation, the detector estimates the probability of role overstepping from the current activation representation, which is then used to dynamically adjust the steering strength and promote role adherence. Extensive experiments demonstrate that DRAS reduces the number of role-overstepping trajectories by up to 86.6% compared with Vanilla MAS and consistently outperforms other methods. By reducing role

overstepping, DRAS further improves task quality score by 10.0%–11.1%, achieves a 1.50×–1.61× task-completion speedup, and reduces token consumption by 26.8%–38.7%.

2 PRELIMINARIES

2.1 ROLE OVERSTEPPING IN MULTI-AGENT COLLABORATION

Unlike end-to-end evaluation of overall task performance in a multi-agent system, role overstepping concerns whether each agent properly executes its assigned role within the internal collaborative workflow. In role-based multi-agent systems, agent responsibilities are usually defined by natural-language role instructions in LLM prompts. Formally, consider a multi-agent system with a role set $\mathcal{R} = \{r_1, \ldots, r_K\}$. Each role r_i is associated with a role instruction I_{r_i}, which specifies its responsibilities and behavioral constraints. Given a subtask context x_i, the agent with role r_i generates a behavior trajectory $\tau_i = \mathrm{LLM}(x_i, I_{r_i})$.

Definition: Role r_i exhibits role overstepping if τ_i contains behaviors that fall within the responsibilities described in another role r_j's role instruction I_{r_j}.

For trajectory evaluation, existing studies (Wang et al., 2026; Kim et al., 2026) commonly adopt LLM-as-a-Judge to determine whether each trajectory exhibits role overstepping according to the predefined role division. In addition, Zhou et al. (2026a) introduce a continuous metric based on semantic similarity in the embedding space. By comparing a trajectory with different role descriptions, the metric quantifies its alignment with the assigned role and its degree of shift toward other roles.

2.2 ACTIVATION ENGINEERING

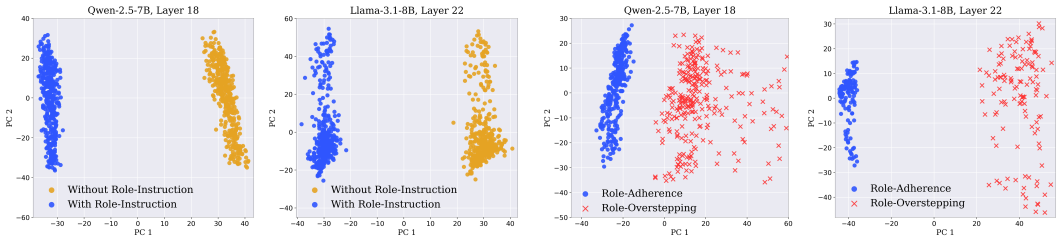
Activation Steering: Emerging research in activation engineering has uncovered that LLMs encode semantic concepts as linear subspaces within internal activations (Liu et al., 2023; Zou et al., 2023; Arditi et al., 2024). This geometric interpretation enables targeted behavioral steering through linear interventions during inference. Rimsky et al. (2024) pioneered activation addition using steering vectors derived from contrastive activations for sentiment and topic control. Following the same principle, Konen et al. (2024) demonstrated that activation vectors can also steer linguistic style, while Zhang et al. (2024) further identified steerable activation directions associated with the truthfulness of model outputs. More recently, Liu et al. (2026); Zhang et al. (2025) extracted activation shifts associated with user personal information to guide personalized generation. In this work, we investigate whether role-specific behaviors exhibit identifiable directional representations in activation space that can be leveraged for role steering in multi-agent collaboration.

Conditional Activation Steering: Applying a steering vector indiscriminately may interfere with model behaviors that do not require correction. For example, strengthening refusal to harmful queries can also suppress appropriate responses to harmless queries (Lee et al., 2025). Therefore, a fine-grained strategy is to condition the steering intervention on the model's current behavioral state. Lee et al. (2025) uses activation differences between harmful and harmless prompts to construct a conditioning vector that determines whether steering should be activated. Wang et al. (2025); Zhang et al. (2024) use labeled activations from truthful and untruthful trajectories to train a lightweight linear probe, which detects the truthfulness state of the current generation and adjusts steering strength accordingly. These studies demonstrate that activation space not only contains directional representations for behavioral steering, but also encodes information that reflects the model's current behavioral state.

3 UNCOVERING ROLE-RELATED PATTERNS IN ACTIVATION SPACE

Building on the insights of activation steering, we investigate whether role-related information exhibits structured and identifiable patterns in the model's internal activation space. Specifically, we extract model activations under contrastive prompts and use principal component analysis (PCA)[1] to

[1]Principal component analysis (PCA) is a linear dimensionality reduction technique. It linearly projects the data onto a new coordinate system, where the axes (principal components) are selected to account for the most significant variance in the data.



(a) PCA projections of activations from prompts with and without role instructions.

(b) PCA projections of activations from role-adherent and role-overstepping trajectories.

Figure 2: PCA projections of internal activations in ChatDev for the CEO role, extracted at selected layers of Qwen2.5-7B and Llama-3.1-8B.

assess the separability of their internal representations. This analysis is useful for examining whether activation differences under contrastive conditions exhibit consistent and identifiable structures (Park et al., 2023).

Observation 1: Role instructions induce consistent activation shifts. We form contrastive prompt pairs using the same role subtask context with and without the corresponding role instruction. For each pair, we perform separate forward passes and extract the last-token residual-stream activations from the same model layer. As shown in Fig. 2a, adding the role instruction produces a clear and consistent shift in the activation distribution at the selected layers. It indicates that role instructions induce systematic activation shifts that may provide an extractable steering direction for role-specific behaviors.

Observation 2: Role-adherent and role-overstepping behaviors exhibit distinct activation patterns. We collect role-adherent and role-overstepping trajectories generated for the same agent role and extract their last-token residual-stream activations from the same model layer. As shown in Fig. 2b, the two behavioral states exhibit clearly distinct activation distributions in the PCA projections at the selected layers. It means that internal activations contain discriminative information about whether an agent is adhering to or overstepping its assigned role.

4 METHOD

Based on the observations above, we propose Dynamic Role Activation Steering (DRAS), an inference-time activation intervention approach for mitigating role overstepping in multi-agent collaboration. For each role, alongside the original role instruction, we introduce a dynamic steering module consisting of a role steering vector and a role-overstepping detector. During generation, the module estimates the probability of role overstepping from activations at a selected LLM layer and dynamically adjusts the steering strength to effectively reduce role overstepping.

4.1 ROLE STEERING VECTOR CONSTRUCTION

Following the contrastive activation paradigm, we construct a role steering vector from activation differences induced by the corresponding role instruction. For each role r, we consider N subtask contexts $\{x_n\}_{n=1}^{N}$. For each x_n, we construct a matched input pair consisting of the subtask context alone, i.e., x_n, and the same context augmented with the role instruction, i.e., $[x_n; I_r]$. Let $h_{n,l}^{\emptyset} \in \mathbb{R}^d$ denote the last-token residual-stream activation at the output of layer l for x_n, and $h_{n,l}^{r} \in \mathbb{R}^d$ denote that for $[x_n; I_r]$, respectively. The role steering vector $\mathbf{v}_r^l$ for role r at layer l is defined as

$$\mathbf{v}_r^l = \frac{1}{N} \sum_{n=1}^{N} \left(h_{n,l}^{r} - h_{n,l}^{\emptyset} \right). \tag{1}$$

Averaging across different subtask contexts reduces task-specific variation and captures the consistent activation shift introduced by the role instruction. In this work, we extract activations and apply steering at a single layer l, which is selected based on probe performance on the validation set.

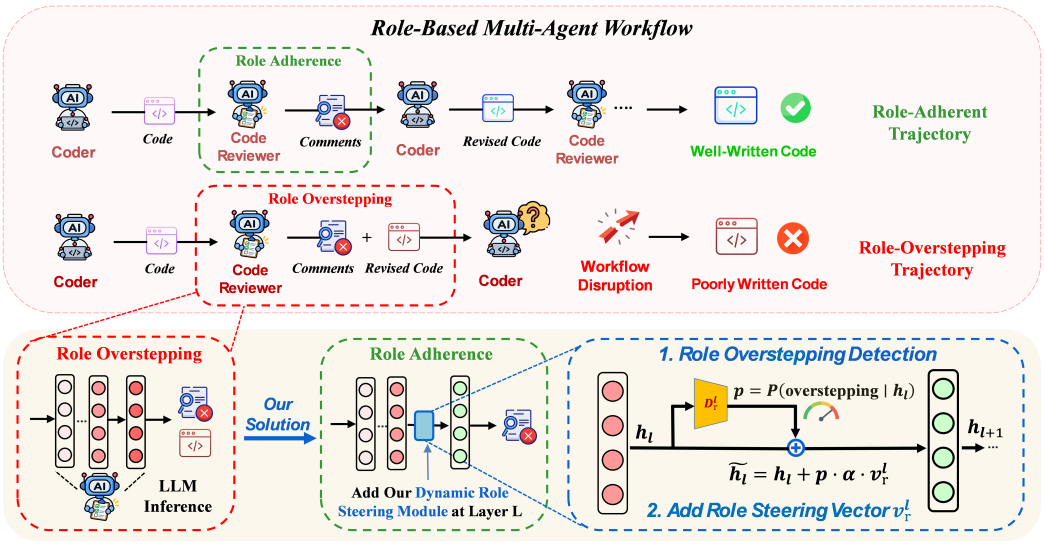


Figure 3: Role overstepping in multi-agent collaboration and our proposed Dynamic Role Activation Steering (DRAS). During generation, DRAS first detects the role-overstepping probability from the current activation state and then dynamically adjusts the steering strength to guide the agent toward its assigned role.

4.2 Role-Overstepping Detector Training

To enable fine-grained role steering and avoid unnecessary activation perturbations on role-adherent generations, we control the steering strength according to the estimated probability of role overstepping. Linear probes are commonly used to understand information encoded in internal representations (Belinkov, 2022; Wang et al., 2025). Based on Observation 2, we employ a lightweight linear probe to differentiate the activation representations of role-adherent and role-overstepping behaviors.

For each role r, we collect N_{adh} role-adherent trajectories $\{\tau_n^{\text{adh}}\}_{n=1}^{N_{\text{adh}}}$ and N_{over} role-overstepping trajectories $\{\tau_m^{\text{over}}\}_{m=1}^{N_{\text{over}}}$. Then we construct two types of inputs by concatenating the subtask context, role instruction I_r, and trajectory, $[x_n; I_r; \tau_n^{\text{adh}}]$ and $[x_m; I_r; \tau_m^{\text{over}}]$, respectively. Let $h_{n,l}^{\text{adh}} \in \mathbb{R}^d$ and $h_{m,l}^{\text{over}} \in \mathbb{R}^d$ denote their last-token residual-stream activations at the output of layer l, respectively. We assign label 0 to role-adherent activations and label 1 to role-overstepping activations, and form the probing dataset as follows.

$$\mathcal{D}_r^{\text{probe}} = \left\{\left(h_{n,l}^{\text{adh}}, 0\right)\right\}_{n=1}^{N_{\text{adh}}} \cup \left\{\left(h_{m,l}^{\text{over}}, 1\right)\right\}_{m=1}^{N_{\text{over}}}. \tag{2}$$

Using this dataset, we independently train the linear probe with a binary cross-entropy objective to differentiate role-adherent activation representations from role-overstepping ones, which has very limited training overhead. The detector for role r at layer l is defined as

$$D_r^l(\mathbf{h}) = \sigma\left((\mathbf{w}_r^l)^\top \mathbf{h} + b_r^l\right), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function, and $\mathbf{w}_r^l \in \mathbb{R}^d$ and b_r^l are the trainable parameters of the detector. The output $D_r^l(\mathbf{h}) \in [0, 1]$ represents the estimated probability of role overstepping for the current activation representation.

4.3 Dynamic Steering During Generation

After constructing the role steering vector and role-overstepping detector, we perform dynamic activation steering during LLM generation. Following Rimsky et al. (2024), we apply the intervention at every decoding step. As illustrated in Fig. 3, let $\mathbf{h}_l$ denote the residual-stream activation at the

last token of the current sequence at the selected layer l. We first feed $\mathbf{h}_l$ into the role-overstepping detector to obtain an estimated probability of role overstepping $p = D_r^l(\mathbf{h}_l) \in [0, 1]$. We then update the activation as

$$\tilde{\mathbf{h}}_l = \mathbf{h}_l + \alpha p \, \mathbf{v}_r^l, \tag{4}$$

where α is a hyperparameter that controls the base steering strength, while p dynamically scales the intervention according to the estimated probability of role overstepping. The modified activation $\tilde{\mathbf{h}}_l$ is then passed through the subsequent Transformer computation. In this way, our DRAS continuously monitors the agent's role behavior and dynamically applies the role activation steering, which keeps reducing role overstepping behaviors throughout the generation process.

5 Experimental Setup

Multi-Agent Systems (MAS) and LLM Backbones: We evaluate DRAS on two representative role-based multi-agent systems, ChatDev (Qian et al., 2024) and MetaGPT (Hong et al., 2024), both of which organize multiple LLM agents with predefined role responsibilities to collaboratively complete software-development tasks. For each system, we preserve its original workflow and role instructions. For the main experiments, we use two representative open-source LLMs, Qwen2.5-7B-Instruct (Yang et al., 2024) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024), as the underlying backbone models for all agents.

Datasets: We evaluate role overstepping and end-to-end task performance on the Software Requirement Description Dataset (SRDD), a curated collection of software requirements introduced in Qian et al. (2024). SRDD contains 1,200 software requirement tasks covering diverse software categories and task types. To avoid evaluation data leakage, we additionally use SWE-Dev (Du et al., 2025), another software-development task dataset, for the offline preparation of different methods, including collecting training data for LoRA FT method and constructing role steering vectors and role-overstepping detectors for DRAS.

Evaluation Metrics: For role overstepping, we report Role-Overstepping Rate (RO Rate), Number of Role-Overstepping Trajectories (RO Traj. #), and Role Clarity Score (RC Score). Specifically, RO Rate is the percentage of tasks with at least one role-overstepping trajectory, and RO Traj. # counts the total number of role-overstepping trajectories. Following Zhou et al. (2026a), RC Score measures role adherence based on the semantic similarity between agent trajectories and their assigned role instruction, with higher values indicating stronger role alignment. Following Qian et al. (2024), task performance is evaluated using Completeness, Executability, and Consistency, and their average is reported as the overall Quality score. More details are provided in Appendix A.2.

Methods for Comparison: We compare DRAS with the following methods. **(i) Vanilla MAS** uses the original multi-agent system with only the predefined role instructions and serves as the baseline. **(ii) Few-shot** augments the original role prompt with both role-adherent and role-overstepping examples to strengthen role constraints. **(iii) Post-hoc** (Wang et al., 2026) uses an LLM-based judge to detect role overstepping after each agent response and regenerates the response when overstepping is detected. **(iv) LoRA FT** (Zhou et al., 2026a) fine-tunes the backbone LLM on role-adherent trajectories using LoRA to improve role adherence. Detailed descriptions and implementation settings for each method are provided in Appendix A.3.

6 Evaluation Results and Analysis

Role-Overstepping Mitigation. We first evaluate different methods for mitigating role overstepping. As shown in Table 1, on ChatDev with Qwen model, DRAS is superior to Vanilla MAS across all three metrics: the RO Rate decreases from 42.67% to 7.75%, RO Traj. # drops from 987 to 132, corresponding to an 86.6% reduction, and the RC Score increases from 0.621 to 0.832. Besides, DRAS also consistently outperforms the other methods across all metrics. For example, on ChatDev-Qwen, DRAS further reduces the number of role-overstepping trajectories by 32.7% compared with the second-best LoRA FT method. These results demonstrate the effectiveness and superiority of DRAS in reducing role overstepping.

Table 1: Role-overstepping mitigation performance of different methods across multi-agent systems and LLM backbones on SRDD. (**RO Rate**: Role-Overstepping Rate; **RO Traj. #**: Number of Role Overstepping Trajectories; **RC Score**: Role Clarity Score)

Method	ChatDev-Qwen			MetaGPT-Llama		
	RO Rate ↓	RO Traj. # ↓	RC Score ↑	RO Rate ↓	RO Traj. # ↓	RC Score ↑
Vanilla MAS	42.67	987	0.621	25.33	423	0.784
Few-shot	25.92	389	0.713	18.75	261	0.786
Post-hoc	15.25	257	0.725	15.33	203	0.813
LoRA FT	11.50	196	0.798	19.33	247	0.834
DRAS (Ours)	**7.75**	**132**	**0.832**	**7.83**	**114**	**0.855**

Table 2: End-to-end task performance and efficiency of different methods across multi-agent systems and LLM backbones on SRDD. (**Compl.**, **Exec.**, **Cons.**, and **Qual.** denote Completeness, Executability, Consistency, and Quality, respectively.)

System & Model	Method	Task Performance				Efficiency	
		Compl. ↑	Exec. ↑	Cons. ↑	Qual. ↑	Compl. Time (s)↓ / (Speedup) ↑	Tokens ↓
ChatDev-Qwen	Vanilla MAS	0.7017	0.8708	0.693	0.7552	341.3 *(1.00×)*	37.7K
	Few-shot	0.7133	0.9292	0.795	0.8125	293.8 *(1.16×)*	31.9K
	Post-hoc	0.7225	0.9375	0.774	0.8113	461.0 *(0.74×)*	51.3K
	LoRA FT	0.7267	0.9358	**0.828**	0.8302	227.9 *(1.50×)*	26.1K
	DRAS (Ours)	**0.7408**	**0.955**	0.819	**0.8383**	**211.4** *(1.61×)*	**24.3K**
ChatDev-Llama	Vanilla MAS	0.7208	0.8433	0.671	0.7450	365.0 *(1.00×)*	39.2K
	Few-shot	0.7300	0.8558	0.731	0.7723	319.7 *(1.14×)*	34.6K
	Post-hoc	0.7533	0.8800	0.788	0.8071	498.2 *(0.73×)*	57.0K
	LoRA FT	0.7425	0.8758	0.793	0.8038	291.1 *(1.25×)*	30.8K
	DRAS (Ours)	**0.7608**	**0.8908**	**0.807**	**0.8195**	**243.6** *(1.50×)*	**26.5K**
MetaGPT-Qwen	Vanilla MAS	0.6217	0.7458	0.631	0.6662	469.2 *(1.00×)*	43.6K
	Few-shot	0.6133	0.7833	0.679	0.6919	423.1 *(1.11×)*	41.7K
	Post-hoc	0.6233	0.7733	0.683	0.6932	543.8 *(0.86×)*	59.4K
	LoRA FT	0.6475	**0.7992**	0.734	0.7269	362.6 *(1.29×)*	38.1K
	DRAS (Ours)	**0.6517**	0.7983	**0.752**	**0.7340**	**312.8** *(1.50×)*	**31.9K**
MetaGPT-Llama	Vanilla MAS	0.6008	0.7108	0.743	0.6849	498.7 *(1.00×)*	47.8K
	Few-shot	0.6633	0.7533	0.756	0.7242	453.3 *(1.10×)*	42.4K
	Post-hoc	0.6817	0.7742	0.779	0.7450	531.4 *(0.94×)*	54.2K
	LoRA FT	0.6433	0.7608	0.763	0.7224	361.0 *(1.38×)*	37.9K
	DRAS (Ours)	**0.6983**	**0.7992**	**0.785**	**0.7608**	**318.5** *(1.57×)*	**29.3K**

End-to-end Task Performance and Efficiency. We further evaluate the end-to-end task performance and runtime efficiency of different methods. As shown in Table 2, compared with Vanilla MAS, DRAS improves the task Quality score by 10.0%–11.1%, achieves a 1.50×–1.61× speedup in task completion, and reduces token consumption by 26.8%–38.7% across the four system-backbone settings. DRAS also consistently outperforms the other compared methods across these metrics. These results indicate that reducing role overstepping within the system can improve both end-to-end task quality and runtime efficiency. It is also worth noting that Post-hoc improves task performance over Vanilla MAS, but incurs longer task completion time and higher token consumption. This highlights the advantage of timely dynamic intervention during generation over detecting and correcting role overstepping only after generation.

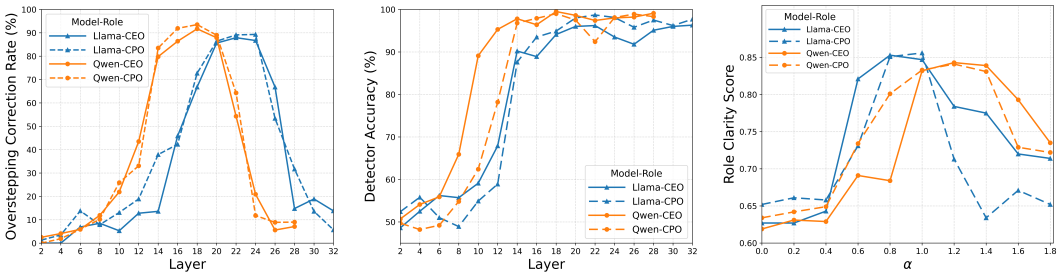
Table 3: Offline preparation cost per role on Qwen2.5-7B.

Method	**Peak Mem. (GB) ↓**	**Computation (PFLOPs) ↓**
LoRA FT	45.3	374.0
DRAS	**16.4**	**42.9**

Since both LoRA FT and DRAS require offline preparation, we also compare their offline preparation costs. As shown in Table 3, for Qwen2.5-7B, LoRA FT requires 45.3 GB of peak memory for model training, while DRAS requires only 16.4 GB, which is at the inference-level memory footprint. In addition, LoRA FT requires 374.0 PFLOPs of computation per role, approximately 8.7× that of DRAS. Overall, DRAS achieves better task performance and runtime efficiency with substantially lower offline preparation cost.

Table 4: Ablation study of dynamic steering. **Static Steering** applies a fixed steering strength at each decoding step, **Binary Gating** converts the detector probability into a binary on/off decision using a threshold, and **DRAS** dynamically adjusts the steering strength according to the predicted role-overstepping probability.

System	Method	RO Rate ↓	RO Traj. # ↓	RC Score ↑	Quality ↑	Compl. Time (s) ↓	Tokens ↓
ChatDev-Qwen	Static Steering	**5.92**	**93**	0.807	0.8207	**206.73**	**22.8K**
	Binary Gating	9.83	151	0.819	0.8223	237.1	27.6K
	DRAS	7.75	132	**0.832**	**0.8383**	211.4	24.3K
MetaGPT-Llama	Static Steering	**6.75**	**92**	0.818	0.7422	**309.1**	**28.7K**
	Binary Gating	10.50	147	0.847	0.7512	324.7	31.4K
	DRAS	7.83	114	**0.855**	**0.7608**	318.5	29.3K



(a) Layer-wise role steering effectiveness.

(b) Layer-wise detector accuracy.

(c) Sensitivity to base steering strength α.

Figure 4: Layer-wise steering and detection performance and sensitivity to the base steering strength in DRAS. Results are reported on the SWE-Dev validation set for the CEO and CPO roles in ChatDev using Qwen2.5-7B and Llama-3.1-8B.

Ablation Study of Dynamic Steering. We further perform an ablation study on the dynamic steering mechanism of DRAS. Static Steering removes the detector-based control and applies the role steering vector with a fixed strength at every decoding step. As shown in Table 4, by avoiding missed interventions caused by detector errors, this strategy achieves slightly lower RO Rate and RO Traj. # than DRAS. However, this static intervention may disturb generation that is already role-adherent. We observe that Static Steering occasionally leads to meaningless or abnormal outputs in such trajectories. Consequently, although it achieves lower role-overstepping rates, both its RC Score and task Quality are noticeably lower than those of DRAS.

We also consider Binary Gating, which adopts the same detector as DRAS but uses 0.5 as a threshold to convert the predicted probability into a binary signal that only determines whether steering is turned on or off. As shown in Table 4, DRAS consistently outperforms Binary Gating across both system-backbone settings in role-overstepping mitigation, task performance, and efficiency. This indicates that continuous adjustment of steering strength based on the estimated role-overstepping probability provides a more effective intervention strategy than a hard binary decision.

Layer-wise Analysis. Figure 4a evaluates role-steering effectiveness across layers, with the role steering vector extracted and applied at the same layer. For both LLMs, effective correction of role overstepping is concentrated in a subset of middle-to-late layers (e.g., layers 16 - 20 in Qwen), while both shallow and very deep layers exhibit relatively weak steering effectiveness. This pattern may arise because role-related representations are not yet sufficiently developed in shallow layers, while deep layers' representations may be less amenable to steering. In addition, different roles show similar layer-wise trends within the same backbone. DRAS therefore uses a shared intervention layer for all roles under each backbone.

Figure 4b presents the layer-wise performance of the role-overstepping detector. Unlike steering effectiveness, detection remains reliable even in deep layers, indicating that role-behavior states remain distinguishable within the deep-layer representations. Moreover, the optimal detection layers are generally close to the optimal steering layers across different models and roles (e.g., both peak at

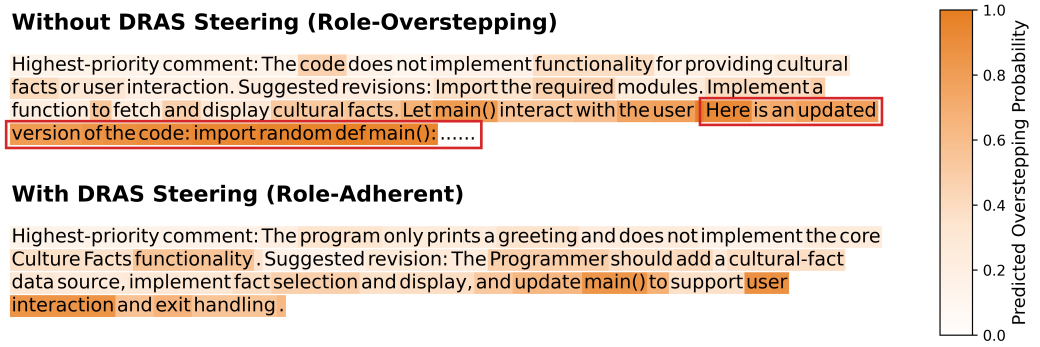


Figure 5: A role-overstepping trajectory and its DRAS-corrected trajectory for the Code Reviewer role in ChatDev-Qwen. The Code Reviewer is instructed to only provide high-priority comments and revision suggestions, and red boxes highlight role-overstepping content. Token colors indicate the role-overstepping probability predicted by the detector.

layer 18 for Qwen-CEO). This allows DRAS to perform role-overstepping detection and activation steering at the same selected layer.

Base Steering Strength Selection. We also evaluate the effectiveness of role steering under different base steering strengths α. As shown in Fig. 4c, increasing α does not consistently lead to better performance. When α is small, the intervention is too weak to effectively reinforce role-adherent behavior. As α increases, the RC Score improves substantially and reaches a favorable range (e.g., α = 0.8 - 1.0 for the Llama model). However, further increasing α leads to performance degradation, indicating potential over-steering and showing that a larger α is not always better. In addition, our empirical results show that different roles exhibit similar effective α ranges within the same backbone. DRAS therefore uses a shared α setting for all roles under each backbone.

Case Study. Figure 5 presents an example of a role-overstepping trajectory and its DRAS-corrected trajectory for the Code Reviewer role in ChatDev-Qwen. We first retain only the role-overstepping detector without applying steering during the generation of the original role-overstepping trajectory to observe its predictions. As shown in the upper example, the predicted role-overstepping probability begins to increase before the model explicitly generates content beyond the reviewer's responsibility. This allows DRAS to perform activation steering before explicit role-overstepping content is generated, rather than reacting only after role overstepping has occurred. When the role steering vector is applied, the predicted probability is used to dynamically adjust the steering strength. As shown in the lower example, relatively high role-overstepping probabilities appear toward the end of the trajectory, triggering stronger steering that redirects the generation toward the assigned role.

7 Conclusion

In this paper, we presented Dynamic Role Activation Steering (DRAS), an inference-time activation intervention approach to effectively reduce role overstepping in multi-agent collaboration. Based on our observations of consistent role-induced activation shifts and distinct activation patterns between role-adherent and role-overstepping behaviors, DRAS jointly uses role steering vectors and lightweight role-overstepping detectors to dynamically control steering strength during generation. Extensive experiments demonstrate that DRAS reduces the number of role-overstepping trajectories by up to 86.6% compared with Vanilla MAS and consistently outperforms other SOTA methods. By reducing role overstepping, DRAS further improves task quality score by 10.0%–11.1%, achieves a 1.50×–1.61× task-completion speedup, and reduces token consumption by 26.8%–38.7%. Moreover, DRAS has much lower offline preparation cost than LoRA-based fine-tuning, making it much easier to deploy in practice. In contrast to post-hoc methods that rely on detection and regeneration after response completion, DRAS performs timely detection and correction during the initial generation process, which enhances both task performance and system efficiency. These results collectively establish DRAS as a lightweight and effective approach for reducing role overstepping in multi-agent collaboration, thereby improving end-to-end task performance and efficiency.

AI USE STATEMENT

In this work, we used the GPT-4o model as an LLM-based role-overstepping evaluator to assess interaction trajectories in multi-agent systems. The corresponding evaluation procedure, prompting strategy, and manual validation process are described in detail in this paper. We did not use generative AI tools to formulate or refine research hypotheses, implement the research methods, clean or reformat datasets, or interpret experimental results. Tasks related to mathematical claims, mathematical proofs, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to improve the readability of the manuscript and assist with formatting. All AI-assisted manuscript edits were reviewed by the authors. We take full responsibility for the final content of this work, including the text, research claims, experimental data, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on improving role adherence and behavioral controllability in LLM-based multi-agent systems. All experiments are conducted on software-development benchmarks and model-generated interaction trajectories. No human subjects were recruited, and no sensitive or personally identifiable information was collected or processed. The manual validation of the LLM-based evaluator was conducted by the authors and did not involve external participants. The datasets used in this work are existing research benchmarks and are used solely for research and evaluation purposes.

REFERENCES

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*, 37:136037–136083, 2024.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *Advances in Neural Information Processing Systems*, 38, 2026.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pp. 20094–20136, 2024.

Yaxin Du, Yuzhu Cai, Yifan Zhou, Cheng Wang, Yu Qian, Xianghe Pang, Qian Liu, Yue Hu, and Siheng Chen. Swe-dev: Evaluating and training autonomous feature-driven software development. *arXiv preprint arXiv:2505.16975*, 2025.

Yicheng Fu, Raviteja Anantha, and Jianpeng Cheng. Camphor: Collaborative agents for multi-input planning and high-order reasoning on device. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pp. 33–51, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Yubin Kim, Chanwoo Park, Taehan Kim, Eugene Park, Samuel Schmidgall, Salman Rahman, Chunjong Park, Cynthia Breazeal, Xin Liu, Hamid Palangi, et al. Teambench: Evaluating agent coordination under enforced role separation. *arXiv preprint arXiv:2605.07073*, 2026.

Kai Konen, Sophie Jentzsch, Diaoulé Diallo, Peer Schütt, Oliver Bensch, Roxanne El Baff, Dominik Opitz, and Tobias Hecking. Style vectors for steering generative large language models. In *Findings of the Association for Computational Linguistics: EACL 2024*, pp. 782–802, 2024.

Bruce W Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering. In *International conference on learning representations*, volume 2025, pp. 90960–90985, 2025.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in neural information processing systems*, 36:41451–41530, 2023.

Jiahong Liu, Wenhao Yu, Quanyu Dai, Zhongyang Li, Jieming Zhu, Menglin Yang, Tat-Seng Chua, and Irwin King. Perfit: Exploring personalization shifts in representation space of llms. In *The Fourteenth International Conference on Learning Representations*, 2026.

Sheng Liu, Haotian Ye, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2023.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Yifu Qiu, Zheng Zhao, Yftah Ziser, Anna Korhonen, Edoardo M Ponti, and Shay B Cohen. Spectral editing of activations for large language model alignment. *Advances in Neural Information Processing Systems*, 37:56958–56987, 2024.

Abhishek Rath. Agent drift: Quantifying behavioral degradation in multi-agent llm systems over extended interactions. *arXiv preprint arXiv:2601.04170*, 2026.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, 2024.

Marco Valentino, Geonhee Kim, Dhairya Dalal, Zhixue Zhao, and André Freitas. Mitigating content effects on reasoning in language models through fine-grained activation steering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33314–33322, 2026.

Dimitri Von Rütte, Sotiris Anagnostidis, Gregor Bachmann, and Thomas Hofmann. A language model's guide through latent space. *arXiv preprint arXiv:2402.14433*, 2024.

Feiyang Wang, Hengguang Cui, Linghao Yang, Chi Shing Lee, Zhongkang Li, and Chenfeiyu Wen. Detecting and repairing role drift in multi-agent collaboration with lightweight protocols. In *2026 International Conference on Communication Networks and Machine Learning (CNML)*, pp. 1420–1424. IEEE, 2026.

Tianlong Wang, Xianfeng Jiao, Yinghao Zhu, Zhongzhi Chen, Yifan He, Xu Chu, Junyi Gao, Yasha Wang, and Liantao Ma. Adaptive activation steering: A tuning-free llm truthfulness improvement method for diverse hallucinations categories. In *Proceedings of the ACM on Web Conference 2025*, pp. 2562–2578, 2025.

Jan Wehner, Sahar Abdelnabi, Daniel Tan, David Krueger, and Mario Fritz. Taxonomy, opportunities, and challenges of representation engineering for large language models. *arXiv preprint arXiv:2502.19649*, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bo-Wen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhen-Ru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024. URL `https://api.semanticscholar.org/CorpusID:274859421`.

Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. Mas-gpt: Training llms to build llm-based multi-agent systems. *arXiv preprint arXiv:2503.03686*, 2025.

Biao Yi, Xueyu Hu, Yurun Chen, Shengyu Zhang, Hongxia Yang, and Fan Wu. Ecoagent: An efficient device-cloud collaborative multi-agent framework for mobile automation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 29838–29846, 2026.

Jinghao Zhang, Yuting Liu, Wenjie Wang, Qiang Liu, Shu Wu, Liang Wang, and Tat-Seng Chua. Personalized text generation with contrastive activation steering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7128–7141, 2025.

Shaolei Zhang, Tian Yu, and Yang Feng. Truthx: Alleviating hallucinations by editing large language models in truthful space. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8908–8949, 2024.

Guoling Zhou, Wenpei Han, Fengqin Yang, Li Wang, Yingcong Zhou, and Zhiguo Fu. Improving role consistency in multi-agent collaboration via quantitative role clarity. *arXiv preprint arXiv:2604.02770*, 2026a.

Yuxiang Zhou, Jichang Li, Yanhao Zhang, Haonan Lu, and Guanbin Li. Mobile-agent-rag: driving smart multi-agent coordination with contextual knowledge empowerment for long-horizon mobile automation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 29939–29947, 2026b.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A Experimental Details

A.1 Dataset Details

Software Requirement Description Dataset (SRDD) Qian et al. (2024): SRDD is a software-development benchmark introduced by ChatDev, consisting of 1,200 natural-language software requirement descriptions. The tasks cover five major domains, including Education, Work, Life, Game, and Creation, and are further organized into 40 subcategories with 30 tasks in each subcategory. The software requirements are constructed from software descriptions collected from major platforms, including Ubuntu, Google Play, Microsoft Store, and Apple Store, followed by LLM-based generation and human-guided refinement. In our experiments, we use SRDD exclusively for the final evaluation of role overstepping and end-to-end task performance.

SWE-Dev (Du et al., 2025): SWE-Dev is a large-scale dataset designed for feature-driven software development in real-world code repositories. It contains 14,000 training instances and 500 test instances, with the test set evenly divided into 250 Easy and 250 Hard tasks. Each instance contains a natural-language Project Requirement Document (PRD), an existing codebase with target functionality removed, and executable developer-authored unit tests for functional verification. In our experiments, we use SWE-Dev only for offline method preparation, including role steering vector construction, role-overstepping detector training, LoRA training-data collection, and validation-based hyperparameter selection.

A.2 Evaluation Metrics

Role-Overstepping Evaluation. Following prior work (Ye et al., 2025; Hu et al., 2025), we use GPT-4o (Hurst et al., 2024) as an LLM-based judge to identify role-overstepping trajectories. For each trajectory, the judge is provided with the role definitions and responsibility boundaries of the multi-agent framework, the current agent role, and its generated trajectory. The judge evaluates the trajectory semantically and determines whether the current agent actually performs behaviors that belong to another role's assigned responsibilities. Importantly, references to, evaluations of, or suggestions about another role's work are not considered role overstepping unless the current agent directly performs that role's responsibility. The judge focuses only on role-boundary adherence and ignores output quality, correctness, completeness, and formatting. A trajectory is labeled as *Role-Overstepping* if it contains at least one explicit overstepping behavior; otherwise, it is labeled as *Role-Adherent*. The prompt used for GPT-4o evaluation is provided in Fig. 6.

To assess the reliability of GPT-4o for role-overstepping evaluation, we conduct a manual validation for the three selected target roles in ChatDev-Qwen by stratified sampling 100 trajectories, including 50 classified as *Role-Adherent* and 50 as *Role-Overstepping* by GPT-4o. The trajectories are manually annotated without access to the GPT-4o predictions, following the same role-boundary criteria used in the automatic evaluation. As shown in Table 5, GPT-4o achieves 96%–100% agreement with the manual annotations across the three roles, indicating high consistency with human judgment.

Table 5: Agreement between GPT-4o and manual annotations on ChatDev-Qwen.

Role	Agreement (%)
CEO	100
CPO	98
Code Reviewer	96

Role Clarity Score. Following Zhou et al. (2026a), we measure role adherence using the semantic similarity between each generated trajectory and its assigned role instruction. Specifically, we use the same backbone LLM to extract the final-layer hidden states of both inputs, apply mean pooling to obtain sequence-level representations, and compute their cosine similarity as the RC Score. The reported RC Score is averaged over trajectories from the selected target roles. A higher RC Score indicates stronger semantic alignment between the generated behavior and the assigned role responsibilities.

SRDD Task Performance Metrics. Following Qian et al. (2024), we evaluate end-to-end task performance using the following four metrics:

- **Completeness** measures the software's ability to fulfill code completion in software development, quantified as the percentage of software without any "placeholder" code snippets. A higher score indicates a higher probability of automated completion.

Prompt for Role-Overstepping Evaluation

You are an impartial evaluator of role consistency in a multi-agent system. Your task is to determine whether the current agent exhibits role overstepping in its generated trajectory. Role overstepping occurs when the current agent actually performs behavior that belongs to the assigned responsibilities of another role rather than remaining within its own responsibility boundary. Judge only whether the agent follows its assigned role boundary, and do not consider output correctness, quality, completeness, formatting, or overall task performance. Judge the behavior semantically and do not classify a trajectory based on isolated keywords such as "code," "test," "design," "API," or role names. Discussing, evaluating, or suggesting what another role should do does not constitute role overstepping; role overstepping occurs only when the current agent actually performs a responsibility assigned to another role. Consider both the current role's responsibilities and the responsibilities assigned to the other roles in the multi-agent system. If the trajectory contains at least one explicit role-overstepping behavior, label the entire trajectory as Role-Overstepping; otherwise, label it as Role-Adherent. First, read the following role definitions and instructions of the multi-agent framework: {Role Instructions}. Then evaluate the following trajectory. Current Role: {Current Role}. Current Role Instruction: {Current Role Instruction}. Generated Trajectory: {Trajectory}. Output only one label: Role-Overstepping or Role-Adherent.

Figure 6: Prompt used for GPT-4o role-overstepping evaluation.

- **Executability** assesses the software's ability to run correctly within a compilation environment, quantified as the percentage of software that compiles successfully and can run directly. A higher score indicates a higher probability of successful execution.
- **Consistency** measures how closely the generated software code aligns with the original requirement description, quantified as the cosine similarity between the semantic embeddings of the textual requirements and the generated software code. A higher score indicates a greater degree of consistency with the requirements.
- **Quality** is a comprehensive metric that integrates various factors to assess the overall quality of software. In our evaluation, Quality is computed as the average of Completeness, Executability, and Consistency. A higher quality score suggests higher overall quality of the generated software.

A.3 Implementation Details

General Experimental Settings. All experiments are conducted on a server equipped with four NVIDIA Quadro RTX 8000 GPUs (48 GB each). We conduct experiments on ChatDev and MetaGPT using Qwen2.5-7B-Instruct and Llama-3.1-8B-Instruct as the backbone LLMs. Unless otherwise specified, we set the generation temperature T to 0.2 for all methods. We first evaluate 2,000 tasks from the SWE-Dev training split to examine the prevalence of role overstepping across different roles in ChatDev and MetaGPT. As shown in Table 6, role overstepping in both multi-agent frameworks is mainly concentrated in a small subset of roles, while the remaining roles exhibit only rare overstepping behavior. Therefore, all compared mitigation methods are applied only to the selected target roles, while the other roles retain their original settings. The statistics of role-overstepping trajectories and the computation of RC Score are also restricted to these target roles.

Few-shot Prompt Augmentation for Role Adherence

Please follow your assigned role responsibilities and avoid role overstepping. An example of role-adherent behavior from another task is: {Role-Adherent Example}. In contrast, an example of role-overstepping behavior from another task is: {Role-Overstepping Example}. Please avoid behavior similar to the role-overstepping example when completing the current task.

Figure 7: Prompt augmentation used for the Few-shot method.

Few-shot. For each selected target role, we append an additional role-boundary instruction to the end of the original role prompt, together with one role-adherent example and one role-overstepping

Table 6: Role-overstepping prevalence across different roles in ChatDev and MetaGPT on 2,000 SWE-Dev tasks using Qwen2.5-7B-Instruct. Checkmarks indicate the selected target roles for role-overstepping mitigation.

Framework	Role	RO Rate (%)	Selected
ChatDev	CEO	35.70	✓
	CPO	31.90	✓
	CTO	1.20	
	Programmer	1.40	
	Code Reviewer	18.50	✓
	Software Test Engineer	0.70	
MetaGPT	Product Manager	19.20	✓
	Architect	0.90	
	Project Manager	15.70	✓
	Engineer	0.60	
	QA Engineer	2.10	

example drawn from other tasks. The additional instruction reminds the agent to remain within its assigned responsibilities, while the two examples explicitly demonstrate representative behaviors that should be followed and avoided, respectively. These examples are used only to illustrate the role boundary and are not associated with the current task being solved. All other role instructions and the original multi-agent workflow remain unchanged. The prompt augmentation used for Few-shot is shown in Fig. 7.

Post-hoc. For each selected target role, we use GPT-4o as a post-hoc judge to examine the generated trajectory after each agent response. The judging prompt is identical to the role-overstepping evaluation prompt described above, including the same role definitions, responsibility boundaries, and judging criteria. The only difference is the final output instruction: instead of requiring GPT-4o to output only the binary label, we replace the final instruction with the one shown in Fig. 8, which additionally asks the judge to briefly summarize the identified role-boundary violation when the trajectory is classified as *Role-Overstepping*. If the trajectory is classified as *Role-Adherent*, the original response is retained without further modification. If it is classified as *Role-Overstepping*, we keep the original role prompt and task context unchanged and append an additional regeneration instruction containing both the previous response and the judge-generated error summary. The augmented prompt is then provided to the same agent to regenerate its response while avoiding the identified role-overstepping behavior. The additional regeneration instruction is shown in Fig. 9.

Post-hoc Judge Output Instruction

Determine whether the generated trajectory is Role-Overstepping or Role-Adherent. If the trajectory is Role-Adherent, output only the label Role-Adherent. If the trajectory is Role-Overstepping, output the label Role-Overstepping and briefly summarize the behavior in the response that exceeds the current role's assigned responsibilities.

Figure 8: Modified output instruction used in the Post-hoc method.

Post-hoc Regeneration Prompt

Your previous response was: {Previous Response}. The response was identified as role-overstepping for the following reason: {Role-Overstepping Error Summary}. Please regenerate your response while following your assigned role responsibilities and avoiding the identified role-overstepping behavior.

Figure 9: Additional instruction appended to the original agent prompt for response regeneration in the Post-hoc method.

LoRA FT. Following the role-consistency fine-tuning strategy of Zhou et al. (2026a), we independently fine-tune the backbone LLM for each selected target role using LoRA and role-adherent

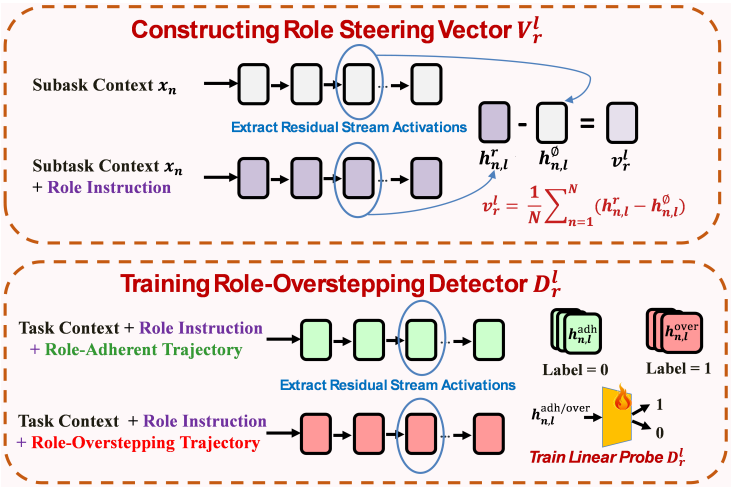


Figure 10: Offline preparation of DRAS, including role steering vector construction and role-overstepping detector training.

Table 7: Implementation configurations of DRAS for different backbone LLMs.

Configuration	Qwen2.5-7B	Llama-3.1-8B
Intervention layer	18	22
Base steering strength α	1.2	0.8

trajectories collected from SWE-Dev. We collect 800 role-adherent trajectories for each role and use them for role-adherence fine-tuning. We further select 300 tasks that exhibit role overstepping under the original backbone model as the validation set to evaluate the effectiveness of fine-tuning in mitigating role overstepping. The best checkpoint is selected based on performance on this validation set. LoRA is applied to the `q_proj` and `v_proj` modules, with rank $r = 16$ and scaling factor $\alpha_{\text{LoRA}} = 16$. The models are trained for 10 epochs with a learning rate of 5×10^{-5} and a batch size of 8. Training is performed in BF16 precision, while the backbone parameters remain frozen and only the LoRA parameters are updated.

Dynamic Role Activation Steering (DRAS). We first run 2,000 tasks from the SWE-Dev training split and, for each selected target role, collect all role-overstepping trajectories generated during these tasks together with an equal number of role-adherent trajectories. The collected trajectories are then split into an extraction set and a validation set with an 8:2 ratio. Following the procedure illustrated in Fig. 10 and described in Sec. 4.1 and Sec. 4.2 , we use the extraction set to construct contrastive samples, extract the corresponding activations, and build the role steering vectors and role-overstepping detectors. The layer-wise analysis and base steering strength α analysis reported in Fig. 4 are conducted on the validation set. For the layer-wise steering analysis in Fig. 4a, we evaluate only the role-overstepping trajectories in the validation set, and define the overstepping correction rate as the percentage of these trajectories that are corrected to Role-Adherent after steering. Unless otherwise specified, all experiments on SRDD use the configurations listed in Table 7. For the same backbone LLM, all selected target roles use the same intervention layer and base steering strength α.