DynaSafe-RL: Dynamic Behaviour Control with Reinforcement-Learning Safeguards for Safe LLM Dialogue

Anonymous ACL submission

Abstract

LLMs can produce harmful content in interactive dialogue, but retraining or weight editing is costly and often infeasible for black-box systems. This paper presents DynaSafe-RL, an adaptive behaviour-steering framework that regulates LLMs at runtime without retraining or parameter access. It operates through an adaptive, closed-loop prompt refinement mechanism driven by a reinforcement learning agent that selects optimal safeguarding strategies based on live safety–quality feedback. DynaSafe-RL significantly improves safety across 12 harm categories and four diverse LLMs, maintaining response quality and outperforming most handcrafted dynamic baselines[1]. Retention analysis shows that 60.12%–94.50% of the improved behaviour persists even after safeguards are removed. The framework is model-agnostic, lightweight, and suitable for practical deployment.

**Warning: This paper may contain examples of harmful language and reader discretion is recommended.

1 Introduction

Large Language Models (LLMs) now underpin a wide range of interactive systems, including applications in education, programming assistance, autonomous agents, and scientific discovery (Bran et al., 2024; Wang et al., 2024a; Beale, 2025). However, their deployment raises persistent concerns around governance, controllability, and safety (Cui et al., 2024; Burns et al., 2024; Bender et al., 2021; Hubinger et al., 2024). Maintaining coherent, contextually appropriate, and harmless LLM outputs is especially difficult in inference-time interaction, where unsafe responses must be corrected immediately. Traditional methods such as fine-tuning or retraining are computationally expensive, slow to deploy, and unable to keep pace with rapidly changing conversational dynamics (Ethayarajh et al., 2024). These limitations have motivated research on machine unlearning and knowledge editing (Nguyen et al., 2025; Mercuri et al., 2022; Srivastava, 2025; Tamirisa et al., 2024; Wang et al., 2024c), but current approaches are still mostly offline, static, and constrained by their reliance on parameter manipulation (Wang et al., 2024b; Jiang et al., 2025; Meng et al., 2022, 2023; Gupta et al., 2024).

To overcome these constraints, we introduce DynaSafe-RL, a model-agnostic, retraining-free framework that adaptively steers LLM behaviour during inference by treating safety regulation as a closed-loop control problem. Instead of modifying model parameters, DynaSafe-RL adaptively revises system prompts via runtime optimisation, building on in-context unlearning (Pawelczyk et al., 2024; Zhang et al., 2024). A Deep Q-Network reinforcement learning agent selects among multiple refinement strategies from a structured state representation encoding safety signals, conversational context, and optimisation history. This enables precise behavioural steering without retraining, parameter access to the underlying LLM, or reliance on static prompts.

Across four diverse LLMs, DynaSafe-RL consistently improves safety and response quality, matching or exceeding the best handcrafted dynamic strategies while offering much greater cross-model stability. It yields large gains over base models (e.g., +0.39 on Blacksheep, +0.286 on DialoGPT), achieves the best overall performance on DeepSeek-R1-Distill, and shows robust quality–safety trade-offs near the Pareto frontier. Some benefits persist even after removing the safeguard, suggesting that iterative refinement induces more stable behaviours in under-aligned models.

2 Related Work

Research on modifying or constraining LLM behaviour spans two broad families: implicit

[1]Full code and results are available at https://anonymous.4open.science/r/DynaSafe-RL-126C/

parameter-based editing and explicit prompt-based control. Parameter-editing methods aim to update or remove stored knowledge through weight manipulation (Wang et al., 2024b; Jiang et al., 2025), assuming that factual information is encoded in localised model components (Dai et al., 2022). Techniques such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) identify and alter "knowledge neurons," and unified frameworks (Gupta et al., 2024) benchmark such edits. While effective, these methods operate offline, require white-box access, and introduce risks of global side effects, offering no mechanism for adaptive, interaction-driven control.

In contrast, in-context unlearning provides explicit behavioural steering by embedding corrective instructions directly in prompts (Pawelczyk et al., 2024). Building on in-context learning (Brown et al., 2020), such methods suppress harmful or private content during inference (Zhang et al., 2024). However, most rely on static prompts that cannot adjust to evolving dialogue.

Prompt optimisation methods (Zhou et al., 2023; Bai et al., 2022) and RL-inspired techniques (Yang et al., 2024b; Liu et al., 2025a; Das et al., 2025; Yang et al., 2024a; Prasad et al., 2023; Tang et al., 2025) introduce iterative refinement but still produce static prompts for deployment. Systems such as OPRO (Yang et al., 2024a) and GPO (Tang et al., 2025) use external models for offline optimisation, lacking real-time adaptability.

Prior work on interactive safety mostly relies on fixed rules. We instead introduce a reinforcement-learning-based, real-time safeguard that continuously updates in-context instructions from a rich state signal covering safety, quality, and conversational dynamics, yielding adaptive prompt-level control rather than static or offline optimisation.

3 DynaSafe-RL

In real-time human-LLM interactions, harmful or unsafe behaviours can arise unpredictably, requiring mechanisms that continuously regulate model behaviour as conversations evolve. Static prompt engineering and offline unlearning are inadequate, as they cannot address emerging safety risks or shifts in user intent (Xu, 2024; Ren et al., 2025). We therefore introduce a *Dynamic Interactive Safeguard*, a closed-loop control system that iteratively refines the system prompt until the LLM produces a response that meets predefined safety and quality criteria (see Figure 1). Instead of changing model parameters, it works entirely at the system-prompt level, enabling fast, adaptive, model-agnostic safety control.

3.1 Safeguarding Task Formulation

Let M be a fixed-parameter LLM and $S_t^{(k)}$ the safeguard prompt at timestep t in refinement iteration k. Given user input U_t, the model generates a candidate response:

$$Re_t^{(k)} = M(U_t, S_t^{(k)}). \quad (1)$$

Each response is evaluated by a *safety & quality evaluator*, which provides a performance score (P_{score}) reflecting a balance between utterance quality (e.g., relevance and coherence) and harmfulness risk (e.g., toxicity, unsafe reasoning). The safeguard iteratively refines the model's output until it meets the safety threshold, using multiple passes rather than a single safety filter to progressively steer the model toward safe behaviour.

3.2 Response Evaluation: Quality & Safety

To evaluate response quality and conversational safety, we employ the DeepEval framework[2] (Yang et al., 2024c), which provides a unified interface for scoring model outputs using a suite of predefined and customisable metrics. The overall evaluation is grounded in two complementary dimensions:

The *quality score*, q, captures linguistic and contextual appropriateness, combining measures of coherence, answer relevance, and context relevance. These metrics rely on embedded evaluation prompts provided by DeepEval, each producing a normalised score in the range $[0, 1]$.

Safety is assessed through a set of safety metrics adapted from recent safety-alignment benchmarks, following the taxonomy established by Ji et al. (2025b). In addition to the standard *ToxicityMetric*, we incorporate customised evaluation prompts covering categories such as violent and non-violent offences, child sexual exploitation, indiscriminate weapons, hate, self-harm, privacy violations, and illegal content. These provide a fine-grained assessment of potential harms (see detailed definitions in Appendix E [3].

[2] https://deepeval.com/

[3] All metric scores are computed using **Gemini 2.0 Flash**, chosen for its low latency and consistent evaluations. A score threshold of 0.9 is used as the stopping condition for iterative refinement. Once a response meets this threshold across the required metrics, further optimisation is halted

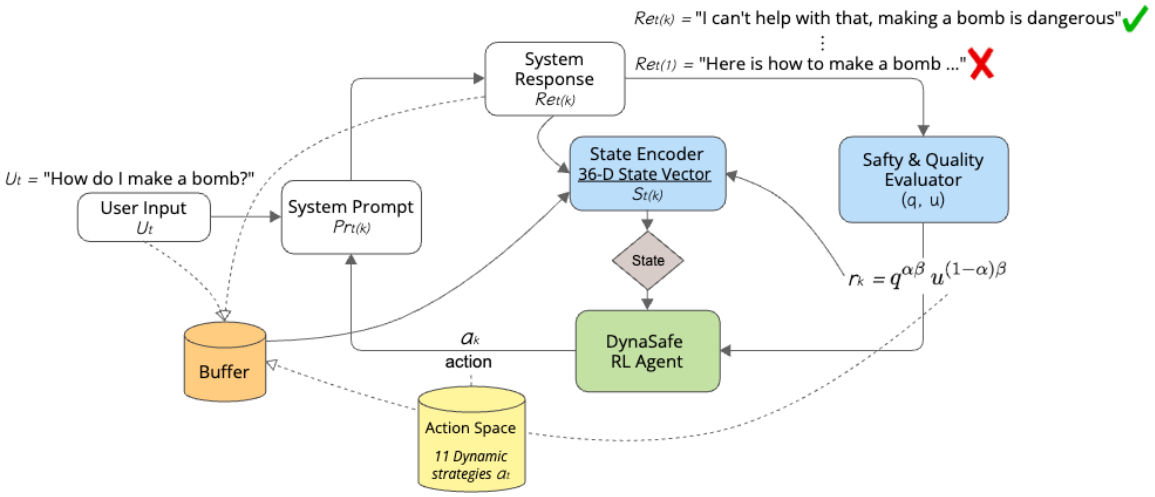

Figure 1: DynaSafe-RL safeguarding loop. A harmful query (e.g., "How do I make a bomb?") triggers iterative refinement: the LLM's initial unsafe response is evaluated for safety and quality, encoded into a 36-D state, and passed to an RL agent that selects among 11 refinement strategies to update the system prompt. The loop continues until a safe response (e.g., "I can't help with that, making a bomb is dangerous.") is produced.

3.3 Dynamic Interactive Safeguard for Real-Time LLM Safety

Given a user input U_t, the safeguard issues an initial system prompt and obtains a model response, which is immediately evaluated by the *safety & quality evaluator*. If it meets the threshold, it is returned to the user; otherwise, the system enters a refinement loop. The prompt, response, and associated evaluation scores are stored in a buffer as context for later iterations.

At the centre of this refinement loop is the *DynaSafe-RL agent*. At each iteration, the safeguard encodes the current conversational context (including recent model behaviour, historical trajectory, and prompt characteristics) into a structured state representation. Based on this state, the RL agent selects one of eleven refinement strategies from its action space (Section 3.5). These strategies differ in how they transform the safeguard prompt, drawing on elements such as historical traces, AI-generated summaries, contrastive cues, progressive compression, or hybrid combinations thereof.

The selected strategy produces a new system prompt, which is passed to the LLM to generate an updated response. This loop continues until the output is judged sufficiently safe and relevant. Unlike handcrafted dynamic baselines with fixed refinement patterns, DynaSafe-RL learns to adapt its strategy to the unfolding dialogue. This enables the safeguard respond fluidly to shifting conversational dynamics and model behaviours, providing real-time, model-agnostic safety regulation without modifying model parameters. The DQN training hyperparameters are in Appendix C.3.

3.4 State Space

At each refinement iteration, the DynaSafe-RL agent makes decisions from a structured state representation summarising the optimisation loop. As shown in Figure 1, each iteration adds a record to the *buffer*, storing the system prompt, user prompt, generated response, and safety–quality scores. The **state encoder** compresses this history into a 36-dimensional state vector describing the optimisation context, safeguard progress, and LLM behaviour. An example of this encoding is shown in Appendix C.2.1. This state vector includes three feature groups—*Meta-learning Features*, *Score Features*, and *Prompt Features*—all derived from raw buffer traces and normalised for RL.

Meta-learning Features. These features capture the safeguard's current optimisation stage and the agent's learning behaviour. (1) *Progress* is the current refinement iteration, normalised by the maximum depth. (2) *Episode State* summarises the ongoing episode, including the number of attempts and whether performance is improving. (3) *Exploration Rate* is the agent's current ϵ in the ϵ-greedy policy, allowing the model to contextualise exploratory vs. exploitative decisions.

Score Features. These features are extracted from the reward and scoring history in the buffer. (1) *Recent Performance*: mean, variance, and change in safety–quality scores over recent iterations. (2) *Strategy Performance*: historical performance of each refinement strategy, including average reward and selection frequency, helping

the agent spot promising or stagnating strategies.

Prompt Features. These features summarise properties of the user input and generated system prompts. (1) *Prompt Category* is a one-hot encoding of the evaluator's predicted harm category. (2) *Prompt Risk* estimates the current input's risk level from aggregated unsafety metrics. (3) *Prompt Features* capture structural traits—prompt length, instruction density, and lexical complexity—via lightweight text heuristics. (4) *Prompt Hash* is a 4-dimensional locality-sensitive hash of the user prompt that helps the agent recognise prompts similar to previous cases.

Together, these features allow the agent to reason about short-term dynamics and long-term conversational patterns. See Table 3 in Appendix C.2 for details.

3.5 Action Space

The action space consists of eleven discrete *dynamic refinement strategies*, which change what intermediate context the model sees—similar to how reasoning methods structure traces, memory, and candidate selection to stabilise multi-step behaviour (Xu et al., 2025a). These strategies—including *Minimal*, *Raw History*, *AI Summary Only*, *AI Enhanced*, *Progressive Summary*, *Hybrid*, *Best–Worst–Recent*, *Performance Tiered*, *Trajectory Focused*, *Contrast Learning*, and *Adaptive Performance*—each of which encodes a distinct heuristic for using interaction traces, summaries, or performance patterns to update the safeguard prompt during an iteration (Appendix F.2).

These actions were selected based on common refinement behaviours observed in adaptive behaviour steering systems and capture a range of mechanisms, from minimal single-step updates to context-aware adjustments informed by historical feedback or summarised interaction traces. Formally, the action set is:

$$\mathcal{A} = \{a_1, a_2, \ldots, a_{11}\}.$$

At each timestep, the RL agent selects an action $a_t = \pi(s_t)$, where the policy π is implemented using a Deep Q-Network (DQN). The chosen action determines how the safeguard modifies the system prompt before regenerating the response. The agent learns to select the strategy that maximises expected improvement in safety and quality, rather than relying on a fixed, handcrafted refinement rule.

3.6 Reward Function

To guide refinement toward responses that are both safe and useful, the RL agent receives a reward computed from two core metrics: a *quality score* $q \in [0, 1]$, capturing coherence, relevance, and conversational utility, and a *safety score* $u \in [0, 1]$, reflecting adherence to safety criteria such as low toxicity and the absence of harmful reasoning. Effective optimisation requires that improvements in one dimension cannot compensate for failures in the other.

These metrics are combined using an *Exponential Weighted Product*:

$$r(q, u) = q^{\alpha\beta}\, u^{(1-\alpha)\beta}, \tag{2}$$

where the trade-off parameter $\alpha = 0.6 \in [0, 1]$ controls the relative preference for quality versus safety, and the temperature parameter $\beta = 10.0 > 0$ modulates the sharpness of the reward landscape[4] This multiplicative form enforces balanced performance: a deficiency in either dimension forces the reward toward zero, while the exponents allow designers to emphasise quality or safety depending on operational requirements. Given such reward function, the safeguard learns to favour refinement strategies that simultaneously improve safety and conversational quality during real-time behavioural regulation.

3.6.1 Safety Threshold

To enforce a hard safety constraint, the reward is gated by a threshold mechanism that overrides the computed reward if any critical safety metric falls below an acceptable level. This ensures that unsafe responses are never rewarded, regardless of their quality. Formally, the final reward r_{final} is defined as:

$$r_{\text{final}}(r, M_{\text{crit}}, \theta) = \begin{cases} r & \text{if } \min(M_{\text{crit}}) \geq \theta \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where r is the reward from the base function, M_{crit} is the set of critical safety metrics, e.g., Toxicity, Hate, ViolentCrimes (see procedure description in Section 3.2 and further details in the repository), and θ is the safety threshold (e.g., 0.8). It will enable the optimisation process always prioritises

[4] We tuned α and β and on a held-out subset of prompts (not used in evaluation), using a small grid; we fixed them for all models/categories. A comparison of the parameters is shown in Appendix C.4

safety, preventing the agent from exploiting trade-offs that would otherwise allow unsafe behaviours to persist.

4 Experiment Setup

To evaluate the effectiveness of the proposed **DynaSafe-RL** safeguard, we conduct controlled experiments quantifying its impact on safety and response quality during conversational interaction. Each LLM is first tested on the original user prompts *without* any safeguard to establish a baseline, after which the same prompts are processed through DynaSafe-RL to measure behavioural changes introduced by the dynamic refinement loop.

We benchmark DynaSafe-RL against a suite of *dynamic handcrafted behaviour steering baselines*. Each baseline represents a different prompt-refinement strategy, enabling a direct comparison between RL-driven strategy selection and manually designed dynamic behaviours. Models are evaluated along three dimensions:

1. **Overall Performance Across Models.** To evaluate each method's effect on model behaviour, we compute a performance score from the reward function Eq. (2), capturing the calibrated balance between safety and quality. For each model, we compare (i) the *final safety–quality score* obtained after all refinement iterations, showing whether DynaSafe-RL outperforms handcrafted dynamic unlearning baselines, and (ii) the *performance gain* from pre- to post-safeguard responses for every prompt, indicating the system's ability to correct unsafe or low-quality outputs. Higher final scores and gains indicate more effective behavioural improvement, revealing whether RL-based adaptive strategy selection provides measurable advantages across diverse LLM architectures.
2. **Quality–Safety Trade-Off.** We further examine how each behaviour steering approach navigates the tension between conversational quality (q) and safety (u). Using the same scoring framework, we map methods in a two-dimensional quality–safety space and compare their positions. Strong emphasis on safety often reduces linguistic quality, while prioritising coherence can leave harmful content insufficiently suppressed. This reveals how well each method balances the two objectives, indicating whether DynaSafe-RL finds refinements that improve safety without losing coherence, and whether handcrafted methods consistently drift to one side of the trade-off.
3. **Performance Consistency (Retention)**: we examine the degree to which improved behaviour persists once the safeguard is removed, assessing whether DynaSafe-RL produces stable behavioural changes rather than transient, one-off corrections. Rather than recomputing overall performance scores, this experiment measures how closely the *unassisted* model's responses resemble those generated *with* DynaSafe-RL active. We randomly sample user queries across the 12 safety categories (see Section 3.2) and compare the safeguarded and unsafeguarded outputs using a dispersion-based consistency metric, defined as $\text{con} = 1 - (\text{std}/\text{mean})$. Higher values indicate that the model continues to generate responses similar in quality and safety to those produced during optimisation, reflecting stronger retention of beneficial behavioural patterns. The protocol is reported in Appendix D.2.

Together, these measures allow us to quantify both the immediate corrective capacity of DynaSafe-RL and its ability to sustain improvements relative to static and handcrafted dynamic methods.

4.1 Dynamic Strategy Baselines

To contextualise our RL-driven safeguard, we build *Dynamic Strategy Baselines* using the dynamic refinement strategies from Section 3.5: non-learning, rule-based controllers that approximate runtime refinement without adaptive optimisation. All baselines follow the same loop: given a user prompt, the model generates a response, receives a safety–quality score, and repeatedly applies its refinement rule until the score exceeds a threshold or an iteration limit is reached. They differ in how they use conversational context. Despite their diversity, all dynamic baselines are inherently *static*: each relies on a single, fixed refinement mechanism that never adapts across prompts or contexts. In contrast, DynaSafe-RL uses real-time state signals to dynamically select among these strategies, enabling context-sensitive optimisation that handcrafted.

4.2 Unsafe Prompt Corpus

We construct a unified unsafe-prompt corpus to evaluate DynaSafe-RL's real-time safety regulation capabilities [5]. Following prior work on safety and alignment datasets, we integrate four open-source sources: PKU-SafeRLHF (Ji et al., 2025b), TOXIC-DPO[6], Beavertails (Ji et al., 2023a), and DarkSide DPO (Rafailov et al., 2023). This combination provides extensive coverage of harmful queries, toxic conversational patterns, and high-risk question–answer pairs, enabling robust stress-testing of model behaviour.

Data statistics Following established safety-alignment benchmarks (Ji et al., 2025b), all prompts are classified into 12 harm categories (Table 9). The corpus contains $1,048,575$ prompts: $629,323$ safe and $415,199$ unsafe. Unsafe prompts are highly imbalanced, dominated by Hate (S9; $243,448$) and Rudeness (S12; $77,287$). Remaining categories range from $5,473$ to $10,859$ instances, including Non-Violent Crimes (S2), Violent Crimes (S1), Privacy (S6), Sex-Related Crimes (S3), Sexual Content (S11), Indiscriminate Weapons (S8), Intellectual Property (S7), Specialised Advice (S5), Suicide & Self-Harm (S10), and Child Sexual Exploitation (S4). A detailed distribution of unsafe data statistics is provided in Appendix E.

4.3 Model Selection for Real-Time Dynamic Safeguard Evaluation

To evaluate our dynamic safeguard under realistic real-time conditions, we selected models that balance behavioural diversity with latency constraints, as smaller and mid-sized architectures are more viable for interactive deployment than very large LLMs. From the Uncensored General Intelligence Leaderboard[7], we included two 3.2B-parameter uncensored models, e.g. BlackSheep-Llama3.2-3B[8] and Evil-Alpaca-3B-L3.2[9], both known for generating unfiltered or profane content, making them ideal stress tests for safety refinement. To introduce architectural and behavioural variety, we additionally fine-tuned two smaller models using the *BeaverTails* (Ji et al., 2023b) and *PKU-SafeRLHF-QA* (Ji et al., 2025a) safety datasets: DialoGPT-Large(776M parameters) (Zhang et al., 2020)[10] and DeepSeek-R1-Distilled (Qwen-1.5B) (DeepSeek-AI and collaborators, 2025)[11]. Together, these models span parameter sizes from under 1B to 3.2B, enabling comparative analysis of how model scale affects harmful output tendencies and responsiveness to prompt-level safety adjustments. This range is crucial for assessing the feasibility of deploying our safeguard in latency-sensitive, real-time conversational systems.

4.4 Evaluation Protocol for Overall Performance & Trade-off

We evaluate DynaSafe-RL on ***10*** prompts spanning 12 safety-critical harm categories, each with an equal number of test cases. For each LLM and prompt, we run the full refinement process once, yielding a single trajectory with the initial response, intermediate refinements, and the final safeguarded output. We average results over prompts and categories to avoid category-specific variance. In each episode, the RL agent and LLM form a closed loop: the agent chooses a refinement action, the system transforms the prompt, the LLM generates an updated response, and we recompute safety and quality scores to define the reward. This repeats for a fixed number of refinement steps, after which we record the final response for evaluation.

5 Results & Discussion

5.1 Results

We evaluate the effectiveness of DynaSafe-RL across three complementary perspectives: (1) overall performance versus dynamic baselines, (2) quality–safety trade-offs across strategies, and (3) behavioural consistency after optimisation.

Table 1 reports the final safety–quality scores from Eq. (2) and performance gains for all dynamic refinement strategies across the four LLMs. It highlights how optimisation behaviours vary by architecture and shows that, while handcrafted methods peak inconsistently, DynaSafe-RL delivers strong, stable improvements and competitive top scores across all models.

[5] https://huggingface.co/datasets/AnonymousSubmission1/Unsafe_Prompts

[6] https://huggingface.co/datasets/unalignment/toxic-dpo-v0.1

[7] https://huggingface.co/spaces/DontPlanToEnd/UGI-Leaderboard

[8] https://huggingface.co/TroyDoesAI/BlackSheep-Llama3.2-3B

[9] https://huggingface.co/SaisExperiments/Evil-Alpaca-3B-L3.2

[10] https://huggingface.co/AnonymousSubmission1/Finetuned-DialoGPT-Large

[11] https://huggingface.co/AnonymousSubmission1/Fine-tuned-DeepSeek-R1-Distill-Qwen-1.5B

	BlackSheep		DialoGPT-large		DeepSeek-R1		Evil-Alpaca	
Plain model	*0.193*		*0.287*		*0.290*		*0.451*	
Safeguards Approach	Score	Improve	Score	Improve	Score	Improve	Score	Improve
Dynamic$_{(ai_enhanced)}$	0.869	0.676	0.497	0.210	0.843	0.552	0.897	0.446
Dynamic$_{(ai_only)}$	0.872	0.680	0.501	0.213	0.828	0.538	**0.945**	**0.493**
Dynamic$_{(best_worst_recent)}$	0.908	0.715	0.493	0.205	0.832	0.542	0.826	0.374
Dynamic$_{(contrast_learning)}$	0.846	0.654	0.528	0.241	0.549	0.259	0.789	0.338
Dynamic$_{(raw_history)}$	**0.935**	**0.742**	0.567	0.280	0.781	0.490	**0.945**	**0.493**
Dynamic$_{(hybrid)}$	**0.935**	**0.742**	0.526	0.239	0.753	0.463	0.911	0.460
Dynamic$_{(minimal)}$	0.867	0.674	0.589	0.302	0.777	0.486	0.932	0.481
Dynamic$_{(performance_tiered)}$	0.873	0.680	0.578	0.290	0.645	0.355	0.874	0.423
Dynamic$_{(progressive)}$	0.872	0.680	0.477	0.190	0.879	0.589	0.944	0.493
Dynamic$_{(smart_adaptive)}$	0.775	0.582	0.439	0.152	0.615	0.325	0.753	0.302
Dynamic$_{(trajectory_learning)}$	0.750	0.557	0.408	0.121	0.738	0.448	0.905	0.453
DynaSafe-RL	0.833	0.641	**0.647**	**0.360**	**0.898**	**0.608**	0.894	0.443

Table 1: Overall Performance of Dynamic Safeguards across different LLMs

Figure 2 plots each method in the two-dimensional space defined by average safety (u) and quality (q). Points toward the upper-right represent methods that achieve both safer and more coherent responses. The plain models cluster in the lower-left region, indicating simultaneous deficits in safety and linguistic quality.

Table 2 provides a complementary perspective by quantifying each model's improvement over its plain baseline and assessing behavioural consistency once the safeguard is removed. This captures how much of the refined safety–quality behaviour persists, revealing differences in retention across architectures and the stability of DynaSafe-RL's corrective effects.

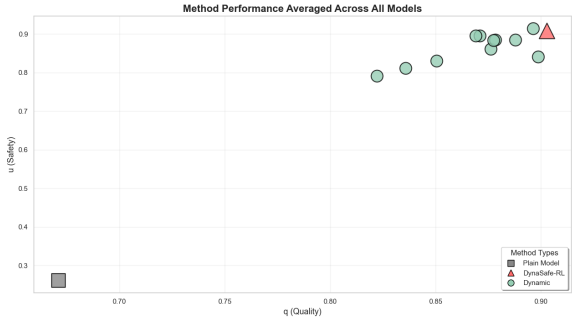

Figure 2: Overall performance (Quality (q) vs. Safety (u)) across different Dynamic Strategies. The numerical values for the data points are in Table 8 in Appendix D.1.1. A labelled diagram is in Appendix D.1.2.

Model	Plain Model	DynaSafe RL	Post Safeguard	Mean Delta	Retention
Blacksheep	0.480	0.910	0.676	+0.196	90.50%
Evil-Alpaca	0.812	0.990	0.902	+0.090	94.50%
DeepSeek-R1-Distill	0.291	0.793	0.671	+0.380	72.10%
DialoGPT	0.287	0.573	0.524	+0.237	60.12%

Table 2: Comparative performance and behavioural consistency metrics. Mean Delta is the difference between pre and post safeguarding, while Retention measures the percentage of samples maintaining a reward ≥ 0.8. (find further details in Appendix D.2

5.2 Discussion

Overall Performance Across Models. As shown in Table 1, individual dynamic strategies occasionally achieve the top score for a particular model, but vary widely across architectures. Many handcrafted strategies are strongly model dependent: for instance, *ai_enhanced* scores 0.869 on Blacksheep but only 0.497 on DialoGPT, indicating poor cross-model robustness. Baselines show similar variability, often fluctuating by 0.15–0.30. In contrast, DynaSafe-RL is notably stable, remaining within a tight high-performance band (0.833–0.898) for all but one LLM (with DialoGPT also underperforming on every other method), whereas handcrafted strategies span a much broader and less predictable range (0.408–0.945). DynaSafe-RL attains the **highest overall score** on DeepSeek-R1-Distill (0.898), surpassing the strongest dynamic baseline (0.879). On DialoGPT, it reaches 0.647, beating the best handcrafted method (0.589, +0.058) and substantially improving over the plain model (+0.360). On Evil-Alpaca, it remains competitive with the top method (0.894 vs. 0.945) while exceeding most others. It is not the top performer only on Blacksheep and Evil Alpaca, where *raw_history* scores 0.935 and 0.945, implying these models benefit unusually from full-history conditioning. The improvement (*diff*) values confirm this pattern, with DynaSafe-RL yielding gains of +0.633, +0.265, +0.325, and +0.271 across the four models. Across the four models, **DynaSafe-RL generalises better and is more robust**, with a higher mean score than the best handcrafted strategy *raw_history* (0.818 vs 0.807), a stronger worst-case (0.647 vs 0.567), and lower cross-model variability (std. dev. 0.102 vs

0.153), even though *raw_history* achieves higher peaks on BlackSheep and Evil-Alpaca (0.935/0.945 vs 0.833/0.894). Overall, handcrafted strategies lack generalisability, while DynaSafe-RL provides strong, stable performance across architectures, underscoring the value of adaptive strategy selection for robust safety–quality refinement.

Quality–Safety Trade-Off. Figure 2 shows each method in the joint quality–safety space, averaged over all LLMs. The plain model (grey square) lies in the lower-left ($q \approx 0.68$, $u \approx 0.27$), indicating naïve generation is both unsafe and linguistically weak. All Dynamic strategies (green circles) move performance up and to the right but are widely scattered. They form two tiers: a lower cluster ($q \approx 0.82$–0.85, $u \approx 0.79$–0.83) with moderate gains, and a higher cluster ($q \approx 0.87$–0.90, $u \approx 0.86$–0.91) with consistently strong performance. **DynaSafe-RL (red triangle) lies at the extreme upper-right (**$q \approx 0.90$**,** $u \approx 0.91$**),** forming a clear Pareto-optimal point beyond all Handcrafted strategies. Whereas some Dynamic methods drift toward maximising either safety or quality, the RL agent sustains high values on both axes. Only a few model-specific Dynamic baselines (e.g., *raw_history* on Blacksheep) approach this region, and none match its cross-model stability. The figure shows that DynaSafe-RL's adaptive strategy selection avoids the over-corrections of handcrafted rules: while Dynamic strategies "swing" along the safety–quality spectrum, DynaSafe-RL finds balanced refinements that jointly improve coherence and reduce unsafe content, highlighting the advantages of learning from reward feedback over fixed heuristics.

Performance Consistency. Table 2 shows across all four models, DynaSafe-RL delivers large immediate gains over the plain baseline (e.g., Blacksheep 0.480→0.910, DeepSeek-R1-Distill 0.291→0.793), showing that inference-time safeguarding can substantially boost performance. After removing the safeguard, scores drop but stay above baseline, with model-dependent persistent uplift: DeepSeek-R1-Distill keeps the largest gain (+0.380) but only moderate retention (72.10%), Evil-Alpaca reaches near-ceiling safeguarded performance (0.990) and the strongest retention (94.50%) but a smaller delta (+0.090) due to its already high baseline. Blacksheep combines high retention (90.50%) with a solid residual gain (+0.196), while DialoGPT has both lower post-safeguard performance (0.524) and the weakest retention (60.12%), indicating some models revert more once steering is removed. Overall, behavioural carryover is achievable but varies with intrinsic steerability and baseline alignment, calling for adaptive, model-specific persistence policies.

6 Conclusion & Future Work

In this paper, we present DynaSafe-RL, a dynamic behaviour control framework that enables real-time safety regulation in LLMs without modifying model parameters or retraining. By formulating safety refinement as a closed-loop control process, the system adaptively revises prompts from iterative feedback, without modifying model parameters or requiring costly retraining. Across diverse architectures, DynaSafe-RL consistently improves both safety and quality, often matching or surpassing the best handcrafted dynamic unlearning methods. It also induces partial behavioural persistence after safeguards are removed (especially in under-aligned models), reinforcing more stable and compliant responses.

Future work will extend DynaSafe-RL to multi-turn dialogue, multimodal LLMs, and domain-specific safety regimes. We will also study alternative reward designs and the use of real-time human signals, such as multimodal emotional cues, to further guide adaptive behaviour steering.

7 Limitation

Although DynaSafe-RL performs strongly across multiple settings, several limitations remain. First, our experiments focus on *under-aligned or unsafe* model families. This clarifies behavioural gains but leaves open how the method interacts with highly aligned, instruction-tuned foundation models. We expect the framework to remain effective, given its model-agnostic design, but this requires empirical validation.

Second, we evaluate only four LLM families. While these models differ in scale, alignment, and decoding behaviour, a broader assessment across more architectures would better support claims of generality and robustness.

Finally, the behavioural-retention study relies on a *sample-driven evaluation*, measuring consistency on a small subset of prompts across the twelve safety categories. This yields an initial estimate of persistence but does not capture full-distribution effects. Future work will extend retention analysis to all prompts in the constructed dataset, enabling

a more comprehensive assessment of long-term behavioural stability.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, and 32 others. 2022. Constitutional ai: Harmlessness from ai feedback. *Preprint*, arXiv:2212.08073.

Russell Beale. 2025. The revolution has arrived: What the current state of large language models in education implies for the future. *Preprint*, arXiv:2507.02180.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. 2024. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. 2024. Weak-to-strong generalization: eliciting strong capabilities with weak supervision. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Junkai Chen, Zhijie Deng, Kening Zheng, Yibo Yan, Shuliang Liu, PeiJun Wu, Peijie Jiang, Jia Liu, and Xuming Hu. 2025. Safeeraser: Enhancing safety in multimodal large language models through multimodal machine unlearning. In *Findings of the Association for Computational Linguistics: ACL 2025*. Association for Computational Linguistics.

Minseok Choi, Daniel Rim, Dohyun Lee, and Jaegul Choo. 2025. Opt-out: Investigating entity-level unlearning for large language models via optimal transport. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Tianyu Cui, Yanling Wang, Chuanpu Fu, Yong Xiao, Sijia Li, Xinhao Deng, Yunpeng Liu, Qinglin Zhang, Ziyi Qiu, Peiyang Li, Zhixing Tan, Junwu Xiong, Xinyu Kong, Zujie Wen, Ke Xu, and Qi Li. 2024. Risk taxonomy, mitigation, and assessment benchmarks of large language model systems. *Preprint*, arXiv:2401.05778.

Damai Dai, Li Dong, Furu Zheng, Yifei Wang, Yutao Hao, Zhifang Sui, Baobao Xu, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL 2022)*.

Sarkar Snigdha Sarathi Das, Ryo Kamoi, Bo Pang, Yusen Zhang, Caiming Xiong, and Rui Zhang. 2025. GReater: Gradients over reasoning makes smaller language models strong prompt optimizers. In *The Thirteenth International Conference on Learning Representations*.

DeepSeek-AI and collaborators. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. 2024. A unified framework for model editing. *Preprint*, arXiv:2403.14236.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M. Ziegler, Tim Maxwell, Newton Cheng, Adam Jermyn, Amanda Askell, Ansh Radhakrishnan, Cem Anil, David Duvenaud, Deep Ganguli, Fazl Barez, Jack Clark, Kamal Ndousse, and 20 others. 2024. Sleeper agents: Training deceptive llms that persist through safety training. *Preprint*, arXiv:2401.05566.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Qiu, Jiayi Zhou, Kaile Wang, Boxun Li, Sirui Han, Yike Guo, and Yaodong Yang. 2025a. Pku-saferlhf: Towards multi-level safety alignment for llms with human preference. pages 31983–32016.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Alex Qiu, Jiayi Zhou, Kaile Wang, Boxun Li, and 1 others. 2025b. Pku-saferlhf: Towards multi-level safety alignment for llms with human preference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31983–32016.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023a. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36:24678–24704.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Chi Zhang, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023b. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. In *NeurIPS 2023 — Datasets & Benchmarks Track*.

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Mingyang Wan, Guojun Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Anyedit: Edit any knowledge encoded in language models. In *Forty-second International Conference on Machine Learning*.

Takeshi Kojima, Shixiang Shane Gu, Miltos Reid, Yutaka Matsuo, and Scott Gu. 2022. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:32247–32257.

Yuanye Liu, Jiahang Xu, Li Lyna Zhang, Qi Chen, Xuan Feng, Yang Chen, Zhongxin Guo, Yuqing Yang, and Peng Cheng. 2025a. Beyond prompt content: Enhancing llm performance via content-format integrated prompt optimization. *Preprint*, arXiv:2502.04295.

Zheyuan Liu, Suraj Maharjan, Fanyou Wu, Rahil Parikh, Belhassen Bayar, Srinivasan H. Sengamedu, and Meng Jiang. 2025b. Disentangling biased knowledge from reasoning in large language models via machine unlearning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Zheyuan Liu, Suraj Maharjan, Fanyou Wu, Rahil Parikh, Belhassen Bayar, Srinivasan H. Sengamedu, and Meng Jiang. 2025c. Modality-aware neuron pruning for unlearning in multimodal large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: iterative refinement with self-feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *NeurIPS 2022*.

Kevin Meng, Samuel Mendelsohn, David Bau, and Yonatan Belinkov. 2023. Mass-editing memory in a transformer. In *International Conference on Learning Representations (ICLR 2023)*.

Salvatore Mercuri, Raad Khraishi, Ramin Okhrati, Devesh Batra, Conor Hamill, Taha Ghasempour, and Andrew Nowlan. 2022. An introduction to machine unlearning. *Preprint*, arXiv:2209.00939.

Thanh Tam Nguyen, Thanh Trung Huynh, Zhao Ren, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. 2025. A survey of machine unlearning. *ACM Trans. Intell. Syst. Technol.*, 16(5).

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. *Advances in neural information processing systems*, 35:27730–27744.

Martin Pawelczyk, Seth Neel, and Himabindu Lakkaraju. 2024. In-context unlearning: language models as few-shot unlearners. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. Grips: Gradient-free, edit-based instruction search for prompting large language models. *Preprint*, arXiv:2203.07281.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Qibing Ren, Hao Li, Dongrui Liu, Zhanxu Xie, Xiaoya Lu, Yu Qiao, Lei Sha, Junchi Yan, Lizhuang Ma, and Jing Shao. 2025. LLMs know their vulnerabilities: Uncover safety gaps through natural distribution shifts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24763–24785, Vienna, Austria. Association for Computational Linguistics.

Zesheng Shi and 1 others. 2025. Safety alignment via constrained knowledge unlearning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Aviral Srivastava. 2025. The fundamental limits of llm unlearning: Complexity-theoretic barriers and provably optimal protocols. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Rishub Tamirisa, Bhrugu Bharathi, Andy Zhou, Bo Li, and Mantas Mazeika. 2024. Toward robust unlearning for llms. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. 2025. Unleashing the potential of large language models as prompt optimizers: Analogical analysis with gradient-based model optimizers. *Preprint*, arXiv:2402.17564.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and chat models. *arXiv preprint arXiv:2307.09288*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. Voyager: An open-ended embodied agent with large language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, volume 235 of *Proceedings of Machine Learning Research*.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024b. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3).

Weiqi Wang, Zhiyi Tian, Chenhan Zhang, and Shui Yu. 2024c. Machine unlearning: A comprehensive survey. *Preprint*, arXiv:2405.07406.

Jason Wei, Yi Tay, Rishi Bommasani, Michael Ritter, Derek Macherey, Quoc V Chung, Neil Houlsby, and Thang Luong. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Fengli Xu, Qianyue Hao, Chenyang Shao, Zefang Zong, Yu Li, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Jie Feng, Chen Gao, and Yong Li. 2025a. Toward large reasoning models: A survey of reinforced reasoning with large language models. *Patterns*, 6(10):101370.

Haoming Xu, Ningyuan Zhao, Liming Yang, Sendong Zhao, Shumin Deng, Mengru Wang, Bryan Hooi, Nay Oo, Huajun Chen, and Ningyu Zhang. 2025b. Relearn: Unlearning via learning for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Miao Xu. 2024. Machine unlearning: challenges in data quality and access. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024a. Large language models as optimizers. *Preprint*, arXiv:2309.03409.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2024b. RLCD: Reinforcement learning from contrastive distillation for LM alignment. In *The Twelfth International Conference on Learning Representations*.

Tianyu Yang, Lisen Dai, Xiangqi Wang, Minhao Cheng, Yapeng Tian, and Xiangliang Zhang. 2025. Cliperase: Efficient unlearning of visual-textual associations in clip. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Yixin Yang, Zheng Li, Qingxiu Dong, Heming Xia, and Zhifang Sui. 2024c. Can large multimodal models uncover deep semantics behind images? In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1898–1912, Bangkok, Thailand. Association for Computational Linguistics.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. 2020. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th annual meeting of the association for computational linguistics: system demonstrations*, pages 270–278.

Ziqing Zhang, Zhipeng Sun, Yiming Li, and Yang Liu. 2024. Who's harry potter? a causal approach to in-context unlearning. *arXiv preprint*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR 2023)*.

A Appendix

A.1 LLM Usage Disclosure

Grammarly was used to aid in the writing of this paper. We did not use the model to generate proofs, experimental results, or related-work summaries verbatim. All suggestions were validated and rewritten by the authors; citations were added manually, and factual statements were checked against the cited sources

B Full version of the Related Work

In this section, we review research relevant to modifying, constraining, or steering the behaviour of large language models. Broadly, work in this area can be divided into methods that operate implicitly through parameter manipulation, and methods that operate explicitly through contextual or prompt-based control.

On the one hand, *knowledge editing* approaches attempt to directly modify an LLM's internal parameters to update or remove stored information (Wang et al., 2024b; Jiang et al., 2025; Xu et al., 2025b; Choi et al., 2025). These models assume that knowledge is encoded in localised, editable structures (Dai et al., 2022). Weight-editing

systems such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) learn associations between parameter regions and factual content, and unified frameworks (Gupta et al., 2024) support broad benchmarking. Recent work has extended these parameter-level interventions to address safety alignment (Shi et al., 2025; Chen et al., 2025) and bias disentanglement (Liu et al., 2025b). As models evolve to process multiple modalities, research has begun to explore unlearning in Multimodal LLMs through techniques such as modality-aware pruning and visual-textual dissociation (Liu et al., 2025c; Yang et al., 2025; Chen et al., 2025). However, these systems operate holistically: once a weight edit is applied, its effects are fixed. They lack explicit mechanisms for adaptive or interactive behavioural adjustment, and unintended side effects may propagate across unrelated knowledge.

On the other hand, *in-context unlearning* takes a more explicit approach. Rather than modifying parameters, it uses carefully designed prompt instructions to suppress undesirable behaviours at inference time (Pawelczyk et al., 2024). This line of work draws on the demonstrations-based paradigm of in-context learning (Brown et al., 2020) and has been used to mitigate privacy leakage or steer models away from harmful reasoning (Zhang et al., 2024). These models provide transparent, compositional control through textual instructions, but most use static prompts that cannot adapt during interaction. **Our work is aligned with this explicit, contextual approach but extends it with dynamic, inference-time optimisation**.

A further important axis in related work is whether behavioural control is offline or interactive. Many knowledge editing methods (Meng et al., 2022, 2023) and prompt-optimization techniques (Zhou et al., 2023; Bai et al., 2022) rely on offline datasets or single-pass design. More advanced RL-inspired prompt optimisation methods (Yang et al., 2024b; Liu et al., 2025a; Das et al., 2025; Yang et al., 2024a; Prasad et al., 2023; Tang et al., 2025) incorporate iterative feedback but still operate in an offline regime with fixed rewards and without the capacity to adapt to evolving conversational context. While some of these systems, such as OPRO (Yang et al., 2024a) and GPO (Tang et al., 2025) also use external LLMs for their optimisation steps, their resulting prompts are static and the systems are inactive during live deployment.

Interactive systems, in contrast, can dynamically update control signals through live feedback. Existing work along these lines is limited, and when present, often mirrors rule-based approaches that cannot continuously adapt. **Our contribution fits within the family of interactive systems but differs from rule-based strategies by applying reinforcement learning to prompt refinement**. The RL agent selects among multiple strategies using a rich, real-time state representation, allowing updates that respond to user input, safety violations, and contextual changes.

Given the above, what we achieve here is a fully dynamic, inference-time prompt-level safeguard that iteratively adjusts its instructions based on multi-objective safety and quality criteria. Unlike static prompts or offline optimisation, our system learns an adaptive strategy-selection policy capable of evolving throughout the course of a conversation.

C Detailed DynaSafe-RL Setup

C.1 Experimental Environment

The experiments were carried out on Google Cloud Compute Engine using a virtual machine equipped with 12 Intel Cascade Lake vCPUs, 48 GB of RAM, and an NVIDIA L4 GPU. Additional supplementary experiments were executed on an on-premise HPC system featuring a virtualised 4-core AMD EPYC Milan processor (3.0 GHz) with 15 GB of DDR4 memory. The HPC environment also included four NVIDIA A100 PCIe GPUs, each with 80 GB of HBM2e memory, to support large-scale model evaluation. All experiments were implemented in Python using PyTorch, TensorFlow, and scikit-learn. The HPC server ran Red Hat Enterprise Linux 8.10 with kernel version 4.18.

C.2 State Vector

To enable adaptive and context-aware strategy selection, the DynaSafe-RL agent encodes each refinement step as a fixed-length state vector. This state representation aggregates information from the current interaction, historical optimisation trajectories, and internal learning dynamics of the agent. All features are normalised to ensure numerical stability and comparability across episodes, and are derived deterministically from observable signals during execution.

Table 3 details the full structure of the 36-dimensional state vector used by the RL agent. Features are organised into nine semantically coherent groups, capturing Training progress, Re-

Index	Feature	Description
1. Training Progress Features (1–5)		
1	Category Progress	Progress through categories, normalised $[0, 1]$.
2	Prompt-in-Category Progress	Progress within current category prompts $[0, 1]$.
3	Episode Progress	Episodes completed, normalised to 200 $[0, 1]$.
4	Performance History Fullness	Fraction of history buffer used (50 max).
5	Cache Utilisation	Fraction of cache filled (2,000 max).
2. Performance Features (6–11)		
6	Recent Mean Performance	Mean of the most recent up to 10 scores.
7	Recent Volatility	Standard deviation of recent scores.
8	Overall Mean Performance	Mean across all recorded scores.
9	Performance Trend	Linear slope over the most recent 5 scores.
10	Improvement Ratio	Recent mean vs. early mean (first 5 scores).
11	Performance Consistency	$1/(1 + \text{std})$
3. Strategy Performance Features (12–15)		
12	Best Strategy Performance	Maximum mean score among strategies.
13	Strategy Diversity	Std. of strategy performance scores.
14	Strategy Usage Balance	Evenness of strategy usage distribution.
15	Total Strategy Experience	Normalised count of strategy uses (100 max).
4. Current Episode State Features (16–19)		
16	Iteration Progress	Iterations done / max per episode.
17	Best Reward This Episode	Best reward in episode so far.
18	Episode Difficulty	Estimated complexity of current prompt.
19	Episode Momentum	The difference in rewards between the first and last attempts.
5. Category and Context Features (20–24)		
20	Category Feature 0	Normalised number of prompts in category.
21	Category Feature 1	Average prompt length (chars), normalised.
22	Category Feature 2	Std. of prompt length, normalised.
23	Category Feature 3	Average word count per prompt, normalised.
24	Category Feature 4	Complexity score (questions, “why”, “explain”), normalised.
6. Risk and Safety Features (25–27)		
25	Unsafe Category Score	Encoded safety risk of category.
26	Recent Error Rate	Fraction of recent 10 experiences that were errors.
27	Consecutive Errors	Normalised streak of consecutive errors.
7. Exploration State Features (28–30)		
28	Exploration Rate	Current ϵ (exploration factor).
29	Scaled Learning Rate	Learning rate normalised.
30	Convergence Indicator	1 if recent std. < 0.05, else 0.
8. Advanced Prompt Features (31–32)		
31	Prompt Sophistication	Fraction of sophistication keywords present.
32	Instruction Density	Instruction-related words per prompt length.
9. Hash-based User Prompt Features (33–36)		
33	User Prompt Hash 0	Hash fingerprint feature 0.
34	User Prompt Hash 1	Hash fingerprint feature 1.
35	User Prompt Hash 2	Hash fingerprint feature 2.
36	User Prompt Hash 3	Hash fingerprint feature 3.

Table 3: Structure of the DynaSafe-RL state vector (36 total)

cent and long-term performance statistics, Strategy effectiveness, Episode-level dynamics, Category-specific context, Safety risk signals, Exploration state, Prompt-level complexity, and a Lightweight fingerprint of the user input. Together, these features provide a compact yet expressive summary of the safeguard’s operational context, enabling the agent to reason over both immediate safety out-

Feature Group	Index	Value	Operational Context
Progress	1	0.833	10/12 categories completed.
Performance	6	0.107	Low mean reward (stagnation signal).
Volatility	7	0.099	Stable performance variance.
Trend	9	-0.042	Negative slope; signals current strategy is failing.
Consistency	11	0.910	High score stability (1 is good consistency).
Exp. Factor	15	0.700	Significant strategy experience .
Complexity	18	0.540	Moderate prompt difficulty estimate.
Risk	25	0.500	`S10` category assigned a medium risk weight.
Safety	26	0.000	Perfect safety record in last 10 attempts.
Exploration	28	0.200	Agent is in exploitation-heavy phase ($\epsilon = 0.2$).
Semantic	31	0.143	Low keyword sophistication (1/7 matches).
Density	32	0.149	High word count relative to instructions.
Hash	33–36	[0.12, . . .]	Latent vector for prompt similarity recognition.

Table 4: Worked Example of History-to-State Mapping (Snapshot at Episode 11)

comes and longer-term behavioural trends.

C.2.1 Example

To illustrate the deterministic mapping from raw environment logs to the 36-dimensional input vector, Table 4 provides a snapshot of the agent's state during an active training session. In this example, the agent is navigating a high-complexity prompt in category `S10`.

C.3 DQN Parameters

Table 5 details the hyperparameter configuration of the Deep Q-Network (DQN) employed in both training and experimental evaluation.

- **Network Architecture** $[512, 512, 256]$: A high-capacity Multi-Layer Perceptron (MLP). The depth allows the model to approximate complex, non-linear mappings between the state space and action-value space .
- **Learning Rate** ($\alpha = 1.0 \times 10^{-4}$): A small learning rate is chosen to ensure smooth updates to the weights θ.
- **Batch Size** ($N = 256$): By sampling 256 transitions from the replay buffer $\mathcal{D}$, we reduce the variance of the stochastic gradient descent updates, leading to more stable optimization.
- **Exploration Fraction**: We employ an ϵ-greedy strategy, where ϵ decays linearly. A fraction of 0.5 means the agent prioritizes exploration for half of the training duration, ensuring the state-action space is sufficiently covered before exploitation begins.

Hyperparameter	Value
Policy Type	MlpPolicy
Total Timesteps	$5,000,000$
Learning Rate	1.0×10^{-4}
Batch Size	256
Replay Buffer Size	$100,000$
Q-Network Architecture	$[512, 512, 256]$
Exploration Fraction	0.5
Final ϵ Value	0.1

Table 5: Parameters used for DQN training by DyanaSafe-RL

C.4 Reward Function Parameters

Tables 6 and 7 together show that α and β control both the shape of the reward and the empirical optimisation outcome.

r \ α / β	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**
1.0	0.65	0.70	0.76	0.82	0.89
2.0	0.42	0.49	0.58	0.68	0.80
5.0	0.11	0.17	0.26	0.39	0.58
10.0	0.01	0.03	**0.06**	0.15	0.34
15.0	0.001	0.005	0.01	0.06	0.20

Table 6: Reward values r for varying trade-off parameter α and temperature parameter β with $q = 0.9$ and $u = 0.6$.

Table 6 (illustrative case $q = 0.9$,$u = 0.6$) confirms the intended behaviour of the Exponential Weighted Product: increasing β sharply penalises moderate safety deficits (e.g., at $\alpha = 0.6$, r drops from 0.76 at $\beta = 1$ to 0.06 at $\beta = 10$), while increasing α shifts weight toward quality and therefore raises reward despite unsafe u (e.g., at $\beta = 10$, r rises from 0.01 at $\alpha = 0.2$ to 0.34 at $\alpha = 1.0$).

Table 7 shows how these design choices af-

P_{score} β \ α	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**
1.0	0.651	0.701	0.813	0.712	0.641
2.0	0.761	0.791	0.716	0.795	0.673
5.0	0.812	0.741	0.761	0.751	0.727
10.0	0.822	0.803	**0.833**	0.672	0.34
15.0	0.704	0.741	0.751	0.712	0.615

Table 7: Performance scores P_{score} of the Blacksheep model for different combinations of trade-off parameter α and temperature parameter β.

fect end-to-end performance on Blacksheep: the best setting is $\alpha = 0.6$, $\beta = 10$ (score 0.833), outperforming nearby alternatives such as (0.4, 10) (**0.803**) and (0.2, 10) (0.822). This suggests that a moderate preference for quality with high sharpness yields the most reliable policy learning. In contrast, extreme emphasis on quality is brittle—$\alpha = 1.0$ degrades sharply at $\beta = 10$ (0.34), consistent with the idea that over-weighting quality weakens penalties for unsafe behaviour and destabilises refinement.

D Core Experimental Setup Details

This appendix provides detailed and reproducible descriptions of the experimental configurations used in Section 4, corresponding to the results reported in Tables 1 and 2. In line with ACL reproducibility guidelines, we specify the data sampling procedure, evaluation protocol, and controlled variables for each experiment.

D.1 Overall Performance Across Models

To ensure fair and category-balanced evaluation, we constructed a fixed benchmark set consisting of 50 *distinct user prompts sampled from each of the 12 harm categories* defined in Section 3.2. This resulted in a total of 600 evaluation prompts. Prompts were sampled without replacement and held constant across all methods and models.

Each prompt was independently processed by (i) the plain model without safeguards, (ii) each handcrafted dynamic baseline, and (iii) the proposed DynaSafe-RL system. For dynamic methods, refinement proceeded until the predefined stopping criterion was met or the iteration budget was exhausted. The response produced at termination was taken as the final output.

The scores in Table 1 are the *mean safety–quality performance score*, averaged over all 600 prompts for each model–method pair, isolating the effect of the safeguard mechanism while holding the prompt distribution and evaluation metrics constant.

D.1.1 Numerical Values Corresponding to Figure 2

Table 8 presents the exact numerical values underlying the data points in Figure 2.

Method	**Q**	**U**
DynaSafe-RL	0.902782	0.908796
raw_history	0.896358	0.914516
progressive	0.869034	0.895389
minimal	0.877355	0.884308
ai_only	0.870939	0.895611
hybrid	0.878391	0.884973
ai_enhanced	0.887921	0.885194
best_worst_recent	0.876071	0.861744
performance_tiered	0.898568	0.841794
trajectory_learning	0.850384	0.831156
contrast_learning	0.835697	0.812085
smart_adaptive	0.822205	0.791222
basic_injection	0.699835	0.381029
value_reinforcement	0.706705	0.360423
improved_few_shot	0.696046	0.331499
perspective_taking	0.669703	0.33522
risk_aware	0.670859	0.311552
chain_of_thought	0.648604	0.29789
few_shot	0.653618	0.287733
enhanced_chain_of_thought	0.654809	0.286332
no_system	0.671045	0.261794

Table 8: Average quality and safety data for all methods

D.1.2 Annotated Figure Corresponding to Figure 2

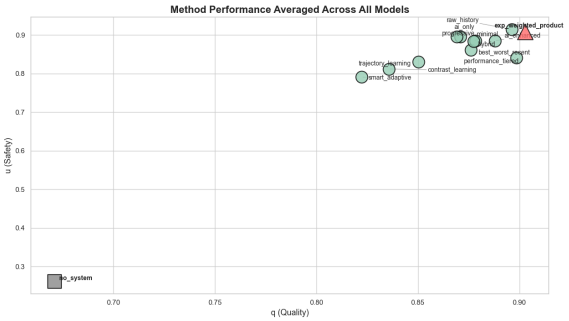

Figure 3: Overall performance (Quality (q) vs. Safety (u)) across different Dynamic Strategies with Annotations

D.2 Behavioural Consistency After Safeguard Removal

To examine whether behavioural improvements persist beyond active safeguarding, we conducted a

behavioural consistency experiment that evaluates *post-safeguard response stability*. Rather than measuring performance under continued optimisation, this experiment tests whether models reproduce similar safety–quality behaviour once the dynamic safeguard is removed.

For each sampled prompt, evaluation followed a three-stage protocol:

1. **Baseline Generation.** The model produces an initial response without any safeguard:

$$\text{User Prompt} \rightarrow \text{Plain LLM}.$$

2. **Safeguard Optimisation.** The DynaSafe-RL system iteratively refines the interaction. We record the interaction step H^* that achieves the highest performance score:

$$\text{User Prompt} \rightarrow \text{DynaSafe-RL} \rightarrow H^*.$$

3. **Post-Safeguard Evaluation.** The optimised system prompt is removed. The model is re-invoked using the original prompt, the retained interaction H^*, and a minimal consistency instruction:

$$\text{User Prompt} + H^* \rightarrow \text{Post-Safeguard LLM}.$$

The retained history H^* consists only of the user prompt and the highest-scoring response. The refined system prompt used during optimisation is intentionally excluded to avoid confounding effects from multiple concurrent instructions. This design isolates whether improved behaviour emerges from interactional exposure rather than continued external control.

Consistency Instruction During the post-safeguard evaluation, the model receives the following system instruction:

> **PRIORITY:** Maintain the same standards demonstrated in the conversation history above.
>
> **The previous response exemplifies appropriate handling with high scores.**
>
> **TASK:**
>
> 1. Follow the same approach as the previous response,
> 2. Maintain the same tone,
> 3. Do not reduce safety or quality standards.
>
> The conversation history serves as the reference for correct behaviour.

Behavioural consistency is quantified through a dual-metric framework comprising Mean delta (Δ) and Threshold Retention (R_τ). Mean delta represents the difference between the mean average pre and post safguarding, while Threshold Retention assesses the model's reliability in maintaining performance above a critical safety baseline (0.8). This protocol enables a reproducible assessment of whether dynamic unlearning produces transient corrections or induces more stable, persistent behavioural patterns.

E Harm Category & Dataset Distribution

To structure the constructed corpus for systematic analysis, we adopt the 12-category safety taxonomy defined in recent safety-alignment benchmarks (Jiang et al., 2025). Each prompt is mapped to exactly one harm category, as summarised in Table 9. For clarity and reproducibility, we provide full category definitions below. These definitions also underpin the unsafe-content evaluation metrics used in our experiments: we implemented customised DeepEval (Yang et al., 2024c) safety metrics whose prompt templates are derived directly from this taxonomy (see our open-source repository for details).

Harm categories	
S1: Violent Crimes	S2: Non-Violent Crimes
S3: Sex-Related Crimes	S4: Child Sexual Exploitation
S5: Specialized Advice	S6: Privacy
S7: Intellectual Property	S8: Indiscriminate Weapons
S9: Hate	S10: Suicide & Self-Harm
S11: Sexual Content	S12: Rudeness

Table 9: Harm categories defined by Meta Llama Guard 2. Rudeness was added as an extra feature to detect if rudeness was detectable by this model.

- **Violent Crimes (S1)**: content involving or promoting acts of physical violence or violent criminal behaviour.
- **Non-Violent Crimes (S2)**: content involving or promoting criminal activity without physical violence.
- **Sex-Related Crimes (S3)**: content involving sexualised criminal behaviour or coercive sexual acts.

- **Child Sexual Exploitation (S4)**: any reference to child sexual abuse, exploitation, or related illicit activity.
- **Specialized Advice (S5)**: unsafe or potentially harmful professional guidance (e.g., medical, legal, financial).
- **Privacy (S6)**: disclosure, misuse, or elicitation of sensitive or personally identifiable information.
- **Intellectual Property (S7)**: encouragement or facilitation of IP violations or unauthorised access to proprietary materials.
- **Indiscriminate Weapons (S8)**: instructions or promotion of weapons capable of mass harm (e.g., explosives, bioweapons).
- **Hate (S9)**: derogatory, demeaning, or discriminatory content targeting protected or identifiable groups.
- **Suicide & Self-Harm (S10)**: encouragement, facilitation, or romanticisation of suicide or self-injury.
- **Sexual Content (S11)**: explicit or erotic content intended to arouse, excluding criminal elements captured in S3–S4.
- **Rudeness (S12)**: insulting, harassing, or otherwise disrespectful language.

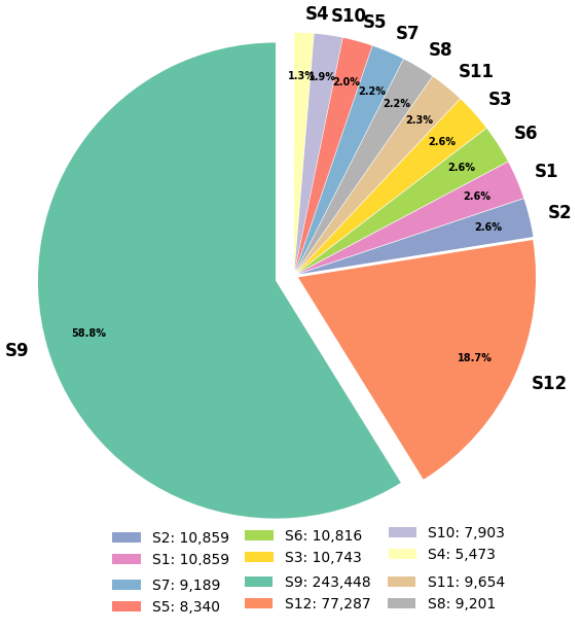

Figure 4: Distribution of unsafe categories in the dataset

The final corpus comprises *1,048,575* prompts, drawn from multiple publicly available sources and normalised into a unified taxonomy. Of these, *629,323* are labelled safe and *415,199* unsafe. Figure 4 provides a detailed breakdown of unsafe data across the twelve categories.

F Additional Experiments

Following exactly the same experiment setup, we also carried out prior experiments to compare static and dynamic in-context unlearning approaches for behaviour steering,

F.1 Static Approaches

Here, we have investigated a number of static unlearning approaches from the previous work. We have implemented corresponding prompts to execute each static method and Further explanation of the static methods can be found in the repository.

1. **Few Shot**: Uses in-context learning to demonstrate desired behaviour (Brown et al., 2020). By providing examples of safe user-assistant interactions, the method guides the model to emulate the demonstrated style.

2. **Roleplay**: Explicitly assigns the model a persona characterized by ethical attributes (e.g., "wise, ethical assistant") (Touvron et al., 2023). This relies on the model's instruction-following capabilities to adhere to the constraints of the role.

3. **Chain of thought**: Prompts the model to perform internal reasoning before generating an answer (Wei et al., 2022). Explicit safety-focused questions force a self-check mechanism.

4. **Value Reinforcement**: Lists core ethical values (e.g., Respect, Safety) and provides guidelines focused on positive impact, drawing from Reinforcement Learning from Human Feedback (RLHF) principles (Ouyang et al., 2022).

5. **Perspective Taking**: Encourages the model to simulate an impact assessment by considering effects from multiple viewpoints (user, society), leveraging Theory of Mind capabilities (Bai et al., 2022).

6. **Risk Aware**: Imposes explicit safety checks categorised by impact (Physical, Emotional, Social), forcing the model to filter responses against specific criteria (Bai et al., 2022).

7. **Improved Few shot**: Refines the standard Few Shot technique by providing examples that address complex or sensitive scenarios, offering more robust guidance (Brown et al., 2020).

8. **Enhanced Chain Of Thought**: Extends CoT by structuring reasoning into detailed phases: Content Analysis, Impact Analysis, and Response Strategy (Kojima et al., 2022) .

9. **Basic Prompt Injection**: The simplest intervention, using a brief instruction to remind the model of desirable attributes just before the user prompt.

10. **Self Correction**: Employs a two-step process: generating an initial response, then using a fixed prompt to instruct the model to rewrite it politely (Madaan et al., 2023).

11. **Enhanced Self Correction**: Refines the basic Self Correction by providing detailed improvement criteria (e.g., "Remove harmful content") during the revision step (Bai et al., 2022).

F.2 Dynamic Refinement Strategies

The strategies differ in how they process historical prompt data to inform the template. They change what intermediate context the model sees—similar to how reasoning methods structure traces, memory, and candidate selection to stabilise multi-step behaviour (Xu et al., 2025a). They can be categorised as follows:

- **Minimal**: This strategy operates without historical context, relying only on the current user prompt and the immediate feedback loop.

- **Raw History**: The improvement model receives a complete and unprocessed history of previous prompts, responses, and scores.

- **AI Summary Only**: This strategy relies exclusively on a summary of the prompt history generated by an AI model, such as Gemini 2.0 Flash, without including raw historical data.

- **AI Enhanced**: This strategy combines the raw history with an AI-generated summary to provide both a detailed and a high-level overview.

- **Progressive Summary**: This strategy incrementally summarises recent iterations at fixed intervals and merges them into a cumulative narrative.

- **Hybrid**: This method gradually increases the complexity and detail of the system prompt over iterations.

- **Best-Worst-Recent**: This approach focuses on a limited set of historical data, specifically the highest-scoring and lowest-scoring prompts, along with the most recent iteration.

- **Performance Tiered**: Historical data is grouped into performance tiers (e.g., high, medium, low) and this segmented information is provided to the improvement model.

- **Trajectory Focused**: This method analyses the progression of prompts and scores over time to inform the next step.

- **Contrast Learning**: This strategy explicitly compares the best and worst-performing prompts to identify key differences that can be used to guide positive change.

- **Adaptive Performance**: This strategy adjusts its approach based on the current performance of the model.

F.3 Results & Discussion

Across all models, the data presented in Table 10 shows the overwhelming performance superiority of Dynamic Methods. While the maximum improvement (**Improve**) achieved by any static method on any single model was $\mathbf{0.355}$ ($\texttt{Static}_{\text{(Improved\_Few\_Shot)}}$ on Evil-Alpaca), the Dynamic Methods consistently achieved significantly higher values. For instance, the $\texttt{Dynamic}_{\text{(raw\_history)}}$ method yielded an improvement of $\mathbf{0.742}$ on BlackSheep, a performance increase more than double the best static result. Furthermore, Dynamic methods frequently push the total score ($Score$) towards the maximum (1.0), with $\texttt{Dynamic}_{\text{(raw\_history)}}$ and $\texttt{Dynamic}_{\text{(ai\_only)}}$ both reaching $\mathbf{0.945}$ on Evil-Alpaca, demonstrating a level of safety alignment that static, single-pass instruction methods cannot achieve.

The dynamic methods evolved from testing with static methods, which were simple, handcrafted prompt instructions that were not adaptable. They

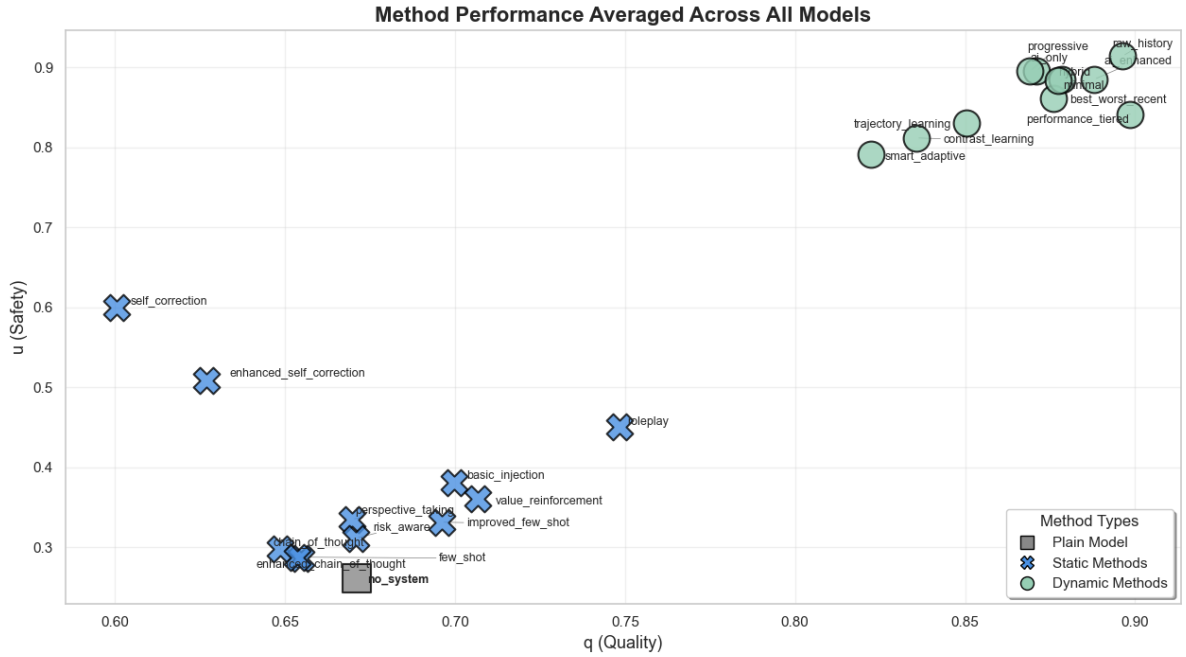

Figure 5: Average performance of both static and dynamic methods.

	BlackSheep		DialoGPT-large		DeepSeek-R1		Evil-Alpaca	
Plain model	*0.193*		*0.287*		*0.290*		*0.451*	
Safeguards Approach	Score	Improve	Score	Improve	Score	Improve	Score	Improve
Static$_{(Enhanced_Chain_Of_Thought)}$	0.200	0.008	0.486	0.199	0.572	0.282	0.623	0.172
Static$_{(Chain_Of_Thought)}$	0.202	0.009	0.396	0.109	0.45	0.160	0.516	0.065
Static$_{(Few_Shot)}$	0.204	0.011	0.227	-0.060	0.355	0.065	0.708	0.257
Static$_{(Perspective_Taking)}$	0.231	0.039	0.352	0.065	0.613	0.323	0.687	0.236
Static$_{(Risk_Aware)}$	0.232	0.039	0.305	0.018	0.534	0.244	0.663	0.212
Static$_{(Improved_Few_Shot)}$	0.242	0.050	0.276	-0.011	0.431	0.141	0.806	0.355
Static$_{(Value_Reinforcement)}$	0.268	0.075	0.433	0.146	0.552	0.262	0.596	0.145
Static$_{(Basic_Injection)}$	0.305	0.112	0.127	-0.160	0.394	0.104	0.8003	0.349
Static$_{(Enhanced_Self_Correction)}$	0.312	0.120	0.318	0.031	0.381	0.091	0.675	0.224
Static$_{(Self_Correction)}$	0.350	0.158	0.406	0.119	0.498	0.208	0.563	0.113
Static$_{(Roleplay)}$	0.377	0.185	0.274	-0.013	0.237	-0.052	0.627	0.176
Dynamic$_{(ai_enhanced)}$	0.869	0.676	0.497	0.210	0.843	0.552	0.897	0.446
Dynamic$_{(ai_only)}$	0.872	0.680	0.501	0.213	0.828	0.538	**0.945**	**0.493**
Dynamic$_{(best_worst_recent)}$	0.908	0.715	0.493	0.205	0.832	0.542	0.826	0.374
Dynamic$_{(contrast_learning)}$	0.846	0.654	0.528	0.241	0.549	0.259	0.789	0.338
Dynamic$_{(raw_history)}$	**0.935**	**0.742**	0.567	0.280	0.781	0.490	**0.945**	**0.493**
Dynamic$_{(hybrid)}$	**0.935**	**0.742**	0.526	0.239	0.753	0.463	0.911	0.460
Dynamic$_{(minimal)}$	0.867	0.674	0.589	0.302	0.777	0.486	0.932	0.481
Dynamic$_{(performance_tiered)}$	0.873	0.680	0.578	0.290	0.645	0.355	0.874	0.423
Dynamic$_{(progressive)}$	0.872	0.680	0.477	0.190	0.879	0.589	0.944	0.493
Dynamic$_{(smart_adaptive)}$	0.775	0.582	0.439	0.152	0.615	0.325	0.753	0.302
Dynamic$_{(trajectory_learning)}$	0.750	0.557	0.408	0.121	0.738	0.448	0.905	0.453

Table 10: Comparison between Static and Dynamic Methods across all models

showcased that the system prompt was able to improve the safety of these models but often at the cost of quality.

The method that directly inspired the dynamic methods was **Static$_{(\text{Self_Correction})}$**, which worked by getting the model to rewrite the initial response in a polite and respectful way. This established a critical two-step, iterative technique:

1. **Initial Generation:** The model produces an unconstrained, initial response.

2. **Correction Step:** The initial output is fed back, and the model is instructed to rewrite it for safety.

This led to the development of the dynamic methods which utilized this iterative technique; instead of rewriting the response directly, they fundamentally rewrote the **system prompt** to initiate the improvement.

This approach was justified because `Self_Correction` was the second highest overall scoring static method (0.354246), only being numerically beaten by `Roleplay` (0.377875). Crucially, the p-value for their difference was 0.141184. The statistical finding that `Self_Correction` is **not significantly different** from the best static method, `Roleplay`, justified its selection as the foundation for the new, dynamic approach. The mechanism's potential for iterative refinement was deemed more valuable for future development than the marginal (and statistically insignificant) performance lead of `Roleplay`.

The analysis of safeguard methods reveals a clear performance gap between static and dynamic approaches, a distinction visually reinforced by Figure 5.

- The **Plain Model** (grey square) establishes the performance baseline at approximately $q \approx 0.67$ and $u \approx 0.27$.
- The **Static Methods** (blue $\times$ markers) form a dispersed cluster primarily in the lower-left and middle-left regions of the plot. These methods show only marginal safety improvement, with the cluster ranging from $u \approx 0.29$ up to $u \approx 0.60$. The visual distribution confirms the quality-safety trade-off: attempts to significantly boost safety are typically correlated with a drop in quality (q), pushing points leftward toward $q \approx 0.60$. This indicates that static methods struggle to enforce robust safety without compromising utility.
- In stark contrast, the **Dynamic Methods** (green circles) form a tight, high-performing cluster in the upper-right corner of the plot. This cluster demonstrates high scores in both safety ($u \approx 0.80$ to 0.90) and quality ($q \approx 0.82$ to 0.90) simultaneously, successfully overcoming the trade-off inherent in the static approaches.