Improving LLMs via Validator-to-Generator Alignment

Juan Diego Rodriguez[♠] **Jocelyn Zhang**[♠] **Katrin Erk**[♢] **Greg Durrett**[♣]

[♠] Department of Computer Science, The University of Texas at Austin
[♢] Departments of Linguistics and Computer Science, University of Massachusetts Amherst
[♣] Department of Computer Science & Center for Data Science, New York University
{juand-r, jocelynzhang}@utexas.edu kerk@umass.edu gdurrett@nyu.edu

Abstract

Large language models are inconsistent: varying prompts or including unrelated information can lead to unexpected changes in model outputs. The generator-validator (G-V) gap is one manifestation of this phenomenon, where LLMs generate responses that they then deem as invalid if re-queried to validate them. In this work, we introduce a new formulation of G-V consistency that involves a principled correction for utterance frequency. Specifically, generators often assign low likelihood to valid strings simply because those strings are a priori unlikely, which makes naive notions of G-V consistency unworkable. We show that under a natural model of rational agents answering questions with multiple answers, consistency of the validator with a frequency-corrected generator score emerges naturally. Our method, *Frequency-corrected Learning of Ordered Rank Alignment* (FLORA), is a training objective implementing frequency-corrected G-V consistency for real-world LLMs. Our experimental results show that training with FLORA substantially improves both G-V consistency and generator performance over prior methods, with gains of up to +27pp in Pearson correlation on IFEval and HumanEval, while preserving validator quality across all evaluated tasks.[1]

1 Introduction

Large language models (LLMs) can be used in two complementary modes: as *generators* that produce candidate responses, and as *validators* that assess response correctness or felicity. These two modes are crucial for usages of LLMs such as self-refinement (Madaan et al., 2023) and backtracking during chain-of-thought (Yao et al., 2023). However, these two modes exhibit divergent behavior, reflecting an underlying inconsistency: even frontier LLMs may generate responses with high probability but judge them to be incorrect, or vice-versa. Accordingly, past work has examined training LMs to be explicitly generator-validator (G-V) consistent in the hope that this will also make them more accurate (West et al., 2024; Li et al., 2024; Rodriguez et al., 2025). But formalizing this turns out to be surprisingly difficult. Past approaches close the gap on G-V correlation (Rodriguez et al., 2025), but these approaches risk contributing to pathological behaviors like suppression of correct responses or contributing to further inconsistency through contradictory training signals.

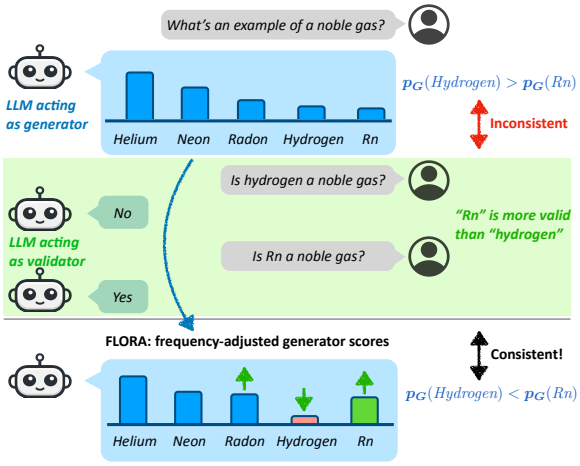

Figure 1: An LLM may generate outputs inconsistent with how it validates them, since low probability options may simply be unlikely to say. FLORA provides a principled correction for generator and allows for training of G-V consistent models.

This work aims to improve G-V consistency through two contributions. First, we axiomatize G-V consistency based on a model of a rational probabilistic agent responding to prompts. Although LLMs are *not* rational agents, this model allows us to examine what the relationship between a generator and a validator should be in cases where the generator may prefer some possible responses over others, but a validator finds them all to be likely. We derive a theoretical relationship between generator and validator probabilities, which implies adjusting generator scores by subtracting off *incorrect probabilities*, or how likely a response is to be sampled when an *incorrect* response is requested. Since this value reflects the base likelihood of the response, we call the final quantity the *frequency-corrected generator score*.

Second, we use this relationship in a training objec-

[1]Code and data available at https://github.com/juand-r/flora

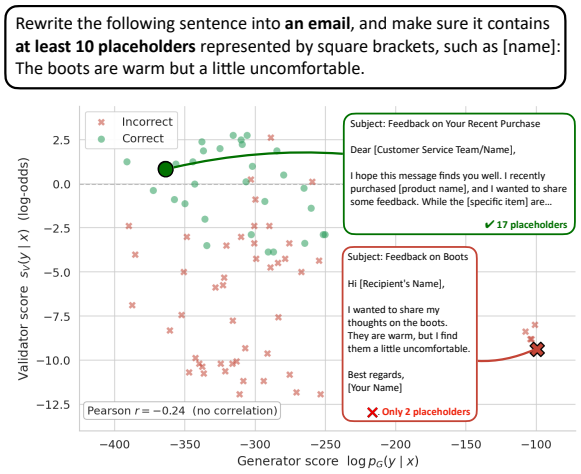

Figure 2: Generator-validator gap on a long-form generation task from IFEval. A short email with few placeholders (red) does not follow the instruction and fails validation but has high generator score. Another email (green) has lower generator score but correctly follows the instruction and is validated as correct.

tive called **Frequency-corrected Learning of Ordered Rank Alignment** (FLORA). This objective encourages rank-alignment of validator scores with frequency-corrected generator scores, representing an approximation of our theoretical G-V consistency for rational agents. We fine-tune LLMs using this objective function in combination with standard losses to ensure generator and validator correctness.

Our experiments focus on tasks where validators outperform generators. Our goal is to observe that the generator improves without degradation of the validator. We particularly focus on *generator AUROC*, or ensuring that the frequency-corrected generator distribution correctly ranks the set of positive responses above the set of negative responses, even down into the tail of the distribution, unlike past work which focuses on the head (Li et al., 2024). This is a strong check of consistency and of generator correctness.

We evaluate on three tasks: instruction-following, coding, and eliciting taxonomic knowledge. Two of these tasks are long-form (several sentences, for IFEval, or Python code), unlike previous work which focused on short answers of one or a couple words (Rodriguez et al., 2025; Li et al., 2024) Across these tasks and three LLMs, we find that our method closes the G-V gap more strongly than previous work, and leads to improved discriminability of correct and incorrect completions. Concretely, FLORA improves generator AUROC by up to +7.3pp and generator-validator correlation by up to +27pp over the strongest prior alignment method, while preserving or improving validator quality.

2 Background and Motivation

The extent to which LLMs have consistent internal processing is an important scientific question (Pres et al., 2026), and the mismatch between generation and validation is an important kind of inconsistency to measure and repair. Much of what a model "knows" never shows up in its sampled outputs, because it lives in discriminative judgments rather than in fluent continuations (Gekhman et al., 2025); this is problematic for evaluations targeting model knowledge (Wang et al., 2024; Biderman et al., 2024). Better aligning generators with an LLM's own validation capability is also important for reward modeling (Yuan et al., 2024) and reranking applications such as mathematical reasoning (Cobbe et al., 2021; Lightman et al., 2024), code generation (Chen et al., 2021), and factual question answering.

Rodriguez et al. (2025) assert that the log-odds of the generator and validator should correlate. However, there are cases where this relationship cannot hold, arising from two main sources: (a) the role of response frequency and (b) aleatoric uncertainty (multiple correct responses).

Motivating Example A well-known issue with using LLMs to generate responses is surface form ambiguity (Holtzman et al., 2021). Given the question *"What do you call it when blood flow to the heart is suddenly blocked?"*, *heart attack* and *myocardial infarction* are both correct answers. A good validator should score them both highly, yet it would be odd to expect a generator to score both highly, since *heart attack* is the more common expression.

This becomes more complex when questions have multiple possible correct answers beyond paraphrases. Consider a question that admits several valid answers, e.g., *"What's an example of a noble gas?"* (Figure 1). There are 7 possible correct answers to this question, plus potential different surface forms of elements (e.g., *"Helium"* vs. *"He"*). It is unlikely that a generator will actually generate *Helium* and *Oganesson* with equal probability when given this question, despite both being correct. Generator scores conflate the correctness of an utterance in response to a prompt with the *frequency* of that utterance.

There is no single "perfect" generator in this setting. One view is that a model should tend towards a uniform distribution over possible options (Zhang et al., 2024), while another view is that the model's distribution over valid answers should match the estimated corpus-frequency distribution (Tomov et al., 2025). A perfect validator, in contrast, should certainly say "Yes" with probability near 1 to each correct option and "No" to incorrect options.

Figure 2 shows this effect in practice on a long-form generation task from IFEval. The validator is imperfect, but most correct responses (green) outscore most incorrect responses (red) on the validator log-odds (y-axis). However, there are incorrect responses that have much higher generator score than correct responses, partially due to being short and generic and sticking close to the prompt.

This example shows that generator and validator

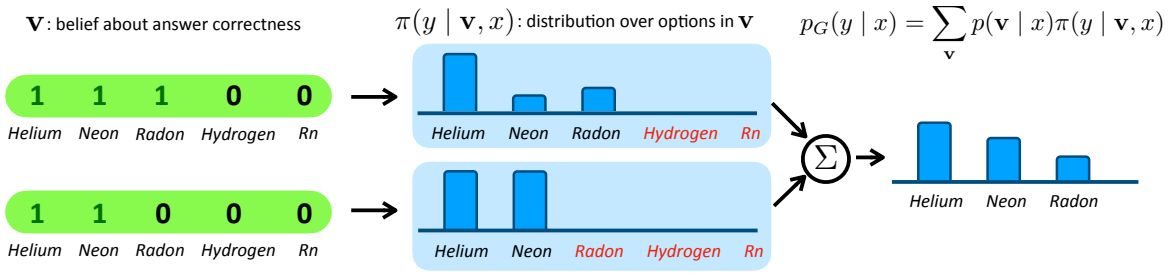

Figure 3: Model of agent response generation. We represent a generator p_G as a mixture distribution over latent validity vectors $\mathbf{v}$, which indicate the subset of responses a model believes to be the correct response. π places a distribution over y conditioned on each $\mathbf{v}$.

scores should not generally be expected to match, or even to correlate linearly, without controlling for competition and frequency effects. Next we derive a relationship between them which accounts for both frequency effects and aleatoric uncertainty.

3 Problem Formulation

Problem Setup We assume a prompt x and a set of possible responses $\mathcal{Y}$.[2] For each $y \in \mathcal{Y}$, we assume that there is a validity label $v(x, y) \in \{0, 1\}$. We assume that validity is binary and unambiguous. We consider an agent, which is an LLM for the purposes of this work, operating in two modes: a generator $p_G(y \mid x)$, and a validator $p_V(v{=}1 \mid x, y)$. For an LLM, p_V and p_G are implemented with different prompts.

Properties of Consistency We would like to choose p_V and p_G to be consistent, under the intuition that improving consistency will also improve accuracy. We can straightforwardly understand p_V as the LM's current "belief" that y is a correct response to x. However, since there are in general many correct responses, p_G additionally has to model competition between these to act as a generator.

We define $\mathbf{v} = \{0, 1\}^{|\mathcal{Y}|}$ as a vector of validities associated with each possible response string, assuming that every possible response is either correct or incorrect, even though the agent may be uncertain as to which. Let $\mathbf{v}_y \in \{0, 1\}$ denote the validity assigned to a particular string y. In the context of Figure 1, a correct $\mathbf{v}$ should assign 1 to the 7 noble gases and all ways of writing them down and 0 to all other strings. Figure 3 shows possible $\mathbf{v}$ values in green.

Assume that for a prompt x, we have a known and fixed $\mathbf{v}$. We define an agent to be *consistent* with its beliefs $\mathbf{v}$ if (1) its validator returns exactly the responses in $\mathbf{v}$; (2) its generator only places mass on valid responses: $p_G(y \mid x) > 0$ iff $\mathbf{v}_y = 1$. We call this a **support constraint** motivated by the maxim of quality (Grice, 1975): conditional on its latent belief state $\mathbf{v}$, the agent never generates a response it believes to be incorrect. However, some options (e.g., xenon and krypton) may be assigned very low probability.

In practice, agents have uncertainty over $\mathbf{v}$. We model this as a distribution $p(\mathbf{v} \mid x) > 0$ reflecting the agent's belief over possible response sets given x. Like in real LLMs, we assume that every $\mathbf{v}$ has nonzero (but possibly tiny) probability mass. (Note that this distribution is a theoretical construct and cannot be materialized in practice.) We can now define *consistency* with respect to this distribution $p(\mathbf{v} \mid x)$. A validator is consistent if it satisfies the following relationship:

$$p_V(y \mid x) := p(v_y = 1 \mid x) = \sum_{\mathbf{v}: v_y = 1} p(\mathbf{v} \mid x) \quad (1)$$

We represent generation through the use of a distribution $\pi(y \mid \mathbf{v}, x)$ that conditions on $\mathbf{v}$. π, although intractable to fully represent in practice, effectively describes how mass should be distributed among the correct options of $\mathbf{v}$. To obey the support constraint, π only assigns nonzero probability to items i where $\mathbf{v}_i = 1$. But within these items, π governs which surface form to use for a given meaning, splitting probability over surface form realizations (Zhang et al., 2024; Holtzman et al., 2021; Zhang et al., 2025), and which response to express when multiple are deemed correct. We define the generator as a mixture model over the $\mathbf{v}$:

$$\begin{aligned} p_G(y \mid x) &:= \sum_{\mathbf{v}} p(\mathbf{v} \mid x)\, \pi(y \mid \mathbf{v}, x) \\ &= \sum_{\mathbf{v}: v_y = 1} p(\mathbf{v} \mid x)\, \pi(y \mid \mathbf{v}, x) \end{aligned} \quad (2)$$

This process is depicted in Figure 3.

3.1 Deriving a G-V relationship

We can now relate the generator and the validator we have defined so far.

We first define one more quantity $\pi'(y \mid \mathbf{v}, x)$. This is a policy for selecting a response when asked for an *incorrect* one, with the related constraint that it never generates an answer it believes to be correct, i.e., $\pi'(y \mid$

[2] Our analysis will require $\mathcal{Y}$ to be finite in size, but it can be very large: Σ^N for a token vocabulary Σ and some N. We do not assume that $\mathcal{Y}$ is tractable.

$\mathbf{v}, x) = 0$ when $v_a = 1$. From that, we can define

$$\begin{aligned} p_{G'}(y \mid x) &:= \sum_{\mathbf{v}} P(\mathbf{v} \mid x)\, \pi'(y \mid \mathbf{v}, x) \\ &= \sum_{\mathbf{v}: v_y = 0} p(\mathbf{v} \mid x)\, \pi'(y \mid \mathbf{v}, x) \end{aligned} \tag{3}$$

Theorem 1 (Main)**.** *Assume a problem x, solution space $\mathcal{Y}$, and generator p_G and validator p_V as defined previously. Further assume that $0 < p_V(y \mid x) < 1$ for all $y \in \mathcal{Y}$. Then*

$$\frac{p_V(y \mid x)}{1 - p_V(y \mid x)} = \frac{p_G(y \mid x)}{p_{G'}(y \mid x)} \cdot r(y, x), \tag{4}$$

where

$$r(y, x) := \frac{\mathbb{E}[\pi'(y \mid \mathbf{v}, x) \mid v_y = 0, x]}{\mathbb{E}[\pi(y \mid \mathbf{v}, x) \mid v_y = 1, x]}.$$

Proof Sketch We give intuition for the proof and a full proof in Appendix A.

The conditional expectation of the policy for picking a correct response from among the set of correct answers given that a is valid is:

$$\begin{aligned} &\mathbb{E}\big[\pi(y \mid \mathbf{v}, x) \mid v_y = 1, x\big] \\ &\quad = \frac{\sum_{\mathbf{v}: v_y = 1} P(\mathbf{v} \mid x)\, \pi(y \mid \mathbf{v}, x)}{P(v_y = 1 \mid x)} \\ &\quad = \frac{p_G(y \mid x)}{p_V(y \mid x)} \end{aligned} \tag{5}$$

Similarly,

$$\begin{aligned} &\mathbb{E}\big[\pi'(y \mid \mathbf{v}, x) \mid v_y = 0, x\big] \\ &\quad = \frac{\sum_{\mathbf{v}: v_y = 0} p(\mathbf{v} \mid x)\, \pi'(y \mid \mathbf{v}, x)}{P(v_y = 0 \mid x)} \\ &\quad = \frac{p_{G'}(y \mid x)}{1 - p_V(y \mid x)} \end{aligned} \tag{6}$$

Since $0 < p_V(y \mid x) < 1$ the main result follows from Eqn 5 and 6.

Intuition First, we note that $0 < p_V(y \mid x) < 1$ is a mild condition in practice. We might reasonably expect that an agent backed by a Transformer produces a distribution $p(\mathbf{v} \mid x)$ with support everywhere due to the nature of softmax; no set of answers is ruled out structurally. Our needed condition follows from this.

In the limit of zero epistemic uncertainty, these probabilities become very small and p places all probability mass on one configuration $\mathbf{v}^\star \in \{0, 1\}^{|A|}$. In this case, the validator probability $p_V(y \mid x)$ will approach 0 or 1, and the generator probabilities become the policies at configuration $\mathbf{v}^\star$: $p_G(y \mid x) = \pi(y \mid \mathbf{v}^\star, x)$ and $p_{G'}(y \mid x) = \pi'(y \mid \mathbf{v}^\star, x)$.

Finally, the value of $r(y, x)$ is very important to reasoning about the generator-validator relationship. The numerator of r is the expected value of the probability of a response being generated as a *incorrect* response conditioned on it being marked as incorrect in $\mathbf{v}$. This involves marginalizing over all possible configurations of $\mathbf{v}$ where $v_y = 0$. We can think of this as saying: in aggregate, what fraction of the π' mass is assigned to y when it's a valid option? Intuitively, this is related to how frequent we are to say y: a string that's simply more frequent will be uttered with higher probability in a larger number of contexts. The denominator of r is similar, but for the case of a correct response.

3.2 Mapping Rational Agent Behavior to LLM Behavior

LLMs are not rational agents of the sort in the previous section. However, we can nevertheless seek to impose the regularity we derived in Equation 4 to their behavior if we can approximate p_V, p_G, and $p_{G'}$. To do so, $p_V(y \mid x)$, $p_G(y \mid x)$ and $p_{G'}(y \mid x)$ can be elicited from an LLM as follows. Given template function $T_V : x, y \to \text{prompt}_V(x, y)$ mapping questions and answers to prompts, and template functions $T_G : x \to \text{prompt}_G(x)$ and $T'_G : x \to \text{prompt}'_G(x)$, asking for correct and incorrect answers to a given question, respectively:

$$\begin{aligned} p_V(y \mid x) &\approx P_{\text{LLM}}\big(\text{Yes} \mid T_V(x, y)\big), \\ p_G(y \mid x) &\approx P_{\text{LLM}}\big(y \mid T_G(x)\big), \\ p_{G'}(y \mid x) &\approx P_{\text{LLM}}\big(y \mid T'_G(x)\big). \end{aligned}$$

There are two sources of error in these approximations. First, LLMs do not have an internally consistent belief $p(\mathbf{v} \mid x)$ about the answers to a problem. Second, even if they did, these prompts do not necessarily elicit rational responses to these.

Nevertheless, we argue that these approximations are useful. Empirically, most of the mass for LLMs when prompted for Yes/No answers lies on the *Yes* and *No* tokens, so at least LLMs can follow these instructions.[3]. $p_G(y \mid x)$ follows closely from how LLMs are trained to answer questions or follow instructions. On the other hand, $p_{G'}(y \mid x)$ is likely less well-approximated, since asking for incorrect responses is not where post-training focuses its efforts; the fidelity of this proxy is an empirical question, especially for smaller or non-instruction-tuned models.

Letting $s_V(y \mid x) := \log \frac{p_V(y|x)}{1 - p_V(y|x)}$ be the validator log-odds, we then have

$$\begin{aligned} s_V(y \mid x) = \log p_G(y \mid x) - \log p_{G'}(y \mid x) \\ + \log r(y \mid x) \end{aligned} \tag{7}$$

Unfortunately, $\log r(y \mid x)$ depends on conditional expectations over quantities that we cannot directly observe. Intuitively, what this term measures is the ratio of the probability of generating y when prompted for a wrong response (and the model thinks y is incorrect) and the probability of generating y when prompted for

[3] In our experiments we aggregate over the tokens `Yes`, `yes`, `YES`, `␣Yes` and `␣yes` for the positive class, and `No`, `no`, `NO`, `␣No` and `␣no` for the negative class.

a correct response (and the model thinks y is correct). Assuming that π and π' are both capturing the frequency of y, and this frequency is not biased by response correctness, this ratio can be approximated as 1.

As a result, disregarding r, Eqn 7 motivates the following adjusted generator score:

$$s_{\text{Adj}}(y \mid x) := \log p_G(y \mid x) - \log p_{G'}(y \mid x). \quad (8)$$

By construction, s_{Adj} tracks the validator log-odds s_V up to the residual $\log r$, and so should correlate better with s_V than the raw generator score $\log p_G$ does. In the next section, we operationalize s_{Adj} as a training objective for our model.

4 Training for Frequency Correction

Data Condition We consider a training setting where we have a dataset $\mathcal{D} = \{(x_i, \{(y_{ij}, v_{ij})\})\}$. That is, each x_i is paired with a set of outputs $\{(y_{ij}, v_{ij})\}$ where the y_{ij} are the outputs themselves and the v_{ij} are correctness labels. We operate in a semi-supervised setting where v_{ij} may not be available for all (x_i, y_{ij}) pairs.

As our training involves imposing consistency, we operate over pairs of points (e.g., similar to *hydrogen* and *Rn* in Figure 1). We define a labeled set $\mathcal{T} = \{((x_i, y_i, v_i), (x_j, y_j, v_j))\}$ of pairs with veracity labels. We define an unlabeled set $\mathcal{U} = \{((x_i, y_i), (x_j, y_j))\}$ of pairs without labels.

Furthermore, throughout this work we require $x_i = x_j$; we sample points *within the same prompt*. We also require $|p_V(y_j \mid x_j) - p_V(y_i \mid x_i)| > \delta$ to avoid noise from too-close validator pairs, following past work (Gekhman et al., 2025; Rodriguez et al., 2025).

Training Objective We use a loss designed to do two things: (1) teach the generator to rank pairs in the same way as the validator, and (2) use the labeled examples to improve the discriminability of generator and validator scores. We train using the following loss, where we let $y_j = y^+$ and $y_i = y^-$ denote the higher- and lower-ranked completions under the validator:

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{\text{pref}}(x_i, y_i, x_j, y_j) + \\ & \lambda_G(v_i \mathcal{L}_{\text{NLL}}(y_i \mid x_i) + v_j \mathcal{L}_{\text{NLL}}(y_j \mid x_j)) + \\ & \lambda_V(\mathcal{L}_{\text{NLL}}(v_i \mid x_i, y_i) + \mathcal{L}_{\text{NLL}}(v_j \mid x_j, y_j)) \quad (9)\end{aligned}$$

The $\mathcal{L}_{\text{pref}}$ loss is always active, while $\mathcal{L}_{\text{NLL}}$ terms are only active for labeled examples. The generator terms only apply to positive examples (i.e., $v_i = 1$ or $v_j = 1$) and the validator terms are only present for labeled examples, which have ground truth Yes/No completions.

Preference Loss We use a pairwise logistic preference loss given by

$$\mathcal{L}_{\text{pref}}(x, y^-, y^+) = -\log \sigma\big(s(y^+|x) - s(y^-|x)\big) \quad (10)$$

where σ is the logistic sigmoid and $s_G(y \mid x)$ is the generator's log-likelihood of completion y given prompt x,

$$s(y \mid x) = \sum_{t=1}^{|y|} \log P_{\text{LLM}}(y_t \mid x, y_{<t}). \quad (11)$$

To account for completion frequency, we consider two empirical estimators of the adjusted score s_{Adj} from Eqn 8 (§3.1). The first, **NegFC**, directly substitutes a negative-prompt elicitation for $p_{G'}$:

$$s_{\text{Neg}}(y \mid x) = s_G(y \mid x) - \log P_{\text{LLM}}(y \mid T'_G(x)). \quad (12)$$

The second, **Unconditional FC**, uses the unconditional log probability (Meister et al., 2023) as a cheaper proxy,

$$s_{\text{PMI}}(y \mid x) = s_G(y \mid x) - \log P_{\text{LLM}}(y), \quad (13)$$

which captures a similar frequency-penalization effect without requiring an elicitation prompt T'_G for incorrect answers.

Sampling strategy Finally, we ensure that the preference losses are consistent with the labels. For any pair $(x, y^-), (x, y^+)$ where $s_V(y^+ \mid x) > s_V(y^-|x) + \delta$, the preference loss will push the generator score of (x, y^-) down and (x, y^+) up; this is incorrect if y^- is a positive example or y^+ is a negative example. In addition, if y^- is positive the preference loss and generator $\mathcal{L}_{\text{NLL}}$ will compete in opposite directions. To avoid these problems, we only keep a labeled (x, y^-) when it is negative, and similarly we only keep a labeled (x, y^+) when it is positive.

5 Experimental Setup

5.1 Tasks and Datasets

We evaluate on three datasets which we adapted from previous work to test our method on instruction following, coding, and conceptual knowledge. We use IFEval (Zhou et al., 2023) for instruction following, HumanEval (Chen et al., 2021) for Python coding, and a mix of datasets to probe for knowledge of taxonomic category relations (hyponymy) (Rosch, 1975; Banks and Connell, 2023; Stoinski et al., 2024; Castro et al., 2021; Van Overschelde et al., 2004; Uyeda and Mandler, 1980). Each dataset has a different notion of correctness, detailed below. These datasets are described here, with further details on datasets given in Appendix C. Prompt templates for each of these tasks are shown in Appendix D.

Instruction Following This task uses prompts from IFEval (Zhou et al., 2023), e.g., "*Write a poem that's at least 350 words about the beauty of eucalyptus trees and their many uses.*" which specify explicit content and format constraints. A correct response should follow all constraints in the prompt, while an incorrect response violates one or more. We generate, using GPT-4o mini, correct responses via paraphrases of the original prompt and incorrect responses by introducing violations (e.g., missing required elements or breaking constraints). Labels are assigned using rule-based checks for verifiable

constraints together with LLM-as-a-judge evaluations for content correctness where deterministic verification is not possible. We use 79 prompts for the training set and 20 for the test set.

Coding This task uses prompts from the HumanEval (Chen et al., 2021) benchmark of Python programming exercises. Each prompt asks to complete a Python function to solve a given problem. Here correct responses are those which pass all the unit tests, while incorrect ones fail one or more. We use various LLMs to generate correct and incorrect responses. To stress-test our method with examples that are correct but unlikely, we systematically convert the variables of all the correct solutions to uppercase.

Hyponymy This task consists of giving exemplars to categories, i.e., answering "An example of a X is a __" for various nouns X. We evaluated on the ten categories used in the original experiments of (Rosch, 1975), supplemented with GPT-5-generated incorrect completions in order to obtain a balanced test set. We decided on a cutoff between items that are members of the category while those that are not. While there is some subjectivity in this decision, our cutoff is validated by the fact that LLMs with validator prompts can successfully distinguish between the positive and negative classes (Table 5).

For all three tasks, we evaluate on a held-out set of prompts. For IFEval and HumanEval, we randomly split the data. For Hyponymy we train on a different set of categories than the ten in (Rosch, 1975). Further details of the dataset composition and construction process is given in Appendix C.

5.2 Target Models for G-V Alignment

As we defined the dataset at the beginning of Section 4, our target models for G-V consistency do not necessarily have to be those from which responses were generated. In fact, having a range of responses (as we generated for each dataset) that we require G-V consistency on allows us to stress-test each model beyond the mode of the distribution that it generates.

We use a range of models including those in the Gemma family, namely Gemma-2-9b-it (**G2-9b-it**) and Gemma-4-31b-it (**G4-31b-it**), and Qwen family, namely Qwen-3.5-9B (**Q3.5-9b**). For each task, we select models in an appropriate performance range: for instance, on IFEval, we only use larger instruction-tuned models because smaller models are generally not capable enough at the task. For each task and model combination, we only proceed with models where validator AUROC is over 65%); this eliminates Gemma-2-9b-it from the HumanEval experiments.

5.3 Evaluation

For all methods, we evaluate both using the raw generator scores $s_G(y \mid x) = \log P_{LLM}(y \mid x)$ as well as the two corrected forms: frequency-corrected scores $s_{\text{PMI}} = s_G(y \mid x) - \log P_{LLM}(y)$, and negative prompting-corrected scores $s_{\text{Neg}} = s_G(y \mid x) - s'_G(y \mid x)$. All metrics are computed independently for each prompt, and averaged over prompts.

Accuracy We evaluate the validator and generator performance at discriminating correct and incorrect answers through AUROC scores of the associated classifiers: $\boldsymbol{p_G}(y \mid x) > \tau_G$ and $s_V(y \mid x) > \tau_V$ for thresholds τ_G and τ_V. Each defines an ROC curve and we refer to the corresponding AUROC scores as $\mathbf{ROC_G}$ and $\mathbf{ROC_V}$. For the generator, we use one of several different possible generator scores: $\boldsymbol{p_G}$, s_{PMI} or s_{Neg}. We additionally measure the **validator accuracy** at $\tau_V = 0$ in order to measure how well-calibrated the validator is.

We focus primarily on $\mathbf{ROC_G}$; our focus in this work is on tasks where the validator is stronger than the generator and can be used to improve the generator's performance.

Correlations Following Rodriguez et al. (2025), we measure the Pearson correlation ρ between the validator and (possibly-corrected) generator scores over the full set of candidate answers.

5.4 Baselines

We compare our method against the following baselines in addition to the **Base** (untrained) model:

SFT We sample pairs of completions as in RankAlign, but use a Supervised Fine-Tuning (SFT) loss, i.e., only the negative log-likelihood terms from Eqn. 9.

Consistency FT Li et al. (2024) SFT only on the subset of examples where generator and validator agree. We use the version from Rodriguez et al. (2025), which measures agreement via $\mathbb{1}_{l_G(z,y_A)>t_G} = \mathbb{1}_{l_V(z,y_A)>t_V}$ where t_G and t_V are thresholds set as the average generator and validator scores over the dataset.

RankAlign We use the preference loss of RankAlign (Rodriguez et al., 2025). Compared to our method this (1) does not apply any frequency correction during training; (2) does not enforce sampled pairs to have the same prompt during training, (3) does not use NLL terms.

6 Results

6.1 Frequency Correction Helps at Test Time

Frequency correction improves generator–validator consistency even before training.

We first ask whether frequency correction, applied at test time, improves the generator's ability to discriminate correct from incorrect answers and its consistency with the model's own validator. Table 1 compares the raw generator score s_G with the two corrected variants introduced in Section 4: NegTC (s_{Neg}), which subtracts the log probability when using the negative prompt, and Unconditional FC (s_{PMI}), which subtracts the unconditional log probability of the completion.

Eval-time score	IFEval		HumanEval		Hyponymy	
	ROC_G	ρ	ROC_G	ρ	ROC_G	ρ
s_G (raw)	57.5 ± 1.9	16.3 ± 3.2	65.7 ± 1.5	34.1 ± 2.2	66.9 ± 2.6	31.5 ± 2.6
s_{PMI} (Uncond. FC)	**81.0** ± 1.2	**44.0** ± 2.8	**72.9** ± 1.3	22.4 ± 3.0	77.7 ± 2.2	**50.0** ± 2.7
s_{Neg} (NegFC)	64.5 ± 1.8	25.3 ± 3.4	71.1 ± 1.6	**38.5** ± 2.5	**81.2** ± 4.0	48.0 ± 5.4

Table 1: Eval-time frequency correction applied to Gemma-2-9b-it, except HumanEval, which uses Gemma-4-31b-it. Both NegFC and Unconditional FC improve over the raw generator score, but neither dominates across all tasks. Subsequent sections apply the per-task best eval-time correction to all methods for a fair comparison.

Both corrections improve ROC_G and ρ over the raw generator score on all tasks. This indicates that some amount of apparent G-V gap goes away when using these corrected terms, indicating that they form a better starting point for improving G-V consistency and generator discriminability.

6.2 Main Results

FLORA outperforms prior consistency alignment methods on both ROC_G and ρ.

Table 2 compares methods on ROC_G and ρ for IFEval, HumanEval, and Hyponymy datasets. All methods are evaluated with the per-task best eval-time correction identified in Section 6.1. Full results with standard errors are given in Table 4 in Appendix B.

FLORA achieves a higher ROC_G than the other baselines in five out of six settings (tying RankAlign on Hyponymy with Qwen-3.5-9B), with gains ranging from +0.9 to +7.3pp over the next-best method. G-V consistency improves by a wider margin, with ρ +27pp over the next-best method on both IFEval with Gemma-2-9b-it (FLORA-PMI: 62.8 vs. 35.5 for Base) and HumanEval (FLORA-Neg: 76.3 vs. 49.0 for Consistency FT). On the two Hyponymy settings, FLORA is competitive on both metrics, trailing RankAlign by at most 1.4pp on ρ.

FLORA Preserves Validator Quality *Improvements in G-V consistency do not come at the cost of validator quality.*

A standard concern with preference-style training is *likelihood displacement* (Razin et al., 2025; Xu et al., 2024): the model may appear better aligned only because its validator has degraded. To rule this out, we measure the validator accuracy and AUROC. Table 5 shows the validator accuracies per-method (ROC_V and Acc_V).

The effect of frequency correction Table 3 ablates the frequency correction term applied *during training*. Here the "w/o typ. corr." variant trains on the raw generator log-likelihood. Both variants apply the same eval-time correction, so the comparison isolates the effect of frequency correction during training. Removing it hurts both ROC_G and ρ on every task, with the largest drops on HumanEval (-6.3pp ROC_G, -16.1pp ρ).

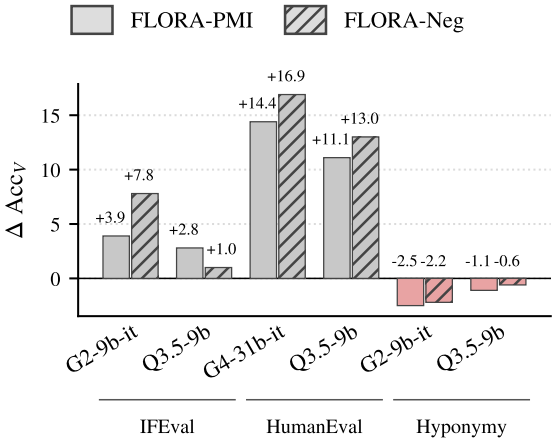

Figure 4: Change in validator accuracy ($\Delta\,\text{Acc}_V$, Method − Base, in percentage points) after FLORA training. **Our primary aim is for validator accuracy not to degrade after FLORA training.** However, FLORA actually preserves or improves validator accuracy across all five settings, with the largest gains on HumanEval (+14–17pp) and IFEval/G2-9b-it (+4–8pp).

6.3 Qualitative Analysis of Alignment

Across many prompts, we observe cleaner consistency between generator scores and validator scores after applying our method. Figure 5 illustrates a case of this for a prompt in IFEval and a set of possible completions. X-axes in the left panel are the raw generator scores, while x-axes in the right two panel show the corrected generator scores s_{PMI}. For the base model (left), the relationship between generator scores and validator scores is noisy: some high-likelihood generations receive low validator scores and vice versa, indicating a misalignment between what the generator prefers and what the validator deems correct. Using frequency correction for the generator scores at test time improves both generator-validator correlation and generator ROC (middle panel). After training, (right panel) the relationship between generator and validator scores becomes even more structured, with a clearer trend in which higher generator scores correspond to higher validator scores.

7 Related Work

Generator-Validator Gap. Generator-validator gaps have been discussed in a number of contexts, including

	IFEval				HumanEval				Hyponymy			
	G2-9b-it		Q3.5-9b		G4-31b-it		Q3.5-9b		G2-9b-it		Q3.5-9b	
Method	ROC_G	ρ	ROC_G	ρ	ROC_G	ρ	ROC_G	ρ	ROC_G	ρ	ROC_G	ρ
Base	78.2	35.5	69.3	21.5	72.9	22.4	65.7	30.4	81.2	48.0	73.5	42.9
SFT	74.4	29.5	73.1	24.1	84.5	28.5	**94.1**	56.3	88.1	64.0	82.6	48.6
Consistency FT	58.7	0.8	69.8	22.4	74.9	49.0	92.6	53.5	85.3	50.8	80.1	56.6
RankAlign	74.9	29.9	75.4	32.8	86.9	40.9	90.2	**69.5**	91.6	**75.2**	88.2	**70.4**
FLORA-PMI	**85.1**	**62.8**	**79.6**	43.3	92.2	66.8	83.2	58.9	**92.5**	73.8	87.3	69.4
FLORA-Neg	60.8	34.4	66.0	**44.7**	**94.2**	**76.3**	88.9	68.5	**92.3**	72.2	**88.2**	69.4

Table 2: **Main results across tasks and models.** ROC_G (generator AUROC) measures generator discriminability, and ρ measures generator–validator Pearson correlation). All methods use the per-task best eval-time correction.

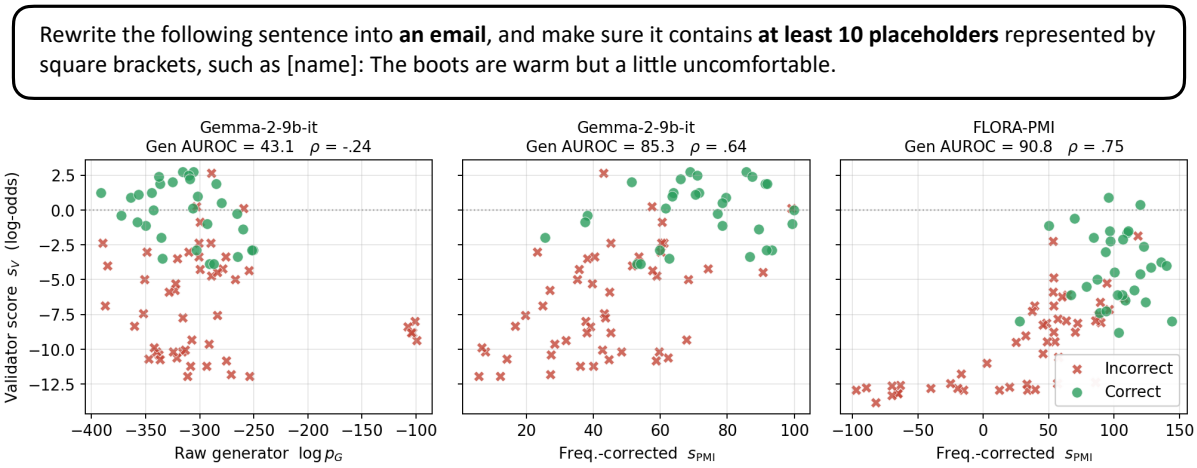

Figure 5: Generator score versus validator log-odds for candidate responses to a single IFEval prompt (base Gemma-2-9b-it; green: correct, red: incorrect). Each panel plots that model's own validator on the y-axis. *Left:* the raw generator score $\log p_G$ is uncorrelated with correctness and does not separate correct from incorrect completions (Gen AUROC 43.1, ρ=−.24). *Middle:* applying the frequency (PMI) correction at test time already recovers most of the alignment (AUROC 85.3, ρ=.64). *Right:* after FLORA-PMI training, the corrected generator score ranks correct above incorrect completions and tracks the validator (AUROC 90.8, ρ=.75).

	FLORA (full)		w/o typ. corr.	
Task	ROC_G	ρ	ROC_G	ρ
IFEval	85.1	62.8	78.5	61.1
HumanEval	94.2	76.3	87.9	60.2
Hyponymy	92.5	73.8	90.9	73.1

Table 3: Removing frequency correction during training (shown here: Gemma-2-9b-it) has a detrimental effect on both G-V consistency and generator performance. Frequency correction is applied at test time in all cases.

(West et al., 2024), (Li et al., 2024), and (Rodriguez et al., 2025). Our work builds on these notions and is the first to derive a principled relationship between generator probabilities and validator log-odds, which (Rodriguez et al., 2025) treated heuristically.

Preference Learning. The ranking-based preference objective we use is similar to DPO (Rafailov et al., 2023). In our case, the validator produces preferences for the generator, broadly related to self-rewarding LMs (Yuan et al., 2024). However, rather than a general post-training method, we view FLORA as a mechanism to achieve *consistency* in an LLMs' predictions.

Frequency correction. Past work has explored similar ideas in frequency correction. Holtzman et al. (2021) correct for frequency in multiple-choice settings. Their domain-conditional PMI is similar to our PMI objective, but it is motivated as a heuristic. Liu et al. (2021) uses the idea of "anti-experts" for controlled text generation, which gives rise to a similar form (subtracting log probabilities from the anti-expert). Li et al. (2023) then extend this idea to contrasting strong and weak language models. However, none of these approaches targets consistency per se.

Theoretical Formulation. We are not the first to analyze LLM behavior by decomposing output distributions over a latent variable. Xie et al. (2022) formulate in-context learning as posterior inference over a latent document concept, and Bigelow et al. (2025) similarly adopt a latent variable formulation to unify in-context learning and activation steering. Our derivation in §3.1 uses a similar idea, but applies it to a different question:

the discrepancy between a single model's generator and validator distributions on a given prompt.

8 Conclusion

In this paper, we presented FLORA, a method for improving LLMs via validator-to-generator alignment. We derive a relationship between generator probabilities and validator log-odds based on a model of how a rational agent might generate a response. This relationship suggests a frequency-based correction factor. When trained to align validator and generator with this factor, we see improved results across taxonomic categorization (Hyponymy) and two long-form generation problems, IFEval and HumanEval.

Limitations

One limitation of this work is a gap between the theoretical motivation and practical implementation. We rely on an assumption that the intractable r term in Equation 4 is close to 1. Although we believe this is a reasonable assumption as discussed in the text, it is difficult to validate this as r is intractable to estimate even on relatively simple examples. Nevertheless, we see that our theoretically-derived training objective performs well empirically, lending credence to our method.

A second limitation is the focus on evaluating fixed sets of outputs. We focus on evaluating the tails of the model's distribution: samples which are potentially low likelihood but which we still believe the model should rank correctly. We believe this setting is relevant, as LLM judges are frequently asked to reckon about fixed responses from other models, and various kinds of self-consistency or best-of-N approaches use a related kind of scoring. However, we do not demonstrate an impact on the 1-best answers produced by greedy decoding.

Furthermore, our framework assumes that correctness is a binary notion. Relaxing this to accommodate partially-correct answers would be an interesting direction for future.

Finally, we note that evaluating within-prompt rather than across a diverse set of prompts prevents us from evaluating directly on the datasets used to measure the G-V gap in (Rodriguez et al., 2025), because most prompts in that work only have a small set of completions. To still enable a comparison, we re-implement RankAlign as a baseline and evaluate it on our datasets alongside FLORA (Tables 4, 5).

Acknowledgments

We thank Jacob Andreas for insightful discussion around this work. This work was supported by NSF CAREER Award IIS-2145280, NSF grant IIS-2433071, the NSF AI Institute for Foundations of Machine Learning (IFML), and the NSF under Cooperative Agreement 2421782 and the Simons Foundation grant MPS-AI-00010515 awarded to the NSF-Simons AI Institute for Cosmic Origins — CosmicAI, https://www.cosmicai.org/. We also thank members of the TAUR Lab for helpful feedback.

References

Briony Banks and Louise Connell. 2023. Category production norms for 117 concrete and abstract categories. *Behavior Research Methods*, 55(3):1292–1313.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, and 11 others. 2024. Lessons from the Trenches on Reproducible Evaluation of Language Models. *CoRR*, abs/2405.14782.

Eric Bigelow, Daniel Wurgaft, YingQiao Wang, Noah Goodman, Tomer Ullman, Hidenori Tanaka, and Ekdeep Singh Lubana. 2025. Belief dynamics reveal the dual nature of in-context learning and activation steering. *arXiv preprint arXiv:2511.00617*.

Nichol Castro, Taylor Curley, and Christopher Hertzog. 2021. Category norms with a cross-sectional sample of adults in the united states: Consideration of cohort, age, and historical effects on semantic categories. *Behavior research methods*, 53(2):898–917.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating Large Language Models Trained on Code. *CoRR*, abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Zorik Gekhman, Eyal Ben-David, Hadas Orgad, Eran Ofek, Yonatan Belinkov, Idan Szpektor, Jonathan Herzig, and Roi Reichart. 2025. Inside-out: Hidden factual knowledge in llms. In *Proceedings of the Conference on Language Modeling (COLM)*.

H. P. Grice. 1975. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, New York.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*,

pages 7038–7051, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive Decoding: Open-ended Text Generation as Optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.

Xiang Lisa Li, Vaishnavi Shrivastava, Siyan Li, Tatsunori Hashimoto, and Percy Liang. 2024. Benchmarking and improving generator-validator consistency of language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. 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)*, pages 6691–6706, Online. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2023. Locally typical sampling. *Trans. Assoc. Comput. Linguistics*, 11:102–121.

Itamar Pres, Belinda Z Li, Laura Ruis, Zifan Carl Guo, Keya Hu, Mehul Damani, Isha Puri, Ekdeep Singh Lubana, and Jacob Andreas. 2026. Position: It's time to optimize for self-consistency.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Noam Razin, Sadhika Malladi, Adithya Bhaskar, Danqi Chen, Sanjeev Arora, and Boris Hanin. 2025. Unintentional unalignment: Likelihood displacement in direct preference optimization. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Juan Diego Rodriguez, Wenxuan Ding, Katrin Erk, and Greg Durrett. 2025. RankAlign: A Ranking View of the Generator-Validator Gap in Large Language Models. In *Proceedings of the Conference on Language Modeling (COLM)*.

Eleanor Rosch. 1975. Cognitive representations of semantic categories. *Journal of experimental psychology: General*, 104(3):192.

Laura M Stoinski, Jonas Perkuhn, and Martin N Hebart. 2024. Thingsplus: New norms and metadata for the things database of 1854 object concepts and 26,107 natural object images. *Behavior Research Methods*, 56(3):1583–1603.

Tim Tomov, Dominik Fuchsgruber, Tom Wollschläger, and Stephan Günnemann. 2025. The Illusion of Certainty: Uncertainty quantification for LLMs fails under ambiguity. *CoRR*, abs/2511.04418.

Katherine M Uyeda and George Mandler. 1980. Prototypicality norms for 28 semantic categories. *Behavior Research Methods & Instrumentation*, 12(6):587–595.

James P Van Overschelde, Katherine A Rawson, and John Dunlosky. 2004. Category norms: An updated and expanded version of the norms. *Journal of memory and language*, 50(3):289–335.

Xinpeng Wang, Bolei Ma, Chengzhi Hu, Leon Weber-Genzel, Paul Röttger, Frauke Kreuter, Dirk Hovy, and Barbara Plank. 2024. "My Answer is C": First-Token Probabilities Do Not Match Text Answers in Instruction-Tuned Language Models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7407–7416, Bangkok, Thailand. Association for Computational Linguistics.

Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D. Hwang, Liwei Jiang, Jillian Fisher, Abhilasha Ravichander, Khyathi Raghavi Chandu, Benjamin Newman, Pang Wei Koh, Allyson Ettinger, and Yejin Choi. 2024. The Generative AI Paradox: "What It Can Create, It May Not Understand". In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 55204–55224. PMLR / OpenReview.net.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 57905–57923. PMLR / OpenReview.net.

Yiming Zhang, Harshita Diddee, Susan Holm, Hanchen Liu, Xinyue Liu, Vinay Samuel, Barry Wang, and Daphne Ippolito. 2025. NoveltyBench: Evaluating Language Models for Humanlike Diversity. In *Proceedings of the Conference on Language Modeling (COLM)*.

Yiming Zhang, Avi Schwarzschild, Nicholas Carlini, Zico Kolter, and Daphne Ippolito. 2024. Forcing Diffuse Distributions out of Language Models. In *Proceedings of the Conference on Language Modeling (COLM)*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *CoRR*, abs/2311.07911.

A Proof of Theorem 1

Step 1: Rewrite p_G and $p_{G'}$. By definition,

$$p_G(y \mid x) = \sum_{\mathbf{v}} p(\mathbf{v} \mid x)\, \pi(y \mid \mathbf{v}, x).$$

Split the sum into $\{\mathbf{v} : v_y = 1\}$ and $\{\mathbf{v} : v_y = 0\}$. By the support constraint on π, every term in the second sum vanishes, so

$$p_G(y \mid x) = \sum_{\mathbf{v}: v_y = 1} p(\mathbf{v} \mid x)\, \pi(y \mid \mathbf{v}, x). \tag{14}$$

By the same reasoning,

$$p_{G'}(y \mid x) = \sum_{\mathbf{v}: v_y = 0} p(\mathbf{v} \mid x)\, \pi'(y \mid \mathbf{v}, x). \tag{15}$$

Step 2: Express with conditional expectation. For any function $f(\mathbf{v})$ and set $A \subseteq \{0,1\}^{|\mathcal{Y}|}$ with $P(\mathbf{v} \in A) > 0$,

$$\mathbb{E}[f(\mathbf{v}) \mid \mathbf{v} \in A] = \frac{\sum_{\mathbf{v} \in A} p(\mathbf{v} \mid x)\, f(\mathbf{v})}{\sum_{\mathbf{v} \in A} p(\mathbf{v} \mid x)}.$$

Applying this with $f(\mathbf{v}) = \pi(y \mid \mathbf{v}, x)$ and $A = \{\mathbf{v} : v_y = 1\}$, the fact that $p_V(y \mid x) > 0$ ensures that the denominator is greater than 0, as the denominator equals $p_V(y \mid x)$ by (1). The numerator equals $p_G(y \mid x)$ by (14). Hence

$$\mathbb{E}[\pi(y \mid \mathbf{v}, x) \mid v_y = 1, x] = \frac{p_G(y \mid x)}{p_V(y \mid x)}. \tag{16}$$

The analogous calculation with $f(\mathbf{v}) = \pi'(y \mid \mathbf{v}, x)$ and $A = \{\mathbf{v} : v_y = 0\}$, using $p_V(y \mid x) < 1$ to ensure $1 - p_V(y \mid x) > 0$, yields

$$\mathbb{E}[\pi'(y \mid \mathbf{v}, x) \mid v_y = 0, x] = \frac{p_{G'}(y \mid x)}{1 - p_V(y \mid x)}. \tag{17}$$

Step 3: Take the ratio. The denominators on the right-hand sides of (16) and (17) are strictly positive by the hypothesis $0 < p_V(y \mid x) < 1$. The numerators are nonnegative as sums of nonnegative terms.

Dividing (17) by (16),

$$\frac{p_{G'}(y \mid x)/(1 - p_V(y \mid x))}{p_G(y \mid x)/p_V(y \mid x)} = \frac{\mathbb{E}[\pi'(y \mid \mathbf{v}, x) \mid v_y = 0, x]}{\mathbb{E}[\pi(y \mid \mathbf{v}, x) \mid v_y = 1, x]}$$

The left-hand side simplifies to

$$\frac{p_V(y \mid x)}{1 - p_V(y \mid x)} \cdot \frac{p_{G'}(y \mid x)}{p_G(y \mid x)},$$

and rearranging gives

$$\frac{p_V(y \mid x)}{1 - p_V(y \mid x)} = \frac{p_G(y \mid x)}{p_{G'}(y \mid x)} \cdot r(y, x).$$

B Additional Results

Full results on generator ROC (ROC_G) and Pearson correlation (ρ) are given in Table 4. Each entry is the average score across prompts, while $\pm$ indicates the standard error.

The full results on validator accuracy (Acc_V) and validator AUROC (ROC_V) are given in Table 5.

C Dataset construction Details

C.1 IFEval

IFEval (Zhou et al., 2023) is an instruction-following benchmark designed to evaluate whether model outputs satisfy explicit, verifiable constraints specified in a prompt. To construct our dataset, we sample and filter prompts from IFEval to ensure they are well-formed, English-only, and support the generation of sufficiently diverse outputs. For each prompt, we generate candidate responses by introducing prompt variations, including

	IFEval				HumanEval				Hyponymy			
	G2-9b-it		Q3.5-9b		G4-31b-it		Q3.5-9b		G2-9b-it		Q3.5-9b	
Method	ROC_G	ρ	ROC_G	ρ	ROC_G	ρ	ROC_G	ρ	ROC_G	ρ	ROC_G	ρ
Base	$78.2_{\pm 2.2}$	$35.5_{\pm 7.5}$	$69.3_{\pm 3.7}$	$21.5_{\pm 7.8}$	$72.9_{\pm 1.3}$	$22.4_{\pm 3.0}$	$65.7_{\pm 1.6}$	$30.4_{\pm 2.4}$	$81.2_{\pm 4.0}$	$48.0_{\pm 5.4}$	$73.5_{\pm 1.1}$	$42.9_{\pm 2.5}$
SFT	$74.4_{\pm 2.6}$	$29.5_{\pm 6.3}$	$73.1_{\pm 3.4}$	$24.1_{\pm 4.8}$	$84.5_{\pm 1.1}$	$28.5_{\pm 3.0}$	$94.1_{\pm 0.7}$	$56.3_{\pm 2.2}$	$88.1_{\pm 1.3}$	$64.0_{\pm 2.8}$	$82.6_{\pm 1.9}$	$48.6_{\pm 2.9}$
Consistency FT	$58.7_{\pm 3.3}$	$0.8_{\pm 7.9}$	$69.8_{\pm 3.6}$	$22.4_{\pm 6.6}$	$74.9_{\pm 1.4}$	$49.0_{\pm 2.3}$	$92.6_{\pm 0.8}$	$53.5_{\pm 2.0}$	$85.3_{\pm 1.4}$	$50.8_{\pm 3.4}$	$80.1_{\pm 1.9}$	$56.6_{\pm 2.2}$
RankAlign	$74.9_{\pm 3.6}$	$29.9_{\pm 7.6}$	$75.4_{\pm 3.9}$	$32.8_{\pm 7.7}$	$86.9_{\pm 1.0}$	$40.9_{\pm 3.0}$	$90.2_{\pm 1.3}$	$69.5_{\pm 1.7}$	$91.6_{\pm 1.3}$	$\mathbf{75.2}_{\pm 2.7}$	$88.2_{\pm 1.8}$	$\mathbf{70.4}_{\pm 3.0}$
FLORA-PMI	$\mathbf{85.1}_{\pm 2.7}$	$\mathbf{62.8}_{\pm 4.4}$	$\mathbf{79.6}_{\pm 3.4}$	$43.3_{\pm 7.3}$	$92.2_{\pm 0.8}$	$66.8_{\pm 1.8}$	$83.2_{\pm 1.2}$	$58.9_{\pm 2.0}$	$\mathbf{92.5}_{\pm 1.1}$	$73.8_{\pm 2.3}$	$87.3_{\pm 2.1}$	$69.4_{\pm 2.6}$
FLORA-Neg	$60.8_{\pm 4.3}$	$34.4_{\pm 6.4}$	$66.0_{\pm 3.1}$	$\mathbf{44.7}_{\pm 4.9}$	$\mathbf{94.2}_{\pm 1.5}$	$\mathbf{76.3}_{\pm 1.3}$	$88.9_{\pm 1.1}$	$68.5_{\pm 1.5}$	$\mathbf{92.3}_{\pm 1.5}$	$72.2_{\pm 3.1}$	$\mathbf{88.2}_{\pm 2.1}$	$69.4_{\pm 2.9}$

Table 4: Main results across tasks and models. ROC_G measures generator discriminability, and ρ measures generator–validator correlation. All methods use the per-task best eval-time correction. Models: G2-9b-it= Gemma-2-9b-it, Q3.5-9b = Qwen-3.5-9B, G4-31b-it = Gemma-4-31b-it.

	IFEval				HumanEval				Hyponymy			
	G2-9b-it		Q3.5-9b		G4-31b-it		Q3.5-9b		G2-9b-it		Q3.5-9b	
Method	ROC_V	Acc_V	ROC_V	Acc_V	ROC_V	Acc_V	ROC_V	Acc_V	ROC_V	Acc_V	ROC_V	Acc_V
Base	$78.4_{\pm 3.5}$	$59.7_{\pm 3.5}$	$82.1_{\pm 2.8}$	$70.8_{\pm 4.0}$	$90.8_{\pm 1.1}$	$71.4_{\pm 1.6}$	$81.7_{\pm 1.3}$	$65.3_{\pm 1.4}$	$94.9_{\pm 1.9}$	$87.1_{\pm 2.5}$	$95.6_{\pm 1.8}$	$87.9_{\pm 2.9}$
SFT	$78.4_{\pm 3.7}$	$56.4_{\pm 2.9}$	$82.8_{\pm 3.1}$	$73.1_{\pm 3.8}$	$91.9_{\pm 1.0}$	$79.6_{\pm 1.5}$	$\mathbf{86.5}_{\pm 1.0}$	$\mathbf{78.6}_{\pm 1.2}$	$95.0_{\pm 1.8}$	$\mathbf{85.3}_{\pm 2.5}$	$95.1_{\pm 2.0}$	$84.8_{\pm 3.3}$
Consistency FT	$77.7_{\pm 3.8}$	$62.6_{\pm 4.0}$	$81.4_{\pm 3.3}$	$69.2_{\pm 4.0}$	$93.8_{\pm 0.8}$	$88.2_{\pm 1.0}$	$84.6_{\pm 1.1}$	$76.5_{\pm 1.2}$	$94.7_{\pm 1.9}$	$80.1_{\pm 3.1}$	$95.0_{\pm 2.0}$	$84.3_{\pm 3.3}$
RankAlign	$70.5_{\pm 4.0}$	$61.4_{\pm 3.6}$	$82.7_{\pm 2.8}$	$62.8_{\pm 4.5}$	$91.6_{\pm 1.0}$	$76.8_{\pm 1.6}$	$83.2_{\pm 1.2}$	$74.3_{\pm 1.2}$	$\mathbf{95.1}_{\pm 1.8}$	$85.1_{\pm 2.7}$	$95.6_{\pm 1.8}$	$\mathbf{87.5}_{\pm 3.2}$
FLORA-PMI	$\mathbf{79.4}_{\pm 3.2}$	$63.6_{\pm 2.8}$	$83.1_{\pm 3.3}$	$\mathbf{73.6}_{\pm 3.8}$	$93.3_{\pm 0.9}$	$85.8_{\pm 1.4}$	$86.3_{\pm 1.1}$	$76.4_{\pm 1.3}$	$94.5_{\pm 1.9}$	$84.6_{\pm 2.4}$	$\mathbf{96.0}_{\pm 1.7}$	$86.8_{\pm 2.8}$
FLORA-Neg	$79.0_{\pm 3.5}$	$\mathbf{67.5}_{\pm 3.9}$	$\mathbf{83.2}_{\pm 3.3}$	$71.8_{\pm 3.8}$	$\mathbf{94.0}_{\pm 0.8}$	$\mathbf{88.3}_{\pm 1.1}$	$86.1_{\pm 1.1}$	$78.3_{\pm 1.1}$	$94.6_{\pm 2.0}$	$84.9_{\pm 2.7}$	$95.6_{\pm 1.7}$	$87.3_{\pm 3.0}$

Table 5: Validator performance across tasks and models. ROC_V measures validator ROC-AUC and Acc_V measures validator accuracy at threshold 0. Both are independent of the eval-time frequency correction (they only use the validator log-odds against the gold label). FLORA matches or exceeds the base model on most tasks, indicating no validator-side likelihood displacement.

equivalent rephrasings (to produce positive samples) and violations of content or format constraints (to produce negative samples). This yields a set of responses with nontrivial separability. Ground-truth labels are assigned using a combination of IFEval-style rule-based verification for format constraints (e.g., keyword counts, structural requirements) and prompt-induced correctness assumptions for content constraints (e.g., relevance or factual consistency). A response is labeled as correct only if it satisfies both criteria. We further divide the dataset into in-domain and out-of-domain splits, where the former evaluates performance on seen prompts with held-out responses, and the latter measures generalization to entirely unseen prompts.

C.2 Hyponymy

Hyponymy is the relationship between a higher-level category and it's exemplars (e.g., (furniture, table)). We adopted the data used by (Rosch, 1975), consisting of ten classes, each with 30 example items. These examples ranged from very prototypical (e.g., table) to less usual (e.g., lamp), with the least prototypical not arguably not really belonging to the category at all. Since most of the examples were positive (actual examples of a category), we used GPT-5 to sample additional examples.

In order to train with a disjoint set of categories, we combined the datasets from (Banks and Connell, 2023; Stoinski et al., 2024; Castro et al., 2021; Van Overschelde et al., 2004; Uyeda and Mandler, 1980), in order to have a diverse set. Then we manually removed the categories which overlapped with the ten in (Rosch, 1975).

C.3 HumanEval

We generated solutions to the 164 Python problems in the HumanEval benchmark, requiring at least 10 passing and 10 failing solutions per problem. We used a diverse set of LLMs to generate solutions from: gpt-3.5-turbo, gpt-4, gpt-4o, gpt-4o-mini, gpt-4.1, gpt-5, and gpt-5.5, as well as Llama-3.1-8B-Instruct, deepseek-coder-1.3b-instruct, Phi-3-mini-4k-instruct, Mistral-7B-Instruct-v0.3, and OLMo-2-0425-1B-Instruct. Each generation was prompted under one of three difference persona strategies: normal, beginner, unusual, refactorable.

Finally, we filtered the solutions that were too short or too long (less than 10 characters or more than 900 characters). 82 problems were randomly sampled to be the held-out test set.

D Prompts

E Dataset Licenses

We summarize the licenses of the datasets used in this work and confirm that our use is consistent with their intended terms.

HumanEval. HumanEval (Chen et al., 2021) is released by OpenAI under the MIT License (see `https://github.com/openai/human-eval`). The MIT License permits use, modification, and redistribution, including for research purposes such as ours, provided the original copyright and license notice are retained. Our

IFEval: Example Prompt and Candidate Responses

IFEval Prompt

Write a song about a man who rents a room in a house with a bunch of other people, and he absolutely hated it. Your song should contain at least 10 words in all capital letters that are adjectives or verbs. Commas are not allowed in the song.

Positive Candidate Response

Title: Cramped and Cursed

Waking up in this WORN-OUT place
Roommates scatter like RATS in a race
Socks and dishes piling HIGH
Voices echo and SHOUT from the sky

Every night's a MESS that never ends
Panic and noise I CAN'T defend
No space to breathe it FEELS so tight
In this gloomy box my dreams take flight

Life here is FROZEN like a bad mistake
I search for peace but ALL I do is ache
Slamming doors and DRIFTING dreams
I'm haunted by all these NOISY screams

A hopeful heart that's TRAPPED inside
Yearning to escape I can't CONFIDE
Longing for laughter not DISTRACTION
But I keep STUMBLING on this chain reaction
Fingers crossed I'll BREAK these chains
From this cursed house with all its pains
Someday I'll find that FREEDOM so true
But for now it's just ME and this messy crew

Negative Candidate Response

Room to Love

In a HOUSE filled with laughter
She found a SPACE so bright
A heart full of wonder
In the soft morning LIGHT

DREAMS linger on the walls
With FRIENDS who understand
They SHOUT and dance in the halls
Creating memories hand in hand

The KITCHEN filled with chatter
Cups of coffee in the air
Sharing STORIES and secrets
Showing love everywhere

Every NOOK is a treasure
Each moment feels so free
A PLACE where her heart soars
As happy as can be
Living in a FLAIR so vivid
Where life's a grand DISPLAY
Together they SHINE like stars
In their beautiful ballet

Figure 6: Example prompt and candidate responses from IFEVAL

HumanEval: Example Prompt and Candidate Responses

HumanEval Prompt

```
def string_to_md5(text):
    """
    Given a string 'text', return its md5 hash equivalent string.
    If 'text' is an empty string, return None.

    >>> string_to_md5('Hello world') == '3e25960a79dbc69b674cd4ec67a72c62'
    """
```

Positive Candidate Response

```
if not text:
    return None
import hashlib
HASH_OBJECT = hashlib.md5(text.encode())
return HASH_OBJECT.hexdigest()
```

Negative Candidate Response

```
if not text:
    return None

m = hashlib.md5()
m.update(text.encode('utf-8'))
return m.hexdigest()
```

Figure 7: Example prompt and candidate responses from HUMANEVAL

Hyponym: Example Hypernym and Candidate Responses

Hypernym

clothing

Positive Candidate Response

leotard

Negative Candidate Response

purse

Figure 8: Example hypernym and candidate responses (hyponyms) from HYPONYM

Prompts: IFEval Templates

Generator Template

<*ifeval_prompt*>

Validator Template

You are a strict binary validator.

You will be given a prompt and a response. Determine if the response follows ALL instructions in the prompt exactly.

Rules:
- Every constraint must be satisfied perfectly.
- If ANY constraint is violated, output NO.
- Do not give partial credit.
- Do not infer or assume missing requirements.
- If you are uncertain, output NO.

Prompt: <*ifeval_prompt*>

Response: <*candidate_response*>

Output exactly one word: YES or NO.

Figure 9: Prompt templates for IFEVAL.

Prompts: HumanEval Templates

Generator Template

Complete the following Python function:

<*humaneval_prompt*>
Solution:

Validator Template

Is this a correct solution to the programming problem?

Problem:
<*humaneval_prompt*>
Solution:
<*candidate_response*>

Answer Yes or No:

Figure 10: Prompt templates for HUMANEVAL.

Prompts: Hyponym Templates

Generator Template

Complete the sentence: An example of a <*hypernym*> is a

Validator Template

Answer whether the first word is a type/kind of the second word.

Is a dog a kind of animal? Answer: Yes
Is a car a kind of fruit? Answer: No
Is an apple a kind of fruit? Answer: Yes
Is a chair a kind of vehicle? Answer: No

Is a <*candidate_response*> a kind of <*hypernym*>? Answer:

Figure 11: Prompt templates for HYPONYM.

derived dataset of model-generated solutions is built on top of these problems and is used solely for non-commercial research.

IFEval. IFEval (Zhou et al., 2023) is released by Google Research under the Apache License 2.0 (see https://github.com/google-research/google-research/tree/master/instruction_following_eval and https://huggingface.co/datasets/google/IFEval). Apache 2.0 permits use, modification, and redistribution for research and other purposes, subject to the standard attribution and notice requirements, which we follow.

Hyponymy stimuli (Rosch, 1975). The category–exemplar lists we use for the Hyponymy task originate from the published norms in Rosch (1975), which appear as tables of example items within the article itself. We are not aware of an explicit data license accompanying the original paper; our use of these short lists of category exemplars for non-commercial academic research, with full citation, is consistent with fair use as commonly applied to stimulus materials reported in published psychology research. The additional category datasets used to construct disjoint training categories (Banks and Connell, 2023; Stoinski et al., 2024; Castro et al., 2021; Van Overschelde et al., 2004; Uyeda and Mandler, 1980) are likewise drawn from items reported in published academic articles or accompanying supplementary materials; THINGSplus (Stoinski et al., 2024) in particular is released under CC BY 4.0. We use only the textual category–exemplar pairs and cite each source.