GREAT EXPECTATIONS UNDER LANGUAGE MODELS WITH POTENTIALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many applications of language models hinge not on individual samples but on the expectation of a test functional under the model. Estimating such expectations reliably can be computationally expensive. In this paper, we show how to make estimation more efficient by exploiting the next-token conditional probabilities which are available as a by-product of sampling. We do so through potentials: real-valued functions on prefixes that decompose the test functional additively. We construct an estimator whose variance depends on the chosen potential, and derive conditions under which a potential reduces this variance. We then develop practical potentials for several estimands and applications, and show that for every estimand we study our estimator achieves lower variance than existing baselines at comparable computational cost.

Code Repository

1 INTRODUCTION

A language model is a probability distribution over strings. The most common application of such a model is to generate free-form text from a prompt (Radford et al., 2019; Brown et al., 2020). However, for some applications, the quantity of interest is not the sample itself but rather the expectation of a test functional under the model. For instance,

- In reinforcement learning from human feedback, proximal policy optimization (PPO; Schulman et al., 2017) regularizes the policy by penalizing its Kullback–Leibler divergence from a reference model (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022), a divergence that is itself an expectation under the policy.
- In AI safety, one is often interested in the probability that a model emits a harmful or otherwise undesirable output (Jones et al., 2025; Angell et al., 2026).
- In distillation, one fits a simpler surrogate, such as an n-gram model, to a large language model by matching expectations such as the cross-entropy under the teacher (Hinton et al., 2015; Agarwal et al., 2024).
- In language-model watermarking, one may be interested in the probability that a generated text is detected as watermarked; that can be viewed as an expectation under the model's distribution (Kirchenbauer et al., 2023).
- In interpretability, distributional functionals quantify how much a given context shifts a model's predictions (Pezeshkpour, 2023; Du et al., 2024; Stoehr et al., 2024).

More precisely, given a language model p and a test function f, we wish to compute the expectation of f under p. Unless p and f exhibit exploitable structure,[1] this expectation must be approximated. *Monte Carlo methods* offer a well known approach: sample strings, evaluate f on them, and average the results. Drawing independent strings directly from p yields the standard Monte Carlo estimator; drawing instead from a proposal distribution and reweighting by the corresponding likelihood ratio yields *importance sampling*. These estimators are unbiased but can have high variance, so achieving a desired level of precision may require generating many samples at substantial computational cost.

[1] When p is a probabilistic finite-state automaton (Rabin, 1963) or a probabilistic context-free grammar (Booth & Thompson, 1973) and f is additively decomposable, one can compute such expectations with dynamic programming (Eisner, 2002; Li & Eisner, 2009; Zmigrod et al., 2021).

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.

In this paper, we introduce a variance-reduction framework for unbiased estimators that exploits the next-token distributions computed as a by-product of sampling. The idea is to write f as an additive decomposition of token-level increments along the string and to replace each increment with its conditional expectation given the prefix, taken over the next token. Applied to an existing unbiased estimator, this procedure always preserves unbiasedness while making the variance depend on the decomposition. The choice of decomposition is thus the central design choice in the framework.

For a given f, we parametrize the family of its additive decompositions using *potentials*: real-valued functions on prefixes. We borrow this construction from potential-based reward shaping in reinforcement learning, where potentials redistribute rewards along a trajectory to facilitate learning (Ng et al., 1999). Here, we repurpose potentials to redistribute the function value over marginalizable token level increments and refer to the resulting estimator as the *potential estimator*.

We first show that there exists an oracle potential yielding a zero-variance estimator. Since it is generally intractable, we derive an exact criterion for when an arbitrary potential reduces variance. We then use the oracle potential to guide potential design: when f has known additive structure, we use components of the oracle potential that can be computed directly from f; for arbitrary functions over strings, we approximate the oracle potential by learning it from samples; given a finite set of candidate potentials, we propose a procedure that adopts the best potential only if it improves on the original estimator, while preserving unbiasedness and using in-sample information.

Our experiments demonstrate substantial gains in statistical efficiency across four applications. For losses and divergences, simple prefix potentials reduce variance in all tested configurations, yielding up to $17\times$ the effective sample size of Monte Carlo. Their gradient counterparts also improve on Monte Carlo in every evaluated comparison, with variance reductions of up to 89%. For expected n-gram counts and frequencies, marginalizing future occurrences produces even larger gains: among the targets included in the variance comparison, half attain the same variance using less than 1% of the Monte Carlo compute, even after accounting for the additional model evaluations required to compute the potential. For non-additive event probabilities, approximations to the oracle potential are also effective: an analytical potential reduces the variance of watermark-detection estimates by a median 36%, while learned potentials selected through nested cross-fitting reduce the variance of the importance-sampling estimator introduced by (Dorman et al., 2026) by a median 12%. In the latter experiment, this corresponds to approximately $900{,}000$ additional baseline generations. These results establish potential design as a practical strategy for improving expectation estimates after sampling, across both direct Monte Carlo and importance sampling. We discuss related work in App. A.

2 Expectations under Autoregressive Language Models

2.1 Language Models

Let Σ be an alphabet, a finite, non-empty set of symbols. We additionally fix a distinguished **terminal symbol** $\textsc{eos} \notin \Sigma$ and write $\Sigma \cup \{\textsc{eos}\} \stackrel{\text{def}}{=} \Sigma \cup \{\textsc{eos}\}$. A **string** is a finite sequence of symbols from Σ. Let Σ^* denote the (countably infinite) set of all such strings. For $\boldsymbol{y}, \boldsymbol{y}' \in \Sigma^*$, we write $\boldsymbol{y} \succeq \boldsymbol{y}'$ when $\boldsymbol{y}'$ is a prefix of $\boldsymbol{y}$, and $\boldsymbol{y}' \prec \boldsymbol{y}$ when $\boldsymbol{y}'$ is a strict prefix of $\boldsymbol{y}$.

A **language model** p is a probability distribution over complete strings taking values in Σ^*, i.e., $p \in \triangle(\Sigma^*)$, where $\triangle(\mathcal{X})$ denotes the set of all probability distributions over a countable set $\mathcal{X}$. We write $\boldsymbol{Y} \sim p$ for a random string drawn from p, so that $\boldsymbol{Y} \succeq \boldsymbol{y}$ is the event that $\boldsymbol{Y}$ extends $\boldsymbol{y}$. For any distribution p, we use the following notation for the **prefix probability** of a string $\boldsymbol{y} \in \Sigma^*$, which gives the marginal probability that a string sampled from p extends $\boldsymbol{y}$:

$$\overrightarrow{p}(\boldsymbol{y}) \stackrel{\text{def}}{=} \Pr[\boldsymbol{Y} \succeq \boldsymbol{y}] = \sum_{\boldsymbol{z} \in \Sigma^*} p(\boldsymbol{y}\boldsymbol{z}). \tag{1}$$

The **conditional prefix probability** is then $\overrightarrow{p}(\boldsymbol{z} \mid \boldsymbol{y}) \stackrel{\text{def}}{=} \Pr[\boldsymbol{Y} \succeq \boldsymbol{y}\,\boldsymbol{z} \mid \boldsymbol{Y} \succeq \boldsymbol{y}] = \overrightarrow{p}(\boldsymbol{y}\,\boldsymbol{z})/\overrightarrow{p}(\boldsymbol{y})$, and we extend $\overrightarrow{p}(\cdot \mid \boldsymbol{y})$ to the end-of-string symbol $\textsc{eos}$ by $\overrightarrow{p}(\textsc{eos} \mid \boldsymbol{y}) \stackrel{\text{def}}{=} p(\boldsymbol{y})/\overrightarrow{p}(\boldsymbol{y})$. Whenever $\overrightarrow{p}(\boldsymbol{y}) = 0$, we use the convention that $\overrightarrow{p}(\boldsymbol{z} \mid \boldsymbol{y}) \stackrel{\text{def}}{=} 0$ for every $\boldsymbol{z} \in \Sigma$ and $\overrightarrow{p}(\textsc{eos} \mid \boldsymbol{y}) \stackrel{\text{def}}{=} 1$, so that $\overrightarrow{p}(\cdot \mid \boldsymbol{y})$ remains a valid conditional distribution over $\Sigma \cup \{\textsc{eos}\}$. Using the conditional prefix probabilities, the probability of any string $\boldsymbol{y} = y_1 \cdots y_N \in \Sigma^*$ under a language model admits a

left-to-right autoregressive factorization:

$$p(\boldsymbol{y}) = \vec{p}(\text{EOS} \mid \boldsymbol{y}) \prod_{n=1}^{|\boldsymbol{y}|} \vec{p}(y_n \mid \boldsymbol{y}_{<n}), \tag{2}$$

where $\boldsymbol{y}_{<n} \stackrel{\text{def}}{=} y_1 \cdots y_{n-1}$. Each $\vec{p}(\cdot \mid \boldsymbol{y}_{<n})$ in Eq. (2) is a probability distribution over $\Sigma \cup \{\text{EOS}\}$.

2.2 Problem Statement

Let $p \in \triangle(\Sigma^*)$ be a language model and let $f\colon \Sigma^* \to \mathbb{R}$ be a real-valued **test function**. We study the problem of estimating

$$\theta \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{Y}\sim p}[f(\boldsymbol{Y})] = \sum_{\boldsymbol{y}\in\Sigma^*} p(\boldsymbol{y}) f(\boldsymbol{y}), \tag{3}$$

assuming throughout that θ is finite. For modern language models, this expectation is rarely available in closed form. The string space Σ^* is countably infinite, and summing exactly over all strings is generally infeasible. A standard solution is to approximate such expectations using Monte Carlo methods. We review the canonical Monte Carlo estimator and importance sampling in the remainder of this section. In the next section, we introduce potential-based estimation, an additional estimation stage that can reduce the variance of these estimators after generation by reusing the sampled strings and the model's next-token distributions retained at sampling time.

2.3 Monte Carlo Estimation

The most direct estimator of θ is an empirical average over sampled strings. We assume throughout the paper that $\mathbb{E}_p\big[f(\boldsymbol{Y})^2\big] < \infty$ and let $\boldsymbol{Y}^{(1)}, \ldots, \boldsymbol{Y}^{(M)} \overset{\text{i.i.d.}}{\sim} p$. The **Monte Carlo estimator** and its variance are

$$\widehat{\theta}_{\text{MC}} \stackrel{\text{def}}{=} \frac{1}{M}\sum_{m=1}^{M} f(\boldsymbol{Y}^{(m)}), \qquad \text{Var}_p(\widehat{\theta}_{\text{MC}}) = \frac{1}{M}\left(\mathbb{E}_p\big[f(\boldsymbol{Y})^2\big] - \theta^2\right). \tag{4}$$

Because the samples are drawn i.i.d. from p and θ is finite, $\widehat{\theta}_{\text{MC}}$ is unbiased, $\mathbb{E}[\widehat{\theta}_{\text{MC}}] = \theta$. Moreover, the strong law of large numbers guarantees that $\widehat{\theta}_{\text{MC}}$ is *strongly consistent*: it converges almost surely to θ as $M \to \infty$ (Owen, 2013; Robert & Casella, 2004).

2.4 Importance Sampling

Importance sampling estimates an expectation under p using samples from a proposal distribution q. Assume that p is absolutely continuous with respect to q, so that $q(\boldsymbol{y}) = 0$ implies $p(\boldsymbol{y}) = 0$. For $\boldsymbol{y} \in \Sigma^*$, define $f_q(\boldsymbol{y}) \stackrel{\text{def}}{=} \frac{p(\boldsymbol{y})}{q(\boldsymbol{y})} f(\boldsymbol{y})$. Then $\theta = \mathbb{E}_{\boldsymbol{Y}\sim p}[f(\boldsymbol{Y})] = \mathbb{E}_{\boldsymbol{Y}\sim q}[f_q(\boldsymbol{Y})]$.

Thus, for M independent samples $\boldsymbol{Y}^{(1)}, \ldots, \boldsymbol{Y}^{(M)} \overset{\text{i.i.d.}}{\sim} q$, the **importance-sampling estimator** is simply the Monte Carlo estimator of f_q under q, such that

$$\widehat{\theta}_{\text{IS}} \stackrel{\text{def}}{=} \frac{1}{M}\sum_{m=1}^{M} \frac{p(\boldsymbol{Y}^{(m)})}{q(\boldsymbol{Y}^{(m)})} f(\boldsymbol{Y}^{(m)}), \qquad \text{Var}_q(\widehat{\theta}_{\text{IS}}) = \frac{1}{M}\big(\mathbb{E}_q[f_q(\boldsymbol{Y})^2] - \theta^2\big). \tag{5}$$

This estimator is unbiased for every such proposal distribution q, but its variance may be lower or higher than that of $\widehat{\theta}_{\text{MC}}$ in Eq. (4) (Robert & Casella, 2004; Owen, 2013). Its variance is small when the weighted contribution $\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})} f(\boldsymbol{Y})$ varies little under q. The variance-minimizing proposal is $q^\star(\boldsymbol{y}) \stackrel{\text{def}}{\propto} p(\boldsymbol{y})|f(\boldsymbol{y})|$, which assigns probability in proportion to each string's absolute contribution to the expectation and makes the magnitude of the weighted contribution constant. Proposition 3 makes this precise.

3 Estimating Expectations with Potentials

We seek to construct a lower-variance estimator by exploiting the target model's next-token distributions along sampled strings, which are already computed when sampling from p or evaluating $p(\boldsymbol{Y})$ for importance weighting under a proposal q. For notational simplicity, we assume $q = p$ throughout this section; the general case $q \neq p$ is given in App. B.2 (Potentials under a Proposal).

3.1 Additive decompositions over prefixes: potentials

Our construction begins with a potential, a prefix-level function $\vec{\phi}\colon \Sigma^* \to \mathbb{R}$ assigning a real value to each prefix. We use its increments to define token-level contributions along each sampled string. Specifically, for the fixed test function f, we define the increment operator Δ, which maps a potential $\vec{\phi}$ to its increment function $\Delta\vec{\phi}$, given by

$$\Delta\vec{\phi}(a \mid \boldsymbol{y}) \stackrel{\text{def}}{=} \vec{\phi}(\boldsymbol{y}a) - \vec{\phi}(\boldsymbol{y}) \quad (a \in \Sigma), \qquad \Delta\vec{\phi}(\text{EOS} \mid \boldsymbol{y}) \stackrel{\text{def}}{=} f(\boldsymbol{y}) - \vec{\phi}(\boldsymbol{y}). \tag{6}$$

For notational ease, we write $\Delta\vec{\phi}(a \mid \boldsymbol{y})$ for $[\Delta\vec{\phi}](a \mid \boldsymbol{y})$, and use the same convention whenever an operator acts on a function. Using this notation, given the random string $\boldsymbol{Y} \sim p$ and its length $|\boldsymbol{Y}|$, we can rewrite $f(\boldsymbol{Y})$ as

$$f(\boldsymbol{Y}) = \vec{\phi}(\varepsilon) + \sum_{t=1}^{|\boldsymbol{Y}|} \Delta\vec{\phi}(Y_t \mid \boldsymbol{Y}_{<t}) + \Delta\vec{\phi}(\text{EOS} \mid \boldsymbol{Y}). \tag{7}$$

This representation expresses $f(\boldsymbol{Y})$ as a sum of random token-level increments $\Delta\vec{\phi}(Y_t \mid \boldsymbol{Y}_{<t})$ along the sampled string $\boldsymbol{Y}$. At each realized prefix $\boldsymbol{y}_{<t}$, we already have the next-token distribution $\vec{p}(\cdot \mid \boldsymbol{y}_{<t})$, which provides the probabilities needed to compute the next increment's conditional expectation by averaging over possible next tokens. In the next subsection, we show how such expected value can be used for variance reduction.

3.2 The Potential Estimator

The **Bellman operator** gives the expected potential after the next token:

$$\mathbf{B}\vec{\phi}(\boldsymbol{y}) \stackrel{\text{def}}{=} \sum_{a \in \Sigma} \vec{p}(a \mid \boldsymbol{y})\, \vec{\phi}(\boldsymbol{y}a) + \vec{p}(\text{EOS} \mid \boldsymbol{y})\, f(\boldsymbol{y}). \tag{8}$$

For notational ease, we write $\mathbf{B}\vec{\phi}(\boldsymbol{y})$ for $[\mathbf{B}\vec{\phi}](\boldsymbol{y})$, and use the same convention whenever an operator acts on a function. The *expected potential increment* at prefix $\boldsymbol{y}$ is then

$$\mathbf{B}\Delta\vec{\phi}(\boldsymbol{y}) \stackrel{\text{def}}{=} \mathbb{E}_{A \sim \vec{p}(\cdot \mid \boldsymbol{y})}\left[\Delta\vec{\phi}(A \mid \boldsymbol{y})\right] = \mathbf{B}\vec{\phi}(\boldsymbol{y}) - \vec{\phi}(\boldsymbol{y}). \tag{9}$$

Substituting these expectations in Eq. (7) gives the following estimator.

Definition 1 (Potential estimator)**.** *Let $\boldsymbol{Y}^{(1)}, \ldots, \boldsymbol{Y}^{(M)} \overset{\text{i.i.d.}}{\sim} p$. Given a potential $\vec{\phi}$, define*

$$\widehat{\theta}(\vec{\phi}) \stackrel{\text{def}}{=} \vec{\phi}(\varepsilon) + \frac{1}{M}\sum_{m=1}^{M} \sum_{t=1}^{|\boldsymbol{Y}^{(m)}|+1} \mathbf{B}\Delta\vec{\phi}(\boldsymbol{Y}^{(m)}_{<t}), \qquad \mathrm{Var}_p\big(\widehat{\theta}(\vec{\phi})\big) = \frac{1}{M}\mathrm{Var}_p\left(\sum_{t=1}^{|\boldsymbol{Y}|+1} \mathbf{B}\Delta\vec{\phi}(\boldsymbol{Y}_{<t})\right). \tag{10}$$

The inner sum includes the final prefix $\boldsymbol{Y}^{(m)}$, at which EOS *is drawn.*

The estimator is unbiased for any potential $\vec{\phi}$ (Proposition 5), but its variance depends on $\vec{\phi}$. The next proposition identifies the oracle potential, which drives the variance to zero, and characterizes exactly when an arbitrary potential improves on an unbiased estimator baseline.

Proposition 1 (Oracle potential and variance comparison)**.** *The **oracle potential**, defined on prefixes of positive probability, satisfies*

$$\vec{\phi}^{\star}(\boldsymbol{y}) = \mathbb{E}_p[f(\boldsymbol{Y}) \mid \boldsymbol{Y} \succeq \boldsymbol{y}] \quad \Longrightarrow \quad \widehat{\theta}(\vec{\phi}^{\star}) = \theta, \qquad \mathrm{Var}_p\Big(\widehat{\theta}(\vec{\phi}^{\star})\Big) = 0.$$

Let $\widehat{\theta}_*$ be any unbiased baseline estimator under p. For every fixed potential $\vec{\phi}$,

$$\mathrm{Var}_p\big(\widehat{\theta}(\vec{\phi})\big) \leq \mathrm{Var}_p(\widehat{\theta}_*) \iff \mathrm{Cov}_p\big(\widehat{\theta}_*, \widehat{\theta}_* - \widehat{\theta}(\vec{\phi})\big) \geq \frac{1}{2}\mathrm{Var}_p\big(\widehat{\theta}_* - \widehat{\theta}(\vec{\phi})\big).$$

Despite the above condition need not hold for a chosen potential, we can exploit the unbiasdness of $\widehat{\theta}_*$ and $\widehat{\theta}(\vec{\phi})$ to build a control-variate estimator with variance equal or lower than $\widehat{\theta}_*$.

Definition 2 (Potential control-variate estimator)**.** *Given a potential $\vec{\phi}$ and an unbiased baseline estimator $\widehat{\theta}_*$, define*

$$\widehat{\theta}(\vec{\phi}, \alpha) \stackrel{\text{def}}{=} \widehat{\theta}_* - \alpha\big(\widehat{\theta}_* - \widehat{\theta}(\vec{\phi})\big), \qquad \alpha \in \mathbb{R}. \tag{11}$$

The coefficient α controls the contribution of the correction. Setting $\alpha = 0$ recovers the baseline estimator, while $\alpha = 1$ recovers the potential estimator. The following result determines the choice that minimizes variance.

Proposition 2 (Unbiasedness and optimal coefficient)**.** *For every fixed potential $\vec{\phi}$ and fixed coefficient α, the estimator is unbiased: $\mathbb{E}_p[\widehat{\theta}(\vec{\phi}, \alpha)] = \theta$. If both $\widehat{\theta}_*$ and $\widehat{\theta}(\vec{\phi})$ have finite second moments and $\mathrm{Var}_p\big(\widehat{\theta}_* - \widehat{\theta}(\vec{\phi})\big) > 0$, the variance-minimizing coefficient satisfies*

$$\alpha^\star = \frac{\mathrm{Cov}_p\big(\widehat{\theta}_*, \widehat{\theta}_* - \widehat{\theta}(\vec{\phi})\big)}{\mathrm{Var}_p\big(\widehat{\theta}_* - \widehat{\theta}(\vec{\phi})\big)} \quad \Longrightarrow \quad \mathrm{Var}_p\big(\widehat{\theta}(\vec{\phi}, \alpha^\star)\big) \le \mathrm{Var}_p(\widehat{\theta}_*). \tag{12}$$

If the difference has zero variance, the two estimators agree almost surely, and the choice of coefficient has no effect.

In the next section, we describe how to design and select potentials in practice, and how to estimate estimator parameters (e.g., α^*) from data, without using additional samples nor sacrificing unbiasedness.

4 Potential Design in Practice: Applications and Experiments

In this section, we show how to construct potentials across several estimands and applications, and ask whether the resulting potential estimators achieve lower variance at comparable computational cost. Depending on the structure of f, we follow one of two complementary approaches: we either exploit known additive structure to derive a potential analytically, or learn a potential from data. We present three applications in the main text and a fourth in App. C.1 (Analysis of Watermark Detection).

4.1 Estimating Proper Losses, Divergences, and Gradients

We estimate seven losses frequently used in language model training and evaluation along with their gradients: entropy (Dai et al., 2018; Braverman et al., 2020), KL divergence (Schulman et al., 2017; Ouyang et al., 2022), Brier loss and power-score regrets (Shao et al., 2024) with exponents $\beta \in \{1.25, 1.5, 1.75, 2\}$.

Table 1: Entropy

$f(\boldsymbol{Y}) = -\log p(\boldsymbol{Y})$
$f(\boldsymbol{x}\boldsymbol{z}) = -\log \vec{p}(\boldsymbol{x}) - \log \vec{p}(\boldsymbol{z} \mid \boldsymbol{x}) \quad \forall\, \boldsymbol{x}, \boldsymbol{z} \in \Sigma^*$
$\vec{\phi}^\star(\boldsymbol{x}) = -\log \vec{p}(\boldsymbol{x}) + \mathbb{E}_p[-\log \vec{p}(\boldsymbol{Z} \mid \boldsymbol{x}) \mid \boldsymbol{Z} \succeq \boldsymbol{x}]$
$\vec{\phi}(\boldsymbol{x}) = -\log \vec{p}(\boldsymbol{x})$
$\Delta\vec{\phi}(a \mid \boldsymbol{x}) = -\log \vec{p}(a \mid \boldsymbol{x})$
$\Delta\vec{\phi}(\textsc{eos} \mid \boldsymbol{x}) = -\log \vec{p}(\textsc{eos} \mid \boldsymbol{x})$

Defining a potential analytically using the function value at the observed prefix.
The above losses and their gradients have a natural additive structure over prefixes; for any such function, the value of the function at each observed prefix yields a potential. Tab. 1 illustrates this for entropy. By the chain rule, $-\log p(\boldsymbol{Y})$ splits into a prefix term $-\log \vec{p}(\boldsymbol{x})$ and a continuation term $-\log \vec{p}(\boldsymbol{z} \mid \boldsymbol{x})$. The oracle potential adds the expected continuation term to the prefix term; keeping only the prefix term thus approximates the oracle potential by its observed component. Here the increments are the per-token surprisals $-\log \vec{p}(a \mid \boldsymbol{x})$ (with a final increment at EOS), which are computed during sampling anyway, so the potential is free. We define the potentials for the other quantities and their gradients in the appendix: App. C.2.1 gives those for KL divergence, the Brier loss and the power-score regrets (Tab. 7 to 9), and App. C.3.1 those for the gradients (Tab. 14).

For KL, our potential estimator recovers that of Amini et al. (2025), whose experiment we reproduce in App. C.2.4 (Reproducing Amini et al. (2025)). We nevertheless include KL in our experiments because their claimed variance ordering with Monte Carlo does not hold in general, as Proposition 1

shows and App. C.2.4 demonstrates with a counterexample. For all quantities and their gradients, we therefore examine how consistently the estimator reduces variance across settings and whether fitting the control-variate coefficient yields further gains.

4.1.1 Experiments and results

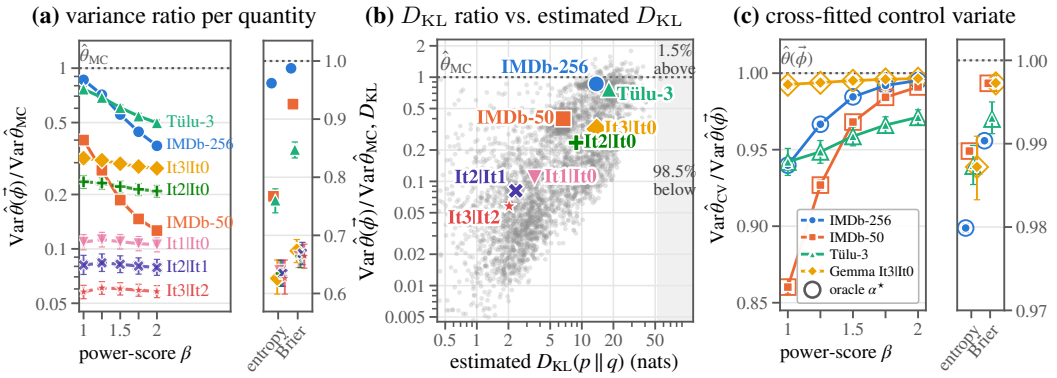


Figure 1: Variance reduction in the proper-loss experiment (lower is better). **(a)** Variance ratio of the potential estimator to Monte Carlo across power scores ($\beta \to 1$ gives D_{KL}); entropy and Brier are shown on the right. Each series is one policy–reference pair (see Tab. 10: solid lines are the four main policy–reference pairs and dashed lines are consecutive Gemma-2 SPPO pairs. **(b)** For D_{KL}, variance ratio against the estimated $D_{KL}(p\|q)$, with one colored marker per pair and grey markers for individual prompts from all pairs. The shaded regions indicate prompts below and above the Monte Carlo variance. **(c)** Variance ratio of the cross-fitted control-variate estimator to the potential estimator; hollow markers use the full-sample oracle coefficient.

The potential estimator reduces variance in every configuration, by up to $17\times$ in effective sample size. We evaluate the seven quantities above on eight configurations from three post-training families (full details in App. C.2.2, Experimental setup). Across all 56 pair–quantity combinations, the potential reduces variance relative to Monte Carlo (Fig. 1a), with ratios between 0.058 and 0.987 and every bootstrap upper bound below one. These correspond to effective sample-size multipliers of $1.01\times$–$17.3\times$.

The gains are largest when the policy and reference are close. The variation across policy–reference pairs is larger than across quantities. For KL at horizon 256, the variance ratio is 0.86 on GPT-2 IMDb, 0.77 on Tülu-3, and 0.32 on Gemma-2, dropping to 0.06–0.11 between consecutive Gemma SPPO iterations (equivalent to $9.1\times$–$16.7\times$ the Monte Carlo sample size). The same pattern appears across prompts: smaller KL is associated with larger variance reductions (Fig. 1b).

Variance reduction holds for the vast majority of prompts, but not all of them: Monte Carlo has lower variance on 1.5% of all 4,096 prompts, including 12.3% of Tülu-3 prompts (95% interval 9.6%–15.2%). This provides empirical evidence that the general KL variance ordering of Amini et al. (2025) does not hold universally; Fitting the control-variate coefficient yields ratios of 0.86–1.00 relative to the untuned potential estimator (Fig. 1c); among KL prompts where the potential estimator increases variance, fitting α lowers the median variance ratio relative to Monte Carlo from 1.134 to 0.974.

App. C.3 repeats the comparison for the gradient of each quantity with respect to the policy parameters along a DPO trajectory of the GPT-2 sentiment policy: the potential removes most of the gradient variance early in training, and the fitted coefficient rescues the quantities the potential alone barely helps.

4.2 Estimating Expected Counts and Frequencies of n-grams

We study expected counts and frequencies of word n-grams, i.e., contiguous sequences of n words. These statistics arise in applications such as model distillation (Suresh et al., 2021; Krishnan et al., 2023) and computational psycholinguistics (**?**Kiegeland et al., 2026). More broadly, n-gram counts are one instance of a general class of functions $f(\boldsymbol{Y}) = \#\{\text{occurrences of substring } \boldsymbol{w} \text{ in } \boldsymbol{Y}\}$, encompassing quantities such as the number of forbidden keywords in a string, unsafe API calls, or other keyword-based patterns emitted by a model.

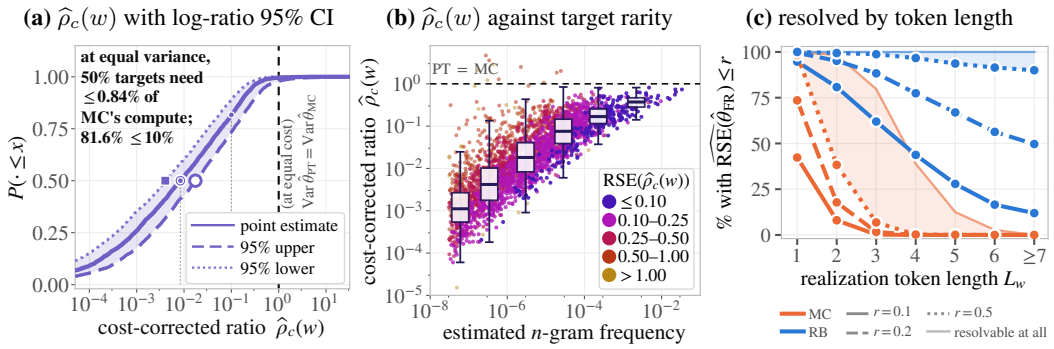


Figure 2: Cost-adjusted variance ratio $\widehat{\rho}_c(w)$ for n-gram frequency estimation; $\widehat{\rho}_c(w) < 1$ means that the potential estimator requires less compute than Monte Carlo to attain the same variance. **(a)** Empirical cumulative distribution of $\widehat{\rho}_c(w)$ across targets, together with 95% confidence bounds. **(b)** Ratio versus estimated n-gram frequency, illustrating how variance reduction changes with target rarity. **(c)** Fraction of targets whose frequency estimate reaches a given relative-standard-error threshold, grouped by realization token length, for Monte Carlo and the potential estimator. Dashed reference lines indicate equal performance at equal compute.

Designing potentials that marginalize over multiple future tokens. We start by observing that, for any target substring $\boldsymbol{w} \in \Sigma^*$, the count function f admits two natural decompositions:

$$f(\boldsymbol{Y}) = \sum_t Z^{\text{end}}_{\boldsymbol{w},t} = \sum_t Z^{\text{start}}_{\boldsymbol{w},t}, \qquad Z^{\text{end}}_{\boldsymbol{w},t} = \mathbf{1}\{\boldsymbol{w} \text{ ends at } t\}, \qquad Z^{\text{start}}_{\boldsymbol{w},t} = \mathbf{1}\{\boldsymbol{w} \text{ starts at } t\}.$$

The $Z^{\text{end}}_{\boldsymbol{w}}$ decomposition yields a potential that is completely determined by evaluating f at prefix x. Its expected increment, however, is nonzero only at prefixes that already match the substring up to its final token, so marginalization is useful only at those few prefixes.

TABLE 2: SUBSTRING COUNTS, LOOKAHEAD OCCURRENCE POTENTIAL

$f(\boldsymbol{Y}) = \#\{\text{occurrences of } \boldsymbol{w} \text{ in } \boldsymbol{Y}\}$
$f(\boldsymbol{Y}) = \sum_t Z^{\text{start}}_{\boldsymbol{w},t},\ Z^{\text{start}}_{\boldsymbol{w},t} = Z^{\text{can}}_{\boldsymbol{w},t} + Z^{\text{noncan}}_{\boldsymbol{w},t}$
$\vec{\phi}^{\star}(\boldsymbol{x}) = \sum_{t \le \lvert\boldsymbol{Y}\rvert} \mathbb{E}_p\big[Z^{\text{start}}_{\boldsymbol{w},t} \mid \boldsymbol{Y} \succeq \boldsymbol{x}\big]$
$\vec{\phi}(\boldsymbol{x}) = \sum_{t \le \lvert\boldsymbol{x}\rvert} \mathbb{E}_p\big[Z^{\text{can}}_{\boldsymbol{w},t} \mid \boldsymbol{Y}_{\le t} = \boldsymbol{x}_{\le t}\big]$
$\Delta\vec{\phi}(a \mid \boldsymbol{x}) = \mathbb{E}_p\big[Z^{\text{can}}_{\boldsymbol{w},t} \mid \boldsymbol{Y}_{\le t} = \boldsymbol{x}a\big],\ t = \lvert\boldsymbol{x}\rvert + 1$
$\Delta\vec{\phi}(\text{EOS} \mid \boldsymbol{x}) = \sum_{t \le \lvert\boldsymbol{x}\rvert} \big(Z^{\text{can}}_{\boldsymbol{w},t} - \mathbb{E}_p[Z^{\text{can}}_{\boldsymbol{w},t} \mid \boldsymbol{Y}_{\le t}]\big) + \sum_t Z^{\text{noncan}}_{\boldsymbol{w},t}$

By contrast, $Z^{\text{start}}_{\boldsymbol{w}}$ is generally still unobserved at the current prefix. We therefore replace it by its full conditional expectation—the probability that the occurrence starting at that position is eventually completed—and take the cumulative expected count over starting positions as the potential. In practice, we marginalize over the canonical tokenization $\boldsymbol{w} = (w_1, \dots, w_L)$, with $w_i \in \Sigma$. At each sampled prefix, this requires L additional token evaluations: the remaining target tokens after the next token, plus the word boundary. Thus, if C denotes the cost of generating one sample from model p, the additional marginalization cost is $O(LC)$.

Tab. 2 gives the mathematical definition of the potential using $Z^{\text{start}}_{\boldsymbol{w}}$ and its increments. This example illustrates the flexibility of the telescoping construction in Eq. (7): the potential may marginalize only a selected component of f—here, occurrences under the canonical tokenization—while the terminal increment recovers the remainder, so the decomposition still equals f exactly at termination.[2]

4.2.1 EXPERIMENTS & RESULTS

Setup. We construct 7,647 word n-gram targets ($n = 1, \dots, 6$) from the English texts of MECO (Siegelman et al., 2022), with canonical tokenization lengths ranging from 1 to 10 tokens. We ask whether the potential estimator that uses the potential defined in Tab. 2 reduces variance at comparable computational cost with MC, and how the gains vary with target rarity;

[2] In this application, part of the terminal increment has expectation zero under p and can therefore be dropped without changing the estimator's expectation, yielding $\widetilde{\Delta}\vec{\phi}(\text{EOS} \mid \boldsymbol{x}) = \sum_t Z^{\text{noncan}}_{\boldsymbol{w},t}$.

For each target, we estimate its expected count and frequency (the ratio estimator of Eq. (28), App. C.4.1) with the potential estimator using 4,000 samples from GPT-2, and we estimate the Monte Carlo variance from an independent 4×10^6-sample bank drawn from the same model. We report the cost-adjusted variance ratio $\widehat{\rho}_c(w)$ (Eq. (29), App. C.4.2), obtained by scaling the potential estimator's variance by the target's canonical token length, to account for its $O(L_w C)$ marginalization cost, and by rescaling the Monte Carlo variance to 4,000 samples. Thus, $\widehat{\rho}_c(w) < 1$ indicates that the potential estimator has lower variance than Monte Carlo at equal model-evaluation cost. To characterize target rarity independently of the variance comparison, we further estimate each target's frequency from a separate 2×10^6-sample bank. We report results for frequencies here, since they are the more interpretable quantity, and defer results for counts to App. C.4; the two agree throughout. The appendix also reports variance results for the simpler observed-prefix potential (defined in §4.1), which yields virtually no variance reduction.

Half the targets reach equal variance on less than 1% of the Monte Carlo compute. Panel Fig. 2a shows the cost-adjusted variance ratio $\widehat{\rho}_c$ for the 3,700 targets whose ratio can be estimated with relative standard error at most 2 (estimated via bootstrapping), together with a per-target 95% bootstrap confidence interval. Interpreting $\widehat{\rho}_c$ as the fraction of Monte Carlo compute required to attain the same variance, half of the targets require at most 0.84% of the Monte Carlo budget, and 81.6% require at most 10%. These gains remain large after accounting for estimation uncertainty: replacing each ratio by the upper bound of its 95% interval, half of the targets still require at most 1.75% of the Monte Carlo budget, 83.4% require at most 20%, and 97.8% remain below parity with Monte Carlo.

The gains are largest for rare targets. We restrict to targets whose frequency estimates have nominal RSE at most 20% under a Poisson approximation, equivalently at least 25 expected occurrences in the independent 2×10^6-sample frequency bank, leaving 2,831 targets. The gains are largest for rare targets: the compute ratio increases strongly with frequency (Spearman $r_s = 0.83$), with its median rising from 1.1×10^{-3} in the rarest frequency decade to 0.37 among the most frequent targets. This trend persists when stratifying by n-gram order and token length (Tab. 17).

At the same sample size ($M = 4{,}000$), the potential estimator yields substantially more precise frequency estimates, with 79.7% of targets reaching RSE $\leq 20\%$ and 96.5% reaching RSE $\leq 50\%$, compared with 11.4% and 19.0% for Monte Carlo. The gap grows with token length: for targets of length 4 or more, Monte Carlo almost never reaches RSE $\leq 50\%$, whereas the potential estimator does so for at least 90% of targets in every length group. For $L_w \geq 7$, Monte Carlo does not observe any target even in the 4×10^6-sample reference bank, while the potential estimator gives a finite estimate for all of them.

4.3 Event analysis with language models

Dorman et al. (2026) estimate event probabilities $\Pr_p[\mathrm{ARI}(\boldsymbol{Y}) \in D_j]$, where $\{D_j\}_{j=1}^J$ partitions the range of the Automated Readability Index (ARI; Eq. (30)), a sequence-level readability score. To make rare outcomes observable, they construct an importance-sampling estimator: they sample from distributions tilted toward different values of the ARI and reweight the resulting trajectories to estimate probabilities under the original language model. We augment this estimator with potentials for the indicator targets $f_j(\boldsymbol{Y}) = \mathbf{1}\{\mathrm{ARI}(\boldsymbol{Y}) \in D_j\}$.

Learning potentials from samples. We note that these indicators need not admit a useful additive decomposition over prefixes. A simple baseline is the *terminal potential*, which assigns zero to unfinished prefixes and the indicator value $f_j(\boldsymbol{Y})$ at the string termination. To exploit information from earlier prefixes, we instead learn an approximation to the oracle potential: the conditional probability of the $f_j(\boldsymbol{Y})$ given the observed prefix. Each sampled prefix $\boldsymbol{x}_i$, $i = 1, \ldots, B$, is paired with its trajectory's outcome $f_j(\boldsymbol{y}_i)$, $\boldsymbol{y}_i \succeq \boldsymbol{x}_i$. Using prefix features $\boldsymbol{\xi}(\boldsymbol{x}) \in \mathbb{R}^d$, we fit a predictor[3] $g_j(\cdot;\boldsymbol{\eta})\colon \mathbb{R}^d \to [0,1]$ with parameters $\boldsymbol{\eta} \in \mathbb{R}^m$. We can define the potential

$$\vec{\phi}_j(\boldsymbol{x};\boldsymbol{\eta}) = g_j(\boldsymbol{\xi}(\boldsymbol{x});\boldsymbol{\eta}) \approx \vec{\phi}_j^{\star}(\boldsymbol{x}) = \Pr_p[\mathrm{ARI}(\boldsymbol{Y}) \in D_j \mid \boldsymbol{Y} \succeq \boldsymbol{x}]. \tag{13}$$

[3]When trajectories are obtained through importance sampling as in this application, we weight each prefix's prediction loss by its trajectory's importance weight, so that fitting targets conditional event probabilities under the original language model.

Unbiased estimation with potentials learned in-sample. Because a learned potential is fitted using the sampled outcomes, fitting and applying it to the same sample can introduce bias. One simple way to avoid this is to use K-fold cross-fitting (Chernozhukov et al., 2018). For example, when $K = 2$, we split the sample into two folds, fit the potential on the first fold and apply it to the second, then fit it on the second fold and apply it to the first. Averaging the two fold-specific estimates preserves unbiasedness while using the full sample.

Potential selection with nested cross-fitting. In practice, we may have several potential models that we want to try, for example one fitted via logistic regression and one via a mixture model, and wish to use whichever reduces variance the most. A naive approach would be to cross-fit both candidates as above, compare their performance on the full sample, and then choose the better one. This final selection step can itself introduce bias, however, because one would use the same sample to both choose the potential and to form the final estimate (Varma & Simon, 2006). To avoid this, we use *nested cross-fitting* (Cawley & Talbot, 2010): for each held-out fold, we select the candidate potential using only the remaining data, through an additional inner cross-validation layer, and then apply the selected potential to the held-out fold. We formalize this procedure in Alg. 1 in the appendix.

4.3.1 Experiments & Results

Setup. We replicate the main experiment of Dorman et al. (2026), estimating the distribution of the Automated Readability Index (ARI) over texts generated by TinyStories-8M. During sampling, we additionally retain the next-token distributions for subsequent potential fitting and evaluation. We consider three candidate potentials: the terminal potential, which marginalizes over the final token, and two learned potentials based on a simple logistic regression and a mixture model, respectively. We defer their details to App. C.5.2 (Potential definitions). We also include a control-variate variant of the terminal potential with a learned coefficient α, giving four candidates, among which we select using nested cross-fitting with $K = 5$ outer folds. Throughout, we refer to the original estimator of Dorman et al. (2026) by the acronym MBAR.

Learned potentials reduce variance by $\sim$12%, equivalent to $\sim 0.9 \times 10^6$ additional MBAR samples.

Tab. 3 summarizes the variance ratio of each potential estimator relative to MBAR; lower values indicate a more efficient estimator. The learned potentials give the largest reductions: the logistic and mixture-model potentials attain median variance ratios of 0.90 and 0.87, respectively, and nested selection attains 0.88, closely matching the best individual candidate. The terminal potential gives a smaller but still consistent reduction, with a median ratio of 0.92. It exploits cases where membership in a bin D_j is determined only at the final token.

Estimator	Var. ratio vs. MBAR [95% CI]
Fitted selection	0.88 [0.86, 0.91]
Terminal	0.92 [0.90, 0.94]
Terminal, α	0.92 [0.90, 0.95]
Logistic	0.90 [0.86, 0.91]
Mixture model	0.87 [0.84, 0.90]

Table 3: variance relative to MBAR, median over the 64 populated ARI bins; 95% intervals from 1,000 paired bootstrap replicas. Full version in Tab. 19.

The experiment uses $N = 6.68 \times 10^6$ generations, matching the original experiments. Assuming variance scales inversely with sample size, the median ratio of 0.88 corresponds to an MBAR budget requiring 0.9×10^6 more generations than our potential estimator. We used only CPUs for potential estimation, as detailed in App. C.5.4 (Compute costs and additional results).

5 Conclusion

We introduced a general framework for reducing the variance of expectation estimates under autoregressive language models using next-token distributions available as a by-product of sampling. By repurposing potentials from reinforcement learning, we construct an unbiased estimator whose variance depends on the chosen potential. Across several estimands and applications, we showed how practical potentials can improve statistical efficiency at comparable computational cost. More broadly, our results establish potential design as a general strategy for constructing lower-variance estimators under language models.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experimental design, assist with the interpretation of experimental results, and help check and refine mathematical statements and derivations. Additionally, we used generative AI tools to assist with software development, and drafting and editing portions of the manuscript for clarity, structure, and readability. AI-assisted text and code were reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Afra Amini, Tim Vieira, and Ryan Cotterell. Better estimation of the Kullback–Leibler divergence between language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a2e9be3d64bcc196731d0a4e0e2c33ae-Abstract-Conference.html.

Rico Angell, Raghav Singhal, Zachary Horvitz, Zhou Yu, Rajesh Ranganath, Kathleen McKeown, and He He. Estimating tail risks in language model output distributions. *arXiv preprint arXiv:2604.22167*, 2026.

Taylor L Booth and Richard A Thompson. Applying probability measures to abstract languages. *IEEE transactions on Computers*, 100(5):442–450, 1973.

Alex Boyd, Samuel Showalter, Stephan Mandt, and Padhraic Smyth. Predictive querying for autoregressive neural sequence models. *Advances in Neural Information Processing Systems*, 35: 23751–23764, 2022.

Mark Braverman, Xinyi Chen, Sham Kakade, Karthik Narasimhan, Cyril Zhang, and Yi Zhang. Calibration, entropy rates, and memory in language models. In *International Conference on Machine Learning*, pp. 1089–1099. PMLR, 2020.

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

Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, 11:2079–2107, 2010.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters, 2018.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.

William G. Cochran. *Sampling Techniques*. John Wiley & Sons, New York, 3 edition, 1977.

Zihang Dai, Qizhe Xie, and Eduard Hovy. From credit assignment to entropy regularization: Two new algorithms for neural sequence prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1672–1682, 2018.

Pierre Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. 2004. URL https://doi.org/10.1007/978-1-4684-9393-1.

Jake McAllister Dorman, Edward Gillman, Dominic C Rose, Jamie F Mair, and Juan P Garrahan. Rare event analysis of large language models. *arXiv preprint arXiv:2602.06791*, 2026.

Kevin Du, Vésteinn Snæbjarnarson, Niklas Stoehr, Jennifer White, Aaron Schein, and Ryan Cotterell. Context versus prior knowledge in language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13211–13235, 2024.

Jason Eisner. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 1–8, 2002.

J. M. Hammersley. Conditional Monte Carlo. *Journal of the ACM*, 3(2), 1956. URL https://doi.org/10.1145/320825.320827.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Erik Jones, Meg Tong, Jesse Mu, Mohammed Mahfoud, Jan Leike, Roger Grosse, Jared Kaplan, William Fithian, Ethan Perez, and Mrinank Sharma. Forecasting rare language model behaviors. *arXiv preprint arXiv:2502.16797*, 2025.

Samuel Kiegeland, Vésteinn Snæbjarnarson, Tim Vieira, and Ryan Cotterell. On the proper treatment of units in surprisal theory. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32202–32224, 2026.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International conference on machine learning*, pp. 17061–17084. PMLR, 2023.

Aravind Krishnan, Jesujoba Alabi, and Dietrich Klakow. On the n-gram approximation of pre-trained language models. *arXiv preprint arXiv:2306.06892*, 2023.

Alexander K. Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K. Mansinghka. Sequential Monte Carlo steering of large language models using probabilistic programs. In *ICML Workshop on Sampling and Optimization in Discrete Space (SODS)*, 2023. URL https://openreview.net/forum?id=Ul2K0qXxXy.

Zhifei Li and Jason Eisner. First-and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 40–51, 2009.

Zixuan Liu, Fangzheng Wu, Brian Summa, and Zizhan Zheng. Adaptive multilevel twisted sequential Monte Carlo for rare events estimation in language models, 2026. URL https://arxiv.org/abs/2608.21736.

João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara, Marjorie Freedman, Jason Eisner, Ryan Cotterell, Vikash Mansinghka, Alexander K. Lew, Tim Vieira, and Timothy J. O'Donnell. Syntactic and semantic control of large language models via sequential Monte Carlo. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=xoXn62FzD0.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

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.

Art B. Owen. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/, 2013.

Pouya Pezeshkpour. Measuring and modifying factual knowledge in large language models. In *2023 international conference on machine learning and applications (ICMLA)*, pp. 831–838. IEEE, 2023.

Michael O Rabin. Probabilistic automata. *Information and control*, 6(3):230–245, 1963.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. 2nd edition, 2004. URL https://doi.org/10.1007/978-1-4757-4145-2.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Chenze Shao, Fandong Meng, Yijin Liu, and Jie Zhou. Language generation with strictly proper scoring rules. *arXiv preprint arXiv:2405.18906*, 2024.

Noam Siegelman, Sascha Schroeder, Cengiz Acartürk, Hee-Don Ahn, Svetlana Alexeeva, Simona Amenta, Raymond Bertram, Rolando Bonandrini, Marc Brysbaert, Daria Chernova, et al. Expanding horizons of cross-linguistic research on reading: The multilingual eye-movement corpus (meco). *Behavior research methods*, 54(6):2843–2863, 2022.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Niklas Stoehr, Kevin Du, Vésteinn Snæbjarnarson, Robert West, Ryan Cotterell, and Aaron Schein. Activation scaling for steering and interpreting language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 8189–8200, 2024.

Ananda Theertha Suresh, Brian Roark, Michael Riley, and Vlad Schogol. Approximating probabilistic models as weighted finite automata. *Computational Linguistics*, 47(2):221–254, 2021.

Sudhir Varma and Richard Simon. Bias in error estimation when using cross-validation for model selection. *BMC bioinformatics*, 7(1):91, 2006.

Nick Whiteley and Anthony Lee. Twisted particle filters. *The Annals of Statistics*, 42(1), 2014. URL https://doi.org/10.1214/13-AOS1167.

Eric Wiewiora. Potential-based shaping and Q-Value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19, 2003. URL https://doi.org/10.1613/jair.1190.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Baker Grosse. Probabilistic inference in language models via twisted sequential Monte Carlo. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024. URL https://proceedings.mlr.press/v235/zhao24c.html.

Ran Zmigrod, Tim Vieira, and Ryan Cotterell. Efficient computation of expectations under spanning tree distributions. *Transactions of the Association for Computational Linguistics*, 9:675–690, 2021.

Appendix Contents

A Related Work **14**

B Additional Propositions and Proofs **15**

B.1 Importance Sampling and the Optimal Proposal 15

B.2 Potentials under a Proposal 15

B.3 Unbiasedness of the Potential Estimator 17

B.4 Proof of Proposition 1 17

B.5 Proof of Proposition 2 18

C Additional Analysis and Experimental Details **18**

C.1 Analysis of Watermark Detection 18

C.1.1 Estimands and potential definitions 18

C.1.2 Experiments and results 19

C.2 Losses and Divergences: Details and Further Results 20

C.2.1 Potential definitions 20

C.2.2 Experimental setup 21

C.2.3 Additional results 22

C.2.4 Reproducing Amini et al. (2025) 23

C.3 Gradients of Losses and Divergences 24

C.3.1 Potential definitions 24

C.3.2 Control-variate coefficient for vector-valued functions 25

C.3.3 Experiments and results 25

C.4 n-gram Counts and Frequencies 26

C.4.1 Definition of frequency estimator 26

C.4.2 Definition of cost-adjusted variance ratio 26

C.4.3 Observed-prefix potential 27

C.4.4 Results for expected counts 27

C.4.5 Additional diagnostics 28

C.5 Event Analysis of (Dorman et al., 2026) 30

C.5.1 The Automated Readability Index 30

C.5.2 Potential definitions 30

C.5.3 Potential selection algorithm 31

C.5.4 Compute costs and additional results 32

LIMITATIONS

Our framework has several limitations. First, variance reduction depends on the choice of potential. Although the oracle potential yields zero variance under direct sampling, practical potentials only approximate it and are not guaranteed to improve over the baseline for every individual prompt. The control-variate formulation provides a safeguard against poorly chosen potentials, although estimating its coefficient or selecting among learned potentials requires cross-fitting to preserve unbiasedness. Thus, the strong performance of the potentials studied here should not be interpreted as implying that arbitrary potentials will yield similar gains.

Second, the computational advantage depends on the cost of evaluating the potential. In several of our applications, the required quantities are already available during sampling or can be computed relatively cheaply, whereas more elaborate potentials may require additional model evaluations or fitting. At the same time, this tradeoff suggests a natural direction for future work: designing potentials that achieve stronger variance reduction while remaining computationally inexpensive.

Third, our approach requires access to the model's next-token probability distribution along sampled prefixes. It therefore applies directly when logits or token probabilities are available, but not to black-box generation APIs that expose only sampled outputs. Finally, our empirical conclusions are necessarily limited to the estimands, models, and experimental configurations considered here; the magnitude of the variance reductions may differ in other settings.

A RELATED WORK

Importance sampling and sequential Monte Carlo for language models. Several methods reduce variance by modifying the sampling distribution to favor events of interest and correcting through importance weighting (Angell et al., 2026; Dorman et al., 2026). For example, Boyd et al. (2022) combine query-specific importance sampling with beam search to estimate probabilities of sequence events. Sequential Monte Carlo (Del Moral, 2004) and its twisted variants (Whiteley & Lee, 2014) have also been applied to controlled generation (Lew et al., 2023; Zhao et al., 2024; Loula et al., 2025) and rare-event estimation (Liu et al., 2026). In such methods, a *twist* assigns a nonnegative weight to each prefix, guiding sampling and resampling toward prefixes judged more likely to produce the desired outcome. For event-probability estimation, the optimal twist weights each prefix according to the probability that completing it under the original language model will satisfy the event. It is therefore proportional to our oracle potential $\vec{\phi}^{\star}$, which represents the same conditional probability (Zhao et al., 2024). Our framework uses prefix potentials to construct additive, zero-mean corrections that enable variance reduction after sampling, exploiting the next-token distribution created at generation. We demonstrate its complementarity with importance sampling by augmenting the importance sampling estimator of Dorman et al. (2026), who sample from exponentially tilted sequence distributions using Markov chain Monte Carlo.

Conditional Monte Carlo, Rao–Blackwellization, and control variates. Replacing a finite-variance estimator by its conditional expectation cannot increase its variance, the principle underlying conditional Monte Carlo (Hammersley, 1956) and Rao–Blackwellization (Robert & Casella, 2004; Owen, 2013). For KL estimation under language models, Amini et al. (2025) replace each sampled token log-likelihood ratio by the exact KL divergence between the next-token distributions at the sampled prefix, and derive an analogous gradient estimator. Although their KL estimator is unbiased, the variance-reduction guarantee in their Theorem 2 does not hold in general: averaging individual token contributions can alter their covariances and increase the variance of the sum. Our framework recovers their KL estimator as a particular choice of potential and extends the construction to arbitrary test functions, explicitly distinguishing unbiasedness from variance reduction. The difference between a potential estimator and any unbiased baseline defines a zero-mean control variate (Owen, 2013). This correction preserves unbiasedness for any fixed coefficient, while the population-optimal coefficient yields variance no greater than that of the baseline.

Potentials and reward shaping in reinforcement learning. Potential-based reward shaping in reinforcement learning uses changes in a function of the state to redistribute rewards while preserving optimal policies (Ng et al., 1999; Wiewiora, 2003). Value functions, which predict future rewards, provide natural potentials: the resulting shaped rewards measure prediction errors and are used in

generalized advantage estimation (Schulman et al., 2015). Our construction uses the same telescoping principle, with text prefixes as states and the test function's value as the terminal reward. We then average the potential increments over the next-token distribution to construct unbiased expectation estimates after sampling. The ideal potential predicts the expected terminal reward given each prefix; its averaged increments vanish, yielding our zero-variance oracle (Proposition 1).

B Additional Propositions and Proofs

Throughout, $\boldsymbol{Y} \sim p$ denotes a single sampled string, $|\boldsymbol{Y}|$ its length excluding EOS, and $\boldsymbol{Y}_{<t}$ its prefix of length $t-1$, so that $\boldsymbol{Y}_{<1} = \varepsilon$ and $\boldsymbol{Y}_{<|\boldsymbol{Y}|+1} = \boldsymbol{Y}$. All expectations are assumed finite.

B.1 Importance Sampling and the Optimal Proposal

Proposition 3 (Importance-sampling variance and optimal proposal)**.** *Let q be a proposal with respect to which p is absolutely continuous, and let $\widehat{\theta}_{\mathrm{IS}}$ be the importance-sampling estimator of Eq. (5). Then $\mathbb{E}_q[\widehat{\theta}_{\mathrm{IS}}] = \theta$ and*

$$\mathrm{Var}_q(\widehat{\theta}_{\mathrm{IS}}) = \frac{1}{M}\left(\sum_{\boldsymbol{y}\in\Sigma^*} \frac{p(\boldsymbol{y})^2 f(\boldsymbol{y})^2}{q(\boldsymbol{y})} - \theta^2\right). \tag{14}$$

If $\mathbb{E}_p\,|f(\boldsymbol{Y})| > 0$, the variance is minimized uniquely by

$$q^\star(\boldsymbol{y}) = \frac{p(\boldsymbol{y})\,|f(\boldsymbol{y})|}{\mathbb{E}_p\,|f(\boldsymbol{Y})|}, \qquad \mathrm{Var}_{q^\star}(\widehat{\theta}_{\mathrm{IS}}) = \frac{1}{M}\Big(\big(\mathbb{E}_p\,|f(\boldsymbol{Y})|\big)^2 - \theta^2\Big), \tag{15}$$

and the minimum is zero if and only if f has constant sign on the support of p.

Proof. The summands of $\widehat{\theta}_{\mathrm{IS}}$ are i.i.d., so it suffices to treat $M = 1$. Absolute continuity makes the weight $p(\boldsymbol{Y})/q(\boldsymbol{Y})$ well defined on the support of q, and

$$\mathbb{E}_q\left[\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})} f(\boldsymbol{Y})\right] = \sum_{\boldsymbol{y}:\, q(\boldsymbol{y})>0} q(\boldsymbol{y}) \frac{p(\boldsymbol{y})}{q(\boldsymbol{y})} f(\boldsymbol{y}) = \sum_{\boldsymbol{y}\in\Sigma^*} p(\boldsymbol{y}) f(\boldsymbol{y}) = \theta,$$

where the middle equality uses that $p(\boldsymbol{y}) = 0$ whenever $q(\boldsymbol{y}) = 0$. The same computation for the second moment gives $\mathbb{E}_q[(p(\boldsymbol{Y})f(\boldsymbol{Y})/q(\boldsymbol{Y}))^2] = \sum_{\boldsymbol{y}} p(\boldsymbol{y})^2 f(\boldsymbol{y})^2/q(\boldsymbol{y})$, which yields Eq. (14).

For the optimal proposal, write $c \stackrel{\text{def}}{=} \mathbb{E}_p\,|f(\boldsymbol{Y})| = \sum_{\boldsymbol{y}} p(\boldsymbol{y})|f(\boldsymbol{y})|$. By the Cauchy–Schwarz inequality applied to the vectors $\big(p(\boldsymbol{y})|f(\boldsymbol{y})|/\sqrt{q(\boldsymbol{y})}\big)_{\boldsymbol{y}}$ and $\big(\sqrt{q(\boldsymbol{y})}\big)_{\boldsymbol{y}}$ over the support of q,

$$c^2 = \Big(\sum_{\boldsymbol{y}} \frac{p(\boldsymbol{y})|f(\boldsymbol{y})|}{\sqrt{q(\boldsymbol{y})}} \sqrt{q(\boldsymbol{y})}\Big)^2 \le \sum_{\boldsymbol{y}} \frac{p(\boldsymbol{y})^2 f(\boldsymbol{y})^2}{q(\boldsymbol{y})} \sum_{\boldsymbol{y}} q(\boldsymbol{y}) = \sum_{\boldsymbol{y}} \frac{p(\boldsymbol{y})^2 f(\boldsymbol{y})^2}{q(\boldsymbol{y})}.$$

Hence $\mathrm{Var}_q(\widehat{\theta}_{\mathrm{IS}}) \ge (c^2 - \theta^2)/M$ for every admissible q, with equality if and only if the two vectors are proportional, that is, $q(\boldsymbol{y}) = p(\boldsymbol{y})|f(\boldsymbol{y})|/c = q^\star(\boldsymbol{y})$ on the support of q; since $q^\star$ sums to one, equality forces $q = q^\star$. Substituting $q^\star$ into Eq. (14) confirms the value in Eq. (15). Finally, $c^2 - \theta^2 = (c + |\theta|)(c - |\theta|) \ge 0$ because $c = \mathbb{E}_p\,|f(\boldsymbol{Y})| \ge |\,\mathbb{E}_p\, f(\boldsymbol{Y})| = |\theta|$, with equality if and only if $f(\boldsymbol{Y})$ has almost surely constant sign under p. □

B.2 Potentials under a Proposal

We now drop the assumption $q = p$ of §3. Let q be a proposal with respect to which p is absolutely continuous, as in §2.4, and let $\boldsymbol{Y}^{(1)}, \ldots, \boldsymbol{Y}^{(M)} \overset{\text{i.i.d.}}{\sim} q$. The only aspect that changes from §3 is that we now care about estimating the expected value of $f_q(\boldsymbol{Y})$ rather than $f(\boldsymbol{Y})$, and any computation in §3 involving the distribution p should now be carried out over the new proposal dsitribution q. Given a potential $\vec{\phi}$, we use the following notation for the modified potential under the proposal,

$$\vec{\phi}_{\mathrm{IS}}(\boldsymbol{y}) \stackrel{\text{def}}{=} \frac{p(\boldsymbol{y})}{q(\boldsymbol{y})} \vec{\phi}(\boldsymbol{y}), \qquad \boldsymbol{y} \in \Sigma^*$$

Definition 3 (Importance-weighted potential estimator)**.** *Let* $\boldsymbol{Y}^{(1)}, \dots, \boldsymbol{Y}^{(M)} \overset{i.i.d.}{\sim} q$*. Given a potential* $\vec{\phi}$*, define*

$$\widehat{\theta}_{\mathrm{IS}}(\vec{\phi}) \overset{\text{def}}{=} \widehat{\theta}(\vec{\phi}_{\mathrm{IS}})$$

Setting $\vec{\phi} \equiv 0$ reduces the inner sum to the terminal increment $f(\boldsymbol{Y}^{(m)})$ and recovers $\widehat{\theta}_{\mathrm{IS}}$ of Eq. (5). The next proposition is the counterpart of Proposition 1 under a proposal.

Proposition 4 (Unbiasedness and variance under a proposal)**.** *Let* $\vec{\phi}_{\mathrm{IS}}$ *satisfy the summability condition of Proposition 5, and assume that* $\widehat{\theta}_{\mathrm{IS}}(\vec{\phi})$ *has a finite second moment. Then* $\mathbb{E}_q[\widehat{\theta}_{\mathrm{IS}}(\vec{\phi})] = \theta$*. Moreover, the oracle potential does not yield a zero-variance estimator unless* $q = p$*:*

$$\mathrm{Var}_q\big(\widehat{\theta}_{\mathrm{IS}}(\vec{\phi}^\star)\big) = \frac{\theta^2}{M}\left(\sum_{\boldsymbol{y}\in\Sigma^*}\frac{p(\boldsymbol{y})^2}{q(\boldsymbol{y})} - 1\right).$$

Proof. Unbiasedness. It suffices to treat $M = 1$.

$$\mathbb{E}_q[\widehat{\theta}_{\mathrm{IS}}(\vec{\phi})] = \mathbb{E}_q[\widehat{\theta}(\vec{\phi}_{\mathrm{IS}})] \overset{(i)}{=} \mathbb{E}_q\left[\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})}\widehat{\theta}(\vec{\phi})\right] = \mathbb{E}_p[\widehat{\theta}(\vec{\phi})] \overset{(ii)}{=} \theta$$

(i) holds since $\widehat{\theta}$ is linear by Def. 1. (ii) is by unbiasedness of $\widehat{\theta}(\vec{\phi})$ by proposition 5.
Variance. Similarly,

$$M\mathrm{Var}_q(\widehat{\theta}_{\mathrm{IS}}(\vec{\phi}^\star)) = M\mathrm{Var}_q(\widehat{\theta}(\vec{\phi}^\star_{\mathrm{IS}})) = \mathrm{Var}_q\left(\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})}\widehat{\theta}(\vec{\phi}^\star)\right)$$

$$= \mathbb{E}_q\left[\frac{p(\boldsymbol{Y})^2}{q(\boldsymbol{Y})^2}\widehat{\theta}(\vec{\phi}^\star)^2\right] - \theta^2$$

$$= \mathbb{E}_p\left[\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})}\widehat{\theta}(\vec{\phi}^\star)^2\right] - \theta^2$$

$$= \mathbb{E}_p\left[\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})}\widehat{\theta}(\vec{\phi}^\star)^2\right] - \mathbb{E}_p\left[\widehat{\theta}(\vec{\phi}^\star)^2\right] + \mathbb{E}_p\left[\widehat{\theta}(\vec{\phi}^\star)^2\right] - \theta^2$$

$$\overset{(i)}{=} \mathbb{E}_p\left[\left(\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})} - 1\right)\widehat{\theta}(\vec{\phi}^\star)^2\right]$$

$$= \mathbb{E}_p\left[\left(\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})} - 1\right)\left(\vec{\phi}^\star(\varepsilon) + \sum_{t=1}^{|\boldsymbol{Y}|+1}\mathbf{B}\vec{\phi}^\star(\boldsymbol{Y}_{<t}) - \vec{\phi}^\star(\boldsymbol{Y}_{<t})\right)^2\right]$$

$$\overset{(ii)}{=} \mathbb{E}_p\left[\left(\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})} - 1\right)\left(\theta + \sum_{t=1}^{|\boldsymbol{Y}|+1}\mathbb{E}_{\boldsymbol{Y}_{t+1}\sim\vec{p}^\star(.|\boldsymbol{Y}_{<t})}\vec{\phi}^\star(\boldsymbol{Y}_{\le t}) - \vec{\phi}^\star(\boldsymbol{Y}_{<t})\right)^2\right]$$

$$= \mathbb{E}_p\left[\left(\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})} - 1\right)\left(\theta + \sum_{t=1}^{|\boldsymbol{Y}|+1}\mathbb{E}_{\boldsymbol{Y}_{t+1}\sim\vec{p}^\star(.|\boldsymbol{Y}_{<t})}\mathbb{E}_p[f(\boldsymbol{Y})|\boldsymbol{Y}\succeq\boldsymbol{Y}_{\le t}] - \mathbb{E}_p[f(\boldsymbol{Y})|\boldsymbol{Y}\succeq\boldsymbol{Y}_{<t}]\right)^2\right]$$

$$= \mathbb{E}_p\left[\left(\frac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})} - 1\right)\left(\theta + \sum_{t=1}^{|\boldsymbol{Y}|+1}\mathbb{E}_p[f(\boldsymbol{Y})|\boldsymbol{Y}\succeq\boldsymbol{Y}_{<t}] - \mathbb{E}_p[f(\boldsymbol{Y})|\boldsymbol{Y}\succeq\boldsymbol{Y}_{<t}]\right)^2\right]$$

$$= \theta^2\left(\sum_{\boldsymbol{y}\in\Sigma^*}\frac{p(\boldsymbol{y})^2}{q(\boldsymbol{y})} - 1\right)$$

(i) holds since $\mathrm{Var}_p\widehat{\theta}(\vec{\phi}^\star) = \mathbb{E}_p\left[\widehat{\theta}(\vec{\phi}^\star)^2\right] - \theta^2 = 0$ by proposition 1. (ii) is since $\mathbb{E}_{\boldsymbol{Y}_{t+1}\sim\vec{p}^\star(.|\boldsymbol{Y}_{<t})}\mathbb{E}_p[f(\boldsymbol{Y})|\boldsymbol{Y}\succeq\boldsymbol{Y}_{\le t}] - \mathbb{E}_p[f(\boldsymbol{Y})|\boldsymbol{Y}\succeq\boldsymbol{Y}_{<t}$ by definition of $\widehat{\theta}^\star$. Thus we have

the expression for the variance of the estimator based on oracle potential under the proposal distribution. Suppose $q \neq p$. Then,

$$\sum_{\boldsymbol{y}\in\Sigma^*} \frac{p(\boldsymbol{y})^2}{q(\boldsymbol{y})} \geq \sum_{\boldsymbol{y}\in\Sigma^*} p(\boldsymbol{y}) \left(\log \frac{p(\boldsymbol{y})}{q(\boldsymbol{y})} + 1\right) = KL(p|q) + 1 > 1$$

Thus the variance cannot be equal to 0 when $q \neq p$. In case $q = p$, substituting the same in the expression for $\mathrm{Var}_q(\widehat{\theta}_{\mathrm{IS}}(\vec{\phi}^\star))$, it is easy to see that the variance is equal to 0. □

The control-variate estimator of Def. 2 is formed with the baseline $\widehat{\theta}_* = \widehat{\theta}_{\mathrm{IS}}$ and $\widehat{\theta}(\vec{\phi})$ replaced by $\widehat{\theta}_{\mathrm{IS}}(\vec{\phi})$,

$$\widehat{\theta}_{\mathrm{IS}}(\vec{\phi}, \alpha) \stackrel{\text{def}}{=} \widehat{\theta}_{\mathrm{IS}} - \alpha\big(\widehat{\theta}_{\mathrm{IS}} - \widehat{\theta}_{\mathrm{IS}}(\vec{\phi})\big), \qquad \alpha \in \mathbb{R}. \tag{16}$$

The proof of Proposition 2 uses only the unbiasedness and finite second moments of the two estimators, so it applies verbatim: $\widehat{\theta}_{\mathrm{IS}}(\vec{\phi}, \alpha)$ is unbiased for every α, and at $\alpha^\star$ its variance is at most that of $\widehat{\theta}_{\mathrm{IS}}$.

B.3 Unbiasedness of the Potential Estimator

Proposition 5 (Unbiasedness and Variance)**.** *Let $\vec{\phi}$ be any potential whose increments are absolutely summable in expectation, $\mathbb{E}_p\big[\sum_{t=1}^{|\boldsymbol{Y}|+1} |\Delta\vec{\phi}(Y_t \mid \boldsymbol{Y}_{<t})|\big] < \infty$. Then the potential estimator of Eq. (10) is unbiased: $\mathbb{E}_p[\widehat{\theta}(\vec{\phi})] = \theta$.*

Proof. It suffices to treat $M = 1$. Write $T \stackrel{\text{def}}{=} |\boldsymbol{Y}| + 1$ for the index of the EOS step, so that the event $\{t \leq T\}$ is determined by $\boldsymbol{Y}_{<t}$: it holds exactly when $\boldsymbol{Y}_{<t}$ has not yet been terminated. For each t, conditioning on $\boldsymbol{Y}_{<t}$ and using that $Y_t \sim \vec{p}(\cdot \mid \boldsymbol{Y}_{<t})$ on this event,

$$\mathbb{E}_p\big[\mathbf{1}\{t \leq T\}\, \Delta\vec{\phi}(Y_t \mid \boldsymbol{Y}_{<t})\big] = \mathbb{E}_p\big[\mathbf{1}\{t \leq T\}\, \mathbb{E}[\Delta\vec{\phi}(Y_t \mid \boldsymbol{Y}_{<t}) \mid \boldsymbol{Y}_{<t}]\big] = \mathbb{E}_p\big[\mathbf{1}\{t \leq T\}\, \mathbf{B}\Delta\vec{\phi}(\boldsymbol{Y}_{<t})\big],$$

by Eq. (9). Absolute summability lets us sum over t inside the expectation on both sides, giving

$$\mathbb{E}_p\Big[\sum_{t=1}^{T} \Delta\vec{\phi}(Y_t \mid \boldsymbol{Y}_{<t})\Big] = \mathbb{E}_p\Big[\sum_{t=1}^{T} \mathbf{B}\Delta\vec{\phi}(\boldsymbol{Y}_{<t})\Big] = \mathbb{E}_p[\widehat{\theta}(\vec{\phi})] - \vec{\phi}(\varepsilon).$$

The left-hand side equals $\mathbb{E}_p[f(\boldsymbol{Y})] - \vec{\phi}(\varepsilon) = \theta - \vec{\phi}(\varepsilon)$ by the telescoping identity Eq. (7), which proves the claim.

□

B.4 Proof of Proposition 1

Proof of Proposition 1. Oracle potential. For a prefix $\boldsymbol{y}$ with $\vec{p}(\boldsymbol{y}) > 0$, the oracle potential is the continuation value

$$\vec{\phi}^\star(\boldsymbol{y}) = \mathbb{E}_p[f(\boldsymbol{Y}) \mid \boldsymbol{Y} \succeq \boldsymbol{y}] = \frac{1}{\vec{p}(\boldsymbol{y})} \sum_{\boldsymbol{z}\in\Sigma^*} p(\boldsymbol{y}\boldsymbol{z})\, f(\boldsymbol{y}\boldsymbol{z}),$$

which is finite whenever $\mathbb{E}_p\,|f(\boldsymbol{Y})| < \infty$. In particular $\vec{\phi}^\star(\varepsilon) = \theta$. Splitting the continuation $\boldsymbol{z}$ according to its first symbol gives the one-step recursion

$$\vec{\phi}^\star(\boldsymbol{y}) = \sum_{a\in\Sigma} \vec{p}(a \mid \boldsymbol{y})\, \vec{\phi}^\star(\boldsymbol{y}a) + \vec{p}(\text{EOS} \mid \boldsymbol{y})\, f(\boldsymbol{y}), \tag{17}$$

where terms with $\vec{p}(a \mid \boldsymbol{y}) = 0$ are read as zero. The right-hand side is precisely $\mathbf{B}\vec{\phi}^\star(\boldsymbol{y})$, so the expected increment of Eq. (9) vanishes at every prefix of positive probability:

$$\mathbf{B}\Delta\vec{\phi}^\star(\boldsymbol{y}) = \mathbf{B}\vec{\phi}^\star(\boldsymbol{y}) - \vec{\phi}^\star(\boldsymbol{y}) = 0.$$

Every prefix $\boldsymbol{Y}_{<t}^{(m)}$ visited by a sampled string has positive probability, so all inner sums in Eq. (10) vanish and

$$\widehat{\theta}(\vec{\phi}^\star) = \vec{\phi}^\star(\varepsilon) + \frac{1}{M}\sum_{m=1}^{M} \sum_{t=1}^{|\boldsymbol{Y}^{(m)}|+1} 0 = \theta$$

pathwise. A constant has zero variance, which proves the first claim.

Variance comparison. Writing $\widehat{\theta}(\vec{\phi}) = \widehat{\theta}_* - \big(\widehat{\theta}_* - \widehat{\theta}(\vec{\phi})\big)$ and expanding,

$$\mathrm{Var}_p\big(\widehat{\theta}(\vec{\phi})\big) = \mathrm{Var}_p(\widehat{\theta}_*) + \mathrm{Var}_p\big(\widehat{\theta}_* - \widehat{\theta}(\vec{\phi})\big) - 2\mathrm{Cov}_p\big(\widehat{\theta}_*, \widehat{\theta}_* - \widehat{\theta}(\vec{\phi})\big).$$

Therefore $\mathrm{Var}_p(\widehat{\theta}(\vec{\phi})) \le \mathrm{Var}_p(\widehat{\theta}_*)$ holds if and only if $\mathrm{Var}_p(\widehat{\theta}_* - \widehat{\theta}(\vec{\phi})) \le 2\mathrm{Cov}_p(\widehat{\theta}_*, \widehat{\theta}_* - \widehat{\theta}(\vec{\phi}))$, which is the stated condition. □

B.5 Proof of Proposition 2

Proof of Proposition 2. Unbiasedness. By linearity,

$$\mathbb{E}_p[\widehat{\theta}(\vec{\phi}, \alpha)] = \mathbb{E}_p[\widehat{\theta}_*] - \alpha\big(\mathbb{E}_p[\widehat{\theta}_*] - \mathbb{E}_p[\widehat{\theta}(\vec{\phi})]\big) = \theta - \alpha(\theta - \theta) = \theta,$$

using that $\widehat{\theta}_*$ is unbiased by assumption and $\widehat{\theta}(\vec{\phi})$ is unbiased by Proposition 5.

Optimal coefficient. Write $D \stackrel{\text{def}}{=} \widehat{\theta}_* - \widehat{\theta}(\vec{\phi})$. Since both estimators have finite second moments,

$$\mathrm{Var}_p\big(\widehat{\theta}(\vec{\phi}, \alpha)\big) = \mathrm{Var}_p(\widehat{\theta}_* - \alpha D) = \mathrm{Var}_p(\widehat{\theta}_*) - 2\alpha\,\mathrm{Cov}_p(\widehat{\theta}_*, D) + \alpha^2\,\mathrm{Var}_p(D). \tag{18}$$

When $\mathrm{Var}_p(D) > 0$, this is a strictly convex quadratic in α, minimized where its derivative vanishes, $\alpha\,\mathrm{Var}_p(D) = \mathrm{Cov}_p(\widehat{\theta}_*, D)$, that is, at $\alpha^\star = \mathrm{Cov}_p(\widehat{\theta}_*, D)/\mathrm{Var}_p(D)$ as in Eq. (12). Substituting $\alpha^\star$ into Eq. (18),

$$\mathrm{Var}_p\big(\widehat{\theta}(\vec{\phi}, \alpha^\star)\big) = \mathrm{Var}_p(\widehat{\theta}_*) - \frac{\mathrm{Cov}_p(\widehat{\theta}_*, D)^2}{\mathrm{Var}_p(D)} \le \mathrm{Var}_p(\widehat{\theta}_*).$$

When $\mathrm{Var}_p(D) = 0$, D is almost surely equal to its mean, which is zero by unbiasedness, so $\widehat{\theta}(\vec{\phi}, \alpha) = \widehat{\theta}_*$ almost surely for every α. □

C Additional Analysis and Experimental Details

This appendix collects the setup behind the experiments of §4: the models and prompt banks, the target-construction protocols, the generation configurations, and the supporting figures and tables. One subsection per application, in the order of §4.

C.1 Analysis of Watermark Detection

A language model watermark embeds a statistical signal in generated text while preserving the model's intended task. Watermark detection then aims to reliably assess whether a text of a given length was generated by a watermarked model. We use the scheme of Kirchenbauer et al. (2023): at each generation step, a pseudorandom **green list** $G(\boldsymbol{x}) \subseteq \Sigma$ is determined from the prompt and generated prefix $\boldsymbol{x}$, and the logits of tokens in this set are increased by a bias δ. Let γ denote the fraction of the vocabulary assigned to the green list, m the generation length, and $\sigma \stackrel{\text{def}}{=} \sqrt{m\gamma(1-\gamma)}$. Detection then counts the number of generated tokens that fall in their respective green lists and standardizes this count,

$$\frac{|\boldsymbol{Y}|_G - \gamma m}{\sigma}, \qquad |\boldsymbol{Y}|_G \stackrel{\text{def}}{=} \#\{t : Y_t \in G(\boldsymbol{Y}_{<t})\}, \tag{19}$$

declaring the string watermarked when this exceeds a prescribed threshold τ, equivalently when $|\boldsymbol{Y}|_G \ge \gamma m + \tau\sigma$.

C.1.1 Estimands and potential definitions

At a fixed prompt and generation length, two expectations characterize the detector. The **expected score** $\mathbb{E}_p\big[(|\boldsymbol{Y}|_G - \gamma m)/\sigma\big]$ measures how strongly the watermark registers on average; the **detection probability** $\Pr_p[|\boldsymbol{Y}|_G \ge \gamma m + \tau\sigma]$ is the detector's true-positive rate, and is the quantity a deployment actually cares about. The two behave differently under our framework. The score has an additive

TABLE 4: EXPECTED WATERMARK SCORE	TABLE 5: PROBABILITY OF WATERMARK DETECTION
$f(\boldsymbol{Y}) = \big(\lvert\boldsymbol{Y}\rvert_G - \gamma m\big)/\sigma$	$f(\boldsymbol{Y}) = \mathbf{1}\{\lvert\boldsymbol{Y}\rvert_G \geq \gamma m + \tau\sigma\}$
$\lvert\boldsymbol{x}\boldsymbol{z}\rvert_G = \lvert\boldsymbol{x}\rvert_G + \lvert\boldsymbol{z}\rvert_G$	not additive in the prefix
$\vec{\phi}^{\star}(\boldsymbol{x}) = \big(\lvert\boldsymbol{x}\rvert_G + \mathbb{E}_p[\lvert\boldsymbol{Z}\rvert_G \mid \boldsymbol{Z} \succeq \boldsymbol{x}] - \gamma m\big)/\sigma$	$\vec{\phi}^{\star}(\boldsymbol{x}) = \Pr_p\big[\lvert\boldsymbol{Y}\rvert_G \geq \gamma m + \tau\sigma \mid \boldsymbol{Y} \succeq \boldsymbol{x}\big]$
$\vec{\phi}(\boldsymbol{x}) = \big(\lvert\boldsymbol{x}\rvert_G - \gamma m\big)/\sigma$	$\vec{\phi}(\boldsymbol{x}, q) = \Pr\big[\mathrm{Bin}(m - \lvert\boldsymbol{x}\rvert, q) \geq k(\boldsymbol{x})\big]$
$\Delta\vec{\phi}(a \mid \boldsymbol{x}) = \mathbf{1}\{a \in G(\boldsymbol{x})\}/\sigma$	$\Delta\vec{\phi}(a \mid \boldsymbol{x}, q) = \vec{\phi}(\boldsymbol{x}a, q) - \vec{\phi}(\boldsymbol{x}, q)$
$\Delta\vec{\phi}(\text{EOS} \mid \boldsymbol{x}) = 0$	$\Delta\vec{\phi}(\text{EOS} \mid \boldsymbol{x}, q) = 0$

structure over prefixes (Tab. 4) so the value of the function at the observed prefix is itself a potential, exactly as for the losses of §4.1. The detection indicator is not additive in the prefix, so it admits no such decomposition and must instead be approached by approximating the oracle potential, as in the event analysis of §4.3. Here that oracle potential is the conditional probability that the detector will eventually fire.

Detection probability: analytical approximation of the oracle potential. For the detection probability a natural analytical approximation of the oracle potential is available. Write $k(\boldsymbol{x}) = \gamma m + \tau\sigma - \lvert\boldsymbol{x}\rvert_G$ for the number of green tokens still needed at the prefix $\boldsymbol{x}$. If the remaining $m - \lvert\boldsymbol{x}\rvert$ tokens were green independently with a common rate q, the detector would fire with probability $\Pr[\mathrm{Bin}(m - \lvert\boldsymbol{x}\rvert, q) \geq k(\boldsymbol{x})]$, which is the potential of Tab. 5. It depends on the single parameter q, which we fit globally for each length m by cross-fitting on 10 folds of the continuations: the q applied to one fold is the average green-list mass of the other nine, pooled over prompts. Computing this potential requires no additional model evaluations, as it uses only the green-list mass $\sum_{a \in G(\boldsymbol{x})} \vec{p}(a \mid \boldsymbol{x})$ from the already-computed next-token distribution.

C.1.2 EXPERIMENTS AND RESULTS

Setup. We use the same setup of Kirchenbauer et al. (2023), with green-list fraction $\gamma = 0.5$ and logit biases $\delta \in \{1, 2, 5\}$. For each configuration, we generate 4,000 continuations per prompt from OPT-1.3B for 512 prompts from C4 RealNewsLike, using temperature 0.7. We evaluate continuation lengths $m \in \{32, 64, 128, 200\}$ and detector thresholds $z_0 \in \{3, 4, 5\}$.

For each of the two estimands, we compare the associated potentials against Monte Carlo. For the watermark-detection probability, we also include the baseline terminal potential, i.e., the corresponding observed-prefix potential for non-additive functions: zero at every nonterminal prefix and equal to the function value at the terminal prefix.

δ	m	median	$[p_{10},\ p_{90}]$	worst prompt
1	32	0.471	[0.413, 0.564]	0.907
	64	0.513	[0.460, 0.597]	0.894
	128	0.561	[0.508, 0.677]	0.931
	200	0.612	[0.544, 0.748]	0.965
2	32	0.538	[0.471, 0.631]	0.836
	64	0.584	[0.520, 0.671]	0.876
	128	0.644	[0.583, 0.748]	0.925
	200	0.699	[0.629, 0.818]	0.962
5	32	0.722	[0.608, 0.807]	0.922
	64	0.743	[0.665, 0.817]	0.960
	128	0.790	[0.725, 0.867]	0.993
	200	0.837	[0.776, 0.910]	0.996

Table 6: Per-prompt variance of $\widehat{\theta}(\vec{\phi})$ against Monte Carlo, 512 prompts.

Detection probability: the binomial potential removes about 44% of the Monte Carlo variance when detection is uncertain, but little once it saturates. Fig. 3(b) shows the variance ratio between the potential estimator and MC for each prompt and configuration (δ, m, z_0). Across the 14,366 pairs for which detection is not deterministic, the median ratio is 0.64, corresponding to a 36% reduction in variance. Since variance scales inversely with the number of samples, this is equivalent to reaching the precision of 4,000 Monte Carlo samples with about 2,560 samples, without additional model evaluations. The gain is largest when detection is uncertain: for $0.001 \leq \widehat{\pi} < 0.5$, the median ratio is 0.56, corresponding to a 44% reduction. It decreases as detection saturates, reaching a median ratio of 0.97 for $\widehat{\pi} \geq 0.99$. Overall, 92.7% of pairs improve over Monte Carlo, with similar behavior across δ, m, and z_0. The terminal potential, by contrast, remains close to Monte Carlo, with a median ratio of 0.973.

Fig. 3(a) gives intuition for the variance reduction. From early in generation, trajectories that are eventually detected overlap with regions of positive potential, whereas undetected trajectories lie

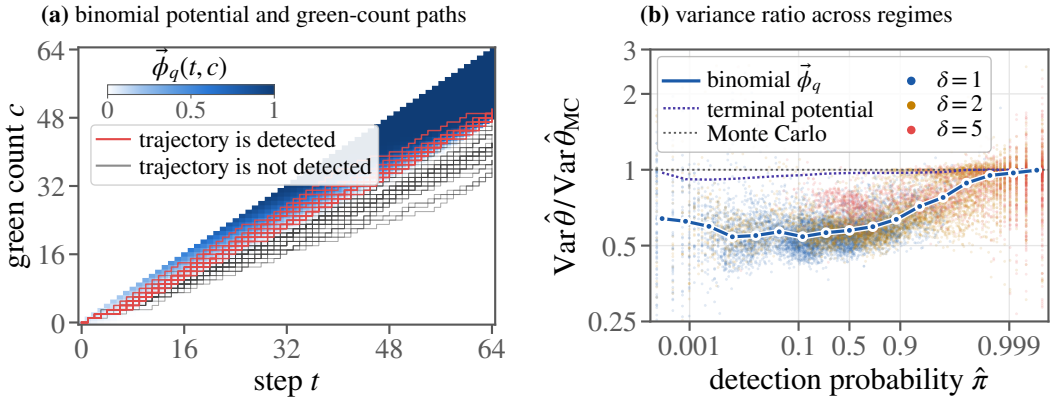


Figure 3: Panel (a) shows how sampled trajectories update the green-token count over 64 generation steps. Red trajectories are those whose final green-token count exceeds the detection threshold, while black trajectories remain below it. The background encodes the potential's assigned probability of eventual detection at each step and green-count state, with white indicating probability zero. Red trajectories tend to pass through higher-probability regions, whereas black trajectories concentrate in low- or zero-probability regions. Panel (b) shows the variance reduction at different detection probabilities

mostly in the white region where the potential is zero. Thus, the potential already contains information about the final detection outcome long before the trajectory ends.

Expected score: every prompt improves, by 16–53% depending on δ and m. Tab. 6 gives the per-prompt variance relative to Monte Carlo over the 512 prompts. The median ratio rises from 0.471 at $\delta = 1, m = 32$ to 0.837 at $\delta = 5, m = 200$: the marginalization helps most when the watermark is weak and the continuation is short, the regime in which a single green indicator carries the most conditional variance. The spread over prompts is narrow, the p_{10}–p_{90} width never exceeding 0.21, and every one of the 512 prompts lies below Monte Carlo in all twelve cells, the worst single prompt reaching 0.996. We also cross-fit a control-variate coefficient on top of but observe that it buys almost nothing here: across the twelve cells the variance against $\widehat{\theta}(\vec{\phi})$ stays within 0.991–0.998 and the median prompt within 0.994–1.000.

C.2 Losses and Divergences: Details and Further Results

C.2.1 Potential definitions

Below we give the potential tables for the losses and divergences of §4.1; the corresponding gradients are in App. C.3.

Table 7: KL divergence	
$f(\boldsymbol{Y}) = \log \dfrac{p(\boldsymbol{Y})}{q(\boldsymbol{Y})}$	Test function
$f(\boldsymbol{xz}) = \log \dfrac{\vec{p}(\boldsymbol{x})}{\vec{q}(\boldsymbol{x})} + \log \dfrac{\vec{p}(\boldsymbol{z} \mid \boldsymbol{x})}{\vec{q}(\boldsymbol{z} \mid \boldsymbol{x})}$	Function structure
$\vec{\phi}^{\star}(\boldsymbol{x}) = \log \dfrac{\vec{p}(\boldsymbol{x})}{\vec{q}(\boldsymbol{x})} + \mathbb{E}_p\left[\log \dfrac{\vec{p}(\boldsymbol{Z} \mid \boldsymbol{x})}{\vec{q}(\boldsymbol{Z} \mid \boldsymbol{x})} \;\middle|\; \boldsymbol{Z} \succeq \boldsymbol{x}\right]$	Oracle potential
$\vec{\phi}(\boldsymbol{x}) = \log \dfrac{\vec{p}(\boldsymbol{x})}{\vec{q}(\boldsymbol{x})}$	Potential
$\Delta\vec{\phi}(a \mid \boldsymbol{x}) = \log \dfrac{\vec{p}(a \mid \boldsymbol{x})}{\vec{q}(a \mid \boldsymbol{x})}$	Increment ($a \in \Sigma$)
$\Delta\vec{\phi}(\mathrm{EOS} \mid \boldsymbol{x}) = \log \dfrac{\vec{p}(\mathrm{EOS} \mid \boldsymbol{x})}{\vec{q}(\mathrm{EOS} \mid \boldsymbol{x})}$	EOS increment
$\mathbf{B}\Delta\vec{\phi}(\boldsymbol{x}) = D_{\mathrm{KL}}\big(\vec{p}(\cdot \mid \boldsymbol{x}) \,\|\, \vec{q}(\cdot \mid \boldsymbol{x})\big)$	Expected increment

TABLE 8: BRIER LOSS
$f(\boldsymbol{Y}) = \sum_t \sum_{b \in \Sigma \cup \{\text{EOS}\}} \big(\vec{p}(b \mid \boldsymbol{Y}_{<t}) - \mathbf{1}\{b = Y_t\}\big)^2$
$\Delta\vec{\phi}(a \mid \boldsymbol{x}) = \sum_{b \in \Sigma \cup \{\text{EOS}\}} \big(\vec{p}(b \mid \boldsymbol{x}) - \mathbf{1}\{b = a\}\big)^2 = 1 - 2\,\vec{p}(a \mid \boldsymbol{x}) + \sum_{b \in \Sigma \cup \{\text{EOS}\}} \vec{p}(b \mid \boldsymbol{x})^2$
$f(\boldsymbol{xz}) = \sum_{t \le \lvert\boldsymbol{x}\rvert} \Delta\vec{\phi}(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}) + \sum_{t \le \lvert\boldsymbol{z}\rvert} \Delta\vec{\phi}(\boldsymbol{z}_t \mid \boldsymbol{x}\boldsymbol{z}_{<t})$
$\vec{\phi}^{\star}(\boldsymbol{x}) = \sum_{t \le \lvert\boldsymbol{x}\rvert} \Delta\vec{\phi}(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}) + \mathbb{E}_p\big[\sum_{t > \lvert\boldsymbol{x}\rvert} \Delta\vec{\phi}(\boldsymbol{Z}_t \mid \boldsymbol{Z}_{<t}) \;\big|\; \boldsymbol{Z} \succeq \boldsymbol{x}\big]$
$\vec{\phi}(\boldsymbol{x}) = \sum_{t \le \lvert\boldsymbol{x}\rvert} \Delta\vec{\phi}(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t})$
$\Delta\vec{\phi}(\text{EOS} \mid \boldsymbol{x}) = 1 - 2\,\vec{p}(\text{EOS} \mid \boldsymbol{x}) + \sum_{b \in \Sigma \cup \{\text{EOS}\}} \vec{p}(b \mid \boldsymbol{x})^2$
$\mathbf{B}\Delta\vec{\phi}(\boldsymbol{x}) = 1 - \sum_{b \in \Sigma \cup \{\text{EOS}\}} \vec{p}(b \mid \boldsymbol{x})^2$

TABLE 9: POWER-SCORE REGRET, EXPONENT β
$f(\boldsymbol{Y}) = \sum_t \big[\beta\big(\vec{p}(Y_t \mid \boldsymbol{Y}_{<t})^{\beta-1} - \vec{q}(Y_t \mid \boldsymbol{Y}_{<t})^{\beta-1}\big) - (\beta-1)\sum_{b \in \Sigma \cup \{\text{EOS}\}} \big(\vec{p}(b \mid \boldsymbol{Y}_{<t})^{\beta} - \vec{q}(b \mid \boldsymbol{Y}_{<t})^{\beta}\big)\big]$
$\Delta\vec{\phi}(a \mid \boldsymbol{x}) = \beta\big(\vec{p}(a \mid \boldsymbol{x})^{\beta-1} - \vec{q}(a \mid \boldsymbol{x})^{\beta-1}\big) - (\beta-1)\sum_{b \in \Sigma \cup \{\text{EOS}\}} \big(\vec{p}(b \mid \boldsymbol{x})^{\beta} - \vec{q}(b \mid \boldsymbol{x})^{\beta}\big)$
$f(\boldsymbol{xz}) = \sum_{t \le \lvert\boldsymbol{x}\rvert} \Delta\vec{\phi}(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}) + \sum_{t \le \lvert\boldsymbol{z}\rvert} \Delta\vec{\phi}(\boldsymbol{z}_t \mid \boldsymbol{x}\boldsymbol{z}_{<t})$
$\vec{\phi}^{\star}(\boldsymbol{x}) = \sum_{t \le \lvert\boldsymbol{x}\rvert} \Delta\vec{\phi}(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}) + \mathbb{E}_p\big[\sum_{t > \lvert\boldsymbol{x}\rvert} \Delta\vec{\phi}(\boldsymbol{Z}_t \mid \boldsymbol{Z}_{<t}) \;\big|\; \boldsymbol{Z} \succeq \boldsymbol{x}\big]$
$\vec{\phi}(\boldsymbol{x}) = \sum_{t \le \lvert\boldsymbol{x}\rvert} \Delta\vec{\phi}(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t})$
$\Delta\vec{\phi}(\text{EOS} \mid \boldsymbol{x}) = \beta\big(\vec{p}(\text{EOS} \mid \boldsymbol{x})^{\beta-1} - \vec{q}(\text{EOS} \mid \boldsymbol{x})^{\beta-1}\big) - (\beta-1)\sum_{b \in \Sigma \cup \{\text{EOS}\}} \big(\vec{p}(b \mid \boldsymbol{x})^{\beta} - \vec{q}(b \mid \boldsymbol{x})^{\beta}\big)$
$\mathbf{B}\Delta\vec{\phi}(\boldsymbol{x}) = \sum_{b \in \Sigma \cup \{\text{EOS}\}} \big[\vec{p}(b \mid \boldsymbol{x})^{\beta} - \beta\,\vec{p}(b \mid \boldsymbol{x})\,\vec{q}(b \mid \boldsymbol{x})^{\beta-1} + (\beta-1)\,\vec{q}(b \mid \boldsymbol{x})^{\beta}\big]$

Entropy is the same construction with $\Delta\vec{\phi}(a \mid \boldsymbol{x}) = -\log \vec{p}(a \mid \boldsymbol{x})$, shown in Tab. 1. The four power-score regrets use $\beta \in \{1.25, 1.5, 1.75, 2\}$; as $\beta \to 1$ the per-token term, divided by $\beta - 1$, tends to the KL term of Tab. 7.

C.2.2 EXPERIMENTAL SETUP

We evaluate the losses and divergences on eight configurations, each defined by a sampling policy, a reference model, and a generation horizon (Tab. 10). These cover seven distinct policy–reference pairs: the GPT-2 sentiment pair is evaluated at two horizons, one of 50 tokens that reuses the setting of Amini et al. (2025) for reproducibility, and one of 256 tokens to match the longer window used for all other models. Model repositories are listed in Tab. 11.

Shared sampling protocol. Each configuration uses 512 prompts and B responses per prompt, with B specified in Tab. 10. Responses are drawn by ancestral sampling at temperature 1, without top-k or top-p filtering, using fp32 precision. Generation stops at EOS or the specified horizon.

Table 10: Policy–reference banks of the proper-loss experiment. Labels are those used in the figures; p is the sampling policy and q the reference; model names are resolved in Tab. 11. All banks have 512 prompts and B responses per prompt.

bank	prompts	sampling policy p	reference q	horizon	B
IMDb-256	IMDb prefixes	gpt2-imdb + DPO	gpt2-imdb	256	4000
IMDb-50	IMDb prefixes	same policy	gpt2-imdb	50	4000
Tülu-3	AlpacaEval 2	Tulu-3-8B-DPO	Tulu-3-8B-SFT	256	100
Gemma It3 vs It0	AlpacaEval 2	SPPO-Iter3	gemma-2-9b-it	256	100
Gemma It1 vs It0	AlpacaEval 2	SPPO-Iter1	gemma-2-9b-it	256	100
Gemma It2 vs It0	AlpacaEval 2	SPPO-Iter2	gemma-2-9b-it	256	100
Gemma It2 vs It1	AlpacaEval 2	SPPO-Iter2	SPPO-Iter1	256	100
Gemma It3 vs It2	AlpacaEval 2	SPPO-Iter3	SPPO-Iter2	256	100

GPT-2-imbd + DPO: policy training. We train the policy p ourself, following the setting of Amini et al. (2025). The reference q is `gpt2-imdb`; we obtain the policy p by fine-tuning it for positive sentiment with direct preference optimization (DPO) on 5,000 IMDb training reviews. Training prompts are 2–8-token prefixes. Four sampled responses per prompt form six response pairs, with

preferences determined by `distilbert-imdb`. We train for one epoch (938 steps), using $\beta_{\mathrm{DPO}} = 0.1$ and a learning rate of 3×10^{-6}.

GPT-2 sentiment: evaluation prompts. We select 512 reviews from the IMDb test split using a fixed seed. For each review, we draw a prefix length uniformly from $\{2, \dots, 8\}$ and retain the corresponding GPT-2 token IDs as the prompt. We draw 4,000 responses per prompt at a primary horizon of 256 tokens and a control horizon of 50 tokens, using the same policy and reference in both configurations.

Chat models: policy–reference comparisons. For Tülu-3, we compare the published DPO policy with its SFT reference. For Gemma-2, we use the three published SPPO iterations, denoting the initial instruction model by iteration 0. Comparing iterations 1, 2, and 3 with iteration 0 measures accumulated drift. We additionally compare iteration 2 with 1 and iteration 3 with 2 to examine local drift between successive updates.

Chat models: evaluation prompts. The chat-model configurations use 512 instructions from the 805-instruction AlpacaEval 2 set. Instructions are rendered with the corresponding model's chat template; within each policy–reference pair, both models receive identical token IDs. Each prompt receives 100 responses, with a horizon of 256 tokens.

Table 11: Model repositories (Hugging Face) for the names used in Tab. 10.

model	repository
gpt2-imdb	`lvwerra/gpt2-imdb`
gpt2-imdb + DPO	trained by us from `lvwerra/gpt2-imdb` (recipe above)
distilbert-imdb (DPO preference judge)	`lvwerra/distilbert-imdb`
Tulu-3-8B-SFT	`allenai/Llama-3.1-Tulu-3-8B-SFT`
Tulu-3-8B-DPO	`allenai/Llama-3.1-Tulu-3-8B-DPO`
gemma-2-9b-it (SPPO Iter0)	`google/gemma-2-9b-it`
SPPO-Iter1	`UCLA-AGI/Gemma-2-9B-It-SPPO-Iter1`
SPPO-Iter2	`UCLA-AGI/Gemma-2-9B-It-SPPO-Iter2`
SPPO-Iter3	`UCLA-AGI/Gemma-2-9B-It-SPPO-Iter3`

C.2.3 Additional results

Variance reduction across quantities. Fig. 4 extends the magnitude–variance comparison of Fig. 1(b) to all seven quantities. For each quantity, it relates the variance ratio to the estimated magnitude of the quantity, showing both configuration-level aggregates and individual prompts.

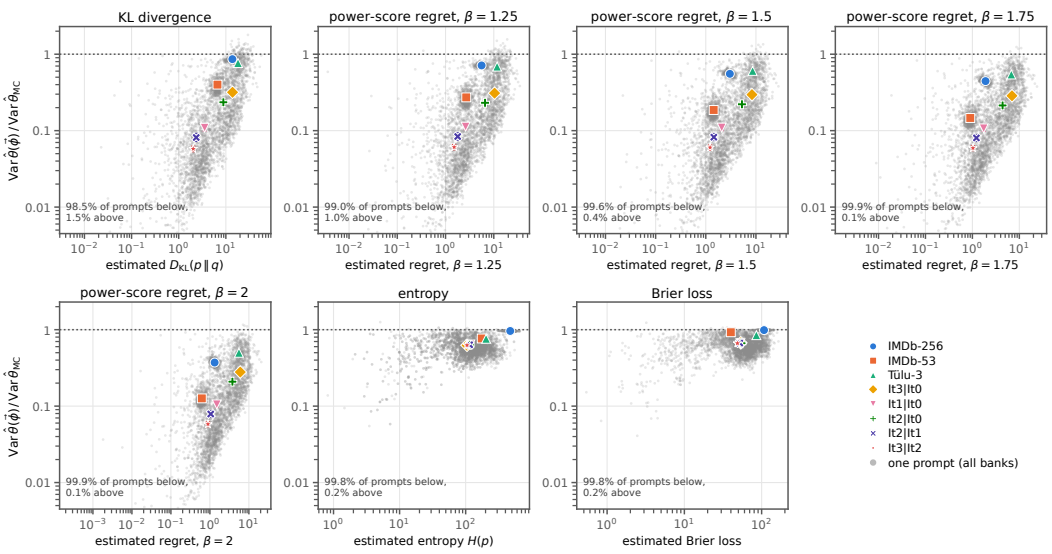


Figure 4: Variance reduction as a function of quantity magnitude for all losses and divergences in the proper-loss experiment. Each panel shows the variance ratio $\mathrm{Var}_p(\widehat{\theta}(\vec{\phi}))/\mathrm{Var}_p(\widehat{\theta}_{\mathrm{MC}})$ against the estimated magnitude of the quantity, with one colored marker per policy–reference bank and grey points for individual prompts. For KL and the power-score regrets, variance reduction becomes stronger as the policy–reference divergence decreases, with this dependence becoming more pronounced for larger β. Entropy and Brier loss instead remain comparatively close to the Monte Carlo baseline across banks.

Prompt-level behavior and control variates. Tab. 12 summarizes the per-prompt variance ratios of the potential estimator and its cross-fitted control-variate variant relative to Monte Carlo. It also reports separately the prompts on which the potential alone increases variance, showing how fitting the control-variate coefficient changes their ratios and how often it brings them below one.

	$\widehat{\theta}(\vec{\phi})$ vs. $\widehat{\theta}_{\mathrm{MC}}$		$\widehat{\theta}(\vec{\phi},\alpha^\star)$ vs. $\widehat{\theta}_{\mathrm{MC}}$		prompts with $\mathrm{Var}_p(\widehat{\theta}(\vec{\phi})) > \mathrm{Var}_p(\widehat{\theta}_{\mathrm{MC}})$			
quantity	median [IQR]	$\% < 1$	median [IQR]	$\% < 1$	n	med. $\widehat{\theta}(\vec{\phi})$	med. $\widehat{\theta}(\vec{\phi},\alpha^\star)$ [95% CI]	$\% < 1$
D_{KL}	0.204 [0.061, 0.503]	98.5	0.204 [0.061, 0.482]	99.5	63	1.134	0.974 [0.945, 0.990]	65.1
regret β=1.25	0.202 [0.063, 0.441]	99.0	0.193 [0.063, 0.436]	99.8	40	1.083	0.951 [0.932, 0.973]	77.5
regret β=1.5	0.164 [0.063, 0.400]	99.6	0.160 [0.063, 0.396]	99.9	15	1.058	0.935 [0.891, 1.044]	66.7
regret β=1.75	0.142 [0.062, 0.365]	99.9	0.141 [0.061, 0.363]	99.9	4	1.190	1.080 [0.925, 1.102]	25.0
regret β=2	0.131 [0.060, 0.330]	99.9	0.129 [0.060, 0.329]	99.9	3	1.125	1.040 [1.009, 1.043]	0.0
entropy	0.633 [0.509, 0.796]	99.8	0.630 [0.498, 0.792]	99.5	10	1.047	1.090 [1.050, 1.096]	0.0
Brier loss	0.694 [0.545, 0.903]	99.8	0.698 [0.543, 0.903]	99.5	9	1.040	1.059 [1.024, 1.080]	0.0

Table 12: Per-prompt variance ratios for the losses and divergences experiments, pooling all eight model pairs. We compare the potential estimator $\widehat{\theta}(\vec{\phi})$ and its cross-fitted control-variate variant $\widehat{\theta}(\vec{\phi},\alpha^\star)$ against Monte Carlo. The rightmost columns restrict attention to prompts for which $\widehat{\theta}(\vec{\phi})$ has higher variance than Monte Carlo and show how often the control variate reduces the ratio below one.

C.2.4 Reproducing Amini et al. (2025)

We first reproduce the KL-estimation experiment of Amini et al. (2025) using our sentiment-tuned policy and compare against the estimators considered in their paper. Their term-wise Rao–Blackwellized estimator is recovered as $\widehat{\theta}(\vec{\phi})$, corresponding to a unit coefficient on the associated zero-mean correction. We reproduce its variance reduction over Monte Carlo (Tab. 13). Fitting the control-variate

coefficient reduces variance further: at $M = 1$, the standard deviation decreases from 2.05 to 1.90. Thus, the unit coefficient is not in general variance-optimal.

		std. dev.		
estimator	mean	M=1	M=5	M=10
$\widehat{\theta}_{\mathrm{MC}}$	6.69	3.32	1.49	1.05
HT	6.69	3.32	1.49	1.05
Schulman CV, $\alpha = 1$	6.47	35.67	15.95	11.28
Schulman CV, fitted α	6.36	11.44	5.11	3.61
$\widehat{\theta}(\vec{\phi})$	6.69	2.05	0.92	0.65
$\widehat{\theta}(\vec{\phi}, \alpha^\star)$	6.69	**1.90**	**0.85**	**0.60**

Table 13: Reproduction of the KL-estimation experiment of Amini et al. (2025): mean estimate and empirical standard deviation (nats) at sample sizes $M = 1, 5, 10$, over 512 prompts with 4,000 samples each. The mean does not depend on M. Lowest standard deviation per column in bold.

Counterexample to the variance guarantee of Amini et al. (2025). Theorem 2 of Amini et al. (2025) claims that their term-wise Rao–Blackwellized estimator never has larger variance than Monte Carlo. Consider language models p and q supported on length-two strings, with

$$p(ax) = p(bx) = \tfrac{1}{2}, \qquad q(ax) = q(bx) = \tfrac{1}{4}, \qquad q(bz) = \tfrac{1}{2},$$

and zero probability elsewhere. Under p, $\log \frac{p(Y)}{q(Y)} = \log 2$ for both possible strings, so $\mathrm{Var}_p(\widehat{\theta}_{\mathrm{MC}}) = 0$. Their estimator, corresponding to $\widehat{\theta}(\vec{\phi})$ in our notation, instead equals $\frac{1}{2}\log\frac{4}{3}$ or $\frac{1}{2}\log\frac{4}{3} + \log 3$ with equal probability. It remains unbiased, but

$$\mathrm{Var}_p\big(\widehat{\theta}(\vec{\phi})\big) = \tfrac{1}{4}(\log 3)^2 > 0 = \mathrm{Var}_p(\widehat{\theta}_{\mathrm{MC}}).$$

This also follows directly from Proposition 1: since $\widehat{\theta}_{\mathrm{MC}}$ is constant, $\mathrm{Cov}_p(\widehat{\theta}_{\mathrm{MC}}, \widehat{\theta}_{\mathrm{MC}} - \widehat{\theta}(\vec{\phi})) = 0 < \frac{1}{2}\mathrm{Var}_p(\widehat{\theta}_{\mathrm{MC}} - \widehat{\theta}(\vec{\phi}))$, so the condition for variance reduction fails. Accordingly, Proposition 2 gives $\alpha^\star = 0$, whereas the estimator of Amini et al. (2025) corresponds to $\alpha = 1$.

C.3 Gradients of Losses and Divergences

C.3.1 Potential definitions

Let $\mathbf{w} \in \mathbb{R}^d$ denote the parameters of p, and write $\nabla \overset{\text{def}}{=} \nabla_{\mathbf{w}}$, holding q fixed. The original test function f is scalar-valued, whereas its gradient with respect to $\mathbf{w}$ is d-dimensional. This gives the vector-valued test function

$$\boldsymbol{f}_\nabla(\boldsymbol{Y}) \overset{\text{def}}{=} \nabla f(\boldsymbol{Y}) + f(\boldsymbol{Y})\, \nabla \log p(\boldsymbol{Y}) \in \mathbb{R}^d. \tag{20}$$

The first term differentiates the function value for a fixed string, while the second accounts for the change in its sampling probability. Under standard regularity conditions,

$$\nabla\ \mathbb{E}_p[f(\boldsymbol{Y})] = \mathbb{E}_p[\boldsymbol{f}_\nabla(\boldsymbol{Y})]. \tag{21}$$

Gradient estimation therefore fits our framework by applying the potential estimator to $\boldsymbol{f}_\nabla$ coordinatewise. Given a scalar potential $\vec{\phi}$ for f, we define the corresponding vector-valued gradient potential

$$\vec{\boldsymbol{\phi}}_\nabla(\boldsymbol{x}) \overset{\text{def}}{=} \nabla\vec{\phi}(\boldsymbol{x}) + \vec{\phi}(\boldsymbol{x})\, \nabla \log \vec{p}(\boldsymbol{x}) \in \mathbb{R}^d. \tag{22}$$

Replacing $\vec{\phi}$ by $\vec{\phi}^\star$ gives the corresponding oracle gradient potential.

Because the next-token probabilities in $\mathbf{B}$ also depend on $\mathbf{w}$, the expected gradient increment satisfies

$$\mathbf{B}\Delta\vec{\boldsymbol{\phi}}_\nabla(\boldsymbol{x}) = \nabla\mathbf{B}\Delta\vec{\phi}(\boldsymbol{x}) + \mathbf{B}\Delta\vec{\phi}(\boldsymbol{x})\, \nabla \log \vec{p}(\boldsymbol{x}). \tag{23}$$

We extend the operators Δ and $\mathbf{B}$ to vector-valued potentials coordinatewise. Tab. 14 gives the resulting constructions for the losses in Tab. 1 and 7 to 9. For KL, $\mathbf{B}\Delta\vec{\phi}(\boldsymbol{x})$ is the KL divergence between the next-token distributions, recovering the local gradient expression of Amini et al. (2025).

TABLE 14: GRADIENT OF A LOSS, $\nabla\ \mathbb{E}_p[f(\boldsymbol{Y})]$	
$\boldsymbol{f}_\nabla(\boldsymbol{Y}) = \nabla f(\boldsymbol{Y}) + f(\boldsymbol{Y})\,\nabla \log p(\boldsymbol{Y})$	TEST FUNCTION
$\Delta\vec{\boldsymbol{\phi}}_\nabla(a \mid \boldsymbol{x}) = \nabla\Delta\vec{\phi}(a \mid \boldsymbol{x}) + \Delta\vec{\phi}(a \mid \boldsymbol{x})\,\nabla \log \vec{p}(\boldsymbol{x}a) + \vec{\phi}(\boldsymbol{x})\,\nabla \log \vec{p}(a \mid \boldsymbol{x})$	INCREMENT ($a \in \Sigma$)
$\boldsymbol{f}_\nabla(\boldsymbol{x}\boldsymbol{z}) = \nabla\vec{\phi}(\boldsymbol{x}) + \vec{\phi}(\boldsymbol{x})\,\nabla \log \vec{p}(\boldsymbol{x}) + \nabla\big[f(\boldsymbol{x}\boldsymbol{z}) - \vec{\phi}(\boldsymbol{x})\big] + \big[f(\boldsymbol{x}\boldsymbol{z}) - \vec{\phi}(\boldsymbol{x})\big]\,\nabla \log p(\boldsymbol{x}\boldsymbol{z}) + \vec{\phi}(\boldsymbol{x})\,\nabla \log \vec{p}(\boldsymbol{z} \mid \boldsymbol{x})$	FUNCTION STRUCTURE
$\vec{\boldsymbol{\phi}}^{\star}_\nabla(\boldsymbol{x}) = \nabla\vec{\phi}^{\star}(\boldsymbol{x}) + \vec{\phi}^{\star}(\boldsymbol{x})\,\nabla \log \vec{p}(\boldsymbol{x})$	ORACLE POTENTIAL
$\vec{\boldsymbol{\phi}}_\nabla(\boldsymbol{x}) = \nabla\vec{\phi}(\boldsymbol{x}) + \vec{\phi}(\boldsymbol{x})\,\nabla \log \vec{p}(\boldsymbol{x})$	POTENTIAL
$\Delta\vec{\boldsymbol{\phi}}_\nabla(\text{EOS} \mid \boldsymbol{x}) = \nabla\Delta\vec{\phi}(\text{EOS} \mid \boldsymbol{x}) + \Delta\vec{\phi}(\text{EOS} \mid \boldsymbol{x})\,\nabla \log p(\boldsymbol{x}) + \vec{\phi}(\boldsymbol{x})\,\nabla \log \vec{p}(\text{EOS} \mid \boldsymbol{x})$	EOS INCREMENT
$\mathbf{B}\Delta\vec{\boldsymbol{\phi}}_\nabla(\boldsymbol{x}) = \nabla\mathbf{B}\Delta\vec{\phi}(\boldsymbol{x}) + \mathbf{B}\Delta\vec{\phi}(\boldsymbol{x})\,\nabla \log \vec{p}(\boldsymbol{x})$	EXPECTED INCREMENT

C.3.2 CONTROL-VARIATE COEFFICIENT FOR VECTOR-VALUED FUNCTIONS

For the gradient we need to fit a global control-variate coefficient when the function value lies in $\mathbb{R}^d$. We describe the construction for a general vector-valued test function $\boldsymbol{f}\colon \Sigma^* \to \mathbb{R}^d$ with a potential $\vec{\boldsymbol{\phi}}\colon \Sigma^* \to \mathbb{R}^d$, both read coordinatewise. Let $\widehat{\boldsymbol{\theta}}_{\mathrm{MC}}$, $\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}) \in \mathbb{R}^d$ denote the Monte Carlo and potential estimators for one trajectory, and define the zero-mean correction

$$\boldsymbol{D} \overset{\text{def}}{=} \widehat{\boldsymbol{\theta}}_{\mathrm{MC}} - \widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}). \tag{24}$$

As in Proposition 2, we use a single scalar coefficient, here shared across all d coordinates:

$$\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}, \alpha) \overset{\text{def}}{=} \widehat{\boldsymbol{\theta}}_{\mathrm{MC}} - \alpha\,\boldsymbol{D}. \tag{25}$$

Thus, $\alpha = 0$ recovers Monte Carlo and $\alpha = 1$ the potential estimator. We choose α to minimize the total variance $\operatorname{tr}\operatorname{Var}_p\big(\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}, \alpha)\big)$, giving

$$\alpha^\star = \frac{\operatorname{tr}\operatorname{Cov}_p\big(\widehat{\boldsymbol{\theta}}_{\mathrm{MC}}, \boldsymbol{D}\big)}{\operatorname{tr}\operatorname{Var}_p(\boldsymbol{D})} = \frac{\mathbb{E}_p\Big[\Big\langle \widehat{\boldsymbol{\theta}}_{\mathrm{MC}} - \mathbb{E}_p[\widehat{\boldsymbol{\theta}}_{\mathrm{MC}}],\ \boldsymbol{D} - \mathbb{E}_p[\boldsymbol{D}] \Big\rangle\Big]}{\mathbb{E}_p[\|\boldsymbol{D} - \mathbb{E}_p[\boldsymbol{D}]\|_2^2]}. \tag{26}$$

Given sampled trajectories indexed by prompt i and response j, we estimate this globally as

$$\widehat{\alpha} = \frac{\sum_{i,j} \Big\langle \widehat{\boldsymbol{\theta}}^{(ij)}_{\mathrm{MC}} - \bar{\widehat{\boldsymbol{\theta}}}_{\mathrm{MC}},\ \boldsymbol{D}^{(ij)} - \bar{\boldsymbol{D}} \Big\rangle}{\sum_{i,j} \|\boldsymbol{D}^{(ij)} - \bar{\boldsymbol{D}}\|_2^2}, \tag{27}$$

where the bars denote means over all trajectories. The coefficient is therefore shared across prompts and coordinates. For the gradient experiments we take $\boldsymbol{f} = \boldsymbol{f}_\nabla$ and $\vec{\boldsymbol{\phi}} = \vec{\boldsymbol{\phi}}_\nabla$ from App. C.3.1, with d the number of policy parameters.

C.3.3 EXPERIMENTS AND RESULTS

Marginalization reduces gradient variance across all 135 defined quantity–checkpoint comparisons. We repeat the comparison for the gradient of each quantity with respect to the policy parameters along a DPO training trajectory of the GPT-2 sentiment model (Fig. 5). We evaluate 20 checkpoints using 512 prompts and 20 responses per prompt, and measure the variance of a gradient estimator $\widehat{\boldsymbol{\theta}}$ by $\operatorname{tr}\operatorname{Var}_p(\widehat{\boldsymbol{\theta}})$. Across all 135 quantity–checkpoint comparisons, the potential estimator $\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}_\nabla)$ reduces variance relative to the centered Monte Carlo gradient $\widehat{\boldsymbol{\theta}}_{\mathrm{MC}}$, with variance ratios $\operatorname{tr}\operatorname{Var}_p(\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}_\nabla))/\operatorname{tr}\operatorname{Var}_p(\widehat{\boldsymbol{\theta}}_{\mathrm{MC}})$ between 0.11 and 0.98.

The gains are largest early in training. The gains are largest early in training and generally decrease as training progresses. For example, the KL gradient retains 35% of the Monte Carlo variance at

step 50 and 92% at step 938, while the $\beta = 2$ regret retains 11% and 65%, respectively. Among the power-score regrets, larger β consistently yields larger reductions at every checkpoint.

Fitting $\alpha^\star$ further improves gradients for entropy and Brier score. For some gradients, the control-variate estimator $\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}_\nabla, \alpha^\star)$ provides an additional gain over $\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}_\nabla)$. The unit-coefficient estimator $\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}_\nabla)$ alone barely improves the entropy and Brier gradients, with variance ratios of 0.97–0.98. Their fitted coefficients are substantially larger, $\alpha^\star \approx 5$ for entropy and $\alpha^\star \approx 11$ for Brier, reducing the ratios to 0.91 and 0.86. For the power-score regrets, by contrast, $\alpha^\star$ remains close to one (0.66–0.99), and fitting it provides at most about a further 3% reduction.

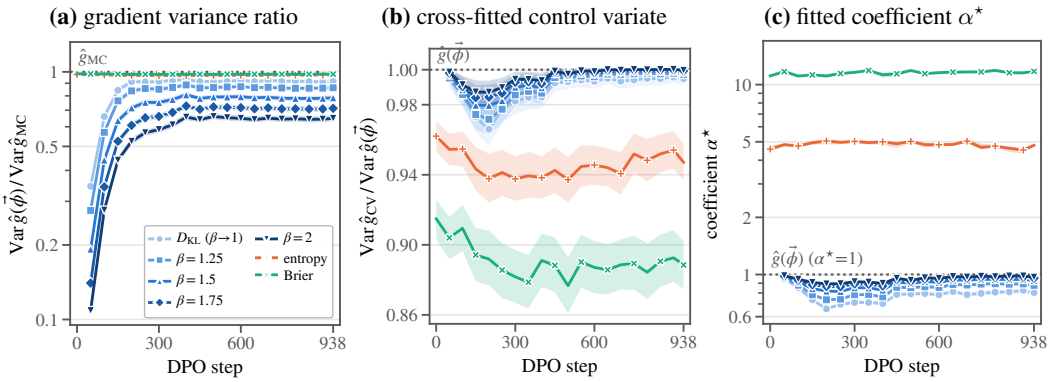


Figure 5: Gradient estimation along a DPO trajectory of the GPT-2 sentiment policy (checkpoints every 50 steps; the variance of an estimator $\widehat{\boldsymbol{\theta}}$ is $\mathrm{tr}\,\mathrm{Var}_p(\widehat{\boldsymbol{\theta}})$ over the 512×20 responses). **(a)** Variance of $\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}_\nabla)$ relative to the centered Monte Carlo gradient $\widehat{\boldsymbol{\theta}}_{\mathrm{MC}}$, one series per quantity (regrets darker for larger β; undefined at step 0). **(b)** Variance of the cross-fitted control-variate estimator $\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}_\nabla, \alpha^\star)$ relative to $\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}_\nabla)$. **(c)** The cross-fitted coefficient, mean over 8 prompt folds with the range as band; $\alpha^\star = 1$ recovers $\widehat{\boldsymbol{\theta}}(\vec{\boldsymbol{\phi}}_\nabla)$. Bands in (a) and (b) are 95% prompt-cluster bootstrap intervals; one training run and one seed.

C.4 n-GRAM COUNTS AND FREQUENCIES

C.4.1 DEFINITION OF FREQUENCY ESTIMATOR

For n-gram statistics one is often interested in the frequency of a target rather than solely its expected count. Let $f(\boldsymbol{Y}) = \#\{\text{occurrences of } w \text{ in } \boldsymbol{Y}\}$ be the count, the test function of Tab. 2, and $N(\boldsymbol{Y}) \overset{\text{def}}{=} \max\{|\boldsymbol{Y}| - n + 1, 0\}$ the number of n-gram slots in $\boldsymbol{Y}$. The frequency of a substring w made of n orthographic words is $\theta_{\mathrm{FR}} \overset{\text{def}}{=} \mathbb{E}[f(\boldsymbol{Y})]/\,\mathbb{E}[N(\boldsymbol{Y})]$, which we estimate with the ratio estimators

$$\widehat{\theta}_{\mathrm{FR}} \overset{\text{def}}{=} \frac{\widehat{\theta}_{\mathrm{MC}}}{\frac{1}{M}\sum_{m=1}^{M} N(\boldsymbol{Y}^{(m)})}, \qquad \widehat{\theta}_{\mathrm{FR}}(\vec{\phi}) \overset{\text{def}}{=} \frac{\widehat{\theta}(\vec{\phi})}{\frac{1}{M}\sum_{m=1}^{M} N(\boldsymbol{Y}^{(m)})}, \tag{28}$$

where $\widehat{\theta}_{\mathrm{MC}}$ and $\widehat{\theta}(\vec{\phi})$ are the Monte Carlo and potential estimators of $\mathbb{E}[f(\boldsymbol{Y})]$ built from Tab. 2, and the denominator is the Monte Carlo estimate of $\mathbb{E}[N(\boldsymbol{Y})]$ over the same M generations. Ratio estimators are biased at finite sample size, but the bias vanishes asymptotically under standard regularity conditions (Cochran, 1977).

C.4.2 DEFINITION OF COST-ADJUSTED VARIANCE RATIO

For each target w, we compare the potential estimator against Monte Carlo at equal sample size and equal compute. For either the expected count or frequency, we define

$$\widehat{\rho}(w) = \frac{\widehat{\mathrm{Var}}\big(\text{potential estimator at } M = 4{,}000\big)}{1000\,\widehat{\mathrm{Var}}\big(\text{Monte Carlo estimator at } M = 4 \times 10^6\big)}, \qquad \widehat{\rho}_c(w) = L_w\,\widehat{\rho}(w). \tag{29}$$

The factor 1000 rescales the Monte Carlo variance from 4×10^6 to 4,000 samples. This rescaling is exact for expected counts, whose estimator is a sample mean. For frequencies, which are ratios of

sample means, the same $1/M$ variance scaling holds asymptotically by the multivariate delta method (**?**, §1.3).

The cost-adjusted variance ratio $\widehat{\rho}_c(w)$ additionally accounts for the cost of marginalization. For a target whose canonical tokenization has L_w tokens, the potential estimator requires approximately L_w times the model evaluation cost per sampled sequence. We can therefore read $\widehat{\rho}_c(w)$ as the fraction of Monte Carlo compute the potential estimator needs to reach the same variance.

C.4.3 Observed-prefix potential

The Z^{end} decomposition gives the observed-prefix potential $\vec{\phi}(\boldsymbol{x}) = f(\boldsymbol{x})$ of Tab. 15. Its expected increment is nonzero only at prefixes that already match the substring up to its final token, so the estimator averages over that token alone. Tab. 16 reports the median variance of the resulting estimator relative to Monte Carlo at equal sample size ($M = 4{,}000$) for the unigram targets of GPT-2, banded by their occurrences in the 4,000 samples (very rare 1–9, rare 10–99, moderate 100–999, frequent $\geq 1{,}000$;

Table 15: Substring counts, observed-prefix potential
$f(\boldsymbol{Y}) = \#\{\text{occurrences of } \boldsymbol{w} \text{ in } \boldsymbol{Y}\}$
$f(\boldsymbol{Y}) = \sum_t Z^{\text{end}}_{\boldsymbol{w},t},\ Z^{\text{end}}_{\boldsymbol{w},t} = \mathbf{1}\{\boldsymbol{w} \text{ ends at } t\}$
$\vec{\phi}^{\star}(\boldsymbol{x}) = \sum_{t \leq \lvert\boldsymbol{Y}\rvert} \mathbb{E}_p\big[Z^{\text{end}}_{\boldsymbol{w},t} \mid \boldsymbol{Y} \succeq \boldsymbol{x}\big]$
$\vec{\phi}(\boldsymbol{x}) = f(\boldsymbol{x}) = \sum_{t \leq \lvert\boldsymbol{x}\rvert} Z^{\text{end}}_{\boldsymbol{w},t}$
$\Delta\vec{\phi}(a \mid \boldsymbol{x}) = \mathbf{1}\{\boldsymbol{w} \text{ ends at } \lvert\boldsymbol{x}\rvert + 1 \text{ in } \boldsymbol{x}a\}$
$\Delta\vec{\phi}(\text{EOS} \mid \boldsymbol{x}) = 0$
$\mathbf{B}\Delta\vec{\phi}(\boldsymbol{x}) = \Pr_p\big[\boldsymbol{w} \text{ ends at } \lvert\boldsymbol{x}\rvert + 1 \mid \boldsymbol{Y} \succeq \boldsymbol{x}\big]$

Table 16: Variance of the observed-prefix estimator relative to Monte Carlo at equal sample size ($M = 4{,}000$): median over the unigram targets of GPT-2, banded by their occurrence count in the 4,000 samples (columns).

model	1–9	10–99	100–999	$\geq 1{,}000$
GPT-2 (854)	0.997	0.996	0.993	0.997

C.4.4 Results for expected counts

Results for expected counts. Fig. 6 repeats Fig. 2 for the expected count $\theta_w = \mathbb{E}[C_w]$ instead of the frequency; the three paragraphs below mirror those of the main text, and support qualitatevely the same conclusions given by the frequency results.

Half the targets reach equal variance on less than 0.6% of the Monte Carlo compute. Fig. 6a shows the cost-adjusted variance ratio $\widehat{\rho}_c$ for the 4,149 targets whose ratio can be estimated with relative standard error at most 2 (3,700 for frequencies), together with the same per-target 95% confidence intervals on the log scale. Interpreting $\widehat{\rho}_c$ as the fraction of Monte Carlo compute required to attain the same variance, half of the targets require at most 0.55% of the Monte Carlo budget, and 82.3% require at most 10%. These gains remain large after accounting for estimation uncertainty: replacing each ratio by the upper bound of its 95% interval, half of the targets still require at most 1.26% of the Monte Carlo budget, 83.7% require at most 20%, and 98.0% remain below parity with Monte Carlo.

The gains are largest for rare targets. We restrict to the same 2,831 targets as for frequencies, those whose count estimates have nominal RSE at most 20% under a Poisson approximation, equivalently at least 25 expected occurrences in the independent 2×10^6-sample frequency bank. The compute ratio increases strongly with the expected count per document (Spearman $r_s = 0.85$), with its median rising from 1.6×10^{-3} in the rarest decade to 0.62 among targets with one to ten expected occurrences per document (Fig. 6b).

At the same sample size ($M = 4{,}000$), the potential estimator yields substantially more precise count estimates, with 79.7% of targets reaching RSE $\leq 20\%$ and 96.5% reaching RSE $\leq 50\%$, compared with 11.4% and 19.0% for Monte Carlo (Fig. 6c). The gap grows with token length: for targets of length 4 or more, Monte Carlo almost never reaches RSE $\leq 50\%$, whereas the potential estimator does so for at least 90% of targets in every length group. For $L_w \geq 7$, Monte Carlo does not observe any target even in the 4×10^6-sample reference bank, while the potential estimator gives a finite estimate for all of them. Every share agrees with the corresponding share for frequencies to within 0.14 percentage points.

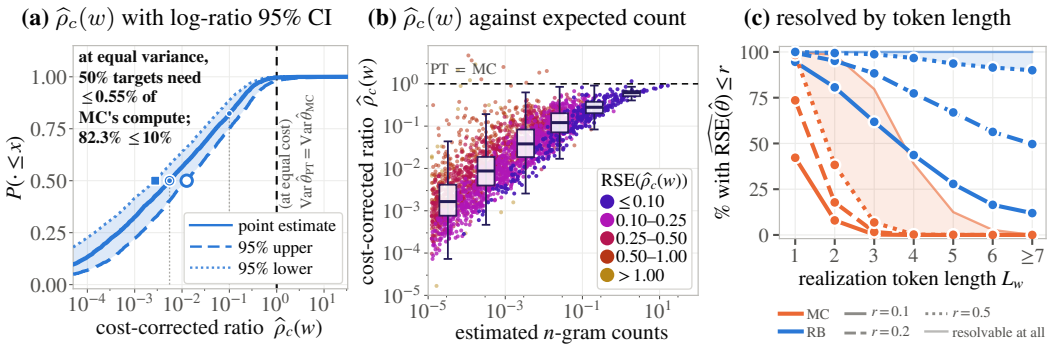
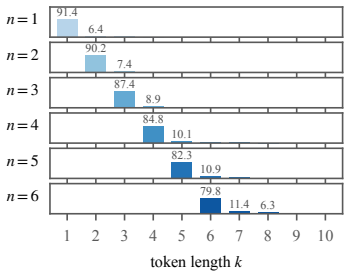


Figure 6: **Fig. 2 for expected counts.** Cost-adjusted variance ratio $\widehat{\rho}_c(w)$ for n-gram counts estimation; $\widehat{\rho}_c(w) < 1$ means that the potential estimator requires less compute than Monte Carlo to attain the same variance. **(a)** Empirical cumulative distribution of $\widehat{\rho}_c(w)$ across targets, together with 95% confidence bounds. **(b)** Ratio versus estimated n-gram counts, illustrating how variance reduction changes with target rarity. **(c)** Fraction of targets whose frequency estimate reaches a given relative-standard-error threshold, grouped by realization token length, for Monte Carlo and the potential estimator. Dashed reference lines indicate equal performance at equal compute.

(a) targets and token lengths per order n

n	N	min	max	mean	med
1	869	1	4	1.110	1
2	1584	2	5	2.128	2
3	1568	3	6	3.174	3
4	1391	4	8	4.219	4
5	1206	5	9	5.266	5
6	1029	6	10	6.320	6
all	7647	–	–	–	–

(b) share of targets at each L_w, per order



(c) same, log scale: tails thin alike across n

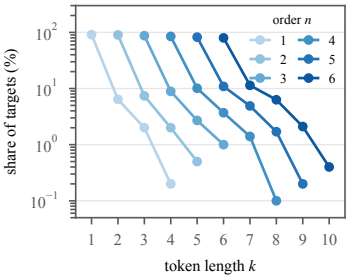


Figure 7: **n-gram target inventory and token-length distribution.** The n-gram task provides a controlled way to study targets of different lengths and, consequently, different marginalization costs, which scale with the number of GPT-2 tokens L_w in the target. (a) Number of targets at each order n, together with the minimum, maximum, mean, and median L_w of their canonical realizations. The inventory contains 7,647 targets in total. (b) Distribution of token lengths within each order. Increasing n shifts the inventory toward longer targets, while GPT-2 tokenization also induces within-order variation in L_w. (c) The same distributions on a log scale, showing that each order additionally contains a progressively thinning tail of targets requiring more tokens, and therefore more costly marginalization.

C.4.5 Additional diagnostics

The variance ratios of Fig. 2 are point estimates, so we also ask how many targets are separated from parity once their own estimation error is taken into account. Tab. 18 reports this across cumulative relative-standard-error subsets.

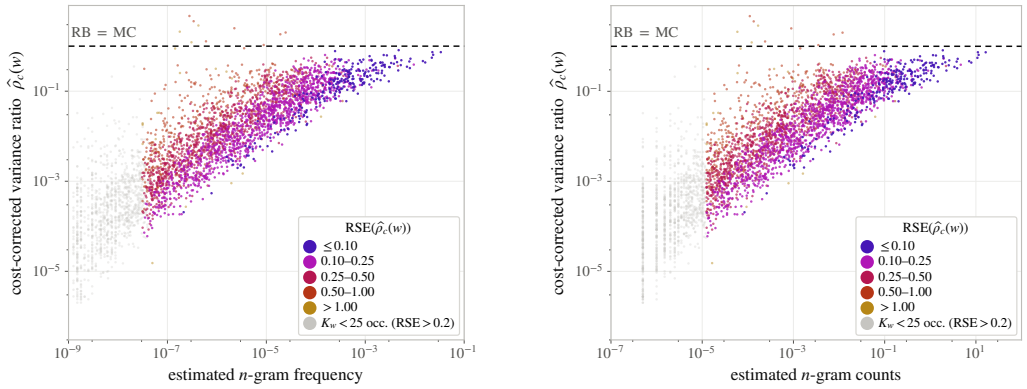


Figure 8: **Cost-adjusted variance ratio against target rarity, including all targets.** We show the same scatter plots as in the main figure, against estimated n-gram frequency (left) and expected count (right), but without filtering targets by the precision of the independently estimated x-axis quantity. In the main figure, we retain only targets whose independent frequency estimate has RSE at most 20%, corresponding to at least 25 expected occurrences in the estimation sample, to restrict the analysis to more reliable estimates. Here, we show the full set of targets for completeness.

Table 17: **Variance reduction is strongest for rare targets, even within fixed n-gram order and token length.** The 2,831 targets of Fig. 2b are grouped by their independently estimated frequency $\widehat{\theta}_{\mathrm{FR}}$. Each entry reports the median cost-adjusted variance ratio $10^3\widehat{\rho}_c$ within that frequency bin; lower values indicate larger gains over Monte Carlo. **(a)** The trend across all targets. **(b–c)** The same comparison within fixed n-gram order and canonical token length L_w, showing that the rarity trend persists after controlling for target length and marginalization cost. The bottom rows give the number of targets in each column and Spearman's correlation r_s between $\widehat{\theta}_{\mathrm{FR}}$ and $\widehat{\rho}_c$. Cells with fewer than 8 targets are omitted.

	(a) all targets		**(b)** n-gram order				**(c)** token length L_w			
frequency $\widehat{\theta}_{\mathrm{FR}}$	# targets	$10^3\widehat{\rho}_c$	1	2	3	4	1	2	3	4
$[10^{-8}, 10^{-7})$	354	1.1	1.5	1.0	1.2	1.0	–	1.1	1.1	1.0
$[10^{-7}, 10^{-6})$	701	4.2	4.6	4.4	3.9	4.1	–	4.4	4.0	4.1
$[10^{-6}, 10^{-5})$	725	18	18	18	19	14	21	17	19	14
$[10^{-5}, 10^{-4})$	684	75	78	74	64	–	78	74	64	–
$[10^{-4}, 10^{-3})$	299	170	170	170	–	–	170	170	–	–
$[10^{-3}, 10^{-1})$	68	370	370	350	–	–	370	350	–	–
targets	2,831	–	865	1,295	537	115	794	1,307	587	124
Spearman r_s	–	0.83	0.71	0.80	0.73	0.69	0.67	0.78	0.74	0.69

Table 18: **Variance reduction remains robust after accounting for uncertainty in the estimated variance ratio.** Each row restricts the analysis to targets whose estimated cost-adjusted variance ratio $\widehat{\rho}_c$ has relative standard error (RSE) at most the indicated cutoff (Max. RSE). For expected-count and frequency estimation, we report the number of retained targets and the percentage for which the approximate 95% confidence interval is guaranteed to lie entirely below 1, the equal-compute Monte Carlo baseline.

	Expected count		Frequency	
Max. RSE	# targets	95% CI below 1	# targets	95% CI below 1
5%	84	98.8%	41	100.0%
10%	317	98.1%	240	100.0%
20%	1,048	98.8%	953	99.6%
50%	2,661	97.9%	2,583	99.0%

C.5 Event Analysis of (Dorman et al., 2026)

C.5.1 The Automated Readability Index

We extend the experiments of (Dorman et al., 2026) that estimate the Automated Readability Index. For a string $\boldsymbol{Y}$ with $c(\boldsymbol{Y})$ characters, $w(\boldsymbol{Y})$ words, and $s(\boldsymbol{Y})$ sentences,

$$\mathrm{ARI}(\boldsymbol{Y}) = 4.71\,\frac{c(\boldsymbol{Y})}{w(\boldsymbol{Y})} + 0.5\,\frac{w(\boldsymbol{Y})}{s(\boldsymbol{Y})} - 21.43, \tag{30}$$

a score calibrated so that its value roughly matches the US school grade needed to read the text. The events D_j are bins of this score, and the test functions are the indicators $\mathbf{1}\{\mathrm{ARI}(\boldsymbol{Y}) \in D_j\}$.

Experimental setup. The experiments follow the authors' setup. The model TinyStories-8M continues a fixed prompt for 100 tokens, and the target is the probability that the completion's ARI falls in each of 80 bins of width 0.29 spanning $[-8, 15]$. The model is tilted by $\exp(-\lambda\,\mathrm{ARI})$ at 19 values of λ; each tilt is sampled by transition-path sampling (200 chains of 32,400 retained states) and complemented by 200,000 direct samples, and the resulting 6.68×10^6 states are combined with MBAR weights, which we retain unchanged. Folds are whole chains, stratified by tilting schedule.

C.5.2 Potential definitions

The events are the $J = 80$ readability bins D_j of App. C.5.1. For each bin we learn a potential $\vec{\phi}_j(\boldsymbol{x};\boldsymbol{\eta}) = g_j(\boldsymbol{\xi}(\boldsymbol{x});\boldsymbol{\eta})$ of the form of Eq. (13), approximating the oracle $\Pr_p[\mathrm{ARI}(\boldsymbol{Y}) \in D_j \mid \boldsymbol{Y} \succeq \boldsymbol{x}]$. The prefix features $\boldsymbol{\xi}(\boldsymbol{x})$ collect the character, word, and sentence counts of $\boldsymbol{x}$ together with its position in the generation window. We consider two models, a logistic model that predicts bin probabilities directly and a mixture model that forecasts the counts underlying the ARI; both are fitted by the cross-fitting scheme introduced in §4.3, with every fitted quantity estimated from the training folds alone.

Logistic potential. The logistic model predicts bin probabilities from the prefix features directly,

$$\vec{\phi}_j^{\log}(\boldsymbol{x};\boldsymbol{\eta}) = \frac{\widehat{p}_j \exp\big(\boldsymbol{\beta}_j^\top \boldsymbol{\xi}(\boldsymbol{x})\big)}{\sum_{j'=1}^{J} \widehat{p}_{j'} \exp\Big(\boldsymbol{\beta}_{j'}^\top \boldsymbol{\xi}(\boldsymbol{x})\Big)}, \qquad \boldsymbol{\eta} = (\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_J), \tag{31}$$

where $\widehat{p}_j$ is the marginal probability of bin D_j estimated from the training data. The feature vector $\boldsymbol{\xi}(\boldsymbol{x})$ here comprises 11 features: an intercept, the normalized prefix position and its square, the inverse remaining length and its inverse square root, the character, word, and sentence counts, and the interactions between each count and the normalized position, so that the counts can play different roles at different stages of generation. We fit $\boldsymbol{\eta}$ by minimizing MBAR-weighted cross-entropy.

Mixture-model potential. At each prefix, we linearly extrapolate the character and word counts in $\boldsymbol{\xi}(\boldsymbol{x})$ to the terminal length. We separately model the number k of additional sentence endings, since a single further sentence break can change the ARI substantially. Each value of k gives a predicted ARI $\mu_k(\boldsymbol{x})$, around which we place a Gaussian with standard deviation $\sigma_k(\boldsymbol{x})$. Combining these components with their predicted probabilities $\pi_k(\boldsymbol{x})$ gives a Gaussian mixture law for the terminal $\mathrm{ARI}(\boldsymbol{Y})$ of the completed string, and the potential is the mass this law assigns to the bin $D_j = [l_j, u_j)$, available in closed form through the standard normal distribution function Φ:

$$\begin{aligned} q_{\mathrm{mix}}(\,\cdot \mid \boldsymbol{x}) &= \sum_k \pi_k(\boldsymbol{x})\,\mathcal{N}\big(\mu_k(\boldsymbol{x}), \sigma_k^2(\boldsymbol{x})\big)\,, \\ \vec{\phi}_j^{\mathrm{mix}}(\boldsymbol{x};\boldsymbol{\eta}) &= q_{\mathrm{mix}}(D_j \mid \boldsymbol{x}) = \sum_k \pi_k(\boldsymbol{x}) \left[\Phi\left(\frac{u_j - \mu_k(\boldsymbol{x})}{\sigma_k(\boldsymbol{x})}\right) - \Phi\left(\frac{l_j - \mu_k(\boldsymbol{x})}{\sigma_k(\boldsymbol{x})}\right)\right]. \end{aligned} \tag{32}$$

Here $\boldsymbol{\eta}$ collects the extrapolation rates, mixture weights, and Gaussian widths. We fit these parameters on the combined direct and transition-path samples, giving each retained sample equal weight. To recover target-model bin probabilities, we multiply the fitted mixture density by the MBAR weight as a function of terminal ARI, integrate the resulting weighted density over each bin using fine-grid quadrature, and normalize once across the full mixture. We cap the number of additional sentence endings at 14, a heuristic upper bound for the 100-token completions used in our experiments.

C.5.3 POTENTIAL SELECTION ALGORITHM

For each potential $\vec{\phi}$, we use the control-variate estimator of **??**. The coefficient α may either be fixed a priori or estimated from data to minimize variance. In particular, $\alpha = 1$ yields the potential estimator, while $\alpha = 0$ recovers the baseline estimator regardless of the potential. Alg. 1 gives the nested cross-fitting procedure used in §4.3 to select both the potential and, when learned, its coefficient using held-out folds.

Algorithm 1 Potential selection by nested cross-fitting

Input: Sampled strings $\mathcal{D}$ with stored next-token distributions; target function f.
Input: P candidate potentials, each fixed or learned, with fitting procedures for learned parameters.
Input: For each candidate, a fixed coefficient α (default: 1) or a coefficient-fitting procedure.
Input: Outer and inner fold counts K and L.
Output: Estimate $\widehat{\theta}$ of $\mathbb{E}_p[f(Y)]$.

Split $\mathcal{D}$ into K equal-sized outer folds $\mathcal{D}_1, \dots, \mathcal{D}_K$.
for $k = 1, \dots, K$ **do**
  $\mathcal{T} \leftarrow \mathcal{D} \setminus \mathcal{D}_k$ ▷ *Outer training data*
  Split $\mathcal{T}$ into L equal-sized inner folds $\mathcal{V}_1, \dots, \mathcal{V}_L$.
  for $j = 1, \dots, P$ **do**
    for $\ell = 1, \dots, L$ **do**
      $\mathcal{A} \leftarrow \mathcal{T} \setminus \mathcal{V}_\ell$
      Fit candidate j's potential on $\mathcal{A}$, if needed, to obtain ϕ.
      Set α to its fixed value, or fit it on $\mathcal{A}$.
      Evaluate $\widehat{\theta}(y_i; \phi, \alpha)$ for every $y_i \in \mathcal{V}_\ell$.
      $v_{j\ell} \leftarrow$ sample variance of these estimates.
    $s_j \leftarrow \frac{1}{L} \sum_{\ell=1}^{L} v_{j\ell}$ ▷ *Validation score*
  $j^\star \leftarrow \arg\min_{j=1,\dots,P} s_j$
  Refit the selected potential on $\mathcal{T}$, if needed, to obtain ϕ_k.
  Set α_k to the selected candidate's fixed value, or refit it on $\mathcal{T}$.
  $\widehat{\theta}_k \leftarrow \dfrac{1}{|\mathcal{D}_k|} \displaystyle\sum_{y_i \in \mathcal{D}_k} \widehat{\theta}(y_i; \phi_k, \alpha_k)$ ▷ *Held-out estimate*
return $\widehat{\theta} = \dfrac{1}{K} \displaystyle\sum_{k=1}^{K} \widehat{\theta}_k$

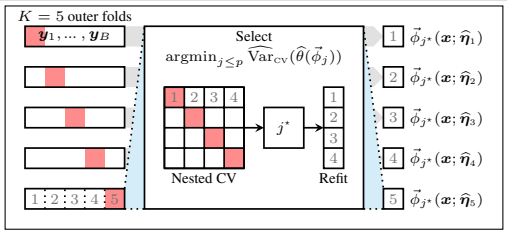


Figure 9: Nested cross-fitting, our in-sample selection among p candidate potentials (Alg. 1). Each of $K = 5$ outer folds is held out in turn (red); on the remaining folds the candidates are compared by inner cross-validation, the winner $j^\star$ is refitted on all remaining data and applied to the held-out fold.

C.5.4 Compute costs and additional results

Compute. All potential fitting, scoring, and selection run on CPUs, with no GPU computation beyond sampling. The logistic potential has 880 coefficients (11 features across 80 bins), while the mixture model stores empirical rates, mixture weights, and residual widths. We use five outer folds, with the four remaining folds serving as the inner folds. Each inner fit uses three folds: exchanging the outer held-out fold and the inner validation fold leaves the training set unchanged. Thus, the $5 \times 4 = 20$ inner evaluations require only $\binom{5}{2} = 10$ distinct fits, which we reuse across outer splits. Together with 5 outer refits and 1 full-data fit, this gives 16 parameter sets per model. Fitting these takes 14 minutes for logistic regression and 4 minutes for the mixture model on a single 32-core node. Scoring both potentials using the stored next-token distributions requires ten parallel jobs, each using 32 cores for 50–95 minutes.

Additional results. Fig. 9 sketches the selection procedure and Fig. 10 shows the variance of the selected estimator over the bins on which MBAR's own variance is well measured, Fig. 11 the same view for each single candidate, Fig. 12 extends the comparison to all 64 populated bins with the accompanying diagnostics, and Tab. 19 lists the chain-level variance, sample-size efficiency, and MBAR-equivalent sample size of every estimator.

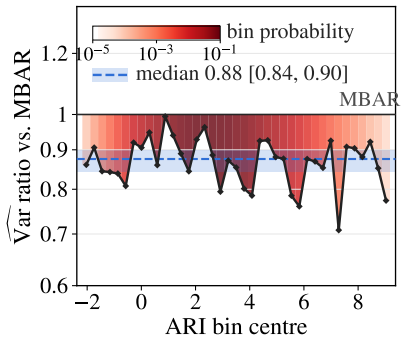


Figure 10: Variance ratio of the selected potential on the event-analysis experiment: chain-level variance of the cross-fitted estimator relative to MBAR, per ARI bin, over the 39 bins 20–58: the 38 bins on which MBAR's own variance is itself measured to within 20%, plus bin 57 (at 25%), kept so that the range is contiguous; shading gives the bin probability, the dashed line the median (0.88) and the band its 95% interval. All 64 populated bins are in Fig. 12.

Estimator	Var. ratio vs. MBAR [95% CI]	Sample-size eff. [95% CI]	MBAR-equiv. samples ($\times 10^6$) [95% CI]
Nested selection	0.88 [0.86, 0.91]	1.14 [1.10, 1.17]	7.58 [7.34, 7.80]
Terminal	0.92 [0.90, 0.94]	1.09 [1.06, 1.11]	7.29 [7.09, 7.40]
Terminal, α	0.92 [0.90, 0.95]	1.09 [1.06, 1.11]	7.27 [7.05, 7.40]
Logistic	0.90 [0.86, 0.91]	1.12 [1.09, 1.16]	7.46 [7.31, 7.75]
Mixture model	0.87 [0.84, 0.90]	1.15 [1.11, 1.19]	7.69 [7.43, 7.92]

Table 19: Event analysis: chain-level variance of each estimator relative to MBAR, median over all 64 populated ARI bins, with 95% intervals from 1,000 paired chain-bootstrap replicas (fitted potentials, selections, and coefficients held fixed). Sample-size efficiency: median over bins of $\mathrm{Var}(\mathrm{MBAR})/\mathrm{Var}(\mathrm{estimator})$; MBAR-equivalent samples: that factor times the actual sample size $N = 6.68 \times 10^6$ states.

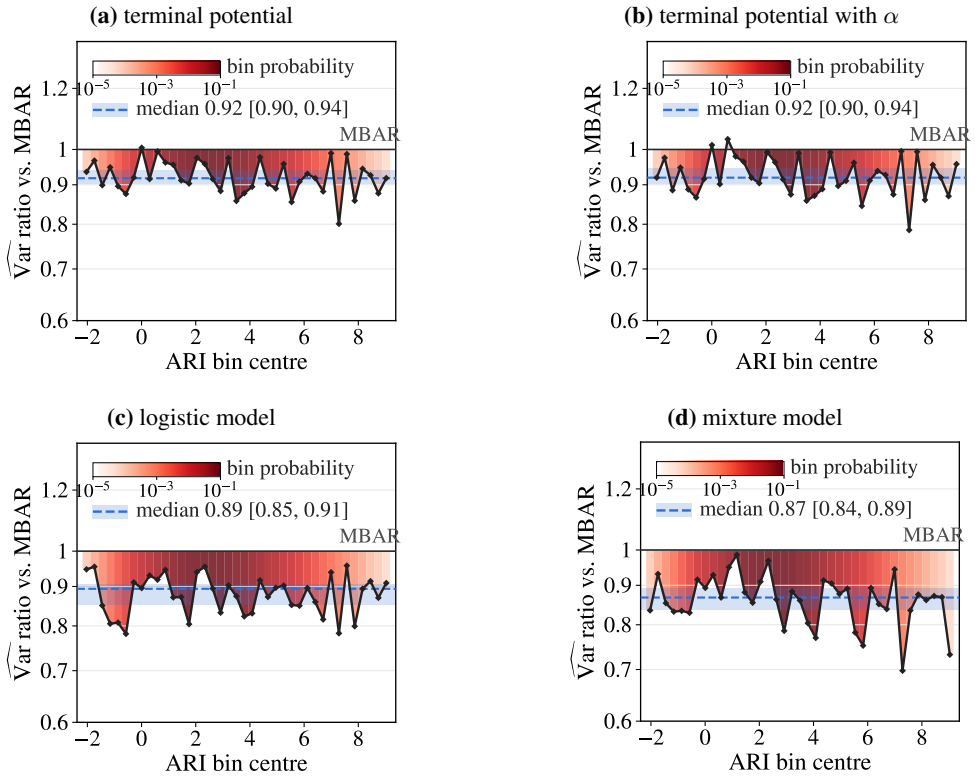


Figure 11: The same view as Fig. 10 for each single candidate, each refitted without the held-out fold and applied to every bin: chain-level variance relative to MBAR per ARI bin over the same 39 bins (38 by the 20% rule plus bin 57), with the bin probability as shading and the candidate's own median over these bins with its 95% interval. The terminal potential and its calibrated version remove 8%, the logistic model 11%, and the mixture model 13%, against 12% for the selection, which picks the mixture model in four folds and the logistic model in one.

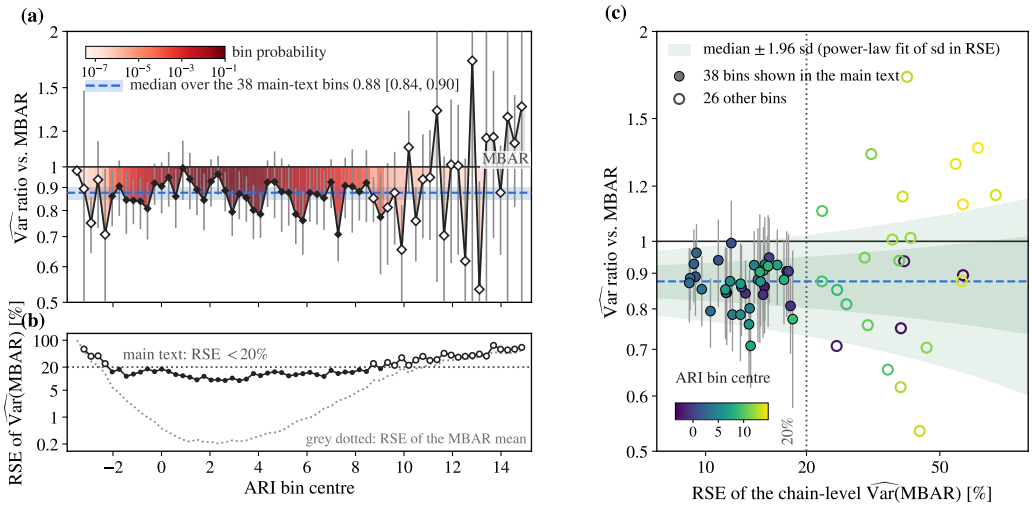


Figure 12: Variance reduction across ARI bins. **(a)** variance of the cross-fitted estimator relative to MBAR for all 64 populated bins. Filled markers denote the 38 bins for which the MBAR variance is estimated with RSE below 20%; over these bins, the median variance ratio is 0.88 with a 95% bootstrap interval of $[0.84, 0.90]$, and over all 64 populated bins it is 0.88 with interval $[0.86, 0.91]$. Ratios in the remaining tail bins are much noisier because the MBAR variance itself is poorly estimated. Whiskers show ± 1.96 bootstrap standard deviations from 1,000 paired chain-bootstrap replicates. **(b)** RSE of the MBAR variance and mean estimates across bins, showing that uncertainty in the variance estimate grows sharply in the tails. **(c)** Variance ratios against the RSE of the MBAR variance estimate. Among the 38 reliable bins, the ratio shows no systematic dependence on baseline uncertainty (Spearman $\rho = 0.05$, $p = 0.78$), indicating that the observed reduction is not driven by noise in the MBAR variance estimate.