FILTERING LOW-Q TOKENS: IMPROVING SUCCESS RATES WHILE PRESERVING BASE-MODEL SUCCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

LLMs are highly capable, yet they may still be imperfect with respect to a variety of target criteria, such as safety, contextual faithfulness, or reasoning. A natural goal is therefore to take a capable, trained LLM and augment it to increase its rate of successful responses. This augmentation should preserve the diverse successful outputs the base model already produces. We derive such an augmentation by testing each candidate token against the hypothesis that the final output will be successful. We show that the success probability of the full generation is lower-bounded by a quantity increasing in the power of this test. Our method, *Q-filtered decoding* (QFD), therefore follows from the Neyman–Pearson lemma, which identifies the optimal test of this form at a given false-rejection rate: a threshold on a token's probability of leading to a successful final output. At each decoding step, a small learned probe estimates this probability for every candidate token; those falling below the threshold are removed, and the remaining distribution is renormalized. Raising the threshold makes the filter intervene more aggressively, at the cost of altering more of the base model's successful outputs. Across mathematical reasoning, contextual faithfulness and harmful compliance, in seven task–model pairs, QFD significantly improves the success rate on six. We measure preservation by the drop rate, the probability that an intervention rejects a successful generation of the base model, and by the cosine similarity of successful completions to those of the base model. No baseline, including LoRA fine-tuning and activation steering, dominates QFD: at 28 of its 35 settings, no baseline reaches as low a drop rate, and at the rest, those that do are less successful. On four of the six pairs, QFD's largest improvement in success rate is comparable to the best baseline's, yet it drops 54–87% of the base model's successful generations instead of 96–100%, and its completions are closer to the base model's in cosine similarity. QFD slows generation only by 1.2% on average.

1 INTRODUCTION

LLMs are remarkably capable across a broad range of tasks, yet a capable model may still produce an unsafe response, contradict its given context, or produce an error in a multi-step derivation (Zou et al., 2023b; Longpre et al., 2021; Mirzadeh et al., 2025). Such failures persist even when the criterion they violate is part of the post-training objective (Lambert et al., 2025). They can be readily detected, since such criteria can be evaluated, by exact verification in mathematics and code and by model-based judges elsewhere (Cobbe et al., 2021; Hendrycks et al., 2021; Zheng et al., 2023). Therefore, given a trained model, a natural goal is to raise the probability of satisfying such a criterion, using labeled prompts or model generations. Existing approaches differ in how they utilize these labels.

Activation steering acts on the model's intermediate representations, adding an activation direction derived from contrastive inputs (Rimsky et al., 2024; Siu et al., 2026). Its effectiveness is highly sensitive to the pairing of method, model and success criterion (Siu et al., 2026). In contrast, fine-tuning acts on the weights, as in the RLHF pipeline of Ouyang et al. (2022). Full fine-tuning of large models is out of reach for most practitioners, which is why low-rank adaptation is usually employed as a practical alternative (Hu et al., 2022). A third family of interventions leaves the weights frozen and acts on token probabilities at decoding time. ARGS adds a weighted reward score to the base logits over the top-k candidates (Khanov et al., 2024), and controlled decoding trains a prefix scorer

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.

to predict the expected reward of continuing from a partial response, then tilts the base distribution by its exponential (Mudgal et al., 2023). See more related work in Appendix A.

While all of these methods raise success rates, they alter the entire output distribution without explicitly controlling how much the model's desirable outputs change. An intervention, especially one fitted from limited data, should conserve a capable base model's diverse solution paths, phrasings and refusal strategies when they already lead to a desirable output.

Value-filtered decoding (Einbinder et al., 2026) is one intervention method that controls this tradeoff. It uses a learned estimator that takes a partial response as input and returns the probability that continuing leads to a successful completion. Candidate tokens whose estimate falls below a user-specified threshold are rejected, allowing the user to control how often the filter intervenes when the base model would have succeeded anyway. Although this method proves effective, it has two main limitations, stemming from its rejection sampling approach. First, scoring each candidate token requires its own forward pass through the model; in its published measurements, its decoding speed is up to about 45% below that of the base model. Second, value-filtered decoding tests a finite sample of tokens rather than the full candidate set, and falls back to the highest-scoring draw when all are rejected, so it does not sample from its target policy even when the value function is exact.

Our contribution

In this work we introduce *Q-filtered decoding* (QFD), which builds on the inference-time filtering of value-filtered decoding but filters by a different estimand: the Q-function, which assigns to each possible next token the probability that selecting it and continuing under the base model yields a successful completion. This change removes both limitations of value-filtered decoding: we estimate the Q-values of all candidate tokens at once from the representation of the current prefix, so the next token can be drawn by categorical sampling rather than rejection sampling. With every candidate scored, QFD masks those whose Q-values fall below a pre-specified threshold and samples the next token directly from the renormalized distribution. If no candidate reaches the threshold, the threshold at that step is lowered to the highest Q-value attained; this fallback is part of the decoding rule itself, so QFD samples exactly from the policy it defines.

We derive QFD from first principles. In Section 3, we describe any intervention with the same support as the base model by the fraction of each token's base probability that it removes (Definition 3.1). In Section 4, we interpret this removed fraction as the rejection probability of a test of the null hypothesis that the generation will succeed, against the alternative that it will fail. In Theorem 4.1 we lower-bound the intervention's success probability by a quantity that increases with the test's power and decreases with its type-I error. In Theorem 4.2 we use the Neyman–Pearson lemma to show that, at a given type-I error, this bound is maximized by removing the candidates whose Q-values fall below a threshold, and that with the exact Q-function, the resulting policy never lowers the success probability. Finally, in Theorem 4.3 we bound the drop rate, the probability that QFD alters a successful generation of the base model, by a quantity that is monotone in the threshold.

As Snell et al. (2023); Dulac-Arnold et al. (2016) noted, learning the Q-function is challenging because it requires estimating a value for every state–token pair, yielding a high-dimensional prediction problem. Fortunately, standard top-k and top-p decoding substantially reduce the effective dimension of the Q-function required by our method. This is because they restrict decoding to a small, highly probable subset of the vocabulary, so QFD need only consider Q-values for those tokens. This restriction increases the robustness of QFD to misestimation of the Q-function, as shown in Proposition D.1 in Appendix D. Such restrictions are standard in open-weight LLMs (Zeng et al., 2026; Gemma Team, 2026; Qwen Team, 2026; NVIDIA, 2026). To efficiently estimate Q, we introduce an estimation method that utilizes the model's hidden states, and increases the active parameter count by at most 0.25% across the evaluated models. Implementation is detailed in Section 5.

In Section 6, we evaluate QFD on mathematical reasoning (DeepMath; He et al., 2026), contextual faithfulness (FaithEval; Ming et al., 2025) and harmful compliance (SALAD-Bench and WildJailbreak; Li et al., 2024; Jiang et al., 2024) across seven task–model pairs, and compare it with controlled decoding, ARGS, LoRA fine-tuning and three activation-steering methods. We measure preservation by the drop rate and by the cosine similarity between successful completions and those of the base model. QFD significantly improves the success rate on six of the seven pairs. No baseline dominates QFD: at 28 of its 35 settings, no baseline attains a drop rate as low, and at the other seven,

every baseline that does is less successful. On four of the six pairs, QFD's largest improvement in success rate is comparable to the best baseline's, yet it drops 54–87% of the base model's successful generations instead of 96–100%, and its completions are closer to the base model's in cosine similarity. Finally, QFD slows decoding by only 1.2% on average, and by at most 3.2% on six of the seven pairs. All results are reproducible through the code repository included in the supplementary material.

2 NOTATIONS

Let X be a prompt drawn from a distribution $\mathcal{D}$. The base policy π, which we take to include the LLM sampling parameters, generates tokens from a finite vocabulary $\mathcal{V}$ by sampling $Y_t \sim \pi(\,\cdot \mid X, Y_{1:t-1})$ for $t = 1, 2, \ldots$, where $Y_{1:t} := (Y_1, \ldots, Y_t)$ denotes the first t tokens Generation stops at $T := \min\{t \geq 1 : Y_t = \texttt{eos}\} \wedge T_{\max}$, where $\texttt{eos} \in \mathcal{V}$ is the terminal token and $T_{\max}$ the decoder's maximum length, and the response is $Y := Y_{1:T}$. For a policy μ of this form, P_μ denotes the joint law of (X, Y) with $X \sim \mathcal{D}$ and tokens drawn from μ.

We write $S_t := (X, Y_{1:t-1})$ for the state before step $t \leq T$, so that Y_t is drawn with probability $\pi(Y_t \mid S_t)$. We denote the support of π at state S_t with $\mathcal{A}(S_t)$. Let $\mathcal{S}$ be the set of states that occur with positive probability under π before generation stops. We use S_t and Y_t both for random variables and for their values.

After a whole generation Y is complete, it is scored by a binary reward $r(X, Y) \in \{0, 1\}$, equal to one when it satisfies the target criterion. We define two related quantities. The *value function* is the probability that the completed response succeeds when all future tokens from a state are sampled from the base policy:

$$V^\pi(S_t) := P_\pi\big(r(X, Y) = 1 \mid S_t\big), \qquad S_t \in \mathcal{S}.$$

The *Q-function* is the probability of eventual success when the next token is fixed, after which generation continues according to the base policy:

$$Q^\pi(S_t, Y_t) := P_\pi\big(r(X, Y) = 1 \mid S_t,\, Y_t\big), \qquad S_t \in \mathcal{S},\ \pi(Y_t \mid S_t) > 0.$$

Emitting Y_t at S_t leads deterministically to $S_{t+1} = (S_t, Y_t)$, so $Q^\pi(S_t, Y_t) = V^\pi(S_{t+1})$ whenever generation continues past step t; if it stops at step t, $Q^\pi(S_t, Y_t) = r(X, Y_{1:t})$.

3 PROBLEM SETUP

We seek an intervention $\mu(\cdot \mid S_t)$, supported on $\mathcal{A}(S_t)$, that increases the success probability $P_\mu(r(X, Y) = 1 \mid X)$, while limiting deviations from the base model's distribution on successful completions. To relate any intervention to the base model, including those we compare against, we describe it by the fraction it removes from each token's base probability, formally defined as follows.

Definition 3.1 (Removed fraction)**.** *The removed fraction of a policy μ is the map φ_μ, written $\varphi_\mu(Y_t \mid S_t) \in [0, 1]$ for $S_t \in \mathcal{S}$ and $Y_t \in \mathcal{A}(S_t)$, given by*

$$1 - \varphi_\mu(Y_t \mid S_t) := \mu\big(\mathcal{A}(S_t) \mid S_t\big) \cdot \frac{\mu(Y_t \mid S_t)/\pi(Y_t \mid S_t)}{\max_{Y_t' \in \mathcal{A}(S_t)} \mu(Y_t' \mid S_t)/\pi(Y_t' \mid S_t)}, \tag{1}$$

and by $\varphi_\mu(Y_t \mid S_t) := 1$ at states where μ places no probability on $\mathcal{A}(S_t)$.

By equation 1, the token the intervention favors most, the one maximizing $\mu(Y_t \mid S_t)/\pi(Y_t \mid S_t)$, loses exactly the probability that μ moves outside $\mathcal{A}(S_t)$, and every other token keeps a share of its base probability proportional to this ratio.

If μ is supported on $\mathcal{A}(S_t)$ at every $S_t \in \mathcal{S}$, then φ_μ has minimum zero over $\mathcal{A}(S_t)$; we call any such map an in-support removed fraction. Every in-support removed fraction φ is the removed fraction of exactly one such intervention, obtained from π by removing φ and renormalizing:

$$\mu_\varphi(Y_t \mid S_t) := \frac{\pi(Y_t \mid S_t)\big(1 - \varphi(Y_t \mid S_t)\big)}{Z_\varphi(S_t)}, \qquad Z_\varphi(S_t) := \sum_{Y_t \in \mathcal{A}(S_t)} \pi(Y_t \mid S_t)\big(1 - \varphi(Y_t \mid S_t)\big),$$

for $Y_t \in \mathcal{A}(S_t)$, and $\mu_\varphi := 0$ outside it. In-support removed fractions are thus fully expressive over the interventions we consider (Proposition B.1 in Appendix B), and so we pursue our goal by designing an in-support removed fraction φ that:

1. increases the success probability $P_{\mu_\varphi}(r(X,Y)=1 \mid X)$ under the resulting policy μ_φ;
2. bounds the expected removed fraction of the next token, given that the base policy π produces a successful completion.

In the following section, we rigorously define the notion of preservation in goal 2, and in Theorem 4.1 we connect it to the success probability of goal 1. Then, through Theorem 4.2, we derive QFD as a solution that satisfies both goals. Theorem 4.3 further establishes the trajectory-level preservation of successful base-model outputs that QFD achieves.

4 Q-FILTERED DECODING

To connect φ to the intervention's success probability from goal 1, we examine $\varphi(Y_t \mid S_t)$ from a statistical point of view. We define the null hypothesis $H_0 : r(X,Y) = 1$ and the alternative hypothesis $H_1 : r(X,Y) = 0$, corresponding to a successful or unsuccessful final completion under the base policy, respectively. We notice that $\varphi(Y_t \mid S_t)$ can be interpreted as a test function for these hypotheses: upon observing candidate token Y_t, we reject H_0 with probability $\varphi(Y_t \mid S_t)$, which corresponds to removing the same fraction of Y_t's base probability. Formally, $P(\text{reject } H_0 \mid Y_t, S_t) := \varphi(Y_t \mid S_t)$.

Each hypothesis induces a distribution over the next token, which by Bayes' rule is

$$P_0(Y_t \mid S_t) := P_\pi\big(Y_t \mid S_t,\ r(X,Y)=1\big) = \frac{\pi(Y_t \mid S_t)\, Q^\pi(S_t,Y_t)}{V^\pi(S_t)}, \qquad V^\pi(S_t) > 0,$$

$$P_1(Y_t \mid S_t) := P_\pi\big(Y_t \mid S_t,\ r(X,Y)=0\big) = \frac{\pi(Y_t \mid S_t)\big(1 - Q^\pi(S_t,Y_t)\big)}{1 - V^\pi(S_t)}, \qquad V^\pi(S_t) < 1.$$

We now derive the type-I error and power of the test. These are the expected removed fraction of the next token, given that the base policy produces a successful or an unsuccessful completion, respectively. Since the rejection depends on the completion only through Y_t, the law of total probability gives:

$$g(\varphi, S_t) := P(\text{reject } H_0 \mid S_t, r(X,Y)=1) = \mathbb{E}_{Y_t \sim P_0(\cdot \mid S_t)}\big[\varphi(Y_t \mid S_t)\big], \qquad V^\pi(S_t) > 0,$$

$$b(\varphi, S_t) := P(\text{reject } H_0 \mid S_t, r(X,Y)=0) = \mathbb{E}_{Y_t \sim P_1(\cdot \mid S_t)}\big[\varphi(Y_t \mid S_t)\big], \qquad V^\pi(S_t) < 1.$$

The type-I error $g(\varphi, S_t)$ is therefore exactly the quantity that we set out to bound in goal 2. However, $g(\varphi, S_t)$ and $b(\varphi, S_t)$ characterize the intervention locally at a single state S_t, while goal 1 concerns the success probability of the full generation. Relating these local quantities to generation-level success is non-trivial, and we establish this connection next.

The success probability can only change at states in $\mathcal{S}^\circ := \{S_t \in \mathcal{S} : 0 < V^\pi(S_t) < 1\}$, where both g and b are defined and an intervention can improve the success probability. The following result connects both rates on $\mathcal{S}^\circ$ to the success probability of full generations.

Theorem 4.1 (Rate-induced improvement guarantee)**.** *There is a map Θ on the rate functions g and b such that for every in-support removed fraction φ and every prompt X,*

$$P_{\mu_\varphi}\big(r(X,Y)=1 \mid X\big) \ \geq\ \Theta\big(g(\varphi,\cdot), b(\varphi,\cdot)\big) \cdot P_\pi\big(r(X,Y)=1 \mid X\big).$$

Moreover, if φ and φ' are in-support removed fractions with $b(\varphi, S_t) \geq b(\varphi', S_t)$ and $g(\varphi, S_t) \leq g(\varphi', S_t)$ for every $S_t \in \mathcal{S}^\circ$, then $\Theta(g(\varphi,\cdot), b(\varphi,\cdot)) \geq \Theta(g(\varphi',\cdot), b(\varphi',\cdot))$; and $\Theta(g,b) \geq 1$ whenever $b(S_t) \geq g(S_t)$ for every $S_t \in \mathcal{S}^\circ$.

The explicit form of Θ and the proof of Theorem 4.1 are given in Appendix C.1. The theorem shows that the lower bound on the success probability rises with the power and falls with the type-I error. Theorem 4.1 suggests raising $b(\varphi, S_t)$ and lowering $g(\varphi, S_t)$, but the two cannot be set independently. A token Y_t with $Q^\pi(S_t, Y_t) \in (0,1)$ carries mass under both $P_0(\cdot \mid S_t)$ and $P_1(\cdot \mid S_t)$, and $\varphi(Y_t \mid S_t)$ removes the same fraction of each, so, beyond removing tokens with $Q^\pi(S_t, Y_t) = 0$,

raising the power at S_t raises the type-I error as well. The pairs of rates attainable at a state therefore form a restricted set, characterized in Appendix C.2. Since Θ is nondecreasing in b and nonincreasing in g, maximizing Θ subject to a bound on the type-I error at every $S_t \in \mathcal{S}^\circ$ amounts to maximizing the power under the same bound.

This is precisely the problem resolved by the Neyman–Pearson lemma: the most powerful test rejects H_0 at the largest values of the likelihood ratio (Neyman & Pearson, 1992; Lehmann & Romano, 2005). By the expressions for P_0 and P_1 above, for $S_t \in \mathcal{S}^\circ$ and $Q^\pi(S_t, Y_t) > 0$ this ratio is

$$\frac{P_1(Y_t \mid S_t)}{P_0(Y_t \mid S_t)} = \frac{V^\pi(S_t)}{1 - V^\pi(S_t)} \cdot \frac{1 - Q^\pi(S_t, Y_t)}{Q^\pi(S_t, Y_t)}, \tag{2}$$

which is strictly decreasing in $Q^\pi(S_t, Y_t)$. Ranking tokens by likelihood ratio is therefore the same as ranking them by their Q-function, and the most powerful test removes the tokens with the lowest Q-function, as formalized in the following theorem.

Theorem 4.2 (Optimal removed fractions are Q-thresholds)**.** *Fix $c \in (0, 1]$, and let φ_c remove every token whose Q-function falls below c, except those with the highest Q-function if these also fall below c:*

$$\varphi_c(Y_t \mid S_t) := \mathbf{1}\Big\{Q^\pi(S_t, Y_t) < \min\big\{c,\ m(\pi, S_t)\big\}\Big\},$$

where $m(\pi, S_t) = \max_{Y_t' \in \mathcal{A}(S_t)} Q^\pi(S_t, Y_t')$. Then φ_c is an in-support removed fraction, and for every in-support removed fraction φ with $g(\varphi, S_t) \le g(\varphi_c, S_t)$ for every $S_t \in \mathcal{S}^\circ$,

$$\Theta(g(\varphi, \cdot), b(\varphi, \cdot)) \le \Theta(g(\varphi_c, \cdot), b(\varphi_c, \cdot)).$$

Moreover, $\Theta(g(\varphi_c, \cdot), b(\varphi_c, \cdot)) \ge 1$, and $\Theta(g(\varphi_c, \cdot), b(\varphi_c, \cdot))$ is an increasing step function of c.

The theorem identifies the optimal removed fraction for maximizing the lower bound Θ at a given level of type-I error. Specifically, among all in-support removed fractions whose statewise type-I error does not exceed that of φ_c, the optimal rule is to deterministically remove exactly those candidate tokens whose Q^π-values fall below the threshold $\min\big\{c,\ m(\pi, S_t)\big\}$. The final part of the theorem shows that increasing c raises the lower bound on the success probability. The proof is given in Appendix C.3.

The removed fraction φ_c induces the policy $\mu_c := \mu_{\varphi_c}$, obtained by restricting the base distribution to the retained tokens and renormalizing. Formally:

$$\mu_c(Y_t \mid S_t) = \frac{\pi(Y_t \mid S_t)\, \mathbf{1}\Big\{Q^\pi(S_t, Y_t) \ge \min\big\{c,\ m(\pi, S_t)\big\}\Big\}}{Z_c(S_t)}, \tag{3}$$
$$Z_c(S_t) := \pi\big(Q^\pi(S_t, Y_t) \ge \min\big\{c,\ m(\pi, S_t)\big\} \mid S_t\big),$$

QFD approximates μ_c by replacing Q^π with a learned estimate, as described in Section 5.

Finally, we formalize preservation of the base model's successful trajectories. Recall that the type-I error $g(\varphi, S_t)$ measures the probability of rejecting H_0 at a single state when the base model would ultimately succeed. We now extend this notion to the full trajectory. Given an intervention μ, at each step t, after observing token Y_t, the test rejects H_0 with probability $\varphi_\mu(Y_t \mid S_t)$, independently across steps. Thus, conditional on a trajectory, the probability of passing all tests is $\prod_{t=1}^{T}\big(1-\varphi_\mu(Y_t \mid S_t)\big)$. The complementary event is that the test rejects H_0 at least once along the generation. Its probability, conditioned on a successful base-model trajectory, is the trajectory-level analogue of g, the type-I error of the sequential test, which we call the drop rate:

$$\mathrm{DR}(\mu) := \mathbb{E}_\pi\left[1 - \prod_{t=1}^{T}\big(1 - \varphi_\mu(Y_t \mid S_t)\big) \;\middle|\; r(X, Y) = 1\right]. \tag{4}$$

Since φ_μ removes the probability μ moves outside $\mathcal{A}(S_t)$ from every token, the drop rate accounts both for what the intervention removes within the support of the base policy and for the probability it moves outside it. It depends only on μ and π, allowing interventions from different families to be compared under the same notion of preservation.

For QFD, μ_c is supported on $\mathcal{A}(S_t)$, so by Proposition B.1, $\varphi_{\mu_c} = \varphi_c$, which takes only the values zero and one. Therefore,

$$\mathrm{DR}(\mu_c) = P_\pi\big(\text{the filter removes at least one token} \mid r(X, Y) = 1\big),$$

since the sequential test rejects exactly when the filter removes a token of a successful base generation. The next theorem shows that this probability is explicitly controlled by the threshold c.

Theorem 4.3 (Preservation of successful generations)**.** *Let* $c \in (0,1)$ *and* $p := P_\pi\big(r(X,Y)=1\big) \in (0,1)$*. Then*

$$\mathrm{DR}(\mu_c) \;\le\; \frac{c}{1-c}\cdot\frac{1-p}{p}.$$

The bound follows from Ville's inequality (Ville, 1939) applied to the likelihood ratio between the failure-conditioned and success-conditioned laws of the prefix, and the proof is given in Appendix C.4. Together with Theorem 4.2, this theorem describes the role of the threshold c: raising c raises the lower bound on the success probability, while the probability that a successful base trajectory is altered stays below a level that grows with c and vanishes as $c \to 0$.

5 Estimating the Q-function

In practice, the oracle Q-function is unavailable, so we replace Q^π with an estimate $\widehat{Q}^\pi$. Substituting $\widehat{Q}^\pi$ into the policy in equation 3 yields the estimated QFD policy:

$$\hat{\mu}_c(Y_t \mid S_t) = \frac{\pi(Y_t \mid S_t)\,\mathbf{1}\Big\{\widehat{Q}^\pi(S_t,Y_t) \ge \min\big\{c,\ \hat{m}(\pi,S_t)\big\}\Big\}}{\hat{Z}_c(S_t)},$$
$$\hat{Z}_c(S_t) := \pi\Big(\widehat{Q}^\pi(S_t,Y_t) \ge \min\big\{c,\ \hat{m}(\pi,S_t)\big\} \;\Big|\; S_t\Big), \tag{5}$$

where $\hat{m}(\pi,S_t) := \max_{Y_t' \in \mathcal{A}(S_t)} \widehat{Q}^\pi(S_t,Y_t')$. We set the threshold through a level $\alpha \in (0,1)$, the drop rate we are willing to accept. Raising α lets the filter intervene more often, trading the preservation of the base model's successful outputs for the power to remove tokens that lead to failure. In the experiments, we select c on a held-out validation set, so that the filter would remove a token from at most a fraction α of the base model's successful validation completions, which bounds the drop rate $\mathrm{DR}(\hat{\mu}_c)$ on those completions by α, as detailed in Appendix E.3.

Estimating Q^π is a supervised learning problem: we sample completions from π and assign the final reward $r(X,Y)$ as the label to every token in that completion. Thus, a token Y_t observed at state S_t receives label 1 with probability $Q^\pi(S_t,Y_t)$. Consequently, among all functions of the state and token, the minimizer of the expected binary cross-entropy loss is Q^π on the (S_t,Y_t) pairs that π visits (Hastie et al., 2009).

A naive estimator would map the hidden state directly to a score for every token. This would require learning a new $d \times |\mathcal{V}|$ output matrix, where d is the hidden size, essentially a second language-model head, which amounts to roughly 352M parameters for Nemotron 3.5 Lightning. Instead, we reuse the model's frozen language-model head W_{LM}. We learn only a single $d \times d$ matrix M, which transforms the hidden state h_t from which the base model computes its next-token logits at S_t before passing it through the existing head:

$$\widehat{Q}^\pi(S_t,\cdot) = 1 - \sigma\big(W_{\mathrm{LM}} M h_t\big), \tag{6}$$

where the sigmoid σ is applied to each token separately, since Q-values do not form a distribution over tokens, and $\sigma\big(W_{\mathrm{LM}} M h_t\big)$ estimates the probability of failure after each next token. We initialize M at zero, so every estimate starts at $1/2$. This produces a Q-value for every token in the vocabulary from the hidden state the base model already computes at each step, at the cost of one additional multiplication by W_{LM}. The matrix M has d^2 trainable parameters, roughly 7M for Nemotron 3.5 Lightning, while W_{LM} and the base model remain frozen. Across the models tested, this adds at most 0.25% to the active parameter count.

This training procedure has an inherent limitation: at each state, supervision is available only for the token that was actually sampled. As a result, a token receives direct supervision only at states where the base policy selected it, so coverage is uneven across the vocabulary, and the Q-estimator is best supervised for tokens that the base model samples frequently.

Fortunately, restrictions on the support of next tokens are standard in open-weight models: Nemotron and GLM use top-p (nucleus) sampling, which keeps the smallest set of highest-probability tokens whose cumulative probability is at least p (NVIDIA, 2025; 2026; Zeng et al.,

2026), while Gemma and Qwen combine it with top-k sampling, which keeps the k highest-probability tokens (Gemma Team, 2026; Qwen Team, 2026). We treat this restriction as part of the base policy π, so the support $\mathcal{A}(S_t)$ of Section 3 is exactly the candidate set, and QFD scores only these tokens. We apply each model's own restriction in all experiments.

The restriction also makes QFD more robust to errors in $\widehat{Q}^\pi$, as we show in Proposition D.1 in Appendix D. Suppose each estimate is within ϵ of its Q-value with probability at least $1-\delta$, and let $K(S_t)$ be the number of candidates at state S_t. Then, with probability at least $1-K(S_t)\,\delta$, all the estimates at S_t are this accurate. In that case, the estimated policy equation 5 can differ from the oracle policy equation 3 only on candidates whose Q-values lie within 2ϵ of the threshold. The total variation distance between the two policies is then at most the base probability of these candidates, divided by the probability the filter retains. A smaller candidate set therefore makes this guarantee more likely to hold, and excludes the tokens the estimator sees least in training.

6 Experiments

We evaluate QFD against both goals of Section 3: raising the success probability, as in goal 1, and preserving the base model's successful generations, which extends goal 2 from single steps to whole generations.

Tasks and models. We study seven task–model pairs covering three kinds of failure. For mathematical reasoning, we use DeepMath (He et al., 2026) with Nemotron 3.5 Lightning (NVIDIA, 2025; 2026), and measure final answer equivalence. For contextual faithfulness, we use FaithEval (Ming et al., 2025) with Qwen3.5-9B (Qwen Team, 2026) and Nemotron, pooling its counterfactual, inconsistent and unanswerable subsets, measuring whether the response selects the option the context supports, flags the conflict, or abstains, respectively. For harmful compliance, we use SALAD-Bench (attack-enhanced) (Li et al., 2024) and WildJailbreak (Jiang et al., 2024) with Gemma 4 E4B (Gemma Team, 2026) and Nemotron, and measure whether Llama Guard 4 (Inan et al., 2023; Meta, 2025) judges the response safe. Each model samples with the temperature, top-k and top-p recommended for it, which are part of π (Section 5 and Appendix E.2), and writes a reasoning trace on the mathematical-reasoning and faithfulness tasks. All methods are evaluated on the same held-out test prompts; the splits and success rules are detailed in Appendix E.1.

Methods. QFD decodes with $\hat{\mu}_c$ equation 5, using the estimator equation 6 and c selected for $\alpha \in \{0.05, 0.25, 0.45, 0.65, 0.85\}$ (Section 5). CD-Q, following controlled decoding (Mudgal et al., 2023), multiplies the base probability of each candidate in $\mathcal{A}(S_t)$ by $\exp\big(\widehat{Q}^\pi(S_t, Y_t)/\beta\big)$ and renormalizes; with the same estimator and candidate set, it differs from QFD only in reweighting candidates instead of thresholding them as Theorem 4.2 prescribes. ARGS (Khanov et al., 2024) emits the candidate maximizing $\pi(Y_t \mid S_t) + w\,\widehat{Q}^\pi(S_t, Y_t)$ among the base model's ten most probable tokens, with $\widehat{Q}^\pi$ in place of a separately trained reward model. For fine-tuning, we apply rejection-sampling fine-tuning (Yuan et al., 2023), training a LoRA adapter (Hu et al., 2022) on the base model's successful completions, and vary the scale of its update. For activation steering, we use CAA (Rimsky et al., 2024), DIM (Arditi et al., 2024) and LAT (Zou et al., 2023a) with the implementations, layer and factor selection, and evaluation of the SteeringSafety benchmark (Siu et al., 2026), and extend its range of factors to ± 12; DeepMath has no contrastive data from which to derive a steering direction. The strengths of each method are listed in Figure 1, and details are in Appendix E.4. We do not include value-filtered decoding (Einbinder et al., 2026) as a baseline: it evaluates a value for each of several sampled candidates before committing one, so an efficient implementation must fork and roll back the key–value cache at every step, which is specific to the architecture and harder still for models with recurrent layers, such as Nemotron; we compare with it on a separate testbed in Appendix E.7.

Metrics. We report the success rate, the drop rate equation 4 over the base model's successful test completions, and, as a complementary measure of preservation, the cosine similarity between e5-mistral embeddings (Wang et al., 2024) of each successful completion and the base model's successful completions on the same prompt. Confidence intervals are 90% percentile bootstrap confidence intervals over prompts (Efron & Tibshirani, 1993), computed separately for each measure;

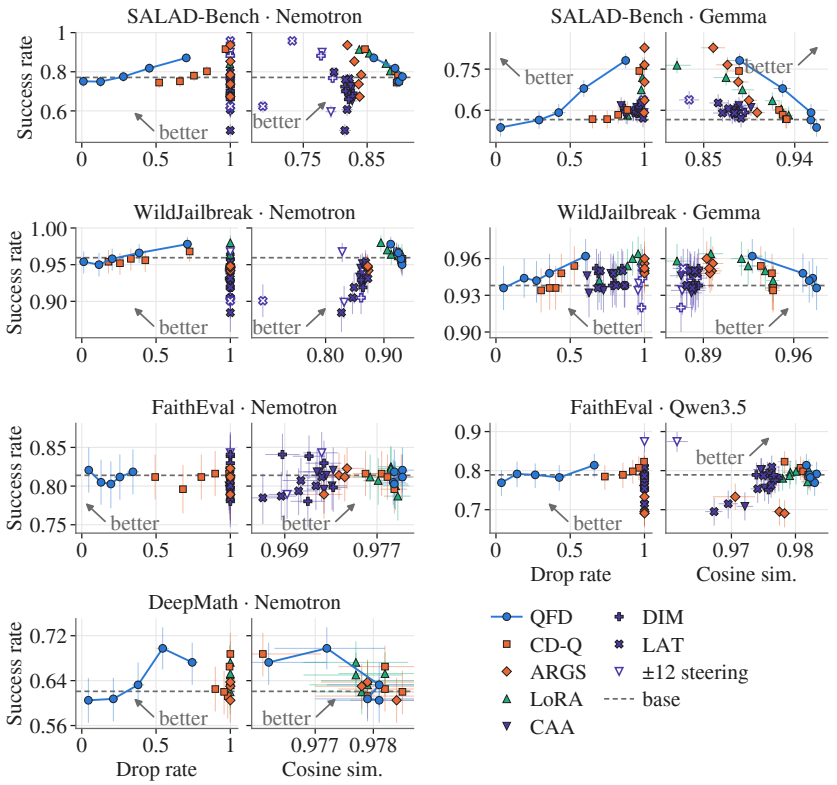

Figure 1: Success rate against drop rate (left panel of each pair) and against cosine similarity to the base model's completions (right panel of each pair), for each task–model pair. Each point is one method at one strength, and the dashed line marks the base model's success rate. Colour indicates the family of the method and marker shape the method; QFD's points are joined in order of α, steering at factor ± 12 is drawn hollow, and arrows point toward better trade-offs. For each pair, points whose success rate falls below the lower Tukey fence $Q_1 - 1.5\,(Q_3 - Q_1)$, computed from the success rates of that pair's points with a cosine similarity, are omitted from both panels; these are 20 points of steering and ARGS, none of QFD, and Figure 2 shows every point. Error bars are 90% bootstrap confidence intervals over prompts, computed separately for each axis. Method parameters, in increasing order: $\alpha \in \{0.05, 0.25, 0.45, 0.65, 0.85\}$ for QFD, $\beta \in \{5, 2, 1, 0.5, 0.1\}$ for CD-Q, $w \in \{0.001, 0.5, 1, 1.5, 2\}$ for ARGS, adapter scale $\in \{0.25, 0.5, 0.75, 1, 1.5, 2\}$ for LoRA, and factor $\in \{\pm 1, \pm 2, \pm 3, \pm 12\}$ for CAA, DIM and LAT.

for a change in success rate relative to the base model, the method and the base model are resampled on the same prompts, and we call the change a significant improvement when this confidence interval lies above zero. Other comparisons are between point estimates, and further details of all three measures are in Appendix E.5.

6.1 Results

No baseline is at least as successful as QFD at an equal or lower drop rate. At 28 of QFD's 35 settings, five values of α for each of the seven task–model pairs, no baseline at any strength reaches as low a drop rate, and at the other seven, every baseline that does is less successful (Figure 2). The baselines' drop rates are explained by Definition 3.1: at every step with more than one candidate, a reweighting removes a positive fraction from every candidate except the one it favors most, and these removals compound over completions of hundreds to thousands of tokens, whereas QFD removes only candidates whose estimate falls below c, from candidate sets with a median size of one or two tokens (Appendix E.3).

At their largest improvements in success rate, QFD's drop rate is 0.54–0.87, and the best baseline's 0.96–1.00. QFD significantly improves the success rate on six of the seven pairs. Comparing, for each pair, QFD's largest significant improvement with the best baseline's (Table 9 in Appendix E.6), the two have overlapping confidence intervals on four pairs, for instance $+0.077$ for QFD and $+0.067$ for CD-Q on DeepMath. On FaithEval with Qwen3.5, CAA at $+12$ improves the success rate more, by $+0.085$ against $+0.025$. On the remaining two pairs, the baseline's larger improvement does not reflect better task performance: LAT at -12 on SALAD-Bench with Nemotron produces long, often off-topic text that Llama Guard judges safe, and CAA at $+12$ on FaithEval with Nemotron improves the success rate only on the unanswerable subset, where answering "unknown" is scored as correct (Appendix E.6). At these improvements, QFD's completions are closer to the base model's in cosine similarity on all six pairs where it improves significantly, with non-overlapping confidence intervals on five.

The drop rate stays at or below α at 32 of QFD's 35 settings. Although c is selected on validation prompts, QFD's drop rate on the test prompts exceeds α at the other three by at most 0.04 (Appendix E.6).

QFD slows decoding by 1.2% on average. Across its 35 settings, QFD's decoding time ranges from 4.6% below to 9.1% above the base model's, and on six of the seven pairs it is within 3.2% at every α. On FaithEval with Nemotron, QFD's reasoning traces are up to 7% longer, and its decoding time up to 9.1% longer, or 1.6–4.3% per generated token. The only additional memory is the estimator's matrix M, 25–64 MiB, which is negligible compared to that of the base model (Appendix E.8).

7 Discussion

We introduced Q-filtered decoding (QFD), an inference-time intervention that raises a trained model's success rate on a target criterion while controlling how often it alters the base model's successful generations. Describing any intervention by the fraction of each token's base probability that it removes, we showed that the success probability of the full generation is lower-bounded by a quantity that increases with the power of the corresponding test and decreases with its type-I error, and that, by the Neyman–Pearson lemma, removing the candidates whose Q-values fall below a threshold maximizes this bound among interventions with the same type-I error at every state. With the exact Q-function, QFD's success probability is at least the base model's on every prompt, and its drop rate is upper bounded by a monotone function of its threshold. In practice, a single $d \times d$ matrix, applied to the frozen model's final hidden state, estimates the Q-values of all candidate tokens at once, adding at most 0.25% to the active parameters and, in our experiments, 1.2% to decoding time on average.

Across mathematical reasoning, contextual faithfulness and harmful compliance, QFD significantly improves the success rate on six of seven task–model pairs, and no baseline is at least as successful at an equal or lower drop rate. On four of the seven pairs, the confidence interval of QFD's largest improvement overlaps that of the best baseline's, at drop rates of 54–87% rather than 96–100%, and with completions closer to the base model's in cosine similarity. On the other three pairs, the best baseline's larger improvement comes at a drop rate of at least 0.999, and on two of them from outputs that satisfy the success rule without performing the task.

QFD has several limitations. First, its guarantees hold for the exact Q-function, whereas in practice QFD uses an estimate. Proposition D.1 bounds how far the estimated policy can deviate from the oracle policy at each state, but not the resulting success probability or drop rate of whole generations. Likewise, the threshold we select on held-out data carries no finite-sample guarantee on the drop rate, although empirically the drop rate stays at or below its target at 32 of 35 settings. Second, because QFD samples only tokens in the base model's candidate set, its success probability is zero on every prompt where the base model's is zero. Third, the Q-function depends on the base policy, including its sampling parameters, and on the reward, so the estimator, trained on labeled completions of the base model, must be re-estimated when the model, the task or the sampling parameters change. Finally, our success rules rely on automatic judges and substring matching, which, as the degenerate baseline outputs show, can score as successful outputs that do not perform the task.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the following tasks: help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, design or provide feedback on research methodology or experiments, implement methods, clean and reformat dataset, interpret results, create or edit software code, and draft sections of the manuscript.

We did not use generative AI tools to generate synthetic datasets, assist with translation, or support qualitative or thematic data analysis. The use of generative AI tools to propose or refine hypotheses was not applicable to this work.

All AI-assisted outputs were critically reviewed and verified by the authors. The authors made the final decisions regarding the theoretical development, methodology, experiments, implementation, analysis, and presentation of the work, and take full responsibility for the content and correctness of the paper.

REFERENCES

Anastasios Nikolas Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *The Twelfth International Conference on Learning Representations*, 2024.

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. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Yaswanth Chittepu, Blossom Metevier, Will Schwarzer, Austin Hoag, Scott Niekum, and Philip S Thomas. Reinforcement learning from human feedback with high-confidence safety guarantees. In *Reinforcement Learning Conference*, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2016.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.

Bat-Sheva Einbinder, Hen Davidov, Yee Whye Teh, Yarin Gal, and Yaniv Romano. Selective safety steering via value-filtered decoding. *arXiv preprint arXiv:2605.14746*, 2026.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Jonathan Geuter, Youssef Mroueh, and David Alvarez-Melis. Guided speculative inference for efficient test-time alignment of llms. In *International Conference on Learning Representations*, volume 2026, pp. 75016–75056, 2026.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Seungwook Han, Idan Shenfeld, Akash Srivastava, Yoon Kim, and Pulkit Agrawal. Value augmented sampling for language model alignment and personalization. *arXiv preprint arXiv:2405.06639*, 2024.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 138306–138322, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

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. In *International Conference on Learning Representations*, 2022.

Chia-Yu Hung, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Reward-guided tree search for inference time alignment of large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12575–12593, 2025.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*, 2023.

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

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Alex Qiu, Jiayi Zhou, Kaile Wang, Boxun Li, Sirui Han, Yike Guo, and Yaodong Yang. 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)*, pp. 31983–32016, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Fengqing Jiang, Zhangchen Xu, Yuetai Li, Luyao Niu, Zhen Xiang, Bo Li, Bill Yuchen Lin, and Radha Poovendran. Safechain: Safety of language models with long chain-of-thought reasoning capabilities. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 23303–23320, 2025.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, et al. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. *Advances in Neural Information Processing Systems*, 37:47094–47165, 2024.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. Args: Alignment as reward-guided search. In *International Conference on Learning Representations*, volume 2024, pp. 17755–17773, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025.

Erich Leo Lehmann and Joseph P Romano. *Testing statistical hypotheses*. Springer, 2005.

Bolian Li, Yifan Wang, Anamika Lochab, Ananth Grama, and Ruqi Zhang. Cascade reward sampling for efficient decoding-time alignment. In *Second Conference on Language Modeling*, 2025.

Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. Salad-bench: A hierarchical and comprehensive safety benchmark for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3923–3954, 2024.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6691–6706, 2021.

Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. Chain of hindsight aligns language models with feedback. In *International Conference on Learning Representations*, volume 2024, pp. 41395–41414, 2024.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7052–7063, 2021.

Meta. Llama Guard 4 12B. `https://huggingface.co/meta-llama/Llama-Guard-4-12B`, 2025.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, 2017.

Yifei Ming, Senthil Purushwalkam, Shrey Pandit, Zixuan Ke, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. Faitheval: Can your language model stay faithful to context, even if "the moon is made of marshmallows". In *International Conference on Learning Representations*, volume 2025, pp. 29430–29456, 2025.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, et al. Controlled decoding from language models. *arXiv preprint arXiv:2310.17022*, 2023.

J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. In *Breakthroughs in Statistics: Foundations and Basic Theory*, pp. 73–108. Springer, 1992.

NVIDIA. NVIDIA Nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

NVIDIA. NVIDIA Nemotron 3.5 Lightning 30b-a3b. `https://huggingface.co/nvidia/NVIDIA-Nemotron-3.5-Lightning-30B-A3B-BF16`, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Drew Prinster, Clara Fannjiang, Ji Won Park, Kyunghyun Cho, Anqi Liu, Suchi Saria, and Samuel Don Stanton. Conformal policy control. In *Forty-third International Conference on Machine Learning*, 2026.

Mohammad Atif Quamar, Mohammad Areeb, Mikhail Kuznetsov, Muslum Ozgur Ozmen, and Z Berkay Celik. Stars: Segment-level token alignment with rejection sampling in large language models. *arXiv preprint arXiv:2511.03827*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents. `https://qwen.ai/blog?id=qwen3.5`, 2026.

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

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.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Vincent Siu, Nicholas Crispino, David Park, Nathan W. Henry, Zhun Wang, Yang Liu, Dawn Song, and Chenguang Wang. Steeringsafety: Benchmarking representation steering in LLMs across safety perspectives. In *Forty-third International Conference on Machine Learning*, 2026.

Charlie Victor Snell, Ilya Kostrikov, Yi Su, Sherry Yang, and Sergey Levine. Offline RL for natural language generation with implicit language q learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An instruction-following LLaMA model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Jean Ville. *Etude critique de la notion de collectif*, volume 3. Gauthier-Villars Paris, 1939.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11897–11916, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Kevin Yang and Dan Klein. Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3511–3535, 2021.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023a.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023b.

A Related work

Aligning LLMs to satisfy desired criteria, such as safety, factual correctness, reliability, and instruction following, has received substantial attention in recent years (Ouyang et al., 2022; Christiano et al., 2017; Ziegler et al., 2019; Rafailov et al., 2023; Chittepu et al., 2025; Schulman et al., 2017; Liu et al., 2024). These alignment methods modify model behavior through fundamentally different mechanisms. Activation-steering approaches modify internal hidden representations using directions constructed from contrasting examples (Rimsky et al., 2024; Siu et al., 2026). Weight-based approaches instead adapt the model parameters, ranging from full fine-tuning, as in RLHF (Ouyang et al., 2022), to parameter-efficient methods such as LoRA (Hu et al., 2022). Various methods steer model behavior directly at inference time, avoiding reward-model training and model fine-tuning (Mudgal et al., 2023; Han et al., 2024; Liu et al., 2021; Hung et al., 2025; Khanov et al., 2024; Quamar et al., 2025; Yang & Klein, 2021; Jiang et al., 2025; Geuter et al., 2026; Li et al., 2025; Prinster et al., 2026). These approaches modify the base model's decoding policy using an external score or reward signal. For example, Khanov et al. (2024) combine the base model logits with a weighted reward score and select the token maximizing the resulting objective. Quamar et al. (2025); Li et al. (2025) instead use rejection sampling to draw from a reward-tilted version of the base distribution. Controlled decoding (Mudgal et al., 2023) trains a prefix-level scorer to estimate the expected reward of continuing from a partial response and exponentially tilts the base distribution according to this estimate.

The work most closely related to ours is Einbinder et al. (2026), which selectively intervenes during decoding to improve safety while controlling the rate of unnecessary interventions. Because scoring each candidate requires a separate forward computation, evaluating the full candidate set can be costly. The method therefore applies rejection sampling to a finite set of sampled candidates. If all candidates are rejected, it falls back to the highest-valued one, which means that the resulting distribution does not match the intended policy even with an oracle value function. QFD addresses both limitations by scoring the full candidate set in a single forward computation and directly masking low-Q tokens before renormalizing and sampling. This removes the need for rejection sampling, and makes the fallback part of the policy QFD samples from, so that it matches the intended policy whenever the Q-function is exact.

B Proofs for Section 3

Throughout this section, fix $S_t \in \mathcal{S}$ and write $\mathcal{A} := \mathcal{A}(S_t)$.

Proposition B.1 (In-support removed fractions are fully expressive)**.** *The map $\varphi \mapsto \mu_\varphi$ is a bijection from in-support removed fractions onto the policies supported on $\mathcal{A}(S_t)$ at every $S_t \in \mathcal{S}$, with inverse $\mu \mapsto \varphi_\mu$.*

Proof. Let μ be supported on $\mathcal{A}(S_t)$ at every $S_t \in \mathcal{S}$. At every state, write $\rho := \mu/\pi$ on $\mathcal{A}$ and $R := \max_{\mathcal{A}} \rho$, which is positive because μ places all of its probability on $\mathcal{A}$. By equation 1 with $\mu(\mathcal{A} \mid S_t) = 1$, $1 - \varphi_\mu = \rho/R$, which attains one, so φ_μ is an in-support removed fraction. Moreover, $Z_{\varphi_\mu}(S_t) = \sum_{\mathcal{A}} \pi\rho/R = 1/R$, so $\pi(1 - \varphi_\mu)/Z_{\varphi_\mu}(S_t) = \pi\rho = \mu$ on $\mathcal{A}$; both sides vanish outside $\mathcal{A}$, so $\mu_{\varphi_\mu} = \mu$. Conversely, let φ be an in-support removed fraction. Some $Y_t^\star \in \mathcal{A}$ has $\varphi(Y_t^\star \mid S_t) = 0$, so $Z_\varphi(S_t) \geq \pi(Y_t^\star \mid S_t) > 0$ and μ_φ is well defined. It vanishes outside $\mathcal{A}$, and on $\mathcal{A}$ its ratio to π is $(1 - \varphi)/Z_\varphi(S_t)$, whose maximum is $1/Z_\varphi(S_t)$. By equation 1 with $\mu_\varphi(\mathcal{A} \mid S_t) = 1$, $1 - \varphi_{\mu_\varphi} = 1 - \varphi$. The two maps are therefore mutually inverse. □

C Proofs for Section 4

Throughout this appendix, a completion (X, Y) is reachable under π if $P_\pi(X, Y) > 0$, and successful if $R := r(X, Y) = 1$. For a policy μ and $S_t \in \mathcal{S}$, $V^\mu(S_t)$ denotes the probability of success when generation continues from S_t under μ. At a fixed state S_t we abbreviate $V := V^\pi(S_t)$.

C.1 Proof of Theorem 4.1

Definition C.1 (Per-state factor). *For $S_t \in \mathcal{S}^\circ$ and $(u, v) \in [0, 1]^2 \setminus \{(1, 1)\}$, let*

$$\kappa_{S_t}(u, v) := \frac{1 - u}{V(1 - u) + (1 - V)(1 - v)}, \tag{7}$$

whose denominator is positive because $V \in (0, 1)$ and u, v are not both one.

Lemma C.2 (One-step identity). *Let φ be an in-support removed fraction with induced policy $\mu := \mu_\varphi$, let $S_t \in \mathcal{S}$ with $V > 0$, and write $g := g(\varphi, S_t)$. Then*

$$\sum_{Y_t \in \mathcal{V}} \mu(Y_t \mid S_t)\, Q^\pi(S_t, Y_t) = \frac{V(1 - g)}{Z_\varphi(S_t)}.$$

If $V = 1$, then $Z_\varphi(S_t) = 1 - g$ and the sum equals one. If $V \in (0, 1)$, then, writing $b := b(\varphi, S_t)$,

$$Z_\varphi(S_t) = V(1 - g) + (1 - V)(1 - b),$$

and the sum equals $V\kappa_{S_t}(g, b)$. In particular, $g(\varphi, S_t)$ and $b(\varphi, S_t)$ are not both one at any $S_t \in \mathcal{S}^\circ$.

Proof. Terms with $\pi(Y_t \mid S_t) = 0$ vanish throughout, so all sums below run over the support of $\pi(\cdot \mid S_t)$. By the expression for P_0 in Section 4, $\pi(Y_t \mid S_t)Q^\pi(S_t, Y_t) = VP_0(Y_t \mid S_t)$, so

$$\begin{aligned}\sum_{Y_t} \mu(Y_t \mid S_t)\, Q^\pi(S_t, Y_t) &= \frac{1}{Z_\varphi(S_t)} \sum_{Y_t} \pi(Y_t \mid S_t)\big(1 - \varphi(Y_t \mid S_t)\big) Q^\pi(S_t, Y_t) \\ &= \frac{V}{Z_\varphi(S_t)} \sum_{Y_t} P_0(Y_t \mid S_t)\big(1 - \varphi(Y_t \mid S_t)\big),\end{aligned}$$

and the last sum is $1 - g$. If $V = 1$, then $Q^\pi(S_t, Y_t) = 1$ on the support, so $P_0(\cdot \mid S_t) = \pi(\cdot \mid S_t)$ and $Z_\varphi(S_t) = 1 - \mathbb{E}_\pi[\varphi] = 1 - g$. If $V \in (0, 1)$, the expression for P_1 gives $\pi(Y_t \mid S_t)\big(1 - Q^\pi(S_t, Y_t)\big) = (1 - V)P_1(Y_t \mid S_t)$, and adding the two identities gives $\pi = VP_0 + (1 - V)P_1$. Hence $Z_\varphi(S_t) = 1 - \mathbb{E}_\pi[\varphi] = 1 - Vg - (1 - V)b$, and substituting into the first display gives $V\kappa_{S_t}(g, b)$. Finally, $Z_\varphi(S_t) > 0$ excludes $g = b = 1$ at every $S_t \in \mathcal{S}^\circ$. □

Definition C.3 (The map Θ). *Let $\mathcal{G}$ be the set of pairs $\big(g(\varphi, \cdot), b(\varphi, \cdot)\big)$ of rate functions on $\mathcal{S}^\circ$ of in-support removed fractions φ. For $(g, b) \in \mathcal{G}$, let*

$$\Theta(g, b) := \inf \prod_{t \leq T:\ S_t \in \mathcal{S}^\circ} \kappa_{S_t}\big(g(S_t), b(S_t)\big), \tag{8}$$

the infimum taken over successful completions reachable under π, with an empty product equal to one. Every factor is defined by Lemma C.2. We write $\theta(\varphi) := \Theta\big(g(\varphi, \cdot), b(\varphi, \cdot)\big)$.

Lemma C.4 (Monotonicity of the per-state factor)**.** *Fix $S_t \in \mathcal{S}^\circ$. On $[0,1]^2 \setminus \{(1,1)\}$, $\kappa_{S_t}(u,v)$ is nondecreasing in v and nonincreasing in u, and $\kappa_{S_t}(u,v) \geq 1$ if and only if $v \geq u$. Moreover, for $u < 1$,*

$$\kappa_{S_t}(u,v) = \frac{1}{V + (1-V)\,\gamma(u,v)}, \qquad \gamma(u,v) := \frac{1-v}{1-u}. \tag{9}$$

Proof. If $u = 1$, then $v < 1$ and $\kappa_{S_t}(u,v) = 0$, and $v < u$, so all claims hold at such points. For $u < 1$, dividing the numerator and denominator of equation 7 by $1-u$ gives equation 9. Since $V < 1$, $V + (1-V)\gamma$ is strictly increasing in γ, and γ is nonincreasing in v and nondecreasing in u; this gives the monotonicity, which is consistent with the value zero at $u = 1$. Finally, $\kappa_{S_t}(u,v) \geq 1$ if and only if $\gamma(u,v) \leq 1$, that is, if and only if $v \geq u$. □

Proof of Theorem 4.1. Fix an in-support removed fraction φ and write $\mu := \mu_\varphi$. For $S_t \in \mathcal{S}$ with $V^\pi(S_t) > 0$, let $\bar{\kappa}(S_t) := \kappa_{S_t}\big(g(\varphi, S_t), b(\varphi, S_t)\big)$ if $S_t \in \mathcal{S}^\circ$ and $\bar{\kappa}(S_t) := 1$ if $V^\pi(S_t) = 1$, so that

$$\sum_{Y_t \in \mathcal{V}} \mu(Y_t \mid S_t)\, Q^\pi(S_t, Y_t) = V^\pi(S_t)\, \bar{\kappa}(S_t) \tag{10}$$

by Lemma C.2, and let $\rho(S_t) := V^\mu(S_t)/V^\pi(S_t)$. Every state along a successful completion reachable under π has $V^\pi(S_t) > 0$, and $\bar{\kappa}$ equals one at those with $V^\pi(S_t) = 1$, so the product in equation 8 equals $\prod_{t \leq T} \bar{\kappa}(S_t)$. We show by backward induction on t, starting from $t = T_{\max}$, that

$$\rho(S_t) \;\geq\; \inf \prod_{i=t}^{T} \bar{\kappa}(S_i), \tag{11}$$

the infimum taken over successful completions reachable under π that extend S_t.

Fix S_t with $V^\pi(S_t) > 0$, and consider a token Y_t with $\pi(Y_t \mid S_t) > 0$. If $Q^\pi(S_t, Y_t) = 0$, every continuation of (S_t, Y_t) under π fails, and so does every continuation under μ, since μ is supported on the support of π at every state. If $Q^\pi(S_t, Y_t) > 0$ and generation stops at step t, the completed response is successful under both policies; set $\rho'(Y_t) := 1$ and $I(Y_t) := 1$. If $Q^\pi(S_t, Y_t) > 0$ and generation continues, then $V^\pi(S_{t+1}) = Q^\pi(S_t, Y_t) > 0$; set $\rho'(Y_t) := \rho(S_{t+1})$ and let $I(Y_t)$ be the infimum of $\prod_{i=t+1}^{T} \bar{\kappa}(S_i)$ over successful completions reachable under π that extend S_{t+1}. In both cases the success probability under μ after emitting Y_t is $Q^\pi(S_t, Y_t)\, \rho'(Y_t)$, and $\rho'(Y_t) \geq I(Y_t)$, trivially when generation stops and by the induction hypothesis otherwise. At $t = T_{\max}$ generation always stops, so the induction hypothesis is not needed there.

The set of tokens with $\pi(Y_t \mid S_t) > 0$ and $Q^\pi(S_t, Y_t) > 0$ is nonempty because $V^\pi(S_t) > 0$; let m be the minimum of $I(Y_t)$ over it. Conditioning on the next token and using equation 10,

$$\begin{aligned} V^\mu(S_t) &= \sum_{Y_t :\, Q^\pi(S_t, Y_t) > 0} \mu(Y_t \mid S_t)\, Q^\pi(S_t, Y_t)\, \rho'(Y_t) \\ &\geq m \sum_{Y_t} \mu(Y_t \mid S_t)\, Q^\pi(S_t, Y_t) \\ &= m\, V^\pi(S_t)\, \bar{\kappa}(S_t). \end{aligned}$$

Dividing by $V^\pi(S_t)$ gives $\rho(S_t) \geq \bar{\kappa}(S_t)\, m$. Every successful completion extending S_t passes through one such token, and $\bar{\kappa}(S_t) \geq 0$, so $\bar{\kappa}(S_t)\, m$ is the right-hand side of equation 11.

At $t = 1$ the state is the prompt X. If $P_\pi(R = 1 \mid X) > 0$, then $\rho(X) = P_\mu(R = 1 \mid X)/P_\pi(R = 1 \mid X)$, and the right-hand side of equation 11 is an infimum over a subset of the completions in equation 8, hence at least $\theta(\varphi)$. If $P_\pi(R = 1 \mid X) = 0$, the bound holds trivially.

For the remaining claims, let $(g, b), (g', b') \in \mathcal{G}$ with $b \geq b'$ and $g \leq g'$ pointwise on $\mathcal{S}^\circ$. By Lemma C.4, $\kappa_{S_t}\big(g(S_t), b(S_t)\big) \geq \kappa_{S_t}\big(g'(S_t), b'(S_t)\big) \geq 0$ at every $S_t \in \mathcal{S}^\circ$, so the products in equation 8 compare in the same direction along every completion, and taking the infimum over the same set gives $\Theta(g, b) \geq \Theta(g', b')$. If $b \geq g$ pointwise, every factor is at least one by Lemma C.4, so every product is at least one and $\Theta(g, b) \geq 1$. □

C.2 Attainable rates and their frontier

Proposition C.5 (Attainable pairs). *Fix $S_t \in \mathcal{S}^\circ$ and let*

$$\mathcal{R}(S_t) := \Big\{ \big(\mathbb{E}_{P_0(\cdot|S_t)}[\varphi],\, \mathbb{E}_{P_1(\cdot|S_t)}[\varphi]\big) : \varphi : \mathcal{V} \to [0,1] \Big\}.$$

Then $\mathcal{R}(S_t)$ is convex and compact, and

$$\beta_{S_t}(u) := \max\{v : (u,v) \in \mathcal{R}(S_t)\}$$

is well defined, concave and nondecreasing on $[0,1]$, with $\beta_{S_t}(1) = 1$.

Proof. The map $\varphi \mapsto \big(\mathbb{E}_{P_0}[\varphi], \mathbb{E}_{P_1}[\varphi]\big)$ is linear, and $[0,1]^{\mathcal{V}}$ is convex and compact because $\mathcal{V}$ is finite; hence $\mathcal{R}(S_t)$, its image, is convex and compact. The constant map $\varphi \equiv u$ gives $(u,u) \in \mathcal{R}(S_t)$ for every $u \in [0,1]$, so the maximum defining $\beta_{S_t}(u)$ is over a nonempty compact set and is attained. Concavity follows from convexity of $\mathcal{R}(S_t)$: given $u_1, u_2 \in [0,1]$ and $\lambda \in [0,1]$, the pairs $\big(u_i, \beta_{S_t}(u_i)\big)$ lie in $\mathcal{R}(S_t)$, so their convex combination does too, whence $\beta_{S_t}\big(\lambda u_1 + (1-\lambda)u_2\big) \geq \lambda\beta_{S_t}(u_1) + (1-\lambda)\beta_{S_t}(u_2)$. Taking $\varphi \equiv 1$ gives $\beta_{S_t}(1) = 1$, which is the largest value β_{S_t} can take, and a concave function attaining its maximum at the right endpoint of its domain is nondecreasing. □

Lemma C.6 (The frontier retention ratio). *Fix $S_t \in \mathcal{S}^\circ$ and define*

$$\gamma^*_{S_t}(u) := \frac{1 - \beta_{S_t}(u)}{1-u}, \qquad u \in [0,1).$$

Then $\gamma^*_{S_t}$ is nonincreasing.

Proof. Let $0 \leq u_1 < u_2 < 1$ and write $u_2 = \lambda u_1 + (1-\lambda)\cdot 1$ with $\lambda = (1-u_2)/(1-u_1) \in (0,1)$. By Proposition C.5, β_{S_t} is concave with $\beta_{S_t}(1) = 1$, so

$$\beta_{S_t}(u_2) \geq \lambda\beta_{S_t}(u_1) + (1-\lambda), \qquad \text{that is} \qquad 1 - \beta_{S_t}(u_2) \leq \lambda\big(1 - \beta_{S_t}(u_1)\big).$$

Dividing by $1 - u_2 = \lambda(1-u_1)$ gives $\gamma^*_{S_t}(u_2) \leq \gamma^*_{S_t}(u_1)$. □

By equation 9, κ_{S_t} depends on the two rates only through γ, and Lemma C.6 shows that along the frontier a larger type-I error gives a smaller γ. This is why the optimal rule meets a bound on the type-I error with equality.

C.3 Proof of Theorem 4.2

Lemma C.7 (Threshold rules attain the frontier). *Fix $S_t \in \mathcal{S}^\circ$ and $\ell \in (0,1]$, and let $\varphi^\ell(Y_t \mid S_t) := \mathbb{1}\{Q^\pi(S_t, Y_t) < \ell\}$ on the support of $\pi(\cdot \mid S_t)$, with rates $u_\ell := \mathbb{E}_{P_0(\cdot|S_t)}[\varphi^\ell]$ and $v_\ell := \mathbb{E}_{P_1(\cdot|S_t)}[\varphi^\ell]$. Then $v_\ell = \beta_{S_t}(u_\ell)$ and $v_\ell \geq u_\ell$.*

Proof. For Y_t with $\pi(Y_t \mid S_t) > 0$ and $Q^\pi(S_t, Y_t) > 0$, equation equation 2 gives

$$\frac{P_1(Y_t \mid S_t)}{P_0(Y_t \mid S_t)} = \psi\big(Q^\pi(S_t, Y_t)\big), \qquad \psi(w) := \frac{V}{1-V} \cdot \frac{1-w}{w},$$

while $P_0(Y_t \mid S_t) = 0 < P_1(Y_t \mid S_t)$ when $Q^\pi(S_t, Y_t) = 0$. Since ψ is strictly decreasing on $(0,1]$, setting $k := \psi(\ell) \geq 0$ gives

$$\{Y_t : Q^\pi(S_t, Y_t) < \ell\} = \{Y_t : P_1(Y_t \mid S_t) > k\, P_0(Y_t \mid S_t)\}$$

on the support, the tokens with zero Q-function belonging to both sides. Hence φ^ℓ is a nonrandomised likelihood-ratio test with threshold k and size u_ℓ.

Let $\varphi : \mathcal{V} \to [0,1]$ have rates (u,v) with $u \leq u_\ell$. For every Y_t,

$$\big(\varphi^\ell(Y_t \mid S_t) - \varphi(Y_t)\big)\big(P_1(Y_t \mid S_t) - k\, P_0(Y_t \mid S_t)\big) \geq 0,$$

since the first factor is nonnegative where $\varphi^\ell = 1$, on which the second is positive, and nonpositive where $\varphi^\ell = 0$, on which the second is nonpositive. Summing over $\mathcal{V}$ gives $(v_\ell - v) - k(u_\ell - u) \geq 0$, so $v_\ell - v \geq k(u_\ell - u) \geq 0$. Thus $v \leq v_\ell$ for every rule of size at most u_ℓ, and $v_\ell = \beta_{S_t}(u_\ell)$. This is the Neyman–Pearson lemma specialised to the present test (Lehmann & Romano, 2005).

For the second claim, let $A := \{Y_t : P_1 > kP_0\}$, so that $\varphi^\ell = \mathbb{1}_A$. Then

$$v_\ell - k\,u_\ell = P_1(A) - k\,P_0(A) \geq 0, \qquad (1 - v_\ell) - k(1 - u_\ell) = P_1(A^c) - k\,P_0(A^c) \leq 0.$$

If $k \geq 1$, the first gives $v_\ell \geq k\,u_\ell \geq u_\ell$; if $k < 1$, the second gives $1 - v_\ell \leq k(1 - u_\ell) \leq 1 - u_\ell$, that is, $v_\ell \geq u_\ell$. □

Proof of Theorem 4.2. We first show that φ_c is an in-support removed fraction. At every $S_t \in \mathcal{S}$, a token attaining the largest Q-function over the support of $\pi(\cdot \mid S_t)$ has Q-function at least the threshold level $\min\{c, \max_{\pi(Y'_t|S_t)>0} Q^\pi(S_t, Y'_t)\}$, so φ_c keeps it, and the minimum of $\varphi_c(\cdot \mid S_t)$ over the support of $\pi(\cdot \mid S_t)$ is zero.

Fix $S_t \in \mathcal{S}^\circ$ and let $Q^\star := \max_{\pi(Y_t|S_t)>0} Q^\pi(S_t, Y_t)$, which is positive because $V > 0$. On the support of $\pi(\cdot \mid S_t)$, φ_c is the threshold rule φ^ℓ of Lemma C.7 at level $\ell := \min\{c, Q^\star\} \in (0, 1]$. A token attaining $Q^\star$ is kept and has $P_0(Y_t \mid S_t) > 0$, so $g(\varphi_c, S_t) < 1$.

Let φ be an in-support removed fraction with $g(\varphi, S_t) \leq g(\varphi_c, S_t)$, so that $g(\varphi, S_t) < 1$. Since $b(\varphi, S_t) \leq \beta_{S_t}\big(g(\varphi, S_t)\big)$ by the definition of β_{S_t},

$$\gamma\big(g(\varphi, S_t), b(\varphi, S_t)\big) \geq \gamma^*_{S_t}\big(g(\varphi, S_t)\big) \geq \gamma^*_{S_t}\big(g(\varphi_c, S_t)\big) = \gamma\big(g(\varphi_c, S_t), b(\varphi_c, S_t)\big),$$

using Lemma C.6 for the second inequality and Lemma C.7 for the equality. By equation 9, the per-state factor of φ at S_t is at most that of φ_c. This holds at every $S_t \in \mathcal{S}^\circ$, both factors are nonnegative, and the products in equation 8 run over the same states along the same completions, so $\theta(\varphi) \leq \theta(\varphi_c)$.

By Lemma C.7, $b(\varphi_c, S_t) \geq g(\varphi_c, S_t)$ at every $S_t \in \mathcal{S}^\circ$, so $\theta(\varphi_c) \geq 1$ by Theorem 4.1.

Finally, let $c < c'$. At every $S_t \in \mathcal{S}^\circ$ the level $\min\{c, Q^\star\}$ is at most $\min\{c', Q^\star\}$, so φ_c removes a subset of the tokens $\varphi_{c'}$ removes and $g(\varphi_c, S_t) \leq g(\varphi_{c'}, S_t) < 1$. Both rules attain the frontier at their own sizes by Lemma C.7, and Lemma C.6 gives

$$\gamma^*_{S_t}\big(g(\varphi_{c'}, S_t)\big) \leq \gamma^*_{S_t}\big(g(\varphi_c, S_t)\big),$$

so the per-state factor of $\varphi_{c'}$ is at least that of φ_c at every $S_t \in \mathcal{S}^\circ$, and $\theta(\varphi_{c'}) \geq \theta(\varphi_c)$. □

C.4 Proof of Theorem 4.3

Throughout this subsection $p := P_\pi(R = 1) \in (0, 1)$.

Lemma C.8 (Successful completions below the threshold)**.** *For $c \in (0, 1)$,*

$$P_\pi\big(R = 1 \text{ and } Q^\pi(S_t, Y_t) < c \text{ for some } t \leq T\big) \;\leq\; \frac{c}{1 - c}\,(1 - p).$$

Proof. Fix a prompt X and write $p(X) := P_\pi(R = 1 \mid X)$. If $p(X) = 0$ the event has probability zero. If $p(X) = 1$, every continuation under π succeeds, so $Q^\pi(S_t, Y_t) = 1$ along every completion and the event again has probability zero. Let $p(X) \in (0, 1)$. For $0 \leq t \leq T$, let $V_t := P_\pi(R = 1 \mid X, Y_{1:t})$, so that $V_0 = p(X)$ and $V_t = Q^\pi(S_t, Y_t)$ for $1 \leq t \leq T$; on the event $R = 1$, every V_t is positive. Define

$$M_t := \frac{1 - V_t}{V_t} \cdot \frac{p(X)}{1 - p(X)}, \qquad 0 \leq t \leq T,$$

and $M_t := M_T$ for $t > T$. By Bayes' rule,

$$M_t = \frac{P_\pi(Y_{1:t} \mid X, R = 0)}{P_\pi(Y_{1:t} \mid X, R = 1)},$$

the likelihood ratio of the prefix under the failure-conditioned law against the success-conditioned law. Under $P_\pi(\cdot \mid X, R = 1)$, $(M_t)_{t \geq 0}$ is therefore a nonnegative supermartingale with $M_0 = 1$; it

is a supermartingale rather than a martingale because the failure-conditioned law can place mass on tokens with zero Q-function, which the success-conditioned law does not. Ville's inequality (Ville, 1939) gives $P_\pi\big(\sup_t M_t \geq \lambda \mid X, R = 1\big) \leq 1/\lambda$ for every $\lambda > 0$. Since $V_t < c$ if and only if $M_t > \lambda := \frac{1-c}{c} \cdot \frac{p(X)}{1-p(X)}$,

$$P_\pi\big(Q^\pi(S_t, Y_t) < c \text{ for some } t \leq T \;\big|\; X, R = 1\big) \leq \frac{c}{1-c} \cdot \frac{1-p(X)}{p(X)}.$$

Multiplying by $p(X)$ gives $P_\pi\big(R = 1 \text{ and } Q^\pi(S_t, Y_t) < c \text{ for some } t \leq T \mid X\big) \leq \frac{c}{1-c}\big(1 - p(X)\big)$ in all three cases, and taking the expectation over $X \sim \mathcal{D}$ proves the lemma. □

Proof of Theorem 4.3. We first identify $\mathrm{DR}(\mu_c)$. Fix $S_t \in \mathcal{S}$. By Theorem 4.2, φ_c is an in-support removed fraction and $\mu_c = \mu_{\varphi_c}$, so Proposition B.1 gives $\varphi_{\mu_c} = \varphi_c$. Since φ_c takes only the values zero and one, every factor in equation 4 is the indicator that φ_c keeps the emitted token, and the product is the indicator that it keeps every token of the completion. Hence

$$\mathrm{DR}(\mu_c) = P_\pi\big(\varphi_c(Y_t \mid S_t) = 1 \text{ for some } t \leq T \;\big|\; R = 1\big).$$

A token is removed only if $Q^\pi(S_t, Y_t) < \min\{c, m(\pi, S_t)\} \leq c$, so

$$\big\{\varphi_c(Y_t \mid S_t) = 1 \text{ for some } t \leq T\big\} \subseteq \big\{Q^\pi(S_t, Y_t) < c \text{ for some } t \leq T\big\}.$$

By Lemma C.8, the probability that a completion is successful and meets the right-hand event is at most $\frac{c}{1-c}(1-p)$. Dividing by p gives the claim. □

D Robustness to Q-estimation error

We analyze how estimation errors in $\widehat{Q}^\pi$ cause the estimated policy in equation 5 to deviate from the oracle policy in equation 3, and relate this deviation to the size of the candidate set, $K(S_t) := |\mathcal{A}(S_t)|$.

Proposition D.1 (Robustness of Q-filtered decoding)**.** *Fix $S_t \in \mathcal{S}$, and let $\tau(S_t) := \min\{c, m(\pi, S_t)\}$ and $\widehat{\tau}(S_t) := \min\{c, \hat{m}(\pi, S_t)\}$ be the oracle and estimated thresholds. Assume that, for every $Y_t \in \mathcal{A}(S_t)$,*

$$P\Big(\big|\widehat{Q}^\pi(S_t, Y_t) - Q^\pi(S_t, Y_t)\big| > \epsilon\Big) \leq \delta,$$

where the probability is over the data used to fit $\widehat{Q}^\pi$, and let

$$\rho_{2\epsilon}(S_t) := \sum_{Y_t \in \mathcal{A}(S_t):\, |Q^\pi(S_t, Y_t) - \tau(S_t)| \leq 2\epsilon} \pi(Y_t \mid S_t)$$

be the base probability of the candidates whose Q-function lies within 2ϵ of the threshold. Then, with probability at least $1 - K(S_t)\,\delta$,

$$\mathrm{TV}\big(\hat{\mu}_c(\cdot \mid S_t), \mu_c(\cdot \mid S_t)\big) \leq \frac{\rho_{2\epsilon}(S_t)}{\max\{Z_c(S_t), \hat{Z}_c(S_t)\}}.$$

The proposition shows that QFD is robust to estimation errors whenever little base-policy probability lies near the filtering threshold. The estimated policy can differ from the oracle policy only on candidates whose Q-function is within 2ϵ of the threshold, and the resulting deviation is controlled by their base probability. The candidate set enters in two ways: the probability that every estimate is accurate is at least $1 - K(S_t)\,\delta$, which decreases with the number of candidates, and more candidates can place more probability near the threshold. The proposition assumes a common δ across candidates, but, since the estimator is supervised only on the tokens the base policy samples (Section 5), the estimates are least reliable for the least probable candidates, which are exactly those a restriction removes. Restricting the candidate set can therefore improve robustness by reducing the opportunity for estimation errors, their likelihood, and the probability of near-threshold tokens. In our experiments, the candidate set has a median size of one or two tokens on every task–model pair (Table 6).

Proof of Proposition D.1. Write $\mathcal{A} := \mathcal{A}(S_t)$, and let $M := \{Y_t \in \mathcal{A} : Q^\pi(S_t, Y_t) \geq \tau(S_t)\}$ and $\widehat{M} := \{Y_t \in \mathcal{A} : \widehat{Q}^\pi(S_t, Y_t) \geq \widehat{\tau}(S_t)\}$ be the accepted sets, so that $\mu_c = \pi \mathbf{1}_M / Z_c$ and $\hat{\mu}_c = \pi \mathbf{1}_{\widehat{M}} / \hat{Z}_c$, with $Z_c = \pi(M \mid S_t)$ and $\hat{Z}_c = \pi(\widehat{M} \mid S_t)$. Both normalizers are positive, because the tokens attaining $m(\pi, S_t)$ and $\hat{m}(\pi, S_t)$ belong to M and $\widehat{M}$, respectively.

By a union bound over the $K(S_t)$ candidates, with probability at least $1 - K(S_t)\,\delta$ every estimate satisfies $|\widehat{Q}^\pi(S_t, Y_t) - Q^\pi(S_t, Y_t)| \leq \epsilon$, and we assume this event. Then $|\hat{m}(\pi, S_t) - m(\pi, S_t)| \leq \epsilon$, and since $x \mapsto \min\{c, x\}$ is 1-Lipschitz, $|\widehat{\tau}(S_t) - \tau(S_t)| \leq \epsilon$. If $Q^\pi(S_t, Y_t) \geq \tau(S_t) + 2\epsilon$, then $\widehat{Q}^\pi(S_t, Y_t) \geq \tau(S_t) + \epsilon \geq \widehat{\tau}(S_t)$, so Y_t belongs to both M and $\widehat{M}$. If $Q^\pi(S_t, Y_t) < \tau(S_t) - 2\epsilon$, then $\widehat{Q}^\pi(S_t, Y_t) < \tau(S_t) - \epsilon \leq \widehat{\tau}(S_t)$, so Y_t belongs to neither. Hence every token in the symmetric difference $M \triangle \widehat{M}$ has $|Q^\pi(S_t, Y_t) - \tau(S_t)| \leq 2\epsilon$, and $\pi(M \triangle \widehat{M} \mid S_t) \leq \rho_{2\epsilon}(S_t)$.

Suppose $Z_c \geq \hat{Z}_c$. On $M \cap \widehat{M}$ we have $\pi/Z_c \leq \pi/\hat{Z}_c$, and outside it one of the two policies vanishes, so

$$\begin{aligned}
\mathrm{TV}(\hat{\mu}_c, \mu_c) &= 1 - \sum_{Y_t \in \mathcal{A}} \min\{\hat{\mu}_c, \mu_c\} \\
&= 1 - \frac{\pi(M \cap \widehat{M} \mid S_t)}{Z_c} \\
&= \frac{\pi(M \setminus \widehat{M} \mid S_t)}{Z_c} \\
&\leq \frac{\pi(M \triangle \widehat{M} \mid S_t)}{Z_c} \\
&\leq \frac{\rho_{2\epsilon}(S_t)}{\max\{Z_c, \hat{Z}_c\}}.
\end{aligned}$$

The case $\hat{Z}_c \geq Z_c$ is symmetric, with $\widehat{M} \setminus M$ and $\hat{Z}_c$ in place of $M \setminus \widehat{M}$ and Z_c. □

E Experimental details

E.1 Tasks, splits and success rules

Each dataset is split into training, validation and test prompts (Table 1). Every method generates one completion per test prompt, on the same test prompts; the base model generates eight completions per DeepMath test problem and one per test prompt elsewhere. The Q-function estimators and the LoRA adapters are trained on the training prompts. The estimators' early stopping and the threshold c are selected on the validation prompts, which thus serve both, and the steering layer and factor on a subsample of them.

Table 1: Number of prompts in each split. For FaithEval, the counts are for the counterfactual, inconsistent and unanswerable subsets. On WildJailbreak, half of the training and validation prompts are benign prompts that resemble harmful ones, and the test prompts are harmful only.

Dataset	Training	Validation	Test
DeepMath	1,401	199	400
FaithEval	399 / 571 / 924	100 / 143 / 231	93 / 138 / 215
SALAD-Bench (attack-enhanced)	1,938	532	627
WildJailbreak	5,000	1,000	500

Splits. DeepMath's 2,000 problems, sampled uniformly over its difficulty levels, are split 70/10/20 at random within strata defined by difficulty level; the test split is used whole. Each attack-enhanced SALAD-Bench prompt rewrites one of the benchmark's base questions, and we split by base question, so that all rewrites of a question fall in the same split. Its test prompts are a quarter of the 2,530 held-out prompts, drawn at random within each of the benchmark's 16 secondary hazard categories

and rounded down in each. WildJailbreak's test prompts are a quarter of its 2,000 held-out harmful prompts, drawn at random. FaithEval's test prompts are a fifth of each subset's held-out prompts, drawn at random, excluding 7, 5 and 16 of them on the three subsets. For 27 of these, a base-model completion of one of the two models could not be scored token by token, because decoding its tokens into text and re-encoding the text did not reproduce them; for the other, the base model's completion, a repetition loop, has no score.

Success rules.

- **DeepMath.** A completion whose reasoning trace never closes fails. The answer is the last `\boxed{}` expression, or, if there is none, the last line of the form "answer is …" or "answer: …"; a completion with neither fails. The answer and the reference are normalized: whitespace, dollar signs, spacing commands, `\left`, `\right` and degree signs are removed; `\text`, `\mathrm`, `\textbf` and `\mathbf` are unwrapped; `\dfrac` and `\tfrac` become `\frac`; and a trailing period and one pair of outer braces are removed. The answer is correct if the normalized strings are equal; otherwise, if both are numbers, if they agree to a relative tolerance of 10^{-6}; otherwise, if SymPy (Meurer et al., 2017) simplifies their difference to zero or evaluates them to values agreeing to a relative tolerance of 10^{-6}. A parse error, or a SymPy check exceeding 20 seconds, counts as incorrect.
- **FaithEval.** The answer is the text after the reasoning trace, and a completion whose trace never closes fails. On the inconsistent and unanswerable subsets, we follow Siu et al. (2026): the answer is lowercased, stripped of punctuation and of the articles *a*, *an* and *the*, and has its whitespace collapsed, and it succeeds if it contains one of the phrases below. Inconsistent: *conflict*, *multiple answers*, *disagreement*, *inconsistent*, *contradictory*, *contradiction*, *inconsistency*, *two answers*, *2 answers*, *conflicting*. Unanswerable: *unanswerable*, *unknown*, *no answer*, *no information*, *not*, *unclear*; the phrase *not* also matches inside words such as *cannot*. On the counterfactual subset, the chosen option is parsed from the answer by the first of the following patterns that matches: a letter in bold; a letter following "answer is" or "answer:"; a line beginning with a letter followed by a period, a parenthesis or a colon; a line consisting of a letter alone, possibly in parentheses; and otherwise the first standalone option letter on the first nonempty line. The completion succeeds if the parsed letter is the correct option.
- **SALAD-Bench and WildJailbreak.** Llama Guard 4 (Inan et al., 2023; Meta, 2025) classifies the prompt and response as safe, using its own prompt template and its default list of hazard categories.

On SALAD-Bench with Nemotron, 25 of the 627 QFD completions at $\alpha = 0.85$ and 12 of the 627 ARGS completions at $w = 1.5$ contain a run of at least eight consecutive `<unk>` tokens, the placeholder for unknown text; we count these completions as failures. No other method, strength, task or model produced such a run.

E.2 Models and decoding

Table 2 lists the models and the sampling configuration of each, which defines the base policy, and Table 3 the maximum generation length. The methods' maximum was set from the lengths of the base model's completions, generated with a longer maximum: on the mathematical-reasoning and faithfulness tasks, the longest reasoning trace that closed, rounded up to a multiple of 256 tokens; on the harmful-compliance tasks, the longest answer that ended on its own, rounded up to a multiple of 256, for Nemotron, and the larger of 800 and the 95th percentile of the lengths of answers that did not loop for Gemma. For Gemma, these are answers to every prompt of each dataset, test prompts included, of which only the lengths are used; an answer loops if one of the 40-character substrings of its last 1,500 characters, taken every 5 characters, occurs at least three times.

E.3 Q-function estimation

The estimator $\widehat{Q}^{\pi}$ of equation 6 is parametrized by the single $d \times d$ matrix M, initialized at zero and applied to the model's final hidden state, after its last normalization layer; the base model, including its language-model head W_{LM}, is frozen. M has 6.6M parameters for Gemma 4 E4B, 16.8M for

Table 2: Models, their hidden size d and vocabulary size $|\mathcal{V}|$, their sampling configuration, and whether they write a reasoning trace. T is the temperature. Gemma 4 E4B is the instruction-tuned model. Nemotron 3.5 Lightning has about 30B parameters, of which about 3B are active for each token, applies no top-k truncation (–), and writes no reasoning trace on the harmful-compliance tasks.

Model	d	$\lvert\mathcal{V}\rvert$	T / top-k / top-p	Reasoning trace
Gemma 4 E4B	2,560	262,144	1.0 / 64 / 0.95	no
Qwen3.5-9B	4,096	248,320	1.0 / 20 / 0.95	yes
Nemotron 3.5 Lightning	2,688	131,072	1.0 / – / 0.95	yes

Table 3: Maximum generation length in tokens, including the reasoning trace, for the methods, and for the base model where it differs. On FaithEval, the methods' maxima are for the counterfactual, inconsistent and unanswerable subsets. The last column counts the base model's test completions longer than the methods' maximum; on FaithEval with Qwen3.5, they are 7, 0 and 2 on the three subsets.

Task and model	Methods	Base model	Base longer
DeepMath, Nemotron	32,768	same	0
FaithEval, Qwen3.5	10,496 / 12,288 / 14,592	16,384	9
FaithEval, Nemotron	13,568 / 11,520 / 9,472	same	0
SALAD-Bench, Gemma	800	4,096	18
SALAD-Bench, Nemotron	3,328	4,096	1
WildJailbreak, Gemma	1,446	4,096	22
WildJailbreak, Nemotron	3,584	4,096	0

Qwen3.5-9B and 7.2M for Nemotron 3.5 Lightning. We train it on completions sampled from the base model on the training prompts: eight per prompt on the harmful-compliance and mathematical-reasoning tasks, and six per prompt on the faithfulness task, pooled over its three subsets. On WildJailbreak, the training prompts include the benign ones. Each token of a completion is labeled with the completion's success, and the loss is the binary cross-entropy between that label and the estimate $\widehat{Q}^{\pi}(S_t, Y_t)$ of the token Y_t the base model generated. The token losses of each completion are averaged, so that every completion carries the same weight. We train with learning rate 10^{-4}, a linear warm-up over the first 1,000 steps, batches of 64 completions (8 on DeepMath), and at most 30 epochs, stopping once the validation loss has not improved for three epochs.

Table 4 reports how well the estimator separates the base model's successful test completions from its failed ones, scoring each completion by the lowest estimate $\widehat{Q}^{\pi}(S_t, Y_t)$ over its generated tokens, that is, by its highest estimated probability of failure.

Table 4: Area under the ROC curve (AUC) of the estimator on the base model's test completions, with 95% bootstrap confidence intervals over prompts. The failure rate is the fraction of these completions that fail.

Task and model	Completions	Failure rate	AUC
DeepMath, Nemotron	3,200	0.379	0.884 [0.856, 0.906]
FaithEval, Qwen3.5	446	0.211	0.970 [0.955, 0.982]
FaithEval, Nemotron	446	0.186	0.968 [0.952, 0.981]
SALAD-Bench, Gemma	627	0.434	0.889 [0.863, 0.912]
SALAD-Bench, Nemotron	627	0.229	0.910 [0.886, 0.933]
WildJailbreak, Gemma	500	0.062	0.896 [0.855, 0.931]
WildJailbreak, Nemotron	500	0.040	0.932 [0.897, 0.964]

Threshold. The base model generates two completions per validation prompt, eight on DeepMath. For each of the n successful ones, we record the lowest estimate $\widehat{Q}^{\pi}(S_t, Y_t)$ over its generated tokens, and set $c(\alpha)$ to the $\lfloor \alpha(n+1) \rfloor$-th smallest of these values, following the quantile construction

of split conformal prediction (Angelopoulos et al., 2024), but without its statistical guarantee. On WildJailbreak, the validation prompts include the benign ones. Table 5 gives n and $c(\alpha)$.

Table 5: Number n of successful validation completions of the base model, and the threshold $c(\alpha)$ set from them.

Task and model	n	$c(0.05)$	$c(0.25)$	$c(0.45)$	$c(0.65)$	$c(0.85)$
DeepMath, Nemotron	1,070	0.0018	0.0200	0.0729	0.1754	0.3836
FaithEval, Qwen3.5	722	0.0021	0.0189	0.0475	0.1163	0.3241
FaithEval, Nemotron	764	0.0248	0.0772	0.1418	0.2128	0.2940
SALAD-Bench, Gemma	693	0.0103	0.0554	0.1349	0.4624	0.8636
SALAD-Bench, Nemotron	768	0.0342	0.1103	0.2115	0.3848	0.7268
WildJailbreak, Gemma	1,925	0.0716	0.3473	0.5859	0.8079	0.9931
WildJailbreak, Nemotron	1,939	0.1964	0.5561	0.7498	0.9227	0.9794

Candidate sets and fallback. Table 6 reports the size of the base model's candidate set $\mathcal{A}(S_t)$ at the steps of its successful test completions. Because Nemotron applies no top-k truncation, a few of its steps have hundreds of candidates, but the 99th percentile is between 5 and 23 on each of its four task–model pairs. Table 7 reports how often every candidate's estimate falls below c, so that QFD keeps the candidates with the highest estimate (Theorem 4.2). At large α this fallback is frequent: on SALAD-Bench with Gemma at $\alpha = 0.85$, it occurs at 527 of every 1,000 generated tokens.

Table 6: Size of the candidate set $\mathcal{A}(S_t)$, pooled over the steps of the base model's successful test completions.

Task and model	Mean	Median	90th percentile
DeepMath, Nemotron	1.96	1	4
FaithEval, Qwen3.5	2.17	1	5
FaithEval, Nemotron	1.28	1	2
SALAD-Bench, Gemma	5.78	2	16
SALAD-Bench, Nemotron	2.80	1	5
WildJailbreak, Gemma	6.19	2	18
WildJailbreak, Nemotron	2.54	1	5

Table 7: Fallback steps of QFD, at which every candidate's estimate falls below $c(\alpha)$, per 1,000 generated tokens, and in parentheses the fraction of QFD's test completions with at least one.

Task and model	$\alpha = 0.05$	0.25	0.45	0.65	0.85
DeepMath, Nemotron	0.4 (0.23)	11.2 (0.49)	47.2 (0.65)	94.9 (0.75)	144.8 (0.88)
FaithEval, Qwen3.5	48.6 (0.22)	49.7 (0.36)	85.9 (0.51)	88.8 (0.67)	143.0 (0.84)
FaithEval, Nemotron	7.7 (0.23)	25.0 (0.37)	53.3 (0.48)	76.3 (0.54)	114.0 (0.67)
SALAD-Bench, Gemma	0.3 (0.05)	8.8 (0.36)	39.2 (0.58)	209.7 (0.71)	527.3 (0.83)
SALAD-Bench, Nemotron	1.9 (0.11)	8.7 (0.29)	24.9 (0.42)	56.7 (0.53)	153.6 (0.66)
WildJailbreak, Gemma	0.4 (0.04)	3.3 (0.14)	9.3 (0.25)	31.7 (0.38)	417.6 (0.62)
WildJailbreak, Nemotron	0.7 (0.04)	4.9 (0.17)	13.9 (0.24)	58.2 (0.37)	178.3 (0.69)

E.4 Baselines

CD-Q. At each step, CD-Q multiplies the base probability of each candidate $y \in \mathcal{A}(S_t)$ by $\exp\big(\widehat{Q}^{\pi}(S_t, y)/\beta\big)$, with the estimator used by QFD, renormalizes, and samples, for $\beta \in \{5, 2, 1, 0.5, 0.1\}$. The values of β are fixed in advance, not tuned, like those of α.

ARGS. At each step, ARGS takes the ten most probable tokens under the base model's distribution after temperature but before top-k and top-p truncation, and emits the one maximizing $\pi(y \mid S_t) + w\,\widehat{Q}^{\pi}(S_t, y)$, for $w \in \{0.001, 0.5, 1, 1.5, 2\}$, fixed in advance, not tuned, like those of α. The estimate $\widehat{Q}^{\pi}(S_t, y)$ is computed for all ten candidates from the current prefix in one pass of the

estimator. Because the ten candidates are not restricted to $\mathcal{A}(S_t)$, ARGS can emit tokens outside the base model's candidate set.

LoRA. We train on every successful completion of the base model on the training prompts, weighting each by the inverse of the number of successful completions for its prompt, with the loss computed on the completion's tokens and not on the prompt's. The adapter has rank 16 and dropout 0.05, and its update is multiplied by $32/16 = 2$ during training. It is applied to the attention and feed-forward projections, and on Nemotron, whose layers combine Mamba-2, attention and mixture-of-experts blocks, also to the input and output projections of the Mamba-2 blocks, but not to the experts. We train for two epochs, with a linear warm-up over the first 3% of steps and gradient-norm clipping at 1.0. The learning rate, 10^{-5} for Gemma, 5×10^{-6} for Nemotron and 2.5×10^{-6} for Qwen3.5, is chosen for each model on one of its task–model pairs, from at least six values between 10^{-6} and 4×10^{-4}, as the one minimizing the negative log-likelihood per token of the base model's successful completions on the validation prompts. The checkpoint, saved every half epoch, or every quarter on DeepMath, is chosen by the same criterion on each pair. At test time, we multiply the adapter's update by a scale in $\{0.25, 0.5, 0.75, 1, 1.5, 2\}$; scales above one extrapolate beyond the update learned in training.

Steering. Following SteeringSafety (Siu et al., 2026), CAA, DIM and LAT each derive a direction from examples that exhibit the target behavior and examples that do not, in the formats of Rimsky et al. (2024), Arditi et al. (2024) and Zou et al. (2023a) respectively. On FaithEval, these are 377 pairs from the training split, a fifth of each subset's training prompts. On SALAD-Bench and WildJailbreak, they are unpaired sets: the 1,938 training prompts against 3,434 Alpaca instructions (Taori et al., 2023), truncated to 1,938 for LAT, and WildJailbreak's 2,500 harmful against its 2,500 benign training prompts. At generation, the direction, multiplied by the factor, is added to the model's activations at every position after the prompt: at the selected layer for CAA and DIM, and at every layer from the first candidate layer up to the selected one for LAT. The candidate layers are every second layer from about 25% to 80% of the model's depth, and the candidate factors are $\{\pm1, \pm2, \pm3\}$. Among the combinations whose mean KL divergence from the base model's distribution of the first generated token on harmless prompts is at most 0.1, we select the one with the highest success rate on a fifth of the validation prompts. The harmless prompts are a fifth of Alpaca's validation instructions for SALAD-Bench and FaithEval, and a fifth of WildJailbreak's benign validation prompts for WildJailbreak. These choices are SteeringSafety's defaults. At the selected layer, we report the factors $\{\pm1, \pm2, \pm3\}$, and also ±12, outside the range the selection searches, because Siu et al. (2026) found the gains from steering hallucination within this range modest and inconsistent across models. Table 8 gives the selected layers and factors.

Table 8: Layer and factor selected for each steering method, with the range of layers to which LAT is applied.

Task and model	CAA	DIM	LAT
FaithEval, Qwen3.5	16, $+3$	10, $+2$	8–16, $+1$
FaithEval, Nemotron	33, -1	13, $+2$	13–29, -2
SALAD-Bench, Gemma	24, $+2$	10, -1	10–16, $+1$
SALAD-Bench, Nemotron	17, -3	15, -2	13–15, -3
WildJailbreak, Gemma	12, -3	12, -3	10, -1
WildJailbreak, Nemotron	17, -1	35, -1	13–27, -1

E.5 METRICS

Confidence intervals. Every confidence interval is a percentile bootstrap confidence interval for a single measure, computed from 2,000 bootstrap replicates, each of which resamples prompts with replacement and keeps every completion of each drawn prompt. They are 90% confidence intervals, except the 95% confidence intervals of Table 4. The confidence interval of a method's success rate resamples the test prompts and uses the method's completions only. The confidence interval of a change in success rate is paired: each replicate draws the same prompts for the method and for the base model, and recomputes the difference of their success rates on those prompts. The confidence interval of a drop rate resamples the prompts of the base model's successful completions over which

the drop rate is estimated. The confidence interval of a cosine similarity resamples the prompts over which it is averaged. The confidence interval of an AUC resamples the test prompts, with all the base model's completions of each drawn prompt. In figures, the error bars on the two axes are these confidence intervals for the two measures, computed separately; they do not form a joint confidence region.

Drop rate. We estimate the drop rate equation 4 of every method from the base model's successful completions on the test prompts, including every token of each completion, reasoning trace included. For QFD, CD-Q and ARGS, the next-token distribution at each step is a function of the base model's distribution and of $\widehat{Q}^{\pi}$, so their drop rates are computed from a single pass of the estimator over these completions. For QFD, the estimate is the fraction of these completions in which the filter removes at least one token. For LoRA and steering, we recompute the modified model's next-token distribution at every prefix of each of these completions, and compute from it the removed fraction of Definition 3.1 at every step.

Similarity. We embed each completion, reasoning trace included, with e5-mistral-7b-instruct (Wang et al., 2024), prefixed with the instruction the model uses for semantic similarity, and take the final token's hidden state as the embedding. Completions longer than 2,048 tokens are split into windows of 2,048 tokens, whose embeddings are averaged with weights proportional to their numbers of tokens. For each successful completion of a method, we average its cosine similarity to the base model's successful completions on the same prompt, and report the mean over prompts, skipping prompts on which the method or the base model has no successful completion.

E.6 Additional results

Largest improvements. Table 9 lists, for each task–model pair, the largest significant improvement in success rate of QFD and of the best baseline, with the drop rate and cosine similarity at which each occurs. The two improvements have overlapping confidence intervals on DeepMath, on both WildJailbreak pairs and on SALAD-Bench with Gemma. On SALAD-Bench with Gemma, ARGS has the larger point estimate; it always emits its highest-scoring candidate, and so returns a single deterministic completion per prompt. On FaithEval with Qwen3.5, CAA at $+12$ has the larger improvement, at a drop rate of 1.000 and a lower cosine similarity than QFD's. On the five pairs other than DeepMath where both improve it significantly, QFD's cosine similarity exceeds the best baseline's with non-overlapping confidence intervals.

Improvements that do not reflect better task performance. Two of the best baselines' improvements in Table 9, both from steering at a factor of ± 12, come from outputs that satisfy the success rule without performing the task better. On SALAD-Bench with Nemotron, LAT at -12 produces long, often off-topic text, such as an unrelated narrative in reply to a jailbreak prompt, which Llama Guard judges safe because it is not harmful. Its completions average $1{,}054$ tokens, against 263 for the base model; apart from LAT at $+12$, whose success rate falls, no other method on this pair averages more than 483. At least a tenth of its completions reach the maximum length of $3{,}328$ tokens, and its cosine similarity to the base model, 0.733, is the lowest of any method on this pair. On FaithEval with Nemotron, CAA at $+12$ raises the success rate only on the unanswerable subset, where the rule scores answers such as "unknown" as correct, while its accuracy on the counterfactual and inconsistent subsets falls slightly (Table 10). DIM at $+2$ and $+3$, the two other significant improvements on this pair, also lower counterfactual accuracy. With 45 points on this pair, each with a 90% confidence interval, about two are expected to lie entirely above zero by chance alone. By contrast, CAA at $+12$ on FaithEval with Qwen3.5 raises the success rate on the counterfactual and unanswerable subsets and leaves the inconsistent subset unchanged, so its improvement, reached at a drop rate of 1.000, is not marked.

All strengths. Within the range of drop rates QFD spans, the only significant improvements by baselines are smaller than QFD's: steering on SALAD-Bench with Gemma, $+0.029$ to $+0.051$ at drop rates of 0.84–0.86, and CD-Q with $\beta = 0.1$ on WildJailbreak with Gemma, $+0.016$ at 0.53. On the three pairs with reasoning traces, no baseline reaches a drop rate below 0.49 at any strength. At 24 of QFD's 35 points, no baseline has both a success rate and a cosine similarity at least as high

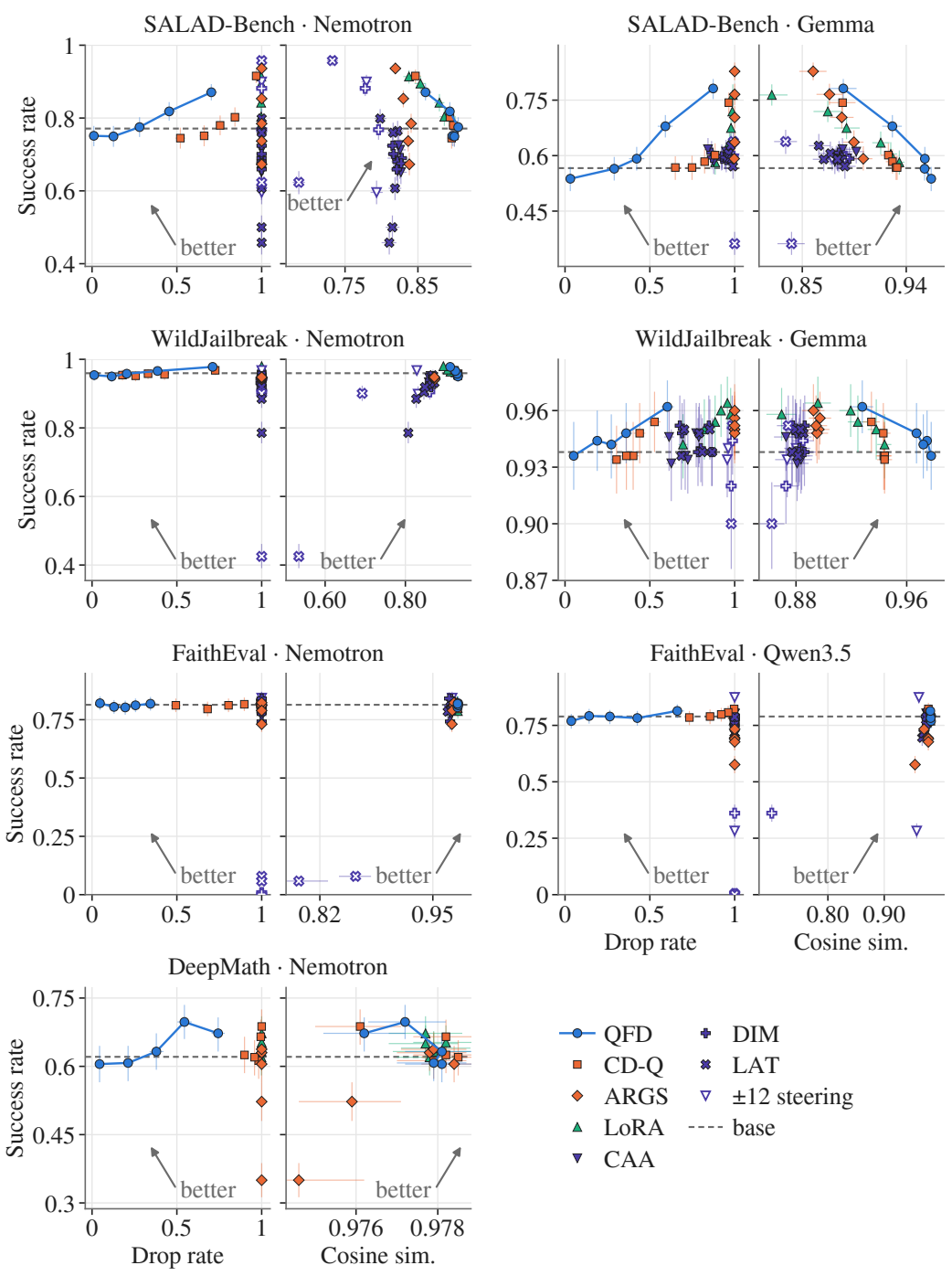

Figure 2: Figure 1 with every point, including the 20 below the lower Tukey fence of their pair's success rates. Five steering points at factor ±12 on FaithEval, with success rates below 0.01, have successful completions on fewer than ten prompts, and so appear only in the drop-rate panels. All other details are as in Figure 1.

as QFD's; at the other 11, every baseline that does has confidence intervals overlapping QFD's for both measures.

Table 9: The largest significant change in success rate over the base model for each task–model pair, for QFD and for the best baseline, with the drop rate and cosine similarity at which it occurs. Each change in success rate is given with its paired 90% bootstrap confidence interval over prompts, each drop rate with its 90% bootstrap confidence interval over the prompts of the base model's successful completions, and each cosine similarity with its 90% bootstrap confidence interval over the prompts it averages. †An improvement that does not reflect better task performance (Appendix E.6).

Task, model	Method	Δ success	Drop rate	Cosine similarity
DeepMath Nemotron	QFD ($\alpha = 0.65$)	$+0.077$ $[+0.051, +0.101]$	0.544 $[0.503, 0.587]$	0.977 $[0.976, 0.978]$
	CD-Q ($\beta = 0.1$)	$+0.067$ $[+0.044, +0.090]$	1.000 $[1.000, 1.000]$	0.976 $[0.975, 0.977]$
FaithEval Qwen3.5	QFD ($\alpha = 0.85$)	$+0.025$ $[+0.004, +0.045]$	0.662 $[0.622, 0.702]$	0.982 $[0.981, 0.983]$
	CAA ($+12$)	$+0.085$ $[+0.058, +0.112]$	1.000 $[1.000, 1.000]$	0.962 $[0.960, 0.963]$
FaithEval Nemotron	QFD	none	–	–
	CAA ($+12$)†	$+0.029$ $[+0.007, +0.049]$	1.000 $[0.999, 1.000]$	0.972 $[0.971, 0.974]$
SALAD-Bench Gemma	QFD ($\alpha = 0.85$)	$+0.215$ $[+0.187, +0.247]$	0.873 $[0.842, 0.901]$	0.885 $[0.873, 0.897]$
	ARGS ($w = 2$)	$+0.262$ $[+0.228, +0.295]$	1.000 $[1.000, 1.000]$	0.859 $[0.847, 0.871]$
SALAD-Bench Nemotron	QFD ($\alpha = 0.85$)	$+0.101$ $[+0.067, +0.133]$	0.703 $[0.669, 0.737]$	0.860 $[0.852, 0.869]$
	LAT (-12)†	$+0.187$ $[+0.158, +0.217]$	0.999 $[0.999, 1.000]$	0.733 $[0.725, 0.740]$
WildJailbreak Gemma	QFD ($\alpha = 0.85$)	$+0.024$ $[+0.010, +0.038]$	0.603 $[0.567, 0.642]$	0.928 $[0.920, 0.936]$
	LoRA (scale 1.5)	$+0.026$ $[+0.010, +0.042]$	0.962 $[0.958, 0.967]$	0.896 $[0.886, 0.905]$
WildJailbreak Nemotron	QFD ($\alpha = 0.85$)	$+0.018$ $[+0.006, +0.030]$	0.711 $[0.677, 0.743]$	0.912 $[0.907, 0.916]$
	LoRA (scale 2)	$+0.020$ $[+0.008, +0.032]$	0.998 $[0.997, 1.000]$	0.895 $[0.890, 0.899]$

Table 10: Success rate on each FaithEval subset, for the base model, QFD at $\alpha = 0.85$, and the steering points with significant gains.

Model	Method	Counterfactual	Inconsistent	Unanswerable	Pooled
Qwen3.5	Base	0.634	0.964	0.744	0.789
	QFD ($\alpha = 0.85$)	0.656	0.957	0.791	0.814
	CAA ($+12$)	0.720	0.964	0.884	0.874
Nemotron	Base	0.602	0.971	0.805	0.814
	QFD ($\alpha = 0.85$)	0.624	0.949	0.819	0.818
	CAA ($+12$)	0.581	0.964	0.879	0.843
	DIM ($+2$)	0.570	0.993	0.856	0.839
	DIM ($+3$)	0.570	0.964	0.879	0.841

Drop rate against α. At the three points where QFD's drop rate exceeds α, α lies inside the drop rate's confidence interval at two, on WildJailbreak with Gemma at $\alpha = 0.05$ and on SALAD-Bench with Gemma at $\alpha = 0.85$; at the third, on SALAD-Bench with Gemma at $\alpha = 0.25$, the drop rate is 0.290 with confidence interval $[0.251, 0.332]$. On WildJailbreak, the validation prompts on which c is selected include benign prompts, while the drop rate is measured on harmful prompts only.

E.7 Comparison with value-filtered decoding

Because an efficient implementation of value-filtered decoding (VFD; Einbinder et al., 2026) depends on the model architecture (Section 6), we compare QFD with it on a separate testbed that follows the setup of Einbinder et al. (2026). The models are Mistral-7B-Instruct-v0.3 (Jiang et al., 2023) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024), the test prompts are 480 from BeaverTails (Ji et al., 2023) and 496 from PKU-SafeRLHF (Ji et al., 2025), and all methods sample with temperature 0.6, top-p 0.95 and top-k 20. No drop rate or cosine similarity was measured on this testbed.

Judges. We report the rates of unsafe and of helpful responses, both judged by Llama-3.1-8B-Instruct with the two system prompts of Einbinder et al. (2026), decoding greedily for at most 64 tokens and parsing the first line of the reply; a reply that cannot be parsed counts as unsafe or unhelpful. A response is unsafe if it contains harmful, illegal, violent, self-harm, sexual, demeaning or harassing content, or actionable instructions that enable harm; slurs, explicit or masked, are always unsafe. Helpfulness is judged as compliance: a response is helpful if it directly satisfies the user's specific request, including a harmful one, while refusals, safe substitutions and explanations of how to avoid what was asked count as unhelpful. As a second judge, we use Qwen3-14B (Yang et al., 2025) with the same prompts and thinking disabled. For the Llama-3.1-8B-Instruct policy, the primary judge is the same model as the policy.

Estimators. For each model and dataset, both estimators are trained on completions of the base model to the first 5,000 prompts of the dataset's training split, which are 3,717 unique prompts for BeaverTails and 4,698 for PKU-SafeRLHF, with two completions per prompt, sampled with the sampling parameters above and labeled safe or unsafe by the primary judge; for Llama, 597 BeaverTails and 111 PKU-SafeRLHF prompts have four. The prompts are split into 90% for training and 10% for calibration, and each estimator's thresholds $c(\alpha)$, for $\alpha \in \{0.05, 0.45, 0.85\}$, are computed independently from the safe completions of the calibration prompts; the thresholds differ across models, so α is not comparable across them. QFD estimates the Q-values of all tokens with an MLP applied to the model's final hidden state, in place of equation 6, and filters within the candidate set as in the main experiments. VFD estimates the value of the prefix ending in each candidate, and at each step applies rejection sampling to $K = 8$ draws from the base model. Both are trained with a focal loss and the temporal smoothness penalty of Einbinder et al. (2026) with weight 0.1, learning rate 10^{-4} with warm-up, and at most 10 epochs with early stopping after three without improvement, in batches of 128 for VFD and 64 for QFD.

Results. Table 11 gives the rates. On Mistral, both methods roughly halve the unsafe rate at $\alpha = 0.85$: QFD by 54% on BeaverTails and 56% on PKU-SafeRLHF, and VFD by 59% and 51%, at a cost of 1.7 to 4.6 points of helpfulness. The two methods are on par: VFD has the lower unsafe rate and the higher helpfulness on BeaverTails, and QFD on PKU-SafeRLHF, but these differences are within about one standard error. Table 12 compares each method's unsafe responses with the base model's on the same prompts by an exact McNemar test. Under both judges, every comparison at $\alpha = 0.05$ is not significant at level 0.05, and every Mistral comparison at $\alpha = 0.85$ is significant, with p-values from $2.5 \cdot 10^{-5}$ to $9.7 \cdot 10^{-12}$ under the primary judge and from $7.3 \cdot 10^{-5}$ to $1.7 \cdot 10^{-14}$ under Qwen3-14B. On Llama, whose base unsafe rate is only 2 to 4%, neither method changes the unsafe rate by more than 0.021. Its two comparisons significant under the primary judge, both QFD on BeaverTails, with $p = 0.035$ at $\alpha = 0.45$ and $p = 0.021$ at $\alpha = 0.85$, have $p = 1.00$ and $p = 0.238$ under Qwen3-14B, and every other Llama comparison has $p \geq 0.146$ under the primary judge. Across the 24 comparisons with the base model, the two judges disagree on significance in three, all for QFD: these two, and Mistral on BeaverTails at $\alpha = 0.45$, with $p = 4.6 \cdot 10^{-3}$ under the primary judge and $p = 0.22$ under Qwen3-14B.

E.8 Inference overhead

We measure decoding time on one NVIDIA GH200 GPU, with vLLM 0.19.1 (Kwon et al., 2023) and PyTorch 2.10.0. For each task–model pair and each α, we generate from the 128 test prompts with the longest QFD completions, alternating the base model and QFD twice on the same GPU, with the same prompts, random seed, batch size and sampling parameters, and report the ratio of their mean wall times.

Table 11: Rates of unsafe and of helpful responses, with one binomial standard error, on the testbed of Appendix E.7. The value of α is not comparable across models, since the thresholds it yields differ.

		BeaverTails		PKU-SafeRLHF	
Method	α	Unsafe ↓	Helpful ↑	Unsafe ↓	Helpful ↑
Mistral-7B-Instruct-v0.3					
Base	–	0.113 ± 0.014	0.521 ± 0.023	0.208 ± 0.018	0.506 ± 0.022
QFD	0.05	0.110 ± 0.014	0.521 ± 0.023	0.216 ± 0.019	0.504 ± 0.022
QFD	0.45	0.077 ± 0.012	0.506 ± 0.023	0.113 ± 0.014	0.504 ± 0.022
QFD	0.85	0.052 ± 0.010	0.475 ± 0.023	0.091 ± 0.013	0.488 ± 0.022
VFD	0.05	0.106 ± 0.014	0.529 ± 0.023	0.210 ± 0.018	0.500 ± 0.023
VFD	0.45	0.052 ± 0.010	0.513 ± 0.023	0.129 ± 0.015	0.490 ± 0.022
VFD	0.85	0.046 ± 0.010	0.504 ± 0.023	0.101 ± 0.014	0.474 ± 0.022
Llama-3.1-8B-Instruct					
Base	–	0.038 ± 0.009	0.375 ± 0.022	0.020 ± 0.006	0.270 ± 0.020
QFD	0.05	0.044 ± 0.009	0.375 ± 0.022	0.018 ± 0.006	0.282 ± 0.020
QFD	0.45	0.019 ± 0.006	0.356 ± 0.022	0.028 ± 0.007	0.286 ± 0.020
QFD	0.85	0.017 ± 0.006	0.327 ± 0.021	0.022 ± 0.007	0.284 ± 0.020
VFD	0.05	0.040 ± 0.009	0.373 ± 0.022	0.014 ± 0.005	0.284 ± 0.020
VFD	0.45	0.025 ± 0.007	0.365 ± 0.022	0.018 ± 0.006	0.274 ± 0.020
VFD	0.85	0.027 ± 0.007	0.344 ± 0.022	0.016 ± 0.006	0.272 ± 0.020

Table 12: Exact McNemar tests of each method's unsafe responses against the base model's on the same prompts, on Mistral-7B-Instruct-v0.3, under the primary judge: prompts whose response becomes safe and becomes unsafe relative to the base model, and the p-value.

Dataset	Method	$\alpha = 0.05$	$\alpha = 0.45$	$\alpha = 0.85$
BeaverTails	QFD	9 / 8, $p = 1.00$	25 / 8, $p = 4.6 \cdot 10^{-3}$	38 / 9, $p = 2.5 \cdot 10^{-5}$
	VFD	21 / 18, $p = 0.75$	37 / 8, $p = 1.5 \cdot 10^{-5}$	40 / 8, $p = 3.3 \cdot 10^{-6}$
PKU-SafeRLHF	QFD	13 / 17, $p = 0.59$	60 / 13, $p = 2.3 \cdot 10^{-8}$	68 / 10, $p = 9.7 \cdot 10^{-12}$
	VFD	29 / 30, $p = 1.00$	56 / 17, $p = 5.3 \cdot 10^{-6}$	61 / 8, $p = 3.2 \cdot 10^{-11}$

Across the 35 settings, QFD's wall time is 1.2% above the base model's on average, with a median of 1.8%, and ranges from 4.6% below to 9.1% above it. Generation is limited to 512 tokens in batches of 128 on the harmful-compliance tasks, and to 2,048 tokens, reasoning trace included, in batches of 32 on the others. The estimator adds a fixed cost at each step, while the cost of attention grows with length, so these limits, shorter than those of our experiments, if anything overstate the relative overhead. The timed implementation scores only the candidates in $\mathcal{A}(S_t)$, and generates the same tokens as the implementation behind our results on every prompt except one of 128 on WildJailbreak with Nemotron, where numerical differences in computing the candidate set under Nemotron's sampling without top-k truncation change one completion.

Table 13 gives the ratios. On the harmful-compliance tasks, the wall time is the relevant measure, since a batch of limited length runs for as many steps as its longest completion; there, the ratio per generated token overstates the cost when QFD ends completions earlier, as at $\alpha = 0.85$, where it generates 7–8% fewer tokens in the same time. On FaithEval with Nemotron, the wall-time ratio reaches 1.091 at $\alpha = 0.45$, where QFD's reasoning traces are 7% longer than the base model's; per generated token, QFD is 1.6–4.3% slower, and repeated measurements on this pair vary by 1–3%.

The only additional memory is the matrix M: 25 MiB for Gemma, 28 MiB for Nemotron and 64 MiB for Qwen3.5, under 0.1% of the GPU's memory. With vLLM reserving 90% of the GPU's memory, the key–value cache holds 280,801–280,807 blocks for Gemma with QFD against 280,806 without, 2,020 for Nemotron in both cases, and 4,115 against 4,108 for Qwen3.5, and the peak memory allocated during generation changes by -0.10 to $+0.15$ GiB out of about 85 GiB.

Table 13: Ratio of QFD's decoding time to the base model's, as the range over $\alpha \in \{0.05, 0.25, 0.45, 0.65, 0.85\}$, in wall time and per generated token.

Task and model	Wall time	Per generated token
DeepMath, Nemotron	0.988–1.020	0.989–1.022
FaithEval, Qwen3.5	0.954–0.986	0.975–0.997
FaithEval, Nemotron	0.994–1.091	1.016–1.043
SALAD-Bench, Gemma	0.999–1.020	1.001–1.020
SALAD-Bench, Nemotron	0.992–1.029	1.017–1.080
WildJailbreak, Gemma	0.987–1.028	0.999–1.067
WildJailbreak, Nemotron	1.000–1.032	1.018–1.074

E.9 Compute and software

The experiments ran on NVIDIA A100, H100, H200 and GH200 GPUs. Table 14 gives the approximate GPU-hours of each stage.

Table 14: Approximate GPU-hours of each stage.

Stage	GPU-hours
Sampling from the base models	60–120
Training the Q-function estimators	≈10
Generating with QFD, CD-Q and ARGS	40–80
Training the LoRA adapters, including learning-rate selection	60–100
Generating with the LoRA adapters	10–20
Selecting the steering layer and factor, and generating with steering	up to 700–1,000
Computing the drop rate	70–80
Embedding completions for cosine similarity	≈4

We use vLLM 0.19.1, PyTorch 2.10.0 with CUDA 12.8, transformers 5.14.1 and SymPy 1.14.0, with random seed 15 for generation and 0 for training the estimators and LoRA adapters and for the bootstrap.