Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

ChiselMark: Efficient and Detectable Watermarking for Large Language Models under Speculative Sampling

Shizhan Cai
Nanyang Technological University
Yiming Li
liyiming.tech@gmail.com
Nanyang Technological University
Yongcheng Jing
Nanyang Technological University
Siyuan Liang
Nanyang Technological University
Dacheng Tao
Nanyang Technological University

Article

Keywords:

Posted Date: November 17th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11525044/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: Competing interest reported. Dacheng Tao serves as a Guest Editor of the *npj Artificial Intelligence* Collection “Securing the Core: Preventing Unauthorized Use and Manipulation of Large Language Models.” Yiming Li has previously collaborated with Bo Li, who also serves as a Guest Editor of the same Collection. These relationships are disclosed as non-financial competing interests. All other authors declare no competing interests.

ChiselMark: Efficient and Detectable Watermarking for Large Language Models under Speculative Sampling

Shizhan Cai **Yiming Li*** **Yongcheng Jing** **Siyuan Liang** **Dacheng Tao**

College of Computing and Data Science
Nanyang Technological University
Singapore, Singapore

shizhan001@e.ntu.edu.sg {liyiming.tech,yongchengjing}@gmail.com
siyuan.liang@ntu.edu.sg dacheng.tao@gmail.com

*Corresponding author: liyiming.tech@gmail.com

Abstract

Generation-time watermarking has become a central safeguard for tracing the outputs of hosted large language models (LLMs) and protecting providers' intellectual property against unauthorized use, yet it must coexist with the increasingly optimized systems through which these models are deployed in production. One such system is speculative sampling, a widely used paradigm for accelerating LLM generation that is now embedded in most production LLM-serving stacks. Here, we study KGW-style logit-reweighting watermarks under speculative sampling and show that straightforward integration systematically erodes the acceleration that speculative sampling is meant to deliver, even when the intended watermarked output distribution is preserved at the token level. We trace this erosion to a latent assumption that current KGW-style methods inherit from standard autoregressive decoding, namely that the distribution being reweighted is also the one whose statistics determine the cost of a write. Under speculative sampling this assumption breaks, because the cost of a write is paid by the proposal side, by the draft distribution at the token level and by the already-constructed candidate tree at the feature level, rather than by the reweighted target. Guided by this understanding, we introduce ChiselMark, a sparse, cost-aware, and architecture-aware watermarking framework that decomposes watermark strength into three distinct controls: where the watermark may be written, whether a particular write is predicted to be locally affordable, and at which stage of the decoding architecture the write enters, while remaining detectable from the generated text, a secret key, and public detector parameters. Across token-level speculative sampling, Medusa, and three generations of EAGLE on multiple model scales and tasks, ChiselMark recovers much of the speculative efficiency lost to dense watermarking while keeping detectability competitive with existing watermarks, and retains detectability under text edits. These findings indicate that practical mechanisms for LLM traceability and controlled deployment should be co-designed with the inference architectures on which they operate, rather than treated as orthogonal interventions at the next-token distribution.

1 Introduction

Large language models (LLMs) have become critical assets whose value depends on massive investments in training data, computation and engineering, with providers typically monetizing frontier models through hosted APIs [1, 2]. However, this exposure also opens the door to unauthorized appropriation. For instance, frontier model providers have recently reported

competing services silently routing user requests through the providers' APIs and presenting the outputs as their own [3]. Although emerging regulations and legislative frameworks increasingly target such unfair competition and unauthorized capability harvesting as model theft [4, 5, 6, 7], enforcing these protections faces an evidentiary bottleneck: without a reliable way to attribute disputed text to its source model, infringement claims can neither be substantiated nor refuted. Output traceability has emerged as an essential safeguard for the intellectual property and accountable deployment of hosted LLMs [8, 9, 10].

A mainstream approach to this problem is *generation-time text watermarking*: at each decoding step, a secret key embeds a statistical signal into the next-token distribution, which is subsequently verified on the generated text through a keyed statistical test [9, 10, 11]. Among these schemes, the KGW method proposed by Kirchenbauer et al. [12] and its variants are the most widely used: at each step, a secret key and the preceding context pseudorandomly partition the vocabulary into a green list and a red list, a small bias δ is added to the logits of green-list tokens, and detection later tests the proportion of green tokens in the text via a z-test. KGW is popular because it acts solely on the next-token logits, requiring neither model retraining nor modifications to the model, so that it can be plugged directly into existing decoding pipelines [13]. Subsequent variants further refine where and how strongly this bias is applied, or how token evidence is weighted during detection [14, 15, 16]. Notably, these KGW-style schemes primarily design their watermark interventions around the next-token distribution, without explicitly accounting for the proposal-side cost introduced by speculative sampling.

Meanwhile, inference acceleration has become mainstream in the industrial LLM serving stack. *Speculative sampling* uses a lightweight draft model to propose several candidate tokens that the target model verifies in a single parallel forward pass, accepting proposals up to the first rejected position [17, 18]; more recent feature-level methods draft directly from the hidden states of the target model to propose a candidate tree for parallel verification, as in Medusa [19] and the EAGLE family [20, 21, 22]. It is widely adopted because it delivers multi-fold speedups without modifying the target model, and the verification rules of token-level speculative sampling and the EAGLE family provably preserve the output distribution; Google, Meta, DeepSeek, and GLM have all deployed it in their products [23, 24, 25, 26]. However, watermarking can interfere with the proposal reuse underlying speculative acceleration, by altering draft–target agreement at the token level or candidate-tree construction and verification at the feature level. This raises a concrete question: *how can we watermark outputs with a KGW-style scheme under speculative sampling so that it remains reliable while the acceleration is largely preserved?*

To the best of our knowledge, post-hoc text watermarks [27] forfeit the lightweight advantage of generation-time watermarking and are not a viable path; existing studies combining watermarking with speculative sampling either examine only unbiased watermarks under token-level speculative sampling [28, 29] or are tied to the speculative decoder of a specific production system [30], and neither covers distribution-shifting watermarks such as KGW under tree-based speculative sampling. Arguably, the most straightforward approach is to apply the same KGW watermark simultaneously to both the draft and target distributions. We prove that this vanilla construction preserves the intended watermarked output distribution at the token level and show how analogous reweighting can be applied to tree-based decoders, which, at first glance, appears suitable for integrating KGW watermarking into speculative sampling. However, we demonstrate that distributional correctness does not translate into efficiency preservation: the vanilla construction consistently erodes the speculative acceleration under both token-level and feature-level decoders. Specifically, we observe a consistent drop in speculative acceptance across nine configurations spanning three target–draft model pairs and three datasets, and at the feature level, Medusa and the EAGLE family have over 60% of their candidate tree nodes replaced at $\delta = 2$. Existing KGW-style variants, including entropy-based position selection (SWEET), entropy-weighted detection (EWD), and distribution-adaptive bias adjustment (MorphMark), fail to recover this efficiency, leaving normalized generation time at the token level 12 to 19 percentage points higher than unwatermarked decoding.

We argue that this failure stems from a latent assumption that the vanilla construction inherits from standard autoregressive decoding: that the distribution being reweighted is

also the one whose statistics determine the cost of a write. Under speculative sampling, however, the distribution that determines the emitted output and the proposal mechanism that determines decoding efficiency play different roles. Even when the same watermark bias is applied to both the target and draft distributions, its efficiency cost depends on how the reweighting changes their compatibility. At the token level, applying the same bias to both models does not affect them symmetrically: the draft and target may assign different probability mass to the green list, so a shared bias is renormalized differently on the two sides and shifts the per-token acceptance threshold, while the reweighting also alters the proposal distribution itself and repeated writes can further reduce acceptance by shifting generation toward contexts with lower baseline acceptance. At the feature level, the proposal side takes the form of a candidate tree constructed before the target verifies it; writing during tree construction perturbs the top-K selection and can replace entire subtrees, and even when the native tree is preserved, verification-side reweighting can still pull target probability mass away from the already-drafted candidates. Together, these mechanisms indicate that an effective solution must control where, whether and to which stage the watermark is written, so that writes are admitted only when a proposal-side cost proxy predicts they are locally affordable, rather than uniformly weakening the bias everywhere.

Motivated by this understanding, we propose **ChiselMark**, a sparse, cost-aware, and architecture-aware framework that decomposes the single lever of watermark strength into three distinct controls: where the watermark may be written, whether a particular write is predicted to be locally affordable, and at which stage of the decoding architecture the write enters. A secret key and the local text context identify reproducible candidate sites, regulating the frequency of possible watermark interventions. At each candidate site, an affordability gate decides whether to write based on a proposal-side cost proxy, and the architecture then determines where the admitted write enters. At the token level, the gate uses the draft's probability mass on the green list, is fixed before proposal sampling to ensure exact sampling from the draft-dependent gated target distribution, and the admitted bias is applied jointly to the draft and target. At the feature level, the native candidate tree is left intact, and the bias is applied only at target verification under a coverage gate that bounds how much target probability mass is pushed away from the already-drafted candidates. The detector reconstructs the candidate sites and green lists from the text, the secret key, and the public detector parameters, calibrating the score against randomly drawn reference keys. Candidate positions vetoed by the cost-aware gate are still included in the detection statistic, although no deliberate watermark bias is applied at those positions. Consequently, the detector does not need to reconstruct the cost-aware gate decisions. We complement this design with theoretical analyses that characterize, under the applicable conditions of each result, token-level sampling exactness, feature-level conditional draft-tree invariance, sequence-level distortion, and the detector's false-positive behavior.

In conclusion, the main contributions of this paper are three-fold. **(1)** We revisit KGW-style watermarking under speculative sampling and reveal a latent assumption that current methods inherit from standard autoregressive decoding, namely that the distribution being reweighted is also the one whose statistics determine the cost of a write; we show that this assumption breaks under speculative sampling, where the cost depends on compatibility between the reweighted target and the proposal mechanism, and that this mismatch is what systematically erodes the speculative acceleration of vanilla constructions under both token-level and feature-level decoders. **(2)** Based on this understanding, we propose ChiselMark, a sparse, cost-aware, and architecture-aware framework that controls where the watermark may be written, whether a particular write is predicted to be locally affordable, and at which stage of the decoding architecture the write enters, while keeping it detectable from the generated text, the secret key, and public detector parameters; we complement this design with theoretical analyses that characterize, under the applicable conditions of each result, token-level sampling exactness, feature-level conditional draft-tree invariance, sequence-level distortion, and finite-sample false-positive control. **(3)** We conduct extensive experiments across token-level speculative sampling, Medusa, and three generations of EAGLE on multiple model scales and tasks, verifying that ChiselMark recovers much of the speculative efficiency lost to dense watermarking while keeping detectability competitive with existing watermarks, and demonstrating robustness to text edits.

2 Related Work

Text Watermarking for LLMs. A language model defines a probability distribution over sequences of tokens from a vocabulary set $\mathcal{V}$. We write $\Delta_{\mathcal{V}} = \{\mathbf{d} \in \mathbb{R}_{\geq 0}^{|\mathcal{V}|} : \sum_{v \in \mathcal{V}} \mathbf{d}(v) = 1\}$ for the probability simplex over $\mathcal{V}$. The watermarking scheme $f_w : \Delta_{\mathcal{V}} \to \Delta_{\mathcal{V}}$ is a reweighting function that maps a probability distribution $\mathbf{d} \in \Delta_{\mathcal{V}}$ to a watermarked distribution $\mathbf{d_w} = f_w(\mathbf{d}) \in \Delta_{\mathcal{V}}$. KGW [12] performs *logit reweighting* on the next-token distribution:

$$\mathbf{d_w}(x) = [f_w(\mathbf{d})](x) = \frac{\mathbf{d}(x)e^{\delta \mathbb{1}_G(x)}}{Z_{\mathbf{d}}(\delta)}, \qquad Z_{\mathbf{d}}(\delta) = \sum_{y \in \mathcal{V}} \mathbf{d}(y)e^{\delta \mathbb{1}_G(y)}, \tag{1}$$

where $G \subset \mathcal{V}$ is a green list seeded by the preceding tokens, $\mathbb{1}_G(\cdot)$ is the indicator function, and $\delta > 0$ is the bias parameter. The follow-up work and complementary work are listed in Appendix A.

Speculative Sampling. Speculative sampling (SpS) [17, 18] is a technique for accelerating the generation of tokens from a target model P by leveraging a faster draft model Q. The output distributions are $\mathbf{p}$ and $\mathbf{q}$.

The key idea is to first sample a draft token $\tilde{x} \in \mathcal{V}$ from the draft model Q, and then accept or reject it based on the ratio of the target and draft probabilities. If the draft token is rejected, a new token is sampled from a residual distribution. Formally, the SpS process generates a token x as follows:

$$\Pr(x = j | \tilde{x} = i) = \begin{cases} \min(1, \frac{\mathbf{p}(i)}{\mathbf{q}(i)}) & \text{if } i = j, \\ \frac{(1 - \frac{\mathbf{p}(i)}{\mathbf{q}(i)})_+ (\mathbf{p}(j) - \mathbf{q}(j))_+}{\sum_{z \in \mathcal{V}} (\mathbf{q}(z) - \mathbf{p}(z))_+} & \text{if } i \neq j, \end{cases} \tag{2}$$

where $(x)_+ = \max(0, x)$. The sampling efficiency, which is the expectation of acceptance, can be measured by the overlap probability $\beta(\mathbf{p}, \mathbf{q}) = \sum_{t \in \mathcal{V}} \min(\mathbf{p}(t), \mathbf{q}(t))$, representing the probability of accepting a draft token at each step. We report acceptance rate (AR) as an observable proxy for speculative reuse; accepted tokens per target-model call and wall-clock time provide complementary efficiency measures in Section 5. With the exact rejection correction, token-level SpS preserves the target sampling distribution while reducing target-model decoding rounds. Feature-level speculative sampling instead exploits the target model's hidden states $\mathbf{h}$ to construct multi-token proposals using lightweight draft modules. The detailed explanation of feature-level speculative sampling is listed in Appendix A.2.

Watermarking under Speculative Sampling. Recent studies have examined watermarking under speculative sampling, primarily through distribution-preserving constructions and token-level verification. Hu and Huang [28] establish, in the token-level setting, a fundamental trade-off between watermark strength and speculative acceptance efficiency, showing that their respective maxima cannot generally be attained simultaneously. He et al. [29] further quantify watermark strength, characterize Pareto frontiers for unbiased watermarking schemes, and introduce pseudorandom draft-token acceptance to attain maximal strength while preserving speculative sampling efficiency. At the production-system level, SynthID-Text [30] integrates watermarking with speculative sampling to support large-scale deployment. However, these results do not directly characterize KGW-style distribution-shifting watermarks under tree-based speculative decoding, where watermark bias interacts with multi-branch proposal construction and verification. A detailed compatibility comparison is provided in Appendix A.3.

3 Revisiting KGW-style Watermark under Speculative Sampling

To examine how KGW watermarking interacts with speculative sampling, we first construct a vanilla baseline, then empirically show that it reduces efficiency in both token-level and feature-level decoders, and finally analyze the mechanisms underlying it.

3.1 Vanilla Watermarks for Speculative Sampling

We construct a vanilla baseline by applying the same KGW transformation during drafting and target verification. This construction preserves the intended watermarked output distribution at the token level. Its extension to feature-level decoders follows the same reweighting principle, with green-list seeds adapted to the drafting architecture.

Vanilla Watermark for Token-level SpS. Following the KGW transformation in Section 2, we construct a vanilla baseline whose intended output distribution is the KGW-transformed target $\mathbf{p}_w$. Since the draft model serves as a proposal distribution, watermarking Q alone leaves the output distribution unchanged when verification and rejection correction use the unwatermarked target P. To obtain the KGW-transformed target distribution $\mathbf{p}_w$, we incorporate the watermark into target-side verification and rejection correction. The vanilla construction applies the same watermark transformation, with a shared green list G and bias δ, to both distributions, yielding $\mathbf{p}_w$ and $\mathbf{q}_w$. The following theorem establishes distributional correctness for this construction.

Theorem 3.1 (Correctness of Vanilla Watermarks for Speculative Sampling)**.** *Fix the decoding context and green list. Let* $\mathbf{p}$ *and* $\mathbf{q}$ *denote the target and draft distributions, respectively, and let* $f_w(\cdot)$ *be the KGW watermark transformation defined in Eq. (1). Define the corresponding watermarked distributions as* $\mathbf{p}_w$ *and* $\mathbf{q}_w$*. If token-level speculative sampling [17] as defined in Eq. (2) is performed with exact rejection correction using* $\mathbf{p}_w$ *as the target distribution and* $\mathbf{q}_w$ *as the draft distribution, then the resulting output token* x *is distributed according to* $\mathbf{p}_w$*; that is,* $x \sim \mathbf{p}_w$*.*

The accepted-proposal mass $\min\{\mathbf{p}_w(x), \mathbf{q}_w(x)\}$ and the rejection-correction mass $(\mathbf{p}_w(x) - \mathbf{q}_w(x))_+$ sum to $\mathbf{p}_w(x)$; the full proof appears in Appendix B.1.

Vanilla Watermark for Feature-level SpS. For feature-level speculative sampling, the same reweighting principle can be applied to drafting and verification, but the green-list seed must be adapted to the topology of the draft proposal. Medusa shares a backbone state across position-specific decoding heads, so a deeper draft node is not conditioned on its parent token, whereas EAGLE drafts autoregressively, with each non-root node inheriting a well-defined parent token. To assign the same green list to the draft and target distributions at each verification node, we choose the seed according to the drafting architecture. Let $G(u)$ denote the green list generated from seed u using the keyed construction defined above. For a Medusa round beginning with accepted prefix $x_{1:t}$, we use the last accepted token $r_t = x_t$ as a shared anchor. For EAGLE, we use the parent token of each draft node. The green list at node ω is therefore

$$G_\omega = \begin{cases} G(r_t), & \text{Medusa}, \\ G(x_{\mathrm{pa}(\omega)}), & \text{EAGLE}, \end{cases} \tag{3}$$

where $x_{\mathrm{pa}(\omega)}$ denotes the parent token of node ω in the EAGLE draft tree. The Medusa anchor is updated to the last accepted token at the beginning of each decoding round. During verification, the target model evaluates the draft tree in parallel, while acceptance is resolved along root-to-leaf paths. At each node ω, we reweight the target distribution using the same green list G_ω used to construct the corresponding draft distribution.

3.2 Acceptance Loss under Vanilla Watermarking

Although the vanilla construction appears feasible, we find that its effectiveness is limited in practice. Specifically, we observe that it reduces speculative acceptance in both token-level and feature-level decoders. Distributional correctness at the token level therefore does not imply preservation of speculative efficiency. We first present the evidence across the two architectures, using measured acceptance to assess speculative reuse.

Token-level Observations. Figure 1 shows that finite-bias KGW reduces per-run speculative acceptance across all nine configurations, covering HumanEval [31], BigCodeBench [32], and TinyStories [33] with three draft-target model pairs. Evaluation settings and aggregation conventions are given in Appendix B.3.

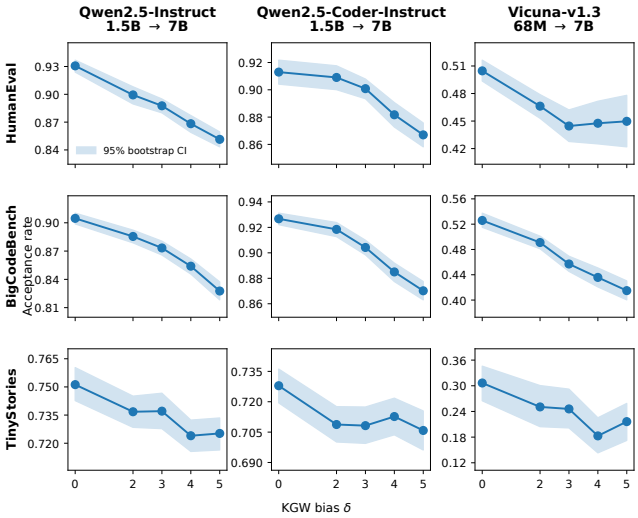


Figure 1: **Watermark bias reduces speculative acceptance.** Speculative acceptance rate as a function of the KGW watermark bias δ across three draft-target model pairs (columns: Qwen2.5-Instruct 1.5B $\rightarrow$ 7B, Qwen2.5-Coder-Instruct 1.5B $\rightarrow$ 7B, Vicuna-v1.3 68M $\rightarrow$ 7B) and three datasets (rows: HumanEval, BigCodeBench, TinyStories). Markers denote mean acceptance rates and shaded regions indicate 95% bootstrap confidence intervals.

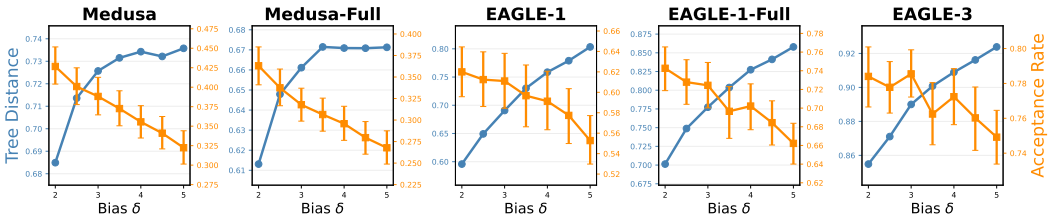


Figure 2: **The tree distance and average acceptance rate across Medusa, EAGLE-1, and EAGLE-3 as watermark strength δ increases.** Medusa and EAGLE-1 denote the original heuristic tree configurations, while "Full" denotes the corresponding complete candidate structures of the draft tree.

Feature-level Observations. Figure 2 shows an overall decline in speculative acceptance as watermark strength increases across Medusa and the evaluated EAGLE configurations. This trend appears with both the original heuristic trees and the complete candidate structures where available. Thus, the acceptance loss also occurs when proposals are constructed from target features. Here, acceptance measures reuse along the verified path; the accompanying Tree Distance measures changes in candidate tokens and is distinct from the acceptance rate. Appendix B.4 gives the architecture-specific comparisons.

These observations show lower acceptance along the generated trajectories across the evaluated architectures. Appendix B.2 studies the local response at zero bias, while the experiments here measure acceptance at finite bias during generation. Appendix B.3.1 explains the aggregation conventions and the prompt-formatting and EOS issue in Vicuna/TinyStories. An intriguing question arises: Why does applying the same watermark during drafting and verification still reduce speculative acceptance?

3.3 Understanding the Acceptance Loss

To understand why applying the same watermark fails to preserve efficiency across the evaluated decoding architectures, we analyze compatibility between the reweighted target

and the proposal mechanism at two levels: token-level overlap and context drift, and feature-level tree construction and verification coverage.

Token-level Analysis. At the token level, consider the acceptance rate for target and draft distributions $\mathbf{p}$ and $\mathbf{q}$ reweighted with the same green list G and bias δ:

$$\beta_w = \beta(\mathbf{p}, \mathbf{q}, G, \delta) = \sum_{x \in \mathcal{V}} \min\{\mathbf{p}_w(x), \mathbf{q}_w(x)\} = 1 - \mathrm{TV}(\mathbf{p}_w, \mathbf{q}_w), \tag{4}$$

where TV denotes total variation distance. The loss can arise from changes in overlap at a fixed context and changes in the contexts encountered during generation. We examine these two mechanisms separately.

Mechanism 1: Static Normalization Mismatch. First, watermarking changes acceptance even when the context is held fixed. For KGW, the shared green-list factor cancels from the likelihood ratio on the common positive support of the target and draft:

$$\log \frac{\mathbf{p}_w(x)}{\mathbf{q}_w(x)} = \log \frac{\mathbf{p}(x)}{\mathbf{q}(x)} - \log \lambda, \qquad \lambda = \frac{Z_p}{Z_q} = \frac{1 + (e^\delta - 1)p_G}{1 + (e^\delta - 1)q_G}, \tag{5}$$

where $p_G = \sum_{x \in G} \mathbf{p}(x)$ and $q_G = \sum_{x \in G} \mathbf{q}(x)$ denote the unwatermarked green-list masses. Thus, the common watermark preserves the relative ordering of likelihood ratios, while disagreement on total green-list mass produces different normalizers and shifts the acceptance threshold. The resulting overlap depends jointly on this shift, the reweighted sampling measure $\mathbf{q}_w$, and within-partition (green/red) target-draft agreement. Static replay on frozen unwatermarked contexts isolates this effect: varying only the scoring bias reduces acceptance over the practically relevant range in nearly all configurations, and the normalization residual relative to a diagnostic with λ set to one has a negative point estimate in all nine configurations at $\delta = 5$. Appendix B.3.2 gives the diagnostic decomposition.

Mechanism 2: Autoregressive Context Drift. Second, watermarking changes the generated trajectory and therefore the contexts at which subsequent target and draft distributions are evaluated. As δ increases, generated continuations generally exhibit higher perplexity under the unwatermarked target and visit states with larger predictive entropy $H(\mathbf{p})$. Changes in draft-target agreement vary across configurations: $\mathrm{KL}(\mathbf{p}\|\mathbf{q})$ and top-1 agreement can either improve or worsen depending on the model-dataset pair. Appendix B.3 decomposes the change in unwatermarked acceptance into an entropy-composition shift (visiting higher-entropy states) and a within-entropy component (Eq. (39)). At $\delta = 5$, the composition term has a negative point estimate across all nine configurations, reflecting a shift toward entropy strata with lower baseline speculative acceptance. The within-entropy term varies across configurations and is often small or positive.

Feature-level Analysis. At the feature level, the proposal side is a candidate tree constructed before target verification. Watermarking can affect reuse at two stages: it can change the candidates during construction and alter their coverage during verification.

Reason 1: Watermark perturbation changes the draft structure. Feature-level speculative sampling relies on a tree-style draft structure. To quantify changes in draft candidates caused by watermarking, let $\mathbf{z} = (z_1, \ldots, z_\mathcal{T})$ and $\mathbf{z}_w = (z_1^{(w)}, \ldots, z_\mathcal{T}^{(w)})$ denote the flattened token sequences of the draft structures before and after watermarking, indexed identically across tree positions. Here, $\mathcal{T}$ is the total number of candidate tokens. For cross-method comparison, we report *Tree Distance*, defined as the relative Hamming distance: $\mathrm{TreeDistance}(\mathbf{z}, \mathbf{z}_w) = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} \mathbb{1}\left[z_t \neq z_t^{(w)}\right]$, where $\mathbb{1}[\cdot]$ is the indicator function. Tree Distance measures the fraction of candidate positions whose tokens change after watermarking. Figure 2 shows that at $\delta = 2$, around 60% or more of candidate positions already change across the evaluated methods, accompanied by an overall loss of speculative acceptance. Greedy or top-K selection turns logit perturbations into candidate changes, which can propagate to descendants in autoregressive tree builders. This identifies a structural cost of writing before the proposal tree is fixed.

Reason 2: Preserving the tree does not preserve its coverage. Even with an unchanged tree, target-side reweighting can move probability mass away from its existing

child candidates. A write can therefore reduce proposal reuse without changing any draft token. Tree preservation addresses the construction-stage perturbation; a separate check of target mass on the available candidates is needed to assess the verification-stage cost. Appendix B.4 details the architecture-specific tree comparisons, while Appendix D.2 derives a fixed-state coverage-loss bound for the specified direct-draw verifier.

Overall Analysis. Taken together, these findings show that preserving the intended watermarked distribution does not preserve speculative efficiency. At the token level, watermarking alters the distributional overlap and shifts the contexts encountered during generation; at the feature level, it perturbs the draft candidates and their coverage under verification. In both cases, the root cause is the same: watermarking disrupts the proposal reuse on which speculative acceleration depends.

4 Proposed ChiselMark Framework

4.1 Overall Pipeline

Motivated by this understanding above, we propose **ChiselMark** to address these controls within a two-phase procedure. Under speculative sampling, the cost of a watermark write depends on compatibility between the reweighted target and the proposal. In **Phase I (embedding)**, a secret key chooses candidate positions, a gate decides which candidates to watermark, and each decoder applies the bias at a suitable stage. This limits how often we write, avoids costly local writes, and leaves feature-level draft trees untouched. In **Phase II (detection)**, the detector reconstructs candidate positions from the output text and secret key, then checks for a watermark without needing the cost-aware gate decisions.

Figure 3 shows the full procedure. For token-level sampling, the gate uses draft probabilities before a token is proposed; an admitted write is applied to both draft and target. For feature-level sampling, the decoder first builds its normal candidate tree; the gate then uses target probabilities during verification. In both cases, detection needs only the text, key, and public settings, not the gate decisions.

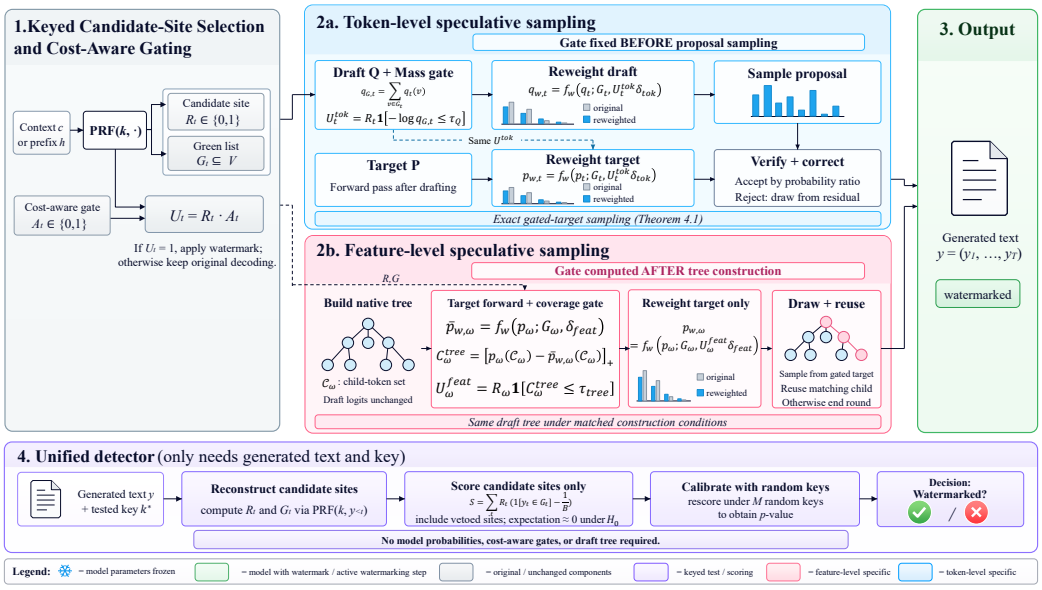


Figure 3: Overall pipeline of ChiselMark. Phase I performs sparse, architecture-aware watermark embedding with keyed candidate sites and cost-aware gating. Token-level speculative sampling co-watermarks the draft and target distributions, whereas feature-level speculative sampling preserves the native draft tree and intervenes only during verification. Phase II reconstructs candidate sites from the generated text, secret key, and public detector parameters and performs unified keyed watermark detection.

4.2 Phase I: Sparse, Cost-Aware, and Architecture-Aware Embedding

We first describe how the key determines candidate positions and green lists. We then explain at which stage each decoder applies the bias: before draft sampling for token-level SpS, and only during verification for feature-level SpS. Throughout, $f_w(\mathbf{d}; G, \delta)$ denotes the reweighting rule in Eq. (1).

Keyed Candidate-Site Selection. We first determine where watermarking is allowed (R_t) and which tokens are favored (G_t). Both are derived from the secret key and preceding generated tokens, allowing the detector to reconstruct them without knowing the cost-aware gate decision A_t. Specifically, let κ have domain-separated subkeys κ_{site}, κ_{sym}, and κ_{tok}. For continuation $x_{1:T}$, use $c_t = (x_{t-\ell}, \ldots, x_{t-1})$ at $t > \ell$ and set $R_t = 0$ for $t \le \ell$, allowing detection without the prompt. Map the site PRF to $[0, 1)$ and define

$$R_t = \mathbb{1}[\mathsf{PRF}_{\kappa_{\text{site}}}(c_t) < \rho], \qquad \rho \in (0, 1]. \tag{6}$$

For $B \ge 2$, define a desired symbol and token labels by

$$b_t = \mathsf{PRF}_{\kappa_{\text{sym}}}(c_t) \bmod B, \quad g_t(v) = \mathsf{PRF}_{\kappa_{\text{tok}}}(c_t, v) \bmod B.$$

We use $B = 2$ for zero-bit detection unless stated otherwise. The green list and actual write indicator are

$$G_t = \{v \in \mathcal{V} : g_t(v) = b_t\}, \qquad U_t = R_t A_t, \tag{7}$$

where $A_t \in \{0, 1\}$ denotes the cost-aware gate. Thus, the detector can reconstruct eligibility without knowing which writes were admitted.

The density parameter ρ controls which positions are eligible for watermarking. The realized write count also depends on the cost-aware gate decisions and the contexts reached during generation. At $\rho = 1$, every position is eligible and the gate alone decides which writes are applied. Identical n-gram contexts c_t reuse the same keyed outputs, which can correlate candidate decisions across positions. We account for these dependencies when calibrating detection in Section 4.3. Using a local context window of length ℓ also allows detection to resynchronize after a text edit once an intact window is reached.

Token-level Gate and Placement. The preceding construction determines where watermarking is eligible, but not whether the bias should be applied. Since watermarking can reduce draft–target overlap, we introduce the draft green-mass gate to selectively activate the bias at eligible positions. The gate is fixed before proposal sampling, and the admitted bias is applied jointly to the draft and target. Specifically, let $\mathbf{p}_t, \mathbf{q}_t$ denote target and draft distributions at the same prefix. The overlap analysis in Section 3.3 depends on both distributions, but a gate based on their full overlap would require target information at proposal time. We instead use the draft's green-list probability mass, available during drafting, as a local compatibility proxy. With $q_{G,t} = \sum_{v \in G_t} \mathbf{q}_t(v)$, compute

$$C_t^Q = -\log q_{G,t}, \qquad A_t^{\text{tok}} = \mathbb{1}[C_t^Q \le \tau_Q], \tag{8}$$

and set $U_t^{\text{tok}} = R_t A_t^{\text{tok}}$, with finite $\tau_Q \ge 0$ and $-\log 0 = +\infty$. The rule admits writes when the draft assigns sufficient mass to the realized green list. Unlike key-agnostic entropy filters (e.g., SWEET), C_t^Q is partition-aware and can admit a peaked, low-entropy state when its green-list mass exceeds the threshold. It requires no additional target forward pass. Cache the pre-proposal gate and apply the same admitted bias to both distributions, with $\delta_{\text{tok}} \ge 0$:

$$\mathbf{q}_{w,t} = f_w(\mathbf{q}_t; G_t, U_t^{\text{tok}} \delta_{\text{tok}}), \quad \mathbf{p}_{w,t} = f_w(\mathbf{p}_t; G_t, U_t^{\text{tok}} \delta_{\text{tok}}). \tag{9}$$

The draft samples from $\mathbf{q}_{w,t}$, and verification uses this actual proposal distribution with target $\mathbf{p}_{w,t}$. The common token-specific bias cancels in their likelihood ratio, while their normalizers and proposal weights still affect acceptance. On rejection, sample from the residual proportional to $[\mathbf{p}_{w,t} - \mathbf{q}_{w,t}]_+$ and discard later speculative tokens. If the block is accepted, sample the bonus token from its own gated target, computing the draft distribution at that prefix when needed; this evaluation is included in runtime cost.

Feature-level Gate and Placement. For feature-level sampling, we address construction-stage perturbation and verification-stage coverage loss separately. We preserve the native

draft tree, then screen target-side writes according to how much probability mass they remove from its child candidates. Specifically, we first construct the native candidate tree $\mathcal{S}(h;\eta)$ using unmodified draft logits, where h is the committed prefix and η encapsulates the tree topology, auxiliary hidden states, and drafting randomness. Deferring watermark intervention to verification yields a critical architectural bonus alongside tree preservation: because the candidate tree is already instantiated before verification begins, every node ω possesses a fully realized ancestral token path. We can therefore seed R_ω and G_ω using the exact path context c_ω (committed prefix plus ancestral tree tokens)—even for non-autoregressive drafters like Medusa that lack parent tokens during drafting—ensuring full context alignment with the text-and-key detector.

With the native tree fixed, the gate evaluates the coverage loss of a candidate write on the available children. Evaluate the unmodified target distribution $\mathbf{p}_\omega$ and let $\mathcal{C}_\omega$ be the unique child-token labels. Form $\bar{\mathbf{p}}_{w,\omega} = f_w(\mathbf{p}_\omega; G_\omega, \delta_{\text{feat}})$. With $\mathbf{d}(\mathcal{C}) = \sum_{v\in\mathcal{C}} \mathbf{d}(v)$, define

$$C_\omega^{\text{tree}} = [\mathbf{p}_\omega(\mathcal{C}_\omega) - \bar{\mathbf{p}}_{w,\omega}(\mathcal{C}_\omega)]_+. \tag{10}$$

For $\tau_{\text{tree}} \geq 0$, set

$$A_\omega^{\text{feat}} = \mathbb{1}[C_\omega^{\text{tree}} \leq \tau_{\text{tree}}], \quad U_\omega^{\text{feat}} = R_\omega A_\omega^{\text{feat}}.$$

This gate uses target probabilities already computed during verification to measure how a write changes the probability mass on the available child candidates. Apply the admitted bias only to target verification:

$$\mathbf{p}_{w,\omega} = f_w(\mathbf{p}_\omega; G_\omega, U_\omega^{\text{feat}}\delta_{\text{feat}}). \tag{11}$$

Algorithms 1 and 2 in Appendix C.1 detail the complete embedding procedures.

4.3 Phase II: Text-and-Key Detection

The gate makes decisions using model probabilities, but a text-and-key detector cannot observe those values. It therefore cannot tell which eligible positions received a watermark. ChiselMark instead scores every eligible position, including those where the gate blocked a write. Eligible positions and green lists depend only on the text and secret key, so the detector can reconstruct them. It then compares the score with scores from random reference keys; Section 4.4.3 explains when this test controls false positives.

Reconstruction and Scoring. The scoring operation measures how often observed tokens match their keyed green lists at reconstructable candidate sites. Specifically, given tokens $y_{1:T}$, use the same tokenizer, ℓ, ρ, and B, with $c_t = (y_{t-\ell}, \ldots, y_{t-1})$ for $t > \ell$. Writing $R^\kappa(c)$, $b^\kappa(c)$, and $g^\kappa(c, v)$ for the keyed maps defined above, reconstruct

$$R_t^\kappa = R^\kappa(c_t), \quad b_t^\kappa = b^\kappa(c_t), \quad \hat{b}_t^\kappa = g^\kappa(c_t, y_t).$$

Each eligible position contributes its centered green-list indicator. Define

$$s_t^\kappa = R_t^\kappa \left(\mathbb{1}[\hat{b}_t^\kappa = b_t^\kappa] - \frac{1}{B} \right), \quad S_\kappa(y_{1:T}) = \sum_{t=\ell+1}^{T} s_t^\kappa. \tag{12}$$

Random-key Calibration. To calibrate the aggregate score despite dependencies between positions, we compare the tested key with reference keys on the same observed text. Specifically, fix the text and public configuration. For tested key $\kappa^\star$, draw M independent reference keys from the same distribution, evaluate the same score for each, and compute

$$p_{\text{wm}} = \frac{1 + \sum_{j=1}^{M} \mathbb{1}[S_{\kappa_j}(y_{1:T}) \geq S_{\kappa^\star}(y_{1:T})]}{M+1}. \tag{13}$$

Reject when $p_{\text{wm}} \leq \alpha$.

4.4 Theoretical Analyses of ChiselMark

Previous parts explain how ChiselMark embeds and detects watermarks. We now establish which properties follow from its design: exact token-level sampling from the gated target, unchanged feature-level draft trees at a fixed state, a bound on text distortion, and false-positive control for the detector. We explain why some write positions are more useful.

4.4.1 Correct Sampling and Preserved Draft Trees

Token-level Sampling. The token-level proposal, correction, and bonus rules preserve the gated target under the following conditions.

Theorem 4.1 (Exactness of Cost-Gated Token-Level ChiselMark)**.** *Assume that the draft distribution and gate are determined by the current prefix, key, and fixed decoding configuration before sampling the proposal. Standard speculative rejection sampling with proposal* $\mathbf{q}_{w,t}$ *and target* $\mathbf{p}_{w,t}$*, together with the correction and bonus rules above, produces exactly the autoregressive gated target defined by Eq. (9). The reference distribution is this gated target, which depends on the draft model through the gate, rather than the ungated KGW target or the original unwatermarked model.*

Theorem 4.1 highlights why fixing A_t^{tok} prior to sampling the draft token is essential: deciding whether to watermark only after observing whether the sampled draft token landed in G_t would violate proposal-target independence and bias the rejection sampler. Under our pre-proposal gate, speculative verification remains distributionally exact with respect to the gated target $\mathbf{p}_{w,t}$ (proof in Appendix D.3).

Feature-level Draft Structure and Coverage. Restricting watermark intervention to verification preserves the draft structure under matched construction conditions.

Proposition 4.2 (Conditional Draft-Tree Invariance)**.** *For an identical prefix, native configuration, auxiliary decoder state, and coupled construction randomness, ChiselMark and the watermark-free decoder construct the same draft tree:* $\mathcal{S}_w(h;\eta) = \mathcal{S}(h;\eta)$*. Thus ChiselMark eliminates direct tree changes caused by watermark perturbations to draft logits.*

Together, Proposition 4.2 and the coverage gate provide two complementary local guarantees at each decoding round: verification-only placement completely eliminates construction-stage tree distortion (Reason 1), while the gate strictly caps the verification-stage probability mass drained from existing child candidates by $[p_\omega(\mathcal{C}_\omega) - p_{w,\omega}(\mathcal{C}_\omega)]_+ \le U_\omega^{\text{feat}} \tau_{\text{tree}}$ (Reason 2). While admitted watermark writes naturally steer subsequent generation toward new prefixes over multi-round trajectories, Appendix D.2 further characterizes the exact verifier conditions under tree-dependent gating.

4.4.2 Text Distortion and Write Selection

The embedding rules raise two complementary questions: how distortion accumulates across realized writes, and how writes should be allocated when speculative cost varies across states. We address the first through a sequence-level bound for the exact gated token-level sampler, and the second through a frozen-state allocation benchmark. For the distortion analysis, write $\delta = \delta_{\text{tok}}$ and $N_{\text{write}} = \sum_{t=\ell+1}^{T} U_t$ for the realized write count.

Theorem 4.3 (Sequence distortion, informal; Appendix D.3)**.** *For the exact gated token-level sampler, the sequence-level distortion satisfies* $D_{\text{KL}}(P^w \,\|\, P) \le (\delta^2/8)\,\mathbb{E}[N_{\text{write}}]$*, where* P^w *and* P *denote the gated and unwatermarked sequence distributions and the expectation is under* P^w*.*

Theorem 4.3 bounds sequence-level KL divergence by a quantity that, for a fixed bias δ, scales linearly with the expected number of realized writes. Candidate selection and cost-aware gating control write activation, linking the embedding policy to cumulative distortion. However, output KL alone does not capture speculative acceptance loss. Remark D.16 in Appendix D.3 provides a fixed-state example in which the output KL is zero, yet changing the draft proposal reduces draft–target acceptance. This motivates accounting for state-dependent speculative costs when selecting watermark writes. We characterize the resulting allocation problem through an idealized frozen-state benchmark:

Theorem 4.4 (Allocation benchmark, informal; Appendix D.4)**.** *Under a frozen-state expected cost budget, the benefit-maximizing activation rule is a threshold on the evidence-per-cost ratio* e_i/c_i*, where* e_i *and* c_i *denote watermark evidence and incremental speculative cost at state* i*.*

Appendix D.5 studies how to spread a fixed total increase in green-list probability when the initial green-list mass is the same at every position. When bias strength can vary, KL

is minimized by assigning the same increase to each position. The required common bias depends on the increase per position. Theorem 4.4 studies a different problem: given the benefit and speculative cost of a write at each state, it selects writes under an expected cost budget. Our gates estimate write cost using draft green-list mass or the target mass on the available tree candidates. At matched TPR (0.907), Table 1 reports lower overhead (3.43% versus 5.68%) and a lower PPL ratio (1.124 versus 1.146) for ChiselMark ($\delta = 4$, $\rho = 0.5$) than for dense KGW ($\delta = 1.5$).

4.4.3 Detection Power and False-Positive Control

Score Centering. The score is centered under H_0, where the text is independent of the tested key. In the ideal-PRF model, for fixed text $y = y_{1:T}$, uniform symbols and domain separation give

$$\mathbb{E}_\kappa[\mathbb{1}[\hat{b}_t^\kappa = b_t^\kappa] \mid y] = 1/B, \quad \mathbb{E}_\kappa[s_t^\kappa \mid y] = 0, \quad \mathbb{E}_\kappa[S_\kappa(y) \mid y] = 0.$$

Repeated contexts can correlate scores across positions.

Random-key Calibration. Random-key calibration compares scores across exchangeable keys on the same observed text. The following theorem states the null assumptions under which this test controls false positives.

Theorem 4.5 (Finite-Sample False-Positive Control by Key Randomization)**.** *Under H_0, suppose that conditional on the observed text the tested and reference keys are exchangeable, and the same fixed scoring rule is applied to every key. Then, for all $\alpha \in [0, 1]$,*

$$\mathbb{P}_{H_0}(p_{\mathrm{wm}} \le \alpha \mid y_{1:T}) \le \alpha.$$

In particular, this holds when all $M + 1$ keys are independently sampled from the same distribution and jointly independent of the null text.

Conditioned on the text, exchangeability makes the tested score's rank a valid basis for calibration; the proof, tie convention, and multiple-testing requirements are given after Algorithm 3 in Appendix C.2.

Evidence from Gated Writes. The detector scores all eligible sites ($R_t = 1$), so its power depends on the evidence from both written and vetoed sites. At a written site, reweighting changes the green-list probability by the increment given in Proposition D.1 in Appendix D.1. The gate depends on the key and can select sites with different score distributions. We therefore evaluate detection power over the full candidate set. Under the null hypothesis, the complete score is centered, and Theorem 4.5 controls false positives under its exchangeability assumptions. Section 5 evaluates detection power together with decoding cost.

5 Experiments

We evaluate ChiselMark across detection, decoding efficiency, and robustness. We first compare its efficiency and detectability across speculative decoders, then test robustness to post-hoc text edits. Next, we evaluate sparse, cost-aware embedding through selected operating-point comparisons and examine the gate mechanisms and their detection trade-offs. A separate Vicuna-13B text-quality evaluation is reported in Appendix E.5.

5.1 Experimental Setup

Datasets and Models. The main comparison uses Vicuna-7B with token-level SpS, Medusa, and EAGLE-1/2 on 400 C4 samples and 80 first-turn MT-Bench prompts, following MarkLLM [13]. Compatible 13B/33B configurations and EAGLE-3 are reported separately in Appendix E.3.

Baselines and Main Settings. Baselines are KGW, SWEET, EWD, and MorphMark. In the main comparison, KGW, SWEET, and EWD use $\delta = 2$; ChiselMark uses $(\delta, \rho) = (5, 0.5)$ at the token level and $(4, 1)$ at the feature level.

Generation Settings. Generation uses temperature 1, top-$p = 1$, top-$k = 0$, up to 256 continuation tokens with EOS stopping, and five generation seeds. Token-level SpS uses a Vicuna-v1.3-68M draft and lookahead 5.

Evaluation Metrics. Detection reports TPR@1% FPR; ChiselMark uses $M = 999$ reference keys and $\alpha = 0.01$ without model access. WD20 denotes detection after deleting 20% of words, and AR denotes speculative acceptance rate. For the main comparison, we report speed retention as t_0/t and AR retention as a/a_0, where t and a denote mean generation time and mean acceptance rate, and t_0 and a_0 are the corresponding watermark-free means under the same backbone, decoder, and dataset. Higher values indicate greater retention of watermark-free efficiency. The complete tables in Appendix E.3 retain the original generation-time normalization by watermark-free token-level SpS and report means and standard deviations over five generation seeds.

Comparison Protocol. The controlled study selects configurations on development data and includes a token-level comparison at matched TPR. Additional cross-model tests evaluate acceptance recovery across the nine model–dataset configurations of Figure 1. Their acceptance-only protocol and results are given in Appendix E.2.1.

Scope and Implementation Details. Our exactness analysis covers the gated token-level sampler. We evaluate Medusa and EAGLE performance empirically. Appendix E.1 gives the full protocols, baseline implementations, seed conventions, and verifier scope; Appendix E.2 lists the hyperparameters.

5.2 Main Results

With the experimental setup in place, we first evaluate whether ChiselMark can recover speculative decoding efficiency while maintaining competitive watermark detectability. We compare it with existing watermarking methods across different speculative decoding architectures, examining generation time, acceptance rate, and detection performance under both clean and edited conditions.

Efficiency and Detectability. Figure 4 reports the Vicuna-7B comparison across token-level SpS, Medusa, and EAGLE-1/2 on MT-Bench and C4. ChiselMark recovers speculative efficiency while retaining competitive detectability. Across all eight configurations, it attains the highest speed retention and AR retention among the evaluated watermarked methods. On token-level SpS/MT-Bench, it increases KGW's speed retention from 89.4% to 97.3% and AR retention from 77.5% to 91.4%, with TPR@1% FPR of 0.695. On EAGLE-1/MT-Bench, AR retention rises from 87.8% to 98.5%, approaching the watermark-free level; recovery is smaller on C4. Numerical results and standard deviations appear in Appendix E.3.

Comparison with EWD. While ChiselMark leads in efficiency, we also compare its detection power with the strongest detection baseline, EWD. On token-level SpS/MT-Bench, EWD has higher clean TPR (0.735 versus 0.695), whereas ChiselMark has higher WD20 TPR (0.540 versus 0.520), higher speed retention, and higher AR retention. Across the eight configurations, ChiselMark's WD20 TPR exceeds the strongest evaluated baseline by 2.00 to 47.45 percentage points, as annotated in Figure 4.

Why Medusa Benefits. Beyond token-level and EAGLE decoders, Figure 4 shows a particularly large gain on Medusa. On Medusa/MT-Bench, clean TPR increases from 0.1150 for vanilla KGW to 0.6825 for ChiselMark, while speed retention rises from 88.5% to 99.0% (recovering 62.1% of the acceptance loss). This detectability gain comes from deferring watermarking to the verification stage. Because Medusa's parallel draft heads are non-autoregressive, watermarking during tree construction forces all candidate nodes in a round to share a single round-level anchor in Eq. (3). As a result, the detector cannot reconstruct the exact green-list seed for each token, and detection collapses. By applying the watermark during target verification after the draft tree is built, ChiselMark seeds each accepted token with its actual preceding tokens, restoring seed alignment with the detector. Across all 24 model–decoder–dataset configurations (spanning 7B/13B/33B backbones and EAGLE-3; see Table 5 in Appendix E.3), ChiselMark recovers an average of 61.5% of the acceptance loss caused by dense KGW.

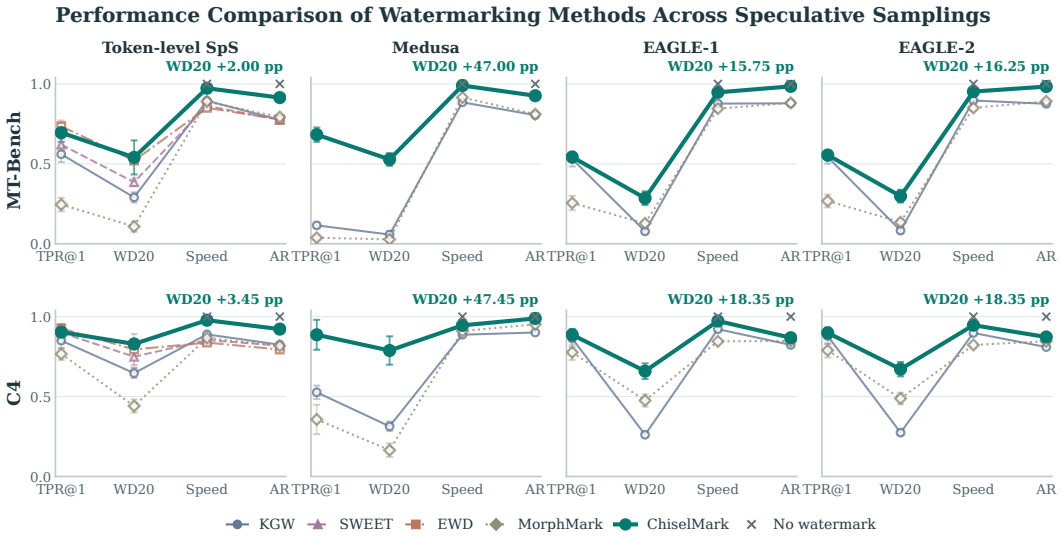


Figure 4: **Watermark detectability and speculative-efficiency retention.** Results on MT-Bench (top) and C4 (bottom) with Vicuna-7B across token-level SpS, Medusa, and EAGLE-1/2 (left to right). TPR@1% FPR and WD20 denote TPR@1% FPR before and after deleting 20% of words, respectively. Speed retention is t_0/t, and AR retention is a/a_0, computed from the mean generation times and acceptance rates relative to watermark-free decoding under the same backbone, decoder, and dataset. Higher values are better for all four metrics; crosses mark watermark-free efficiency at one. Detection rates are means over five generation seeds, with error bars showing one standard deviation. WD20 annotations give ChiselMark's gain over the strongest evaluated baseline in each panel, in percentage points. Complete numerical results and standard deviations are provided in Appendix E.3.

5.3 Robustness to Text Edits

The main comparison includes an initial 20% word-deletion test (WD20). To see whether the watermark survives broader post-hoc modifications, we further test word substitution, word deletion across multiple rates, and neural paraphrasing by Dipper and GPT without access to the secret key. Word-level edits alter local tokens and their surrounding context, while paraphrasing rewrites the phrasing needed to recover candidate positions and green lists. These tests use 80 C4 prompts with 200 generated tokens each. The evaluation protocol is detailed in Appendix E.4 and differs from that of the main comparison.

Figure 5 shows that ChiselMark retains detectable signal after these edits, although TPR falls as editing becomes stronger. Under 30% substitution or deletion, token-level ChiselMark reaches TPRs of 0.463 and 0.475, compared with 0.425 and 0.438 for MorphMark and 0.188 and 0.200 for KGW. ChiselMark and MorphMark perform similarly under paraphrasing: under Dipper-20, their TPRs are 0.463 and 0.475; under GPT rewriting, they are 0.400 and 0.388. Each method uses its standard setting, with clean 200-token TPRs of 0.700 for ChiselMark, 0.675 for MorphMark, and 0.563 for KGW. Figure 5 also reports results for EAGLE-1/3. The editing procedures and evaluation protocol are detailed in Appendix E.4.

5.4 Effectiveness of Sparse, Cost-Aware Embedding

Having evaluated ChiselMark's efficiency and detectability, as well as its robustness to text edits, we compare dense, sparse, and gated watermarking configurations at the token and feature levels. Table 1 reports detection and decoding efficiency for the selected settings.

The held-out study uses 32 C4 development prompts to select configurations and 60 disjoint test prompts for evaluation. All runs generate 192 tokens with EOS stopping disabled; token-level results average five key–seed runs, and EAGLE-1 results average three. We report TPR@1% FPR, accepted tokens per target call, and wall-clock overhead relative

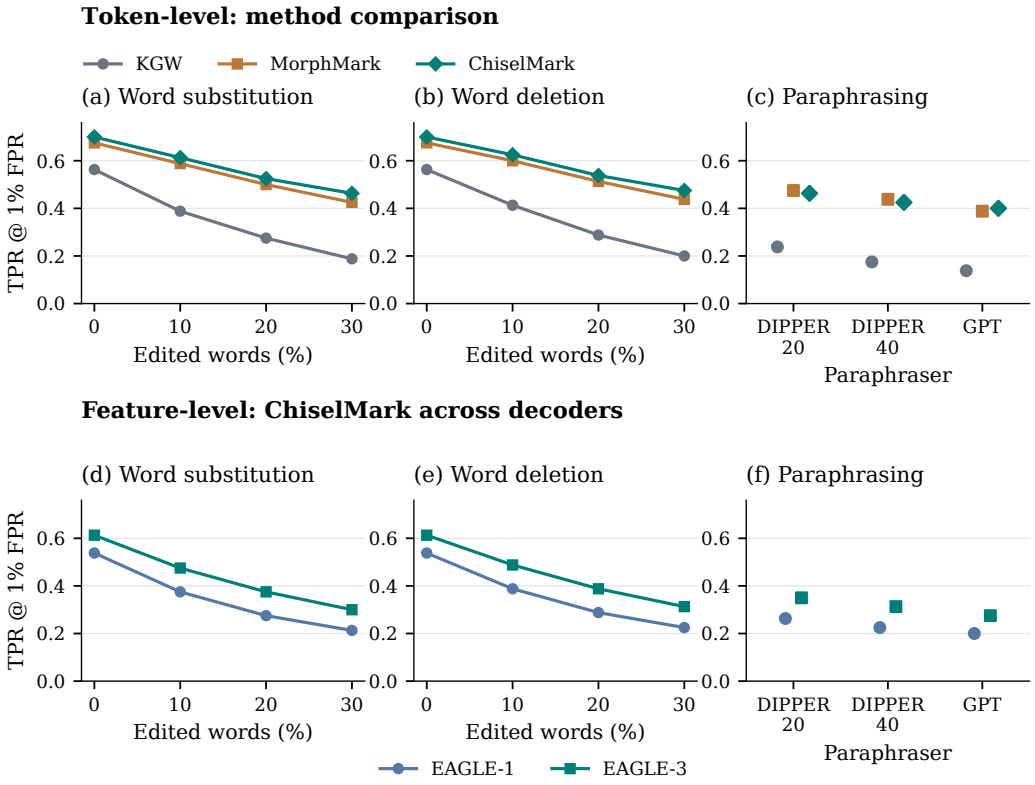


Figure 5: **Robustness to text edits.** TPR@1% FPR on 200-token C4 continuations. Panels (a)–(c) compare KGW, adapted MorphMark (adapted to the speculative verification distribution under identical drafting budgets), and ChiselMark under token-level decoding; panels (d)–(f) show ChiselMark with EAGLE-1/3. Zero editing denotes clean TPR. Dipper-20/40 use lexical diversity settings of 20/40, and GPT denotes GPT-5.4-mini rewriting. Token-level SpS and EAGLE-1 use Vicuna-7B; EAGLE-3 uses Vicuna-13B. Methods are not matched for clean TPR; the editing procedures and evaluation protocol are detailed in Appendix E.4.

to the watermark-free decoder of the same architecture. Each configuration specifies a combination of bias δ, candidate density ρ, and gate thresholds. Several of these parameters vary together, so the comparisons assess complete configurations. The protocol differs from the main comparison. Appendix E.6 gives the selection procedure, additional configurations, and detectability–overhead plots in Figure 9.

Token Level: Sparsity and Draft Green-Mass Gating. We first compare whether limiting eligible writes is enough to protect draft acceptance, or whether cost-aware admission provides a further benefit. At matched TPR 0.907, ChiselMark reduces dense KGW's generation overhead from 5.68% to 3.43% and raises accepted tokens per target call from 1.817 to 1.881. Ungated sparse KGW achieves slightly lower overhead (3.37%) and higher TPR (0.953), but accepts fewer tokens per call (1.793 versus 1.881 for ChiselMark and 1.813 for watermark-free decoding). At these settings, sparse KGW gives the highest TPR and lowest overhead, while ChiselMark gives the most accepted tokens per target call.

Feature Level: Verification-Side Coverage Gating. We next ask whether leaving the feature-level draft tree intact is sufficient to avoid efficiency loss. To focus on the verification stage, all EAGLE-1 variants in Table 1 preserve the native draft tree. At $\delta = 4$, dense verification-only watermarking reaches TPR 1.000 but incurs 19.45% overhead and reduces accepted tokens per call from the watermark-free 1.871 to 1.622. The coverage gate ($\tau_{\text{tree}} = 0.01$) instead reaches TPR 0.961, lowers overhead to 1.70%, and raises accepted

Table 1: **Selected held-out operating points.** Detection, wall-clock overhead, and accepted tokens per target call are reported together because they can move in different directions. PPL ratios provide a separate measure of text quality. Additional tested points are plotted in Figure 9.

Architecture	Method	TPR@1% FPR	Overhead	Tokens/target call	PPL ratio
Token	Watermark-free	—	0.00%	1.813	1.000
Token	Dense KGW, $\delta = 1.5$	0.907	5.68%	1.817	1.146
Token	Sparse KGW, $\delta = 5, \rho = 0.25$	**0.953**	**3.37%**	1.793	1.238
Token	ChiselMark, $\delta = 4, \rho = 0.5, q_G \geq 0.35$	0.907	3.43%	**1.881**	**1.124**
EAGLE-1	Watermark-free	—	0.00%	1.871	1.000
EAGLE-1	Dense verifier, $\delta = 2$	0.978	9.18%	1.776	1.342
EAGLE-1	Dense verifier, $\delta = 4$	1.000	19.45%	1.622	2.184
EAGLE-1	Sparse Verification-Only KGW, $\delta = 4, \rho = 0.25$	0.917	4.89%	1.813	1.183
EAGLE-1	Coverage gate, $\delta = 4, \tau_{\text{tree}} = 0.01$	0.961	**1.70%**	**1.913**	1.296

tokens per target call to 1.913. At these selected points, it also improves TPR, overhead, and acceptance relative to ungated sparse verification ($\rho = 0.25$, TPR 0.917, overhead 4.89%, acceptance 1.813). The benefit therefore extends beyond tree preservation: cost-aware admission can protect verifier-side reuse of already-constructed candidates.

5.5 Gate Mechanisms and Detection Trade-offs

We next examine which candidate writes the draft green-mass gate admits, how watermarking changes the contexts visited during autoregressive generation, and how gating affects the statistical evidence available to the text-and-key detector.

Local Compatibility of the Token-Level Gate. We test whether draft green-list mass predicts the single-step acceptance loss discussed in Section 3.3. Across 6,002 frozen Vicuna-7B/Vicuna-v1.3-68M states at $\delta = 4$, the draft-side score $C_t^Q = -\log q_G$ has Spearman correlation 0.686 with the measured positive-part acceptance loss. At $q_G \geq 0.35$, the mean positive-part loss is 0.018 for admitted states and 0.252 for vetoed states, approximately 14× lower. The gate therefore selects states with lower average positive-part loss using probabilities already available from the draft model.

Why Candidate Sparsity Is Still Needed. The gate assesses each write at its current context. Later acceptance also depends on the contexts reached after that write. In the tested model pair, the decomposition in Appendix E.7 gives negative contributions from same-context reweighting and changes in the mix of entropy levels. A positive within-entropy contribution partly offsets these losses. Candidate density ρ controls how often watermark writes are attempted across the sequence, complementing the gate's local decisions.

Detection Evidence at Candidate Sites. Gating changes the evidence for detection. The detector scores all eligible sites, including those where the gate blocked a write. Because the gate depends on the key, it can select sites with different score distributions. In the held-out study, evidence per candidate can be up to 75% lower than an estimate based only on written sites. We measure detection power over the full candidate set. False-positive control follows separately from the exchangeability assumptions in Theorem 4.5. Appendix E.6 explains the evidence calculation, and Appendix E.7 gives the trajectory analysis.

6 Conclusion

In this paper, we revisited KGW-style watermarking under speculative sampling. We revealed that a vanilla construction, which applies the same logit reweighting to the draft and the target, preserves the intended watermarked distribution at the token level yet systematically erodes speculative acceptance under both token-level and feature-level decoders. We traced this erosion to a latent assumption inherited from standard autoregressive decoding, namely that the reweighted distribution is also the one whose statistics determine the cost of a write. Under speculative sampling, that cost is instead paid on the proposal side, by the draft distribution at the token level and by the already-constructed candidate tree at the feature level. Based on this understanding, we proposed ChiselMark, which decomposes watermark strength

into keyed candidate sites, an affordability gate, and architecture-aware placement, while remaining detectable from the generated text, the secret key, and public detector parameters alone. ChiselMark samples exactly from the gated target at the token level and leaves native draft trees intact at the feature level. Extensive experiments across token-level speculative sampling, Medusa, and three generations of EAGLE verified that ChiselMark recovers much of the speculative efficiency lost to dense watermarking while keeping detectability competitive, as well as its robustness to text edits. We hope that this work can motivate the co-design of generation-time watermarks with the inference architectures on which they are deployed, rather than treating them as orthogonal interventions on the next-token distribution.

7 Data Availability

The publicly available benchmark datasets used in this study, including HumanEval, BigCodeBench, TinyStories, C4, and MT-Bench, can be accessed through their original sources as cited in the manuscript. The experimental results supporting the findings of this study are presented in the main text and Appendix. Additional evaluation results and generated data are available from the corresponding author upon reasonable request.

8 Code Availability

The source code implementing ChiselMark, including the watermark embedding and detection procedures, is provided in Supplementary Materials.

9 Acknowledgements

This research is supported by the National Research Foundation, Singapore, and Cyber Security Agency of Singapore under its National Cybersecurity R&D Programme and CyberSG R&D Cyber Research Programme Office. Any opinions, findings and conclusions or recommendations expressed in these materials are those of the author(s) and do not reflect the views of National Research Foundation, Singapore, Cyber Security Agency of Singapore as well as CyberSG R&D Programme Office, Singapore.

10 Use of LLM

During manuscript preparation, the authors used large language models solely to assist with language polishing and minor structural editing. The authors reviewed and revised the assisted text and take full responsibility for the final content of the manuscript.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[3] Anthropic. Detecting and countering misuse of ai: September 2026. Technical report, Anthropic, September 2026. URL https://www.anthropic.com/threat-intelligence-report-september-2026.

[4] Cyberspace Administration of China et al. Interim measures for the management of generative artificial intelligence services, 2023. Order No. 15, jointly issued by seven agencies, effective 15 August 2023.

[5] White House Office of Science and Technology Policy. Adversarial distillation of american AI models (NSTM-4), 2026. Memorandum, 23 April 2026.

[6] U.S. House of Representatives. Deterring american AI model theft act of 2026, H.R. 8283, 119th congress, 2026. Introduced 15 April 2026.

[7] U.S. Senate. Blocking large-scale adversarial distillation efforts (BLADE) act of 2026, S. 5252, 119th congress, 2026. Introduced 5 August 2026.

[8] Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. Protecting intellectual property of language generation APIs with lexical watermark. In *AAAI*, 2022.

[9] Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Protecting language generation models via invisible watermarking. In *Proceedings of the International Conference on Machine Learning*, 2023.

[10] Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. In *Advances in Neural Information Processing Systems*, 2024.

[11] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 57(2):1–36, 2024.

[12] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023.

[13] Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. MarkLLM: An open-source toolkit for LLM watermarking. In Delia Irazu Hernandez Farias, Tom Hope, and Manling Li, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 61–71, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-demo.7. URL https://aclanthology.org/2024.emnlp-demo.7/.

[14] Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4890–4911. Association for Computational Linguistics, 2024.

[15] Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. An entropy-based text watermarking detection method. *arXiv preprint arXiv:2403.13485*, 2024.

[16] Zongqi Wang, Tianle Gu, Baoyuan Wu, and Yujiu Yang. Morphmark: Flexible adaptive watermarking for large language models. *arXiv preprint arXiv:2505.11541*, 2025.

[17] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[18] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[19] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv: 2401.10774*, 2024.

[20] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, 2024.

[21] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Empirical Methods in Natural Language Processing*, 2024.

[22] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Annual Conference on Neural Information Processing Systems*, 2025.

[23] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Looking back at speculative decoding. Google Research Blog, 2024. 6 December 2024, https://research.google/blog/looking-back-at-speculative-decoding/.

[24] Bangsheng Tang, Carl Chengyan Fu, Fei Kou, Grigory Sizov, Haoci Zhang, et al. Efficient speculative decoding for Llama at scale: Challenges and solutions. *arXiv preprint arXiv:2508.08192*, 2025.

[25] DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[26] GLM-4.5 Team. GLM-4.5: Agentic, reasoning, and coding (ARC) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

[27] Xi Yang, Kejiang Chen, Weiming Zhang, Chang Liu, Yuang Qi, Jie Zhang, Han Fang, and Nenghai Yu. Watermarking text generated by black-box language models. *arXiv preprint arXiv:2305.08883*, 2023.

[28] Zhengmian Hu and Heng Huang. Inevitable trade-off between watermark strength and speculative sampling efficiency for language models. *Advances in Neural Information Processing Systems*, 37:55370–55402, 2024.

[29] Weiqing He, Xiang Li, Li Shen, Weijie Su, and Qi Long. Improving the trade-off between watermark strength and speculative sampling efficiency for language models. *arXiv preprint arXiv:2602.01428*, 2026.

[30] Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.

[31] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[32] Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, volume 2025, pages 66602–66656, 2025.

[33] Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

[34] Yepeng Liu and Yuheng Bu. Adaptive text watermark for large language models. *arXiv preprint arXiv:2401.13927*, 2024.

[35] Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. Unbiased watermark for large language models. *arXiv preprint arXiv:2310.10669*, 2023.

[36] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

[37] Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. A resilient and accessible distribution-preserving watermark for large language models. *arXiv preprint arXiv:2310.07710*, 2023.

[38] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2023.

[39] Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. On the learnability of watermarks for language models. *arXiv preprint arXiv:2312.04469*, 2023.

[40] Minjia Mao, Dongjun Wei, Zeyu Chen, Xiao Fang, and Michael Chau. A watermark for low-entropy and unbiased generation in large language models. *arXiv preprint arXiv:2405.14604*, 2024.

[41] Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, et al. Glide with a cape: A low-hassle method to accelerate speculative decoding. *arXiv preprint arXiv:2402.02082*, 2024.

[42] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification. *arXiv preprint arXiv:2305.09781*, 2023.

[43] Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

[44] Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*, 2024.

[45] Abe Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. Semstamp: A semantic watermark with paraphrastic robustness for text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4067–4082, 2024.

[46] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[47] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67, 2020.

[48] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36: 46595–46623, 2023.

[49] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[50] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36, 2024.

Appendix

The appendices follow the main argument: Appendix A establishes the methodological scope; Appendix B develops the diagnosis; Appendix C specifies the implementation; Appendix D provides the theoretical analyses; and Appendix E collects the experimental protocols and supporting results.

A Extended Related Work and Baseline Compatibility

A.1 Text Watermarking for LLMs

Subsequent logit-reweighting methods differ in how they control this intervention: SWEET [14] selects high-entropy positions for embedding, EWD [15] weights detection evidence by entropy, MorphMark [16] adaptively adjusts watermark strength according to the token distribution, and related adaptive schemes use an auxiliary model to estimate token entropy and determine where watermarking should be applied [34]. All of these KGW-style schemes intervene at the next-token distribution.

A complementary family of *distortion-free* or *unbiased* watermarks modifies the sampling procedure while preserving the original output distribution in expectation over watermark randomness. Unbiased watermarking (UW) employs randomized reweighting and sampling rules [35]; KTH-style watermarking uses keyed pseudorandom sampling to introduce detectable signals [36]; DiPmark develops distribution-preserving watermarking with practical detection and resilience to text modifications [37]; and cryptographically motivated constructions study watermark undetectability under explicit adversarial assumptions [38]. Unlike KGW-style methods that deliberately shift the next-token distribution, these approaches seek to preserve the marginal output law while introducing detectable dependence on secret randomness.

Beyond generation-time watermarking, post-hoc text watermarks [27] embed signals into finished outputs through text-level modifications, forgoing the lightweight nature of generation-time schemes. Robustness and security studies further examine detectability under transformations and adaptive adversaries, including learnability analyses that investigate whether watermark signals can be inferred or neutralized through model adaptation [39], and low-entropy watermarking methods that seek to preserve distributional fidelity in regimes with limited token uncertainty [40].

A.2 Speculative Sampling

Medusa [19] employs parallel decoding heads that predict future tokens from a shared backbone state, forming a draft tree from combinations of their candidates. In contrast, the EAGLE family [20, 21, 22] employs autoregressive feature-level drafting in which successor proposals depend on previously proposed tokens, with later versions improving dynamic tree construction and prediction efficiency. Token-level SpS and EAGLE-style decoding with exact verification and rejection correction can preserve the target distribution; this guarantee does not generally apply to Medusa's typical-acceptance variant. The practical relevance of speculative and multi-token decoding is further reflected in studies of large-scale inference systems [23, 24, 25, 26]. Other decoder variants and evaluation frameworks are covered in the broader literature [41, 42, 43, 44].

A.3 Watermarking under Speculative Sampling and Baseline Selection

Our empirical comparison uses KGW, SWEET, EWD, and MorphMark because they share the next-token logit-reweighting interface examined in the main text. This common interface permits a controlled comparison of watermark interventions across the evaluated speculative architectures; implementation parameters are collected in Appendix E.2.

Baseline selection. Our direct empirical comparison covers methods that share the KGW-style logit-reweighting interface across the evaluated decoders. Existing work integrates

unbiased watermarking with *token-level* speculative sampling by modifying sampling, acceptance, or detection [28, 29]. Applying these methods across Medusa and EAGLE would require corresponding tree-based implementations. Table 2 summarizes the required changes and the available implementations. Semstamp [45] works in a sentence-level semantic space; including it in this comparison would require a compatible speculative implementation.

Table 2: Direct compatibility of representative watermarking methods with speculative-decoding settings.

Watermark Method	**Token-level SpS**	**Feature-level SpS**
DiPmark [37]	Requires redesign	Requires redesign
Reason: DiPmark changes the next-token sampling rule through a key-conditioned transformation. Under standard speculative sampling, draft proposals are subsequently corrected against the verifier's target distribution. Modifying only the proposal distribution therefore does not, in general, preserve the intended watermark behavior, whereas adopting the watermarked distribution at the verifier also changes speculative acceptance. A faithful integration consequently requires jointly redesigning the proposal and verification procedure rather than applying DiPmark as a drop-in sampler.		
KTH [36]	Requires redesign	Requires redesign
Reason: KTH relies on inverse-transform sampling that couples pseudorandom values to token selection. Standard speculative sampling introduces an additional proposal–accept/reject– correction procedure, so the original pseudorandom coupling is not preserved under a naive composition. Maintaining the intended sampling law therefore requires a specialized coupling of the speculative randomness and corresponding modifications to detection. Extending this construction to multi-branch feature-level proposals would require additional design.		
UW [35]	Specialized integration [28] [29]	Requires redesign
Reason: Unbiased watermarking based on several specific sampling has been explicitly studied in token-level speculative sampling [28, 29], demonstrating that compatibility is possible with appropriately redesigned sampling or acceptance procedures. These constructions, however, are specialized to token-level speculative sampling and do not directly provide a verifier-compatible implementation for the tree-structured proposals used by Medusa and the EAGLE family.		
Christ [38]	Requires redesign	Requires redesign
Reason: The method of Christ et al. uses a sequential, context-dependent cryptographic sampling construction. A speculative decoder evaluates multiple hypothetical continuations before committing to a single output path, while feature-level methods additionally construct multi-branch draft trees. Direct integration would therefore require defining consistent keyed randomness across speculative branches and adapting both the verification and detection procedures. We are not aware of an off-the-shelf implementation supporting the feature-level speculative architectures evaluated here.		

B Mechanistic Analysis of Watermarking under Speculative Sampling

B.1 Correctness of the Vanilla Watermark for Token-Level SpS

Fix a decoding context. Let $\mathbf{p}, \mathbf{q} \in \Delta_{\mathcal{V}}$ denote the target and draft next-token distributions, respectively. Applying the KGW watermark transformation f_w with the same green list G and bias δ gives

$$\mathbf{p}_w = f_w(\mathbf{p}), \qquad \mathbf{q}_w = f_w(\mathbf{q}).$$

We show that token-level speculative sampling with target distribution $\mathbf{p}_w$ and draft distribution $\mathbf{q}_w$ produces an output token $x \sim \mathbf{p}_w$. The argument follows the correctness proof of standard speculative sampling [17].

The algorithm first samples a draft token $\tilde{x} \sim \mathbf{q}_w$. Given $\tilde{x} = i$ with $\mathbf{q}_w(i) > 0$, it accepts the draft token with probability

$$\Pr(\text{accept} \mid \tilde{x} = i) = \min\left\{1, \frac{\mathbf{p}_w(i)}{\mathbf{q}_w(i)}\right\}.$$

Thus, the overall acceptance probability is

$$\beta_w = \beta(\mathbf{p}_w, \mathbf{q}_w) = \sum_{i \in \mathcal{V}} \min\{\mathbf{p}_w(i), \mathbf{q}_w(i)\}.$$

If $\beta_w = 1$, then $\mathbf{p}_w = \mathbf{q}_w$, and every draft token is accepted, so the claim follows immediately. Otherwise, upon rejection, the algorithm samples the output token from the residual distribution

$$\mathbf{p}_{w,\text{res}}(j) = \frac{(\mathbf{p}_w(j) - \mathbf{q}_w(j))_+}{\sum_{z \in \mathcal{V}} (\mathbf{q}_w(z) - \mathbf{p}_w(z))_+} = \frac{(\mathbf{p}_w(j) - \mathbf{q}_w(j))_+}{1 - \beta_w}, \qquad j \in \mathcal{V},$$

where $(a)_+ = \max\{0, a\}$. The equality follows from

$$\sum_{z \in \mathcal{V}} (\mathbf{q}_w(z) - \mathbf{p}_w(z))_+ = \sum_{z \in \mathcal{V}} (\mathbf{p}_w(z) - \mathbf{q}_w(z))_+ = 1 - \beta_w.$$

For any $j \in \mathcal{V}$, the output event $x = j$ can occur through two mutually exclusive paths: accepting the draft token or resampling after rejection. The acceptance path contributes

$$\Pr(x = j, \text{accept}) = \min\{\mathbf{q}_w(j), \mathbf{p}_w(j)\}.$$

This expression also holds when $\mathbf{q}_w(j) = 0$, in which case the acceptance path has probability zero.

The rejection path contributes

$$\begin{aligned} \Pr(x = j, \text{reject}) &= (1 - \beta_w)\mathbf{p}_{w,\text{res}}(j) \\ &= (\mathbf{p}_w(j) - \mathbf{q}_w(j))_+. \end{aligned} \tag{14}$$

Combining the two paths yields

$$\begin{aligned} \Pr(x = j) &= \Pr(x = j, \text{accept}) + \Pr(x = j, \text{reject}) \\ &= \min\{\mathbf{q}_w(j), \mathbf{p}_w(j)\} + (\mathbf{p}_w(j) - \mathbf{q}_w(j))_+ \\ &= \mathbf{p}_w(j). \end{aligned} \tag{15}$$

Therefore, $x \sim \mathbf{p}_w$, proving that token-level speculative sampling preserves the watermarked target distribution exactly. □

B.2 First-Order Neutrality and Its Proof

The fixed-state question. To isolate the local effect, hold $\mathbf{p}$ and $\mathbf{q}$ fixed, apply a common green list G, and examine the derivative of β_w at $\delta = 0$. A realized green list can locally raise or lower acceptance. Under equal marginal green-list probabilities and the no-ties assumption stated below, these first-order effects cancel in expectation over G.

Corollary B.1 (First-order neutrality under a random KGW green list)**.** *Under the standard pseudorandomness idealization, suppose that, conditioned on the current target and draft distributions* $\mathbf{p}$ *and* $\mathbf{q}$*, each token has the same marginal probability of being assigned to the green list. Assuming* $\mathbf{p}(x) \neq \mathbf{q}(x)$ *whenever* $\mathbf{p}(x) + \mathbf{q}(x) > 0$*, we have*

$$\mathbb{E}_G\left[\left.\frac{d\beta_w}{d\delta}\right|_{\delta=0} \middle| \mathbf{p}, \mathbf{q}\right] = 0. \tag{16}$$

From Zero Bias to Finite Bias. At finite bias, different green-list masses in the target and draft produce different normalizers and change their overlap. Repeated writes also change the contexts reached during generation. These are the normalization and context-drift mechanisms examined in Section 3.3.

The influence vector derived below satisfies $\sum_x c_x = 0$, which explains the cancellation in Corollary B.1. Appendix B.2.1 gives examples at finite bias. We next derive the influence vector and prove the corollary.

This appendix proves Corollary B.1. Throughout, $\mathbf{p}$ and $\mathbf{q}$ are the target and draft distributions at a fixed decoding position, and $\mathbf{p_w} = f_w(\mathbf{p})$ and $\mathbf{q_w} = f_w(\mathbf{q})$ are their reweightings under Eq. (1) with the same green list $G \subset \mathcal{V}$ and the same bias δ, whose dependence on δ we make explicit where needed. We write

$$L = \{x : \mathbf{p}(x) \le \mathbf{q}(x)\}, \quad a = \sum_{x \in L} \mathbf{p}(x), \quad b = \sum_{x \notin L} \mathbf{q}(x), \quad P_G = \sum_{x \in G} \mathbf{p}(x), \quad Q_G = \sum_{x \in G} \mathbf{q}(x),$$

so that $\beta_w|_{\delta=0} = \beta(\mathbf{p}, \mathbf{q}) = a + b$. The assumption of no nontrivial ties means $\mathbf{p}(x) \neq \mathbf{q}(x)$ whenever $\mathbf{p}(x) + \mathbf{q}(x) > 0$; tokens with $\mathbf{p}(x) = \mathbf{q}(x) = 0$ contribute nothing to Eq. (4).

Lemma B.2 (Local stability of the accept region)**.** *Assume no nontrivial ties. Then there exists $\varepsilon > 0$, depending only on $(\mathbf{p}, \mathbf{q})$, such that for every $G \subset \mathcal{V}$ and every $|\delta| < \varepsilon$,*

$$\{x : \mathbf{p_w}(x) \le \mathbf{q_w}(x)\} = L.$$

Proof. By Eq. (1),

$$\mathbf{p_w}(x) - \mathbf{q_w}(x) = e^{\delta \mathbf{1}_G(x)} \left(\frac{\mathbf{p}(x)}{Z_{\mathbf{p}}(\delta)} - \frac{\mathbf{q}(x)}{Z_{\mathbf{q}}(\delta)} \right),$$

which is continuous in δ and equals $\mathbf{p}(x) - \mathbf{q}(x) \neq 0$ at $\delta = 0$ for every token carrying mass. Hence for each such x and each G there is a neighborhood of $\delta = 0$ on which the sign is preserved. Taking the smallest such neighborhood over the finitely many tokens and the finitely many green lists gives a single $\varepsilon > 0$. □

Proposition B.3 (Token-level first-order influence)**.** *Assume no nontrivial ties. Then*

$$\left. \frac{d\beta_w}{d\delta} \right|_{\delta=0} = \sum_{x \in G} c_x, \tag{17}$$

$$c_x = \min\{\mathbf{p}(x), \mathbf{q}(x)\} - a\,\mathbf{p}(x) - b\,\mathbf{q}(x).$$

and the influence vector $c \in \mathbb{R}^{|\mathcal{V}|}$, which depends only on $(\mathbf{p}, \mathbf{q})$ and not on G, is zero-sum:

$$\sum_{x \in \mathcal{V}} c_x = 0. \tag{18}$$

Proof. By Lemma B.2, for $|\delta| < \varepsilon$ the minimum in Eq. (4) is attained by $\mathbf{p_w}$ exactly on L, so

$$\beta_w = \sum_{x \in L} \mathbf{p_w}(x) + \sum_{x \notin L} \mathbf{q_w}(x) = \frac{A(\delta)}{Z_{\mathbf{p}}(\delta)} + \frac{B(\delta)}{Z_{\mathbf{q}}(\delta)},$$

where $A(\delta) = \sum_{x \in L} \mathbf{p}(x) e^{\delta \mathbf{1}_G(x)}$ and $B(\delta) = \sum_{x \notin L} \mathbf{q}(x) e^{\delta \mathbf{1}_G(x)}$. Evaluating at $\delta = 0$,

$$A(0) = a, \quad A'(0) = \sum_{x \in L \cap G} \mathbf{p}(x), \quad B(0) = b, \quad B'(0) = \sum_{x \notin L,\, x \in G} \mathbf{q}(x), \quad Z_{\mathbf{p}}(0) = Z_{\mathbf{q}}(0) = 1,$$

with $Z'_{\mathbf{p}}(0) = P_G$ and $Z'_{\mathbf{q}}(0) = Q_G$. The quotient rule gives

$$\left. \frac{d\beta_w}{d\delta} \right|_{\delta=0} = \big(A'(0) - aP_G\big) + \big(B'(0) - bQ_G\big) \tag{19}$$

$$= \sum_{x \in G} \min\{\mathbf{p}(x), \mathbf{q}(x)\} - \sum_{x \in G} \big(a\,\mathbf{p}(x) + b\,\mathbf{q}(x)\big) \;=\; \sum_{x \in G} c_x, \tag{20}$$

where the second line uses $A'(0) + B'(0) = \sum_{x \in G} \min\{\mathbf{p}(x), \mathbf{q}(x)\}$, valid because L selects the pointwise minima. Finally,

$$\sum_{x \in \mathcal{V}} c_x = \sum_{x \in \mathcal{V}} \min\{\mathbf{p}(x), \mathbf{q}(x)\} - a \sum_x \mathbf{p}(x) - b \sum_x \mathbf{q}(x) = (a + b) - a - b = 0.$$

□

Equation (18) implies that, unless $c \equiv 0$, the influence vector has entries of both signs; a green list therefore never creates first-order influence but only redistributes it, and whether watermarking locally helps or hurts acceptance depends entirely on which tokens the realized green list contains.

Proof of Corollary B.1. By Proposition B.3, and since c is a deterministic function of $(\mathbf{p}, \mathbf{q})$ alone,

$$\mathbb{E}_G\left[\left.\frac{d\beta_w}{d\delta}\right|_{\delta=0} \,\middle|\, \mathbf{p}, \mathbf{q}\right] = \sum_{x\in\mathcal{V}} \Pr\big(x \in G \mid \mathbf{p}, \mathbf{q}\big)\, c_x = \gamma \sum_{x\in\mathcal{V}} c_x = 0,$$

using the assumption of a common marginal green probability γ and then Eq. (18).

To establish the second-order bound on the expected acceptance change, fix G and let $\varepsilon > 0$ be as in Lemma B.2. On $(-\varepsilon, \varepsilon)$ the function $\delta \mapsto \beta_w$ is a sum of two ratios of finite exponential sums with nonvanishing denominators, hence smooth, so Taylor's theorem gives

$$\beta_w = \beta(\mathbf{p}, \mathbf{q}) + \delta \sum_{x\in G} c_x + R_G(\delta), \qquad |R_G(\delta)| \le \tfrac{1}{2} M_G \delta^2, \qquad M_G = \sup_{|s|\le|\delta|} \left|\frac{d^2\beta_w}{d\delta^2}(s)\right|.$$

Since $\mathcal{V}$ is finite there are finitely many green lists, so $M = \max_G M_G < \infty$ and the bound holds uniformly in G on the common neighborhood $(-\varepsilon, \varepsilon)$. Taking conditional expectations and using the first display,

$$|\mathbb{E}_G[\beta_w \mid \mathbf{p}, \mathbf{q}] - \beta(\mathbf{p}, \mathbf{q})| = |\mathbb{E}_G[R_G(\delta)]| \le \tfrac{1}{2} M \delta^2.$$

□

Remark B.4. The hypothesis of Corollary B.1 constrains only the *marginal* green probability of each token; it does not require the events $\{x \in G\}$ to be independent across the vocabulary. It is nonetheless an idealization, since in KGW the green list is seeded by the preceding tokens, which also determine $\mathbf{p}$ and $\mathbf{q}$ at the current position. We adopt the standard pseudorandomness idealization and validate the prediction empirically.

B.2.1 Toy Examples with Finite Watermark Bias

We provide two two-token examples to illustrate how finite watermark bias affects speculative acceptance.

Example 1: relatively matched target and draft. Consider $p = (0.9, 0.1)$ and $q = (0.8, 0.2)$, with baseline acceptance rate $\beta(0) = \min(0.9, 0.8) + \min(0.1, 0.2) = 0.9$ and token-level first-order influences $(c_1, c_2) = (0.07, -0.07)$. Local analysis predicts that assigning token 1 to the green list G increases acceptance, whereas assigning token 2 decreases it. At $\delta = 0.5$, setting $G = \{1\}$ yields $p^\delta \approx (0.9369, 0.0631)$ and $q^\delta \approx (0.8683, 0.1317)$, giving $\beta(0.5) = \min(0.9369, 0.8683) + \min(0.0631, 0.1317) \approx 0.9315$ ($\beta(0.5) - \beta(0) \approx +0.0315$). In contrast, setting $G = \{2\}$ yields $p^\delta \approx (0.8473, 0.1527)$ and $q^\delta \approx (0.7081, 0.2919)$, giving $\beta(0.5) \approx 0.8630$ ($\beta(0.5) - \beta(0) \approx -0.0370$). Consistent with the signs of (c_1, c_2), the same watermark bias produces opposite effects depending on the green list: $G = \{1\} \Rightarrow \beta(0.5) > \beta(0)$, whereas $G = \{2\} \Rightarrow \beta(0.5) < \beta(0)$.

Example 2: strongly mismatched target and draft. Next, consider $p = (0.9, 0.1)$ and $q = (0.2, 0.8)$. The baseline acceptance rate drops to $\beta(0) = \min(0.9, 0.2) + \min(0.1, 0.8) = 0.3$, while the first-order influence vector remains $(c_1, c_2) = (0.07, -0.07)$. At $\delta = 0.5$, $G = \{1\}$ yields $p^\delta \approx (0.9369, 0.0631)$ and $q^\delta \approx (0.2919, 0.7081)$, increasing acceptance to $\beta(0.5) \approx 0.3550$ ($\beta(0.5) - \beta(0) \approx +0.0550$). Conversely, $G = \{2\}$ yields $p^\delta \approx (0.8473, 0.1527)$ and $q^\delta \approx (0.1317, 0.8683)$, reducing acceptance to $\beta(0.5) \approx 0.2865$ ($\beta(0.5) - \beta(0) \approx -0.0135$). Again, the green-list choice flips the outcome: $G = \{1\} \Rightarrow \beta(0.5) = 0.3550 > 0.3$, while $G = \{2\} \Rightarrow \beta(0.5) = 0.2865 < 0.3$.

Connection to the first-order analysis. For small δ, $\beta(\delta) = \beta(0) + \delta \sum_{i\in G} c_i + O(\delta^2)$. Since $(c_1, c_2) = (0.07, -0.07)$ in both examples, the first-order approximation predicts $\beta(\delta) \approx \beta(0) + 0.07\delta$ for $G = \{1\}$ and $\beta(\delta) \approx \beta(0) - 0.07\delta$ for $G = \{2\}$, matching the finite-bias calculations above.

Why a single-token-per-bucket approximation can fail. In the two-token examples above, each color bucket contains exactly one token, so representing each bucket by its most likely token is exact. For a full language-model distribution, however, approximating each bucket by a single representative token can discard substantial tail mass and distort the predicted acceptance behavior. Consider refining Example 1 ($G = \{2\}$) by splitting token 2 into ten green tail tokens $g_1, \ldots, g_{10}$ while preserving total mass: $\mathbf{p}(r) = 0.9, \mathbf{p}(g_j) = 0.01$ and $\mathbf{q}(r) = 0.8, \mathbf{q}(g_j) = 0.02$ for $j = 1, \ldots, 10$. The aggregate green masses remain $p_G = 0.1$ and $q_G = 0.2$ with identical within-green conditional distributions, leaving the exact bucket-level problem—and $\beta_w \approx 0.8630$ at $\delta = 0.5$—unchanged. However, the two-atom reduction retains only one green tail token and the red mode, replacing the true bucket log-odds $\tilde{a} = \log \frac{p_G}{1-p_G} = \log \frac{0.1}{0.9} \approx -2.197$ and $\tilde{b} = \log \frac{q_G}{1-q_G} = \log \frac{0.2}{0.8} \approx -1.386$ with $a = \log \frac{\mathbf{p}(g_m)}{\mathbf{p}(r_m)} = \log \frac{0.01}{0.9} \approx -4.500$ and $b = \log \frac{\mathbf{q}(g_m)}{\mathbf{q}(r_m)} = \log \frac{0.02}{0.8} \approx -3.689$. Consequently, the true crossing point $\delta^\star = -\frac{\tilde{a}+\tilde{b}}{2} \approx 1.79$ is displaced to $\delta^\star_{\text{atom}} = -\frac{a+b}{2} \approx 4.09$. Thus, the failure stems not from the crossing-window idea itself, but from estimating a *bucket mass* using a single representative token. Splitting fixed tail mass across more tokens preserves exact watermarked acceptance but arbitrarily shrinks the largest tail token's probability, shifting the two-atom prediction. This is precisely the regime of peaked language-model distributions, where a dominant token carries most of the mass and the remainder is spread across a large vocabulary.

B.3 Finite-Bias Normalization and Autoregressive Context Drift

We now study finite-bias acceptance by separating changes in overlap at a fixed context from changes in the contexts visited during generation. The common token-wise bias cancels from the target–draft likelihood ratio, leaving a ratio of their normalizers. Repeated generation also changes the contexts at which the distributions and their overlap are evaluated.

B.3.1 Setup and Acceptance Estimators

Fix a context and let $\mathbf{p}, \mathbf{q} \in \Delta_{\mathcal{V}}$ denote the target and draft next-token distributions over $\mathcal{V}$. For a shared green list $G \subset \mathcal{V}$ and bias δ, define

$$\mathbf{p}_w(x) = \frac{\mathbf{p}(x)e^{\delta \mathbf{1}[x \in G]}}{Z_p}, \qquad \mathbf{q}_w(x) = \frac{\mathbf{q}(x)e^{\delta \mathbf{1}[x \in G]}}{Z_q}. \tag{21}$$

Let

$$\theta := e^\delta - 1, \qquad p_G := \sum_{x \in G} \mathbf{p}(x), \qquad q_G := \sum_{x \in G} \mathbf{q}(x), \tag{22}$$

so that

$$Z_p = 1 + \theta p_G, \qquad Z_q = 1 + \theta q_G. \tag{23}$$

We further define

$$\tilde{a} := \text{logit}(p_G), \qquad \tilde{b} := \text{logit}(q_G), \qquad r(x) := \frac{\mathbf{p}(x)}{\mathbf{q}(x)}, \tag{24}$$

the unwatermarked acceptance rate

$$\beta_0 := \sum_{x \in \mathcal{V}} \min\{\mathbf{p}(x), \mathbf{q}(x)\}, \qquad \beta_0^G := \sum_{x \in G} \min\{\mathbf{p}(x), \mathbf{q}(x)\}, \tag{25}$$

and the partition-function ratio

$$\lambda := \frac{Z_p}{Z_q} = \frac{1 + \theta p_G}{1 + \theta q_G}. \tag{26}$$

Under KGW, G is generated by a keyed hash of the previous ℓ tokens. For the analysis below, we use only that each vocabulary item has marginal green-list probability γ and that the random green list is independent of the current $(\mathbf{p}, \mathbf{q})$. Unless otherwise stated, the experiments use $\gamma = 0.5$, $\ell = 1$, temperature 1, and no truncation.

Comparison with the main-text acceptance sweep. The main-text sweep and the trajectory diagnostics use different acceptance estimators. Figure 1 averages realized acceptance rates uniformly over generation runs, whereas the trajectory diagnostics pool analytic overlaps over all reached verification positions. In Vicuna/TinyStories, an unintended trailing space in the prompts frequently caused immediate EOS termination. These one-token runs had zero accepted draft tokens in the recorded experiments, depressing the per-run average as their frequency increased. In contrast, they receive little weight in the pooled estimate. Recomputing both aggregation statistics on each dataset reproduces the apparent trend reversal, indicating that the discrepancy primarily reflects aggregation rather than contradictory acceptance behavior.

B.3.2 Structural Results and Static Replay

Proposition B.5 (Likelihood-ratio shift)**.** *Because the target and draft use the same green list and the same bias,*

$$\log \frac{\mathbf{p}_w(x)}{\mathbf{q}_w(x)} = \log \frac{\mathbf{p}(x)}{\mathbf{q}(x)} - \log \lambda \qquad \text{for every } x \in \mathcal{V}. \tag{27}$$

Consequently,

$$\beta_w = \mathbb{E}_{x\sim\mathbf{q}_w}\left[\min\left\{\frac{r(x)}{\lambda}, 1\right\}\right]. \tag{28}$$

Proof. From Eq. (21),

$$\frac{\mathbf{p}_w(x)}{\mathbf{q}_w(x)} = \frac{\mathbf{p}(x)e^{\delta \mathbf{1}[x\in G]}/Z_p}{\mathbf{q}(x)e^{\delta \mathbf{1}[x\in G]}/Z_q} = \frac{r(x)}{\lambda},$$

because the shared watermark factor cancels pointwise. Therefore

$$\sum_x \min\{\mathbf{p}_w(x), \mathbf{q}_w(x)\} = \sum_x \mathbf{q}_w(x) \min\left\{\frac{\mathbf{p}_w(x)}{\mathbf{q}_w(x)}, 1\right\},$$

which gives Eq. (28). □

Equation (27) shows that the common KGW bias shifts every target–draft log-likelihood ratio by the same amount, $-\log\lambda$, preserving their relative order. Acceptance also changes through the proposal distribution $\mathbf{q}_w$.

Proposition B.6 (Bucket invariance and closed form)**.** *The watermark leaves the within-bucket conditional distributions unchanged:*

$$\mathbf{p}_w(\cdot \mid G) = \mathbf{p}(\cdot \mid G), \qquad \mathbf{p}_w(\cdot \mid G^c) = \mathbf{p}(\cdot \mid G^c), \tag{29}$$

and analogously for $\mathbf{q}$*. The watermarked green masses are*

$$A_p := \mathbf{p}_w(G) = \frac{e^\delta p_G}{Z_p} = \sigma(\tilde{a} + \delta), \qquad A_q := \mathbf{q}_w(G) = \sigma(\tilde{b} + \delta). \tag{30}$$

Define

$$M(\lambda) := \sum_{x\in\mathcal{V}} \min\{\mathbf{p}(x), \lambda\mathbf{q}(x)\}, \qquad M_G(\lambda) := \sum_{x\in G} \min\{\mathbf{p}(x), \lambda\mathbf{q}(x)\}. \tag{31}$$

Then

$$\beta_w = \frac{\theta M_G(\lambda) + M(\lambda)}{1 + \theta p_G}, \qquad \lim_{\delta\to\infty} \beta_w = \sum_{x\in G} \min\{\mathbf{p}(x \mid G), \mathbf{q}(x \mid G)\}. \tag{32}$$

Proof. For any x,

$$\mathbf{q}_w(x) = \frac{e^{\delta \mathbf{1}[x\in G]}}{Z_p}\lambda\mathbf{q}(x),$$

and therefore

$$\min\{\mathbf{p}_w(x), \mathbf{q}_w(x)\} = \frac{e^{\delta \mathbf{1}[x \in G]}}{Z_p} \min\{\mathbf{p}(x), \lambda \mathbf{q}(x)\}.$$

Summing separately over G and G^c gives

$$\beta_w = \frac{e^{\delta} M_G(\lambda) + M(\lambda) - M_G(\lambda)}{Z_p} = \frac{\theta M_G(\lambda) + M(\lambda)}{1 + \theta p_G}.$$

The within-bucket conditional identities follow because the common multiplicative factor cancels under conditioning. As $\delta \to \infty$, both watermarked distributions concentrate on G while retaining their original conditionals within G, which gives the stated limit. □

Proposition B.6 separates the probability mass of each color group from the distribution within that group. The target and draft distributions within the green and red groups stay unchanged as δ varies, while the group masses A_p and A_q change. Thus (p_G, q_G, δ) determine the group masses, and exact acceptance also depends on target–draft overlap within each group.

The $\lambda = 1$ diagnostic. To isolate the normalization contribution algebraically, we set $\lambda \leftarrow 1$ in Eq. (32) while keeping the observed $(\mathbf{p}, \mathbf{q}, G, p_G)$ fixed. This gives

$$\beta_{\lambda \leftarrow 1}(\delta; G) := \frac{\beta_0 + \theta \beta_0^G}{1 + \theta p_G}. \tag{33}$$

Equivalently,

$$\beta_{\lambda \leftarrow 1}(\delta; G) = A R_G + (1 - A) R_{G^c}, \qquad R_G := \frac{\beta_0^G}{p_G}, \qquad R_{G^c} := \frac{\beta_0 - \beta_0^G}{1 - p_G}, \tag{34}$$

where $A = e^{\delta} p_G / (1 + \theta p_G)$. This is an algebraic diagnostic rather than a realisable sampler: it removes the likelihood-ratio displacement due to λ while retaining the observed green-list allocation.

Static acceptance-change decomposition.

To examine the contribution of normalizer mismatch on frozen $\delta = 0$ contexts, we use the algebraic $\lambda \leftarrow 1$ diagnostic $\beta_{\lambda \leftarrow 1}$ in Eq. (33) to split the fixed-state change from baseline acceptance $\beta_0 = \beta(\mathbf{p}, \mathbf{q})$ into

$$\beta_w - \beta_0 = \underbrace{\beta_{\lambda \leftarrow 1} - \beta_0}_{\text{retained allocation effect}} + \underbrace{\beta_w - \beta_{\lambda \leftarrow 1}}_{\text{normalization residual } N}. \tag{35}$$

The local derivative of the retained-allocation diagnostic is exact:

$$\frac{\partial}{\partial \delta} \beta_{\lambda \leftarrow 1}(\delta; G) \Big|_{\delta = 0} = \beta_0^G - \beta_0 p_G. \tag{36}$$

Because each token has marginal green-list probability γ,

$$\mathbb{E}_G[\beta_0^G] = \gamma \beta_0, \qquad \mathbb{E}_G[p_G] = \gamma,$$

and hence

$$\mathbb{E}_G \left[\frac{\partial}{\partial \delta} \beta_{\lambda \leftarrow 1}(\delta; G) \Big|_{\delta = 0} \right] = 0. \tag{37}$$

For a fixed green list, the diagnostic can change with the bias. Its expected first derivative at $\delta = 0$ is zero under the random green-list construction.

Figure 6 separates changes at a fixed context from changes along a generated trajectory. Panel (a) shows acceptance loss when only the scoring bias changes and the contexts remain fixed. Panel (b) shows a negative normalization residual across the experimental grid.

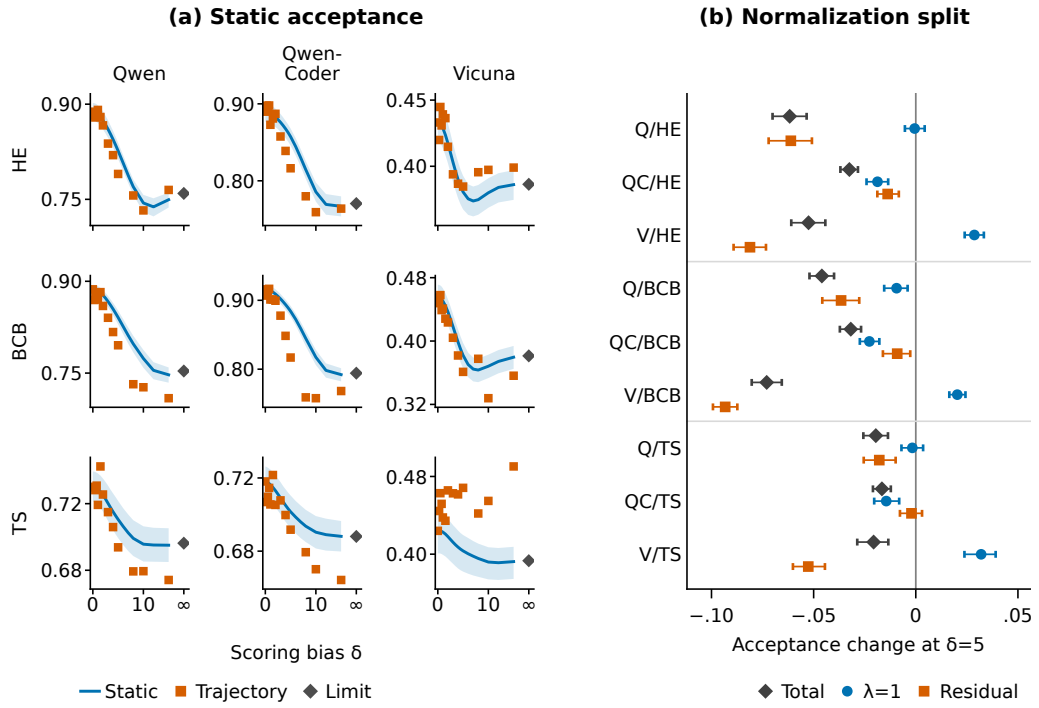


Figure 6: **Static acceptance and normalization decomposition. (a)** Static replay: contexts, target/draft logits, and green lists are frozen at $\delta = 0$, while only the scoring bias is varied. Acceptance decreases over the practically relevant bias range in nearly all configurations and recovers at larger bias toward the within-green conditional overlap in Eq. (32). The trajectory curve is shown for reference. **(b)** Static acceptance change at $\delta = 5$, decomposed into the $\lambda \leftarrow 1$ diagnostic of Eq. (33) and the residual normalization component. The residual has a negative point estimate in all nine configurations. The $\lambda \leftarrow 1$ component varies across configurations and is positive for Vicuna.

B.3.3 Autoregressive Drift: Composition and Within-Entropy Effects

The static analysis above freezes the context. During actual autoregressive generation, however, changing δ also changes the distribution of contexts at which the target and draft models are evaluated. We therefore separately characterize (i) how the visited context distribution changes and (ii) whether draft–target agreement systematically worsens among comparable-entropy states.

Figure 7 shows changes in the contexts reached as watermark bias increases. Continuation perplexity under the unwatermarked target and predictive entropy $H(\mathbf{p})$ increase broadly across the grid. Changes in draft–target agreement vary across configurations: several cells show higher $\mathrm{KL}(\mathbf{p}\|\mathbf{q})$ and lower top-1 agreement, while the original Vicuna/TinyStories run shows the opposite KL trend. We examine its sensitivity to prompt formatting below.

To distinguish a change in context composition from degradation among comparable states, let $\{B_j\}_{j=1}^{J}$ denote entropy bins constructed from the $\delta = 0$ context distribution, and define

$$w_{\delta j} := \Pr_{\pi_\delta}[H(\mathbf{p}) \in B_j], \qquad \mu_{\delta j} := \mathbb{E}_{\pi_\delta}[\beta_0 \mid H(\mathbf{p}) \in B_j]. \tag{38}$$

Then the drift in unwatermarked acceptance admits the exact decomposition

$$\underbrace{\mathbb{E}_{\pi_\delta}[\beta_0] - \mathbb{E}_{\pi_0}[\beta_0]}_{D(\delta)} = \underbrace{\sum_j \mu_{0j}\,(w_{\delta j} - w_{0j})}_{C(\delta)\text{: entropy composition}} + \underbrace{\sum_j w_{\delta j}\,(\mu_{\delta j} - \mu_{0j})}_{W(\delta)\text{: within entropy}}. \tag{39}$$

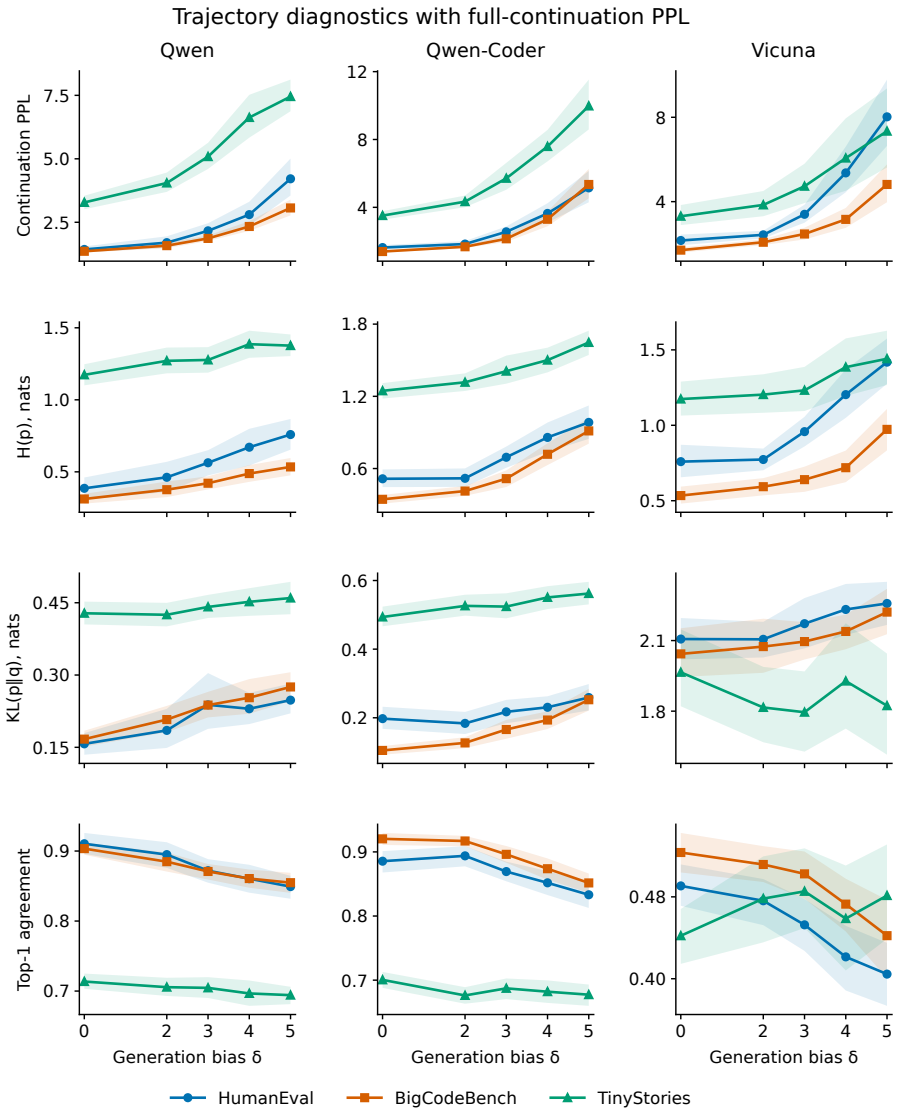


Figure 7: **Trajectory diagnostics under increasing generation bias.** Rows report full-continuation perplexity under the unwatermarked target, predictive entropy $H(\mathbf{p})$, trajectory-averaged $\mathrm{KL}(\mathbf{p}\|\mathbf{q})$, and top-1 agreement between the unwatermarked target and draft. Increasing δ generally moves generation toward contexts with higher perplexity and higher predictive entropy. Draft–target mismatch also changes along the trajectory, but its direction is not universal: several Qwen configurations show increasing KL and decreasing top-1 agreement, whereas Vicuna/TinyStories shows a decrease in KL and a non-monotonic, overall improved top-1-agreement trajectory.

Figure 8 shows a negative point estimate for the entropy-composition term in all nine configurations: watermarking shifts generation toward entropy ranges with lower baseline acceptance. The within-entropy term varies across configurations and is usually small or positive. The negative part of the drift is therefore largely associated with changes in the mix of entropy levels.

The original Vicuna/TinyStories run has positive total drift, with the within-entropy term offsetting the negative composition term. This result is sensitive to the prompt-formatting issue described above. After removing the unintended trailing whitespace, the corrected

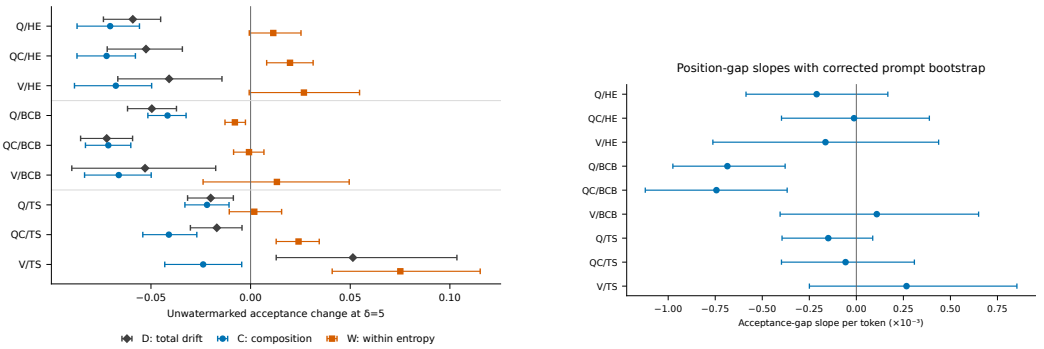


Figure 8: **Autoregressive drift decomposition and position-wise accumulation. Left:** Eq. (39) at $\delta = 5$. The entropy-composition component C has a negative point estimate in all nine configurations, whereas the within-entropy component W is heterogeneous and is often near zero or positive. In Vicuna/TinyStories, a large positive within-entropy component more than offsets the negative composition shift, yielding positive total drift. **Right:** Linear slopes of the acceptance gap versus sequence position with prompt-cluster bootstrap confidence intervals. Seven of nine intervals include zero; the two Qwen/BigCodeBench configurations exhibit significant negative slopes, providing evidence of accumulation in those two cases but not a universal compounding effect.

endpoint evaluation gives empirical acceptance rates of 0.445 at $\delta = 0$ and 0.433 at $\delta = 5$. The confidence interval for the change in unwatermarked overlap includes zero.

The position-wise analysis in Figure 8 finds a detectable negative slope in Qwen/BigCodeBench and Qwen-Coder/BigCodeBench. The other seven configurations show no detectable linear accumulation of the acceptance gap.

The analysis identifies two sources of acceptance change. At a fixed context, watermarking changes the target and draft normalizers. During generation, it also changes the mix of visited contexts. The entropy-composition contribution has a negative point estimate across the tested configurations, while the within-entropy contribution can partly or fully offset it.

B.4 Feature-Level Tree Structure

Medusa-Full and EAGLE-1-Full retain the complete candidate structures under their respective drafting configurations, while Medusa and EAGLE-1 use the original heuristic tree configurations.

Figure 2 shows substantial changes in draft candidates over the tested range $\delta \in [2, 5]$. At $\delta = 2$, Tree Distance is already around 0.6 or higher across the evaluated methods. Tree construction relies on greedy or top-K candidate selection, so changes in token rankings can alter the selected candidate set. For autoregressive tree builders such as EAGLE, such changes can also propagate to descendant proposals through parent-conditioned drafting. As δ increases, the extent of structural change evolves differently across methods. For Medusa and Medusa-Full, Tree Distance gradually approaches a plateau, whereas for EAGLE-1, EAGLE-1-Full, and EAGLE-3, it continues to increase with watermark strength. These differences indicate that sensitivity to watermarking depends on the underlying tree construction strategy adopted by each decoder.

The changes in draft candidates are accompanied by an overall decline in speculative acceptance, with different trends across configurations. For EAGLE-1, the complete-tree variant EAGLE-1-Full achieves higher acceptance than the heuristic variant, whereas Medusa-Full exhibits lower acceptance than Medusa. The effect of retaining the complete candidate structure therefore depends on the decoder. EAGLE-3 also exhibits increasing Tree Distance together with an overall decline in acceptance.

Beyond the structural changes shown above, the interaction between watermarking and tree-based decoding also involves verification-side alignment. At a fixed verification node, reweighting changes the target probability mass assigned to the existing child candidates, pulling it away from the already-drafted set even when the native tree is preserved. The feature-level interaction therefore involves both how watermarking affects candidate selection during drafting and how the resulting target distribution aligns with those candidates during verification; later experiments further decompose these two effects.

C Algorithms and Implementation Details

This appendix specifies how the main-text construction is implemented. Algorithms 1 and 2 give the token-level and feature-level embedding steps; Algorithm 3 gives the common detector. We fix context and stopping conventions first, then state the feature-verifier scope and the random-key calibration requirements. Formal distortion and allocation analyses are collected in Appendix D.

C.1 Shared Conventions and Embedding Procedures

Shared notation and keyed candidate-site selection. We follow Sections 4.2 and 4.3 of the manuscript. The secret key κ has domain-separated subkeys κ_{site}, κ_{sym}, and κ_{tok}. For a full prefix h_t before continuation position t, define $\textsc{KeyedOpportunity}(h_t, t; \kappa, \ell, \rho, B)$ to return (R_t, G_t). If $t \le \ell$, it returns $(0, \varnothing)$. Otherwise, it extracts the last ℓ generated tokens c_t from h_t (excluding the prompt and the next token), and evaluates the keyed site, symbol, and token maps of Section 4.2 to return (R_t, G_t). The site PRF is mapped to $[0, 1)$; symbol mappings are uniform on $\{0, \dots, B-1\}$, with unbiased implementation of the modulo operation. The early-position empty set is a harmless implementation convention: $R_t = 0$ disables reweighting. At tree nodes, the same routine uses the committed continuation followed by the root-to-node path, excluding the child token being verified. Identical contexts reuse identical keyed values. We use the KGW transformation in Eq. (1) and $[z]_+ = \max\{z, 0\}$. Here h denotes a full input prefix, n its number of generated continuation tokens, and L_{d} the configured token-level draft-block length. The two embedding algorithms are decoding steps, repeated until termination; generation stops immediately upon emitting EOS or reaching the output limit. A draft block may be shortened by the native stopping rule, in which case the loop bounds use its realized length.

Likelihood ratio at an active token-level write. Let $p_{G,t} = \sum_{v \in G_t} \mathbf{p}_t(v)$ and $q_{G,t} = \sum_{v \in G_t} \mathbf{q}_t(v)$. For $\mathbf{q}_t(v) > 0$, the shared token-specific factor cancels:

$$\frac{\mathbf{p}_{w,t}(v)}{\mathbf{q}_{w,t}(v)} = \frac{\mathbf{p}_t(v)}{\mathbf{q}_t(v)} \frac{Z_{q,t}}{Z_{p,t}}, \qquad Z_{p,t} = 1 + (e^{\delta_{\text{tok}}} - 1) p_{G,t}, \qquad Z_{q,t} = 1 + (e^{\delta_{\text{tok}}} - 1) q_{G,t}. \tag{40}$$

The normalizers therefore alter the pointwise acceptance test, while the changed proposal weights also enter the average overlap $\sum_v \min\{\mathbf{p}_{w,t}(v), \mathbf{q}_{w,t}(v)\}$. Algorithm 1 caches the pre-proposal gate and uses the actual proposal distribution for verification and correction.

Algorithm 1 Phase I: Token-Level ChiselMark Embedding (One Step)

Require: Prefix h with n generated tokens; models P, Q; key κ; ℓ, ρ, B; bias δ_{tok}; threshold τ_Q; draft length L_{d}.
Ensure: A verified output block extending h.

$h_{n+1} \leftarrow h$
for $t = n+1$ **to** $n + L_{\text{d}}$ **do**
  $q_t \leftarrow Q(\cdot \mid h_t)$; $(R_t, G_t) \leftarrow \textsc{KeyedOpportunity}(h_t, t; \kappa, \ell, \rho, B)$
  $q_{G,t} \leftarrow \sum_{v \in G_t} q_t(v)$; $C_t^Q \leftarrow -\log q_{G,t}$ $\{-\log 0 = +\infty\}$
  $A_t^{\text{tok}} \leftarrow \mathbb{I}[C_t^Q \le \tau_Q]$; $U_t^{\text{tok}} \leftarrow R_t A_t^{\text{tok}}$
  $q_{w,t} \leftarrow f_w(q_t; G_t, U_t^{\text{tok}} \delta_{\text{tok}})$
  Cache $(G_t, U_t^{\text{tok}}, q_{w,t})$ {Gate fixed before proposal sampling}
  $\tilde{x}_t \sim q_{w,t}$; $h_{t+1} \leftarrow h_t \,\|\, \tilde{x}_t$
end for
$(p_t)_{t=n+1}^{n+L_{\text{d}}+1} \leftarrow \textsc{TargetForward}(P, h, \tilde{x}_{n+1:n+L_{\text{d}}})$ $\{p_t = P(\cdot \mid h_t)\}$
for $t = n+1$ **to** $n + L_{\text{d}}$ **do**
  Retrieve $(G_t, U_t^{\text{tok}}, q_{w,t})$
  $p_{w,t} \leftarrow f_w(p_t; G_t, U_t^{\text{tok}} \delta_{\text{tok}})$
  $u \sim \text{Uniform}(0, 1)$
  if $u < \min\{1, p_{w,t}(\tilde{x}_t)/q_{w,t}(\tilde{x}_t)\}$ **then**
    Emit $x_t \leftarrow \tilde{x}_t$
  else
    $r_t(v) \leftarrow \dfrac{[p_{w,t}(v) - q_{w,t}(v)]_+}{\sum_{a \in \mathcal{V}} [p_{w,t}(a) - q_{w,t}(a)]_+}$
    Sample $x_t \sim r_t$ and emit x_t
    Discard later draft tokens; **return** emitted block
  end if
end for
$t \leftarrow n + L_{\text{d}} + 1$ {All proposals accepted: bonus token}
$q_t \leftarrow Q(\cdot \mid h_t)$; $(R_t, G_t) \leftarrow \textsc{KeyedOpportunity}(h_t, t; \kappa, \ell, \rho, B)$
$q_{G,t} \leftarrow \sum_{v \in G_t} q_t(v)$; $C_t^Q \leftarrow -\log q_{G,t}$
$A_t^{\text{tok}} \leftarrow \mathbb{I}[C_t^Q \le \tau_Q]$; $U_t^{\text{tok}} \leftarrow R_t A_t^{\text{tok}}$
$p_{w,t} \leftarrow f_w(p_t; G_t, U_t^{\text{tok}} \delta_{\text{tok}})$
Sample $x_t \sim p_{w,t}$ and emit x_t
return emitted block

Algorithm 2 Phase I: Feature-Level ChiselMark Embedding (One Step)

Require: Prefix h; target model P; native draft constructor $\mathcal{S}$ and state η; key κ; ℓ, ρ, B; bias δ_{feat}; threshold τ_{tree}.
Ensure: A verified output block and the native draft tree $\mathcal{S}(h;\eta)$.

Construct $\mathcal{S}(h;\eta)$ using unmodified draft logits $\{\mathcal{S}_w(h;\eta) = \mathcal{S}(h;\eta)\}$
$\mathcal{P} \leftarrow \textsc{TargetTreeForward}(P, \mathcal{S}(h;\eta))$
$\omega \leftarrow \text{root}(\mathcal{S}(h;\eta))$
loop
  $h_\omega \leftarrow$ prefix represented by node ω; $t \leftarrow 1 + \textsc{GeneratedLength}(h_\omega)$
  $(R_\omega, G_\omega) \leftarrow \textsc{KeyedOpportunity}(h_\omega, t; \kappa, \ell, \rho, B)$ {Exclude the next child token}
  $p_\omega \leftarrow \mathcal{P}[\omega]$; $\bar{p}_{w,\omega} \leftarrow f_w(p_\omega; G_\omega, \delta_{\text{feat}})$
  $\mathcal{C}_\omega \leftarrow$ set of unique child-token labels at ω
  $C_\omega^{\text{tree}} \leftarrow [p_\omega(\mathcal{C}_\omega) - \bar{p}_{w,\omega}(\mathcal{C}_\omega)]_+$
  $A_\omega^{\text{feat}} \leftarrow \mathbb{I}[C_\omega^{\text{tree}} \le \tau_{\text{tree}}]$; $U_\omega^{\text{feat}} \leftarrow R_\omega A_\omega^{\text{feat}}$
  $p_{w,\omega} \leftarrow f_w(p_\omega; G_\omega, U_\omega^{\text{feat}} \delta_{\text{feat}})$
  Sample $x_t \sim p_{w,\omega}$ and emit x_t {Direct-draw verifier; see "Verifier scope" in Appendix C.1}
  if $x_t \notin \mathcal{C}_\omega$ **then**
    return emitted block and $\mathcal{S}(h;\eta)$
  end if
  $\omega \leftarrow \textsc{MatchingChild}(\omega, x_t)$ {Reuse its cached state; resolve duplicates by a fixed native rule}
end loop

Verifier scope. Algorithm 2 draws a token from the gated target and reuses a matching child when available. At a leaf, $\mathcal{C}_\omega = \varnothing$, so coverage loss is zero and the sampled token ends the traversal. The emitted token follows the gated target conditional on the current runtime state. This statement applies to the direct-draw rule in Algorithm 2; other verifier rules require their own sampling analysis. Appendix D.2 relates the direct-draw rule to proposal reuse. Draft-tree invariance is conditional on the same prefix, configuration, auxiliary state, and coupled construction randomness.

Algorithm 3 Phase II: Text-and-Key ChiselMark Detection (Both Frameworks)

Require: Observed tokens $y_{1:T}$; tested key $\kappa^\star$; public parameters ℓ, ρ, B; reference-key count M; significance level α.
Ensure: Score $S_{\kappa^\star}(y_{1:T})$, calibrated p-value p_{wm}, and decision $\mathbb{I}[p_{\text{wm}} \le \alpha]$.

Sample independent reference keys $\kappa_1, \ldots, \kappa_M$ from the same distribution as the tested key
for all $\kappa \in (\kappa^\star, \kappa_1, \ldots, \kappa_M)$ **do**
  Derive $(\kappa_{\text{site}}, \kappa_{\text{sym}}, \kappa_{\text{tok}})$; $S_\kappa(y_{1:T}) \leftarrow 0$
  for $t = \ell + 1$ **to** T **do**
    $c_t \leftarrow (y_{t-\ell}, \ldots, y_{t-1})$
    $R_t^\kappa \leftarrow \mathbb{I}[\text{PRF}_{\kappa_{\text{site}}}(c_t) < \rho]$
    $b_t^\kappa \leftarrow \text{PRF}_{\kappa_{\text{sym}}}(c_t) \bmod B$; $\hat{b}_t^\kappa \leftarrow \text{PRF}_{\kappa_{\text{tok}}}(c_t, y_t) \bmod B$
    $s_t^\kappa \leftarrow R_t^\kappa(\mathbb{I}[\hat{b}_t^\kappa = b_t^\kappa] - 1/B)$
    $S_\kappa(y_{1:T}) \leftarrow S_\kappa(y_{1:T}) + s_t^\kappa$ {Include vetoed candidates; do not reconstruct A or U}
  end for
end for
$$p_{\text{wm}} \leftarrow \frac{1 + \sum_{j=1}^{M} \mathbb{I}[S_{\kappa_j}(y_{1:T}) \ge S_{\kappa^\star}(y_{1:T})]}{M+1}$$
return $S_{\kappa^\star}(y_{1:T})$, p_{wm}, $\mathbb{I}[p_{\text{wm}} \le \alpha]$

C.2 Random-Key Calibration: Proof and Testing Conventions

The following proof establishes Theorem 4.5 for the score used by Algorithm 3.

Proof. Conditioned on the text, the scores are exchangeable. Randomly breaking ties would make the tested score's descending rank uniform on $\{1, \ldots, M+1\}$. Counting all ties as exceedances can only increase that rank and hence its p-value. Thus the rejection probability is at most $\lfloor \alpha(M+1) \rfloor / (M+1) \le \alpha$. □

If $T \le \ell$, every score is zero and $p_{\mathrm{wm}} = 1$. Otherwise the same fixed tokenizer and scoring configuration are used for all keys. Using $\ge$ in Eq. (13) counts ties conservatively. The minimum attainable value is $1/(M+1)$, so rejection at level $\alpha > 0$ requires $M \ge \lceil 1/\alpha \rceil - 1$. The proof uses exchangeability across keys on the fixed observed text. Searching over text windows, detector settings, or tested keys requires calibrating the complete selection procedure or applying an appropriate multiple-testing correction.

D Evidence–Distortion Analyses and Cost-Aware Allocation

D.1 Local Evidence and Distributional Distortion

Let $\mathcal{V}$ be a finite vocabulary and let p be a probability distribution on $\mathcal{V}$. For a set $G \subseteq \mathcal{V}$ and a finite bias $\delta \ge 0$, define

$$p_\delta(v) = f_w(p; G, \delta)(v) := \frac{p(v)e^{\delta \mathbf{1}_G(v)}}{\sum_{u\in\mathcal{V}} p(u)e^{\delta \mathbf{1}_G(u)}}. \tag{41}$$

Throughout, logarithms are natural, $0\log(0/a) = 0$ for $a \ge 0$, and $D_{\mathrm{KL}}(w\|p) = +\infty$ if $w(v) > 0 = p(v)$ for some v. We use

$$\mathrm{TV}(w,p) := \frac{1}{2}\sum_{v\in\mathcal{V}} |w(v) - p(v)|, \qquad D_{\mathrm{KL}}(w\|p) := \sum_{v:w(v)>0} w(v)\log\frac{w(v)}{p(v)}. \tag{42}$$

For $s \in [0,1]$, write

$$\begin{aligned} Z_s(\delta) &= 1 + (e^\delta - 1)s, \qquad r_s(\delta) = \frac{e^\delta s}{Z_s(\delta)}, \\ e_s(\delta) &= \Delta(s,\delta) := r_s(\delta) - s = \frac{(e^\delta - 1)s(1-s)}{1 + (e^\delta - 1)s} \ge 0. \end{aligned} \tag{43}$$

The quantity $e_s(\delta)$ is the increase in green-match probability at a fixed state. With $s = p_G$, it gives the contribution of an admitted write used in the detector discussion. Sequence-level detection also depends on the contexts reached and the scores at vetoed sites. We therefore evaluate power over all reconstructable candidate sites.

Proposition D.1 (Exact local evidence–distortion identities)**.** *Let $s = p(G)$. For every finite $\delta \ge 0$,*

$$\mathrm{TV}(p_\delta, p) = e_s(\delta), \qquad K(s,\delta) := D_{\mathrm{KL}}(p_\delta\|p) = \delta r_s(\delta) - \log Z_s(\delta). \tag{44}$$

For $0 < s < 1$,

$$K(s,\delta) = d_{\mathrm{Ber}}(r_s(\delta)\|s), \qquad d_{\mathrm{Ber}}(r\|s) := r\log\frac{r}{s} + (1-r)\log\frac{1-r}{1-s}. \tag{45}$$

Moreover,

$$K(s,\delta) \le \frac{\delta^2}{8}, \qquad K(s,\delta) \le -\log s \quad \text{if } s > 0. \tag{46}$$

At $s = 0$ or $s = 1$, $p_\delta = p$ and both $e_s(\delta)$ and $K(s,\delta)$ vanish.

To prove Proposition D.1, we first record the signed-mass identity and the derivatives of the scalar normalizer.

Lemma D.2 (Signed-mass representation)**.** *For probability distributions a, b on a finite set, let $d(v) = a(v) - b(v)$ and $A_+ = \{v : d(v) > 0\}$. Then*

$$\mathrm{TV}(a,b) = \sum_v [d(v)]_+ = \sum_v [-d(v)]_+ = a(A_+) - b(A_+) = \sup_A |a(A) - b(A)|. \tag{47}$$

In particular, if $a(v) \ge b(v)$ on G and $a(v) \le b(v)$ on G^c, then $\mathrm{TV}(a,b) = a(G) - b(G)$.

Proof. Since $d = [d]_+ - [-d]_+$ and $\sum_v d(v) = 0$, we have $\sum_v [d(v)]_+ = \sum_v [-d(v)]_+$. Consequently,

$$\frac{1}{2}\sum_v |d(v)| = \frac{1}{2}\left(\sum_v [d(v)]_+ + \sum_v [-d(v)]_+\right) = \sum_v [d(v)]_+. \tag{48}$$

For every $A \subseteq \mathcal{V}$,

$$-\sum_v [-d(v)]_+ \le \sum_{v\in A} d(v) \le \sum_v [d(v)]_+,$$

and the upper bound is attained at A_+. This proves equation (47). Under the additional sign condition, all positive contributions lie in G and all negative contributions lie in G^c, so their sum on G equals the total positive mass. □

Lemma D.3 (Normalizer derivatives)**.** *Let $0 < s < 1$ and define $A_s(u) = \log(1 - s + se^u)$. For all finite u,*

$$A_s'(u) = r_s(u), \qquad A_s''(u) = r_s(u)(1 - r_s(u)), \qquad 0 < A_s''(u) \le \frac{1}{4}. \tag{49}$$

For $\delta \ge 0$,

$$\delta A_s'(\delta) - A_s(\delta) = \int_0^\delta u A_s''(u)\, du. \tag{50}$$

Proof. Differentiating the explicit scalar function, we obtain

$$\begin{aligned} A_s'(u) &= \frac{se^u}{1 - s + se^u}, \\ A_s''(u) &= \frac{se^u(1 - s + se^u) - s^2e^{2u}}{(1 - s + se^u)^2} \\ &= \frac{se^u}{1 - s + se^u}\frac{1 - s}{1 - s + se^u} = r_s(u)(1 - r_s(u)). \end{aligned} \tag{51}$$

The upper bound follows from $x(1 - x) = 1/4 - (x - 1/2)^2 \le 1/4$. Finally, if $F_s(u) = uA_s'(u) - A_s(u)$, then $F_s(0) = 0$ and

$$F_s'(u) = A_s'(u) + uA_s''(u) - A_s'(u) = uA_s''(u).$$

Integrating from 0 to δ proves equation (50). □

Proof of Proposition D.1. First suppose $0 < s < 1$. Splitting the normalization sum into its two buckets gives

$$\sum_v p(v)e^{\delta \mathbf{1}_G(v)} = e^\delta \sum_{v\in G} p(v) + \sum_{v\notin G} p(v) = e^\delta s + 1 - s = Z_s(\delta). \tag{52}$$

Thus $p_\delta(G) = r_s(\delta)$. Since $1 \le Z_s(\delta) \le e^\delta$, the ratios $e^\delta/Z_s(\delta)$ on G and $1/Z_s(\delta)$ on G^c are respectively at least and at most one. Lemma D.2 therefore yields

$$\mathrm{TV}(p_\delta, p) \overset{(a)}{=} \sum_{v\in G}(p_\delta(v) - p(v)) \overset{(b)}{=} \frac{e^\delta s}{Z_s(\delta)} - s = e_s(\delta), \tag{53}$$

where (a) uses the just-established signs and (b) uses equation (52).

Finite tilting preserves the support of p. On that support, $\log(p_\delta(v)/p(v)) = \delta \mathbf{1}_G(v) - \log Z_s(\delta)$. Hence

$$\begin{aligned} D_{\mathrm{KL}}(p_\delta \| p) &= \sum_{v:p(v)>0} p_\delta(v)\big(\delta \mathbf{1}_G(v) - \log Z_s(\delta)\big) \\ &= \delta \sum_{v\in G} p_\delta(v) - \log Z_s(\delta) \sum_v p_\delta(v) \\ &= \delta r_s(\delta) - \log Z_s(\delta). \end{aligned} \tag{54}$$

Using $r_s/s = e^\delta/Z_s$ and $(1-r_s)/(1-s) = 1/Z_s$, we also have

$$\begin{aligned} d_{\mathrm{Ber}}(r_s\|s) &= r_s \log \frac{e^\delta}{Z_s} + (1-r_s)\log\frac{1}{Z_s} \\ &= r_s\delta - r_s \log Z_s - (1-r_s)\log Z_s = \delta r_s - \log Z_s. \end{aligned} \tag{55}$$

This proves equation (44) and equation (45).

By equation (54) and Lemma D.3,

$$K(s,\delta) \overset{(a)}{=} \delta A_s'(\delta) - A_s(\delta) \overset{(b)}{=} \int_0^\delta u r_s(u)(1-r_s(u))\,du \overset{(c)}{\le} \int_0^\delta \frac{u}{4}\,du = \frac{\delta^2}{8}, \tag{56}$$

where (a) uses $A_s' = r_s$, (b) uses equation (50), and (c) uses $u \ge 0$ and equation (49). The same integral shows that $K(s,\delta)$ is nondecreasing in δ. To compute its upper limit, write

$$\begin{aligned} K(s,\delta) &= \delta(r_s(\delta)-1) - \log\bigl(s + (1-s)e^{-\delta}\bigr), \\ 0 \le \delta(1-r_s(\delta)) &= \frac{\delta(1-s)e^{-\delta}}{s+(1-s)e^{-\delta}} \le \frac{1-s}{s}\delta e^{-\delta} \longrightarrow 0. \end{aligned} \tag{57}$$

Therefore $K(s,\delta) \uparrow -\log s$, which proves the second inequality in equation (46). If $s = 0$, every supported token is red and $Z_s = 1$. If $s = 1$, every supported token is green and $Z_s = e^\delta$. Substitution into equation (41) gives $p_\delta = p$ in both cases, completing the proof. □

Theorem D.4 (Minimum KL for a prescribed local evidence gain)**.** *Fix p and G with $s = p(G) \in (0,1)$, and let $r \in (s,1)$. The problem*

$$\inf\{D_{\mathrm{KL}}(w\|p) : w \text{ is a distribution on } \mathcal{V},\ w(G) \ge r\} \tag{58}$$

has the unique minimizer $w_* = p_{\delta_*}$, where

$$\delta_* = \mathrm{logit}(r) - \mathrm{logit}(s), \qquad \mathrm{logit}(x) = \log\frac{x}{1-x}. \tag{59}$$

Its optimal value is $d_{\mathrm{Ber}}(r\|s)$.

For this theorem, we need an explicit decomposition of KL into a bucket term and two conditional terms.

Lemma D.5 (KL decomposition over a partition)**.** *Let $s = p(G) \in (0,1)$ and $w \ll p$. Put $t = w(G)$, $p_G(v) = p(v)/s$ on G, and $p_R(v) = p(v)/(1-s)$ on G^c. For $0 < t < 1$, define w_G, w_R similarly. Then*

$$D_{\mathrm{KL}}(w\|p) = d_{\mathrm{Ber}}(t\|s) + tD_{\mathrm{KL}}(w_G\|p_G) + (1-t)D_{\mathrm{KL}}(w_R\|p_R). \tag{60}$$

At $t = 0$ or $t = 1$, omit the term for the zero-mass bucket. The conditional KL terms are nonnegative; each term with positive weight vanishes precisely when its two conditional distributions agree.

Proof. For $0 < t < 1$, write $w(v) = tw_G(v)$ and $p(v) = sp_G(v)$ on G. The contribution of that bucket is

$$\begin{aligned} \sum_{v\in G: w(v)>0} w(v)\log\frac{w(v)}{p(v)} &= \sum_{v\in G: w(v)>0} tw_G(v)\left(\log\frac{t}{s} + \log\frac{w_G(v)}{p_G(v)}\right) \\ &= t\log\frac{t}{s} + tD_{\mathrm{KL}}(w_G\|p_G). \end{aligned} \tag{61}$$

The analogous red-bucket contribution is $(1-t)\log((1-t)/(1-s)) + (1-t)D_{\mathrm{KL}}(w_R\|p_R)$. Adding them proves equation (60). A zero-mass bucket contributes zero to the original sum, giving the endpoint convention.

To justify the nonnegativity and equality statements, for any distributions $a \ll b$ let $S = \mathrm{supp}(a)$. The inequality $\log x \le x - 1$ gives

$$-D_{\mathrm{KL}}(a\|b) = \sum_{v\in S} a(v)\log\frac{b(v)}{a(v)} \le \sum_{v\in S} a(v)\left(\frac{b(v)}{a(v)} - 1\right) = b(S) - 1 \le 0. \tag{62}$$

Equality forces $b(v)/a(v) = 1$ for all $v \in S$ and $b(S) = 1$, hence $a = b$. The converse is immediate. □

Proof of Theorem D.4. For any feasible w with finite KL, let $t = w(G) \geq r$. By Lemma D.5,

$$D_{\mathrm{KL}}(w\|p) \geq d_{\mathrm{Ber}}(t\|s). \tag{63}$$

For $s < t < 1$, differentiation in the first argument gives

$$\begin{aligned}\frac{\partial}{\partial t} d_{\mathrm{Ber}}(t\|s) &= \left(\log\frac{t}{s} + 1\right) + \left(-\log\frac{1-t}{1-s} - 1\right) \\ &= \log\frac{t(1-s)}{s(1-t)} > 0.\end{aligned} \tag{64}$$

Since $r > s$, this strict monotonicity, extended to $t = 1$ by continuity, implies

$$d_{\mathrm{Ber}}(t\|s) \geq d_{\mathrm{Ber}}(r\|s), \quad \text{with equality if and only if } t = r. \tag{65}$$

Combining equation (63)–equation (65) establishes the lower bound on equation (58).

Equality requires $t = r$, $w_G = p_G$, and $w_R = p_R$, because r and $1 - r$ are both positive. These conditions determine exactly one distribution:

$$w_*(v) = \begin{cases} \dfrac{r}{s} p(v), & v \in G, \\ \dfrac{1-r}{1-s} p(v), & v \notin G. \end{cases} \tag{66}$$

This distribution is feasible. To identify it with the claimed tilt, observe that

$$e^{\delta_*} = \frac{r(1-s)}{s(1-r)}, \qquad Z_s(\delta_*) = 1 - s + s\frac{r(1-s)}{s(1-r)} = \frac{1-s}{1-r}. \tag{67}$$

It follows that $e^{\delta_*}/Z_s(\delta_*) = r/s$ and $1/Z_s(\delta_*) = (1-r)/(1-s)$. Substitution into equation (41) yields equation (66). Distributions with infinite KL cannot improve its finite value.

One may equivalently certify the optimum by the exact identity

$$\begin{aligned} D_{\mathrm{KL}}(w\|p) - d_{\mathrm{Ber}}(r\|s) &\overset{(a)}{=} D_{\mathrm{KL}}(w\|w_*) + \delta_* w(G) - \log Z_s(\delta_*) - d_{\mathrm{Ber}}(r\|s) \\ &\overset{(b)}{=} D_{\mathrm{KL}}(w\|w_*) + \delta_*[w(G) - r] \\ &\overset{(c)}{\geq} 0, \end{aligned} \tag{68}$$

where (a) splits $\log(w/p)$ as $\log(w/w_*) + \log(w_*/p)$, (b) uses Proposition D.1 at $w_*(G) = r$, and (c) uses $\delta_* > 0$, feasibility, and KL nonnegativity. Equality again forces $w = w_*$. □

Corollary D.6 (TV optimality and the largest local gain)**.** *Under the constraint in Theorem D.4, the minimum TV distance is $r - s$ and is attained by p_{δ_*}. For fixed $\delta > 0$,*

$$\max_{s\in[0,1]} e_s(\delta) = \tanh(\delta/4), \qquad \arg\max_{s\in[0,1]} e_s(\delta) = \left\{\frac{1}{1+e^{\delta/2}}\right\}. \tag{69}$$

Proof. Every feasible w satisfies $\mathrm{TV}(w, p) \geq w(G) - p(G) \geq r - s$ by Lemma D.2. Proposition D.1 gives equality for p_{δ_*}. For the maximization over s, let $\theta = e^{\delta} - 1 > 0$. Expanding equation (43),

$$\begin{aligned} e_s(\delta) &= \frac{(1+\theta)s - s(1+\theta s)}{1+\theta s} = \frac{\theta s(1-s)}{1+\theta s}, \\ \partial_s e_s(\delta) &= \frac{\theta(1-2s)(1+\theta s) - \theta^2 s(1-s)}{(1+\theta s)^2} = \frac{\theta(1-2s-\theta s^2)}{(1+\theta s)^2}. \end{aligned} \tag{70}$$

The polynomial $1 - 2s - \theta s^2$ is strictly decreasing on $[0, 1]$, with opposite signs at the two endpoints. Its unique zero is

$$s_* = \frac{\sqrt{1+\theta} - 1}{\theta} = \frac{1}{1+e^{\delta/2}}.$$

Writing $k = e^{\delta/2}$ gives $1 + \theta s_* = k$ and

$$e_{s_*}(\delta) = \frac{(k^2-1)\,k/(1+k)^2}{k} = \frac{k-1}{k+1} = \tanh(\delta/4).$$

The derivative changes from positive to negative at s_*, proving uniqueness. □

D.2 Architecture-specific Local Guarantees

For a target a and proposal b, set $\beta(a,b) = \sum_v \min\{a(v), b(v)\}$. From $2\min\{x,y\} = x + y - |x-y|$,

$$\beta(a,b) = \frac{1}{2}\sum_v (a(v) + b(v) - |a(v) - b(v)|) = 1 - \mathrm{TV}(a,b). \tag{71}$$

Proposition D.7 (Same-prefix token-level acceptance bound)**.** *Fix a prefix, key, and a shared write indicator $U \in \{0,1\}$ before proposal sampling. Let $p_w = f_w(p; G, U\delta)$, $q_w = f_w(q; G, U\delta)$, $e_p = e_{p(G)}(\delta)$, and $e_q = e_{q(G)}(\delta)$. Then*

$$|\beta(p_w, q_w) - \beta(p,q)| \le U(e_p + e_q). \tag{72}$$

Proof. For each token, the scalar inequality $\big||x| - |y|\big| \le |x-y|$ gives

$$\begin{aligned}
|\beta(p_w,q_w) - \beta(p,q)| &\overset{(a)}{=} \frac{1}{2}\left|\sum_v \big(|p_w(v) - q_w(v)| - |p(v) - q(v)|\big)\right| \\
&\overset{(b)}{\le} \frac{1}{2}\sum_v \big||p_w(v) - q_w(v)| - |p(v) - q(v)|\big| \\
&\overset{(c)}{\le} \frac{1}{2}\sum_v |(p_w(v) - p(v)) - (q_w(v) - q(v))| \\
&\overset{(d)}{\le} \frac{1}{2}\sum_v |p_w(v) - p(v)| + \frac{1}{2}\sum_v |q_w(v) - q(v)| \\
&\overset{(e)}{=} U(e_p + e_q).
\end{aligned} \tag{73}$$

Here (a) uses equation (71); (b) and (d) are scalar triangle inequalities; (c) is the displayed reverse triangle inequality; and (e) uses Proposition D.1 if $U = 1$, while both differences vanish if $U = 0$. □

Corollary D.8 (Draft Green-Mass Gate)**.** *Suppose an active state satisfies $-\log q(G) \le \tau_Q < \infty$, where $\tau_Q \ge 0$. Then*

$$D_{\mathrm{KL}}(q_\delta \| q) \le \tau_Q, \qquad \mathrm{TV}(q_\delta, q) \le 1 - e^{-\tau_Q}. \tag{74}$$

If, in addition, $\mathrm{TV}(p,q) \le \eta < e^{-\tau_Q}$ at every active state, then at those states

$$K(p(G), \delta) \le \min\left\{\frac{\delta^2}{8}, -\log(e^{-\tau_Q} - \eta)\right\}, \tag{75}$$

and at both active and inactive states

$$|\beta(p_w, q_w) - \beta(p,q)| \le U\big(2(1 - e^{-\tau_Q}) + \eta\big). \tag{76}$$

Proof. Put $a = e^{-\tau_Q} > 0$. Exponentiating the gate condition gives $q(G) \ge a$. Proposition D.1 and $r_{q(G)}(\delta) \le 1$ imply

$$\begin{aligned}
D_{\mathrm{KL}}(q_\delta \| q) &\le -\log q(G) \le -\log a = \tau_Q, \\
\mathrm{TV}(q_\delta, q) &= r_{q(G)}(\delta) - q(G) \le 1 - q(G) \le 1 - a.
\end{aligned} \tag{77}$$

Under the additional assumption,

$$p(G) \overset{(a)}{=} q(G) + (p(G) - q(G)) \overset{(b)}{\ge} q(G) - \mathrm{TV}(p,q) \overset{(c)}{\ge} a - \eta > 0, \tag{78}$$

where (b) is Lemma D.2 for the event G, and (c) uses both active-state assumptions. Since $-\log x$ decreases in $x > 0$, the claimed KL bound follows from equation (46). Moreover,

$$\begin{aligned}
e_p + e_q &\le (1 - p(G)) + (1 - q(G)) \\
&\le (1 - q(G) + \eta) + (1 - q(G)) \le 2(1 - a) + \eta.
\end{aligned} \tag{79}$$

Multiply by U and apply Proposition D.7. When $U = 0$, the target and proposal are unchanged, so no active-state assumption is needed. □

Corollary D.9 (Feature-level coverage gate)**.** *Condition on a fixed verification state and its set of unique child-token labels $\mathcal{C}$. Let $s = p(G)$, $c = p(\mathcal{C})$, $j = p(\mathcal{C} \cap G)$, and $\theta = e^{\delta} - 1$. Then*

$$p_\delta(\mathcal{C}) = \frac{c + \theta j}{1 + \theta s}, \qquad C^{\text{tree}} := [p(\mathcal{C}) - p_\delta(\mathcal{C})]_+ = \frac{\theta[sc - j]_+}{1 + \theta s}. \tag{80}$$

If $U = R\mathbf{1}\{C^{\text{tree}} \le \tau_{\text{tree}}\}$ with $R \in \{0, 1\}$ and $p_w = f_w(p; G, U\delta)$, then

$$[p(\mathcal{C}) - p_w(\mathcal{C})]_+ = UC^{\text{tree}} \le U\tau_{\text{tree}}, \qquad C^{\text{tree}} \le e_s(\delta). \tag{81}$$

Proof. Since $\mathcal{C} = (\mathcal{C} \cap G) \dot\cup (\mathcal{C} \cap G^c)$,

$$\begin{aligned} p_\delta(\mathcal{C}) &= \sum_{v \in \mathcal{C} \cap G} \frac{e^\delta p(v)}{1 + \theta s} + \sum_{v \in \mathcal{C} \cap G^c} \frac{p(v)}{1 + \theta s} \\ &= \frac{e^\delta j + (c - j)}{1 + \theta s} = \frac{c + \theta j}{1 + \theta s}, \\ p(\mathcal{C}) - p_\delta(\mathcal{C}) &= \frac{c(1 + \theta s) - c - \theta j}{1 + \theta s} = \frac{\theta(sc - j)}{1 + \theta s}. \end{aligned} \tag{82}$$

The factor $\theta/(1 + \theta s)$ is nonnegative, so it can be pulled outside the positive part. Because U is binary, $p_w = (1 - U)p + Up_\delta$, and consequently

$$[p(\mathcal{C}) - p_w(\mathcal{C})]_+ = [U(p(\mathcal{C}) - p_\delta(\mathcal{C}))]_+ = UC^{\text{tree}} \le U\tau_{\text{tree}}. \tag{83}$$

The last inequality holds because $U = 1$ implies the gate condition, and both sides are zero if $U = 0$. Finally,

$$C^{\text{tree}} \le |p(\mathcal{C}) - p_\delta(\mathcal{C})| \overset{\text{(a)}}{\le} \text{TV}(p, p_\delta) \overset{\text{(b)}}{=} e_s(\delta), \tag{84}$$

where (a) is Lemma D.2 and (b) is Proposition D.1. □

For the verifier that draws $X \sim p_w$ and reuses a matching child whenever $X \in \mathcal{C}$, the conditional reuse probability is $p_w(\mathcal{C})$. The corollary therefore bounds its loss at the fixed verification state. Draft-tree invariance follows from using the same unmodified construction rule with the same prefix, auxiliary state, and coupled randomness. Different emitted prefixes may lead to different subsequent trees.

D.3 Sequence Distortion under the Gated Token-Level Target

Theorem 4.1 follows from the one-step rejection identity in Lemma D.11 and the emitted-prefix kernel argument in Lemma D.12 below. We give these arguments once, together with the sequence-distortion proof, so that rejection, bonus-token, and prefix-determinism conditions remain explicit.

Fix the prompt, key κ, finite output horizon T, and decoding configuration. At time t, let $H_t = X_{<t} \in \mathcal{V}^{t-1}$ be the emitted prefix. For every prefix h, assume that $p_t(\cdot \mid h)$, $q_t(\cdot \mid h)$, $G_t(h)$, and $U_t(h) \in \{0, 1\}$ are deterministic functions of h and the fixed quantities. The draft green-mass gate in the main text has this property. Define

$$\begin{aligned} p_{w,t}(v \mid h) &= f_w(p_t(\cdot \mid h); G_t(h), U_t(h)\delta)(v), \\ q_{w,t}(v \mid h) &= f_w(q_t(\cdot \mid h); G_t(h), U_t(h)\delta)(v), \\ s_t(h) &= p_t(G_t(h) \mid h), \\ P(x_{1:T}) = \prod_{t=1}^{T} p_t(x_t \mid x_{<t}), &\qquad P^w(x_{1:T}) = \prod_{t=1}^{T} p_{w,t}(x_t \mid x_{<t}). \end{aligned} \tag{85}$$

At an active proposal site, both the proposal and target use the same cached $U_t(h)$ and $G_t(h)$. On rejection, the correction distribution is proportional to $[p_{w,t} - q_{w,t}]_+$, and later draft tokens are discarded. A bonus token is drawn directly from $p_{w,t}$ at its own prefix, evaluating q_t there if the gate requires it. We assume the standard fixed-block speculative sampler: the acceptance decision at a position does not inspect later draft tokens. These are the exact-sampling rules underlying the main-text construction.

Theorem D.10 (Write-dependent sequence KL bound)**.** *Under the preceding construction, the emitted law is P^w. For $N_{\mathrm{write}} := \sum_{t=1}^{T} U_t(H_t)$, one has*

$$D_{\mathrm{KL}}(P^w \| P) = \sum_{t=1}^{T} \mathbb{E}_{P^w}[U_t(H_t) K(s_t(H_t), \delta)] \le \frac{\delta^2}{8} \mathbb{E}_{P^w} N_{\mathrm{write}}. \tag{86}$$

If $s_t(h) \ge \varepsilon$ for some $\varepsilon \in (0, 1]$ at every active prefix with $P^w(H_t = h) > 0$, then

$$D_{\mathrm{KL}}(P^w \| P) \le \min\left\{ \frac{\delta^2}{8}, \log \frac{1}{\varepsilon} \right\} \mathbb{E}_{P^w} N_{\mathrm{write}}. \tag{87}$$

We prove the theorem in three stages: the one-step rejection identity, the induced autoregressive law, and a finite-sum decomposition of its KL divergence.

Lemma D.11 (One-step rejection identity)**.** *Let a, b be target and proposal distributions on $\mathcal{V}$. Sample $Y \sim b$ and accept Y with probability $\alpha(Y) = \min\{1, a(Y)/b(Y)\}$. On rejection, draw Z from the distribution proportional to $[a - b]_+$. The resulting token X has distribution a.*

Proof. For $b(v) > 0$, the accepted mass at token v is

$$\Pr(Y = v, \text{accept}) = b(v) \min\left\{1, \frac{a(v)}{b(v)}\right\} = \min\{a(v), b(v)\}. \tag{88}$$

For $b(v) = 0$, the accepted mass is zero, which is still the right-hand side of equation (88). No likelihood ratio is needed at such a token. The total rejection probability is

$$\begin{aligned} R &= 1 - \sum_v \min\{a(v), b(v)\} \\ &= \sum_v \big(a(v) - \min\{a(v), b(v)\}\big) = \sum_v [a(v) - b(v)]_+. \end{aligned} \tag{89}$$

If $R > 0$, put $c(v) = [a(v) - b(v)]_+ / R$. Taking $Z \sim c$ conditional on rejection gives

$$\begin{aligned} \Pr(X = v) &\overset{(a)}{=} \Pr(Y = v, \text{accept}) + \Pr(\text{reject}) \Pr(Z = v \mid \text{reject}) \\ &\overset{(b)}{=} \min\{a(v), b(v)\} + R \frac{[a(v) - b(v)]_+}{R} \\ &\overset{(c)}{=} a(v), \end{aligned} \tag{90}$$

where (a) partitions according to acceptance and rejection, (b) uses equation (88)–equation (89), and (c) is $\min\{x, y\} + [x - y]_+ = x$. If $R = 0$, all differences $a(v) - b(v)$ are nonpositive and sum to zero, so $a = b$. The proposal is then always accepted, again giving $X \sim a$. □

Lemma D.12 (From block sampling to an emitted-prefix kernel)**.** *For every t and every reached prefix h, the construction above satisfies*

$$\Pr(X_t = v \mid H_t = h) = p_{w,t}(v \mid h). \tag{91}$$

Consequently its output law is the product P^w in equation (85).

Proof. A draft block can be sampled using fresh independent random variables $\zeta_1, \ldots, \zeta_{L_d}$, where ζ_j produces its jth token from the autoregressive draft conditional at the draft prefix. Acceptance coins are independent of these variables. An acceptance test at position j uses that token, its prefix, and its own acceptance coin; it does not use $\zeta_{j+1}, \ldots, \zeta_{L_d}$.

Therefore one may defer sampling each unused draft suffix until its first token is needed. To see why the deferral preserves the emitted law, fix a draft prefix $y_{<j}$. Conditional on this prefix, the event that the preceding proposals were accepted depends only on earlier acceptance coins and earlier tokens. It does not change the independent variable ζ_j, so

$$\Pr(Y_j = v \mid Y_{<j} = y_{<j}, \text{preceding proposals accepted}) = q_{w,t}(v \mid h), \tag{92}$$

where h is the emitted prefix consisting of the block's initial prefix followed by $y_{<j}$. If a rejection occurs, the unused variables only determine a discarded suffix and hence cannot affect the emitted continuation.

In this equivalent deferred construction, let J_t denote the runtime information immediately before producing the next emitted token, including the block position and past random choices, but excluding the random choices for the next proposal, its acceptance, or its correction. For any (h, j) with positive probability, there are two cases. In a proposal case, equation (92) identifies the proposal with $q_{w,t}(\cdot \mid h)$; its target is $p_{w,t}(\cdot \mid h)$ because the gate is determined by h. Lemma D.11 implies

$$\Pr(X_t = v \mid H_t = h, J_t = j) = p_{w,t}(v \mid h).$$

In a bonus case, the same identity holds by direct sampling from that target. The right-hand side is independent of j. Averaging the conditional identity over J_t gives

$$\begin{aligned}\Pr(X_t = v \mid H_t = h) &= \mathbb{E}\big[\Pr(X_t = v \mid H_t, J_t) \mid H_t = h\big] \\ &= \mathbb{E}[p_{w,t}(v \mid h) \mid H_t = h] = p_{w,t}(v \mid h). \end{aligned} \tag{93}$$

Repeated conditional factorization now yields

$$\Pr(X_{1:T} = x_{1:T}) = \prod_{t=1}^{T} \Pr(X_t = x_t \mid X_{<t} = x_{<t}) = \prod_{t=1}^{T} p_{w,t}(x_t \mid x_{<t}). \tag{94}$$

□

Lemma D.13 (Finite-horizon KL decomposition)**.** *Let $a_t(\cdot \mid h)$ and $b_t(\cdot \mid h)$ be conditional distributions on a finite alphabet, and let $A = \prod_t a_t$, $B = \prod_t b_t$ be their product laws. Assume that a_t and b_t have the same support at every A-reachable prefix, and that $|\log(a_t(v \mid h)/b_t(v \mid h))| \le M$ on that support for a finite M. Define $\rho_t^A(h) = A(H_t = h)$. Then*

$$D_{\mathrm{KL}}(A\|B) = \sum_{t=1}^{T} \sum_{h:\rho_t^A(h)>0} \rho_t^A(h) D_{\mathrm{KL}}(a_t(\cdot \mid h)\|b_t(\cdot \mid h)). \tag{95}$$

Proof. Write $\ell_t(h, v) = \log(a_t(v \mid h)/b_t(v \mid h))$ on the common support. An A-positive sequence has positive b_t at each of its prefixes and hence positive B-mass. On such a sequence,

$$\log \frac{A(x_{1:T})}{B(x_{1:T})} = \log \prod_{t=1}^{T} \frac{a_t(x_t \mid x_{<t})}{b_t(x_t \mid x_{<t})} = \sum_{t=1}^{T} \ell_t(x_{<t}, x_t). \tag{96}$$

All sums are finite, and the summands are bounded by M in absolute value. Grouping first by t, then by the prefix h and token v, gives

$$\begin{aligned} D_{\mathrm{KL}}(A\|B) &= \sum_{x:A(x)>0} A(x) \sum_{t=1}^{T} \ell_t(x_{<t}, x_t) \\ &\overset{(a)}{=} \sum_{t=1}^{T} \sum_{h:\rho_t^A(h)>0} \sum_{v:a_t(v|h)>0} \ell_t(h, v) \sum_{z \in \mathcal{V}^{T-t}} A(h, v, z) \\ &\overset{(b)}{=} \sum_{t=1}^{T} \sum_{h:\rho_t^A(h)>0} \rho_t^A(h) \sum_{v:a_t(v|h)>0} a_t(v \mid h) \ell_t(h, v) \\ &\overset{(c)}{=} \sum_{t=1}^{T} \sum_{h:\rho_t^A(h)>0} \rho_t^A(h) D_{\mathrm{KL}}(a_t(\cdot \mid h)\|b_t(\cdot \mid h)). \end{aligned} \tag{97}$$

Here (a) is a regrouping of the finite sequence sum; (b) uses

$$\sum_{z} A(h, v, z) = \rho_t^A(h) a_t(v \mid h) \underbrace{\sum_{z} \prod_{j=t+1}^{T} a_j(z_j \mid h, v, z_{t+1:j-1})}_{=1}; \tag{98}$$

the underbraced sum equals one by summing its last coordinate first and successively using normalization of each conditional distribution. Step (c) is the definition of conditional KL. □

Proof of Theorem D.10. Lemma D.12 identifies the emitted law. To apply Lemma D.13, fix a reached prefix h. The multiplying factors in equation (41) are strictly positive, so

$$\operatorname{supp}(p_{w,t}(\cdot \mid h)) = \operatorname{supp}(p_t(\cdot \mid h)). \tag{99}$$

For a supported v, the local log likelihood ratio is

$$\log \frac{p_{w,t}(v \mid h)}{p_t(v \mid h)} = U_t(h)\delta \mathbf{1}_{G_t(h)}(v) - \log Z_{s_t(h)}(U_t(h)\delta). \tag{100}$$

Because $0 \le \log Z_s(u) \le u$ for $u \ge 0$, its absolute value is at most $U_t(h)\delta \le \delta$. Thus the lemma applies with $M = \delta$.

Let $\rho_t^w(h) = P^w(H_t = h)$. It is determined recursively by

$$\rho_1^w(\varnothing) = 1, \qquad \rho_{t+1}^w(hv) = \rho_t^w(h) p_{w,t}(v \mid h). \tag{101}$$

Here the expectation uses the contexts reached under the actual gated sequence law. We obtain

$$\begin{aligned}
D_{\mathrm{KL}}(P^w \| P) &\overset{(a)}{=} \sum_{t=1}^{T} \sum_{h} \rho_t^w(h) D_{\mathrm{KL}}(p_{w,t}(\cdot \mid h) \| p_t(\cdot \mid h)) \\
&\overset{(b)}{=} \sum_{t=1}^{T} \sum_{h} \rho_t^w(h) U_t(h) K(s_t(h), \delta) \\
&\overset{(c)}{\le} \frac{\delta^2}{8} \sum_{t=1}^{T} \sum_{h} \rho_t^w(h) U_t(h) \\
&\overset{(d)}{=} \frac{\delta^2}{8} \sum_{t=1}^{T} \mathbb{E}_{P^w} U_t(H_t) \\
&\overset{(e)}{=} \frac{\delta^2}{8} \mathbb{E}_{P^w} \sum_{t=1}^{T} U_t(H_t).
\end{aligned} \tag{102}$$

The sums over h may be restricted to positive-occupancy prefixes. Step (a) is Lemma D.13. In (b), an inactive kernel equals p_t and contributes zero, while an active kernel contributes $K(s_t(h), \delta)$ by Proposition D.1. Step (c) uses its uniform KL bound and the nonnegativity of $\rho_t^w(h)U_t(h)$. Step (d) is the definition of expectation under equation (101); (e) is linearity of a finite sum. This proves both the equality and inequality in equation (86).

If every positive-occupancy active prefix satisfies $s_t(h) \ge \varepsilon$, then at such prefixes

$$K(s_t(h), \delta) \le \min\{\delta^2/8, -\log s_t(h)\} \le \min\{\delta^2/8, \log(1/\varepsilon)\}. \tag{103}$$

Replacing step (c) by this bound proves equation (87). □

Lemma D.14 (Bounded expectations and binary Pinsker inequality)**.** *For distributions μ, ν on a finite set and f taking values in $[0,1]$,*

$$|\mathbb{E}_\mu f - \mathbb{E}_\nu f| \le \mathrm{TV}(\mu, \nu) \le \min\{1, \sqrt{D_{\mathrm{KL}}(\mu \| \nu)/2}\}. \tag{104}$$

Proof. Let $d(x) = \mu(x) - \nu(x)$. Since $0 \le f \le 1$,

$$-\sum_x [-d(x)]_+ \le \sum_x f(x) d(x) \le \sum_x [d(x)]_+.$$

Lemma D.2 proves the first inequality. To prove the second, let $A = \{x : d(x) > 0\}$, $a = \mu(A)$, and $b = \nu(A)$, so $a - b = \mathrm{TV}(\mu, \nu)$. If this distance is zero there is nothing to prove. If

$D_{\mathrm{KL}}(\mu\|\nu) = +\infty$, the square-root bound is vacuous. Otherwise, a positive distance implies $0 < b < 1$, and Lemma D.5, applied to the event A, gives

$$D_{\mathrm{KL}}(\mu\|\nu) \geq d_{\mathrm{Ber}}(a\|b). \tag{105}$$

For fixed $b \in (0,1)$, put $g_b(x) = d_{\mathrm{Ber}}(x\|b) - 2(x-b)^2$. Direct differentiation yields

$$g_b(b) = g_b'(b) = 0, \qquad g_b''(x) = \frac{1}{x(1-x)} - 4 \geq 0 \quad (0 < x < 1). \tag{106}$$

For $x > b$, integrate g_b'' from b to x twice to get $g_b(x) \geq 0$; for $x < b$, the same conclusion follows by integrating in the reverse direction. Continuity gives the endpoint cases. Therefore

$$D_{\mathrm{KL}}(\mu\|\nu) \geq d_{\mathrm{Ber}}(a\|b) \geq 2(a-b)^2 = 2\mathrm{TV}(\mu,\nu)^2. \tag{107}$$

Taking square roots and using $\mathrm{TV}(\mu,\nu) \leq 1$ proves equation (104). □

Corollary D.15 (Separating local overlap change from context drift)**.** *At a common output prefix, define*

$$\beta_{0,t}(h) = \beta(p_t(\cdot \mid h), q_t(\cdot \mid h)), \qquad \beta_{w,t}(h) = \beta(p_{w,t}(\cdot \mid h), q_{w,t}(\cdot \mid h)),$$

and $K_{<t} = D_{\mathrm{KL}}(P^w_{<t}\|P_{<t})$*. Write* $e_{p,t}(h) = e_{s_t(h)}(\delta)$ *and* $e_{q,t}(h) = e_{q_t(G_t(h)|h)}(\delta)$*. Then*

$$\begin{aligned} &|\mathbb{E}_{P^w}\beta_{w,t}(H_t) - \mathbb{E}_P\beta_{0,t}(H_t)| \\ &\quad \leq \mathbb{E}_{P^w}\left[U_t(H_t)(e_{p,t}(H_t) + e_{q,t}(H_t))\right] + \min\{1, \sqrt{K_{<t}/2}\}. \end{aligned} \tag{108}$$

Proof. Let $\rho_t^0(h) = P(H_t = h)$ and $b(h) = \beta_{0,t}(h) \in [0,1]$. The exact algebraic decomposition is

$$\begin{aligned} &\sum_h \rho_t^w(h)\beta_{w,t}(h) - \sum_h \rho_t^0(h)b(h) \\ &\quad = \sum_h \rho_t^w(h)(\beta_{w,t}(h) - b(h)) + \sum_h (\rho_t^w(h) - \rho_t^0(h))b(h). \end{aligned} \tag{109}$$

Taking the absolute value of the first term and applying Proposition D.7 gives

$$\left|\sum_h \rho_t^w(h)(\beta_{w,t}(h) - b(h))\right| \leq \sum_h \rho_t^w(h)U_t(h)(e_{p,t}(h) + e_{q,t}(h)). \tag{110}$$

By Lemma D.14, the absolute value of the second term is bounded by $\mathrm{TV}(\rho_t^w, \rho_t^0) \leq \min\{1, \sqrt{K_{<t}/2}\}$. Adding the bounds proves equation (108). Theorem D.10, truncated at $t-1$, also gives the explicit estimate

$$\sqrt{K_{<t}/2} \leq \frac{\delta}{4}\left(\mathbb{E}_{P^w}\sum_{j<t} U_j(H_j)\right)^{1/2}. \tag{111}$$

□

These overlap comparisons average over emitted prefixes. A runtime average over proposal attempts also accounts for discarded proposals, rejection events, and bonus tokens.

Remark D.16 (Output KL and runtime acceptance)**.** For $p = (1,0)$, $q = (1/2, 1/2)$, $G = \{2\}$, and an active write,

$$p_\delta = p, \qquad q_\delta = \frac{(1, e^\delta)}{1 + e^\delta}.$$

Thus $D_{\mathrm{KL}}(p_\delta\|p) = 0$, but $\beta(p,q) = 1/2$ and $\beta(p_\delta, q_\delta) = 1/(1+e^\delta)$. A draft green-mass gate admits this example whenever the site is a candidate and $\tau_Q \geq \log 2$. The comparison cannot be bounded by output KL alone because it also changes the proposal, rather than merely changing the distribution of a common statistic of the emitted text.

Remark D.17 (Additional gate randomization)**.** The prefix-deterministic condition is used in step (b) of equation (102). If an additional random gate is used, set $a_t(h) = \Pr(U_t = 1 \mid H_t = h)$. Assuming the conditional emitted law given (H_t, U_t) is the corresponding gated target, its marginal kernel at h is

$$m_t(\cdot \mid h) = (1 - a_t(h))p_t(\cdot \mid h) + a_t(h)(p_t)_\delta(\cdot \mid h).$$

For $F(x) = x \log x$, $F''(x) = 1/x > 0$. On the support of p_t, convexity gives

$$\begin{aligned} D_{\mathrm{KL}}(m_t \| p_t) &= \sum_v p_t(v) F\left((1 - a_t) + a_t \frac{(p_t)_\delta(v)}{p_t(v)}\right) \\ &\le \sum_v p_t(v) \left((1 - a_t)F(1) + a_t F\left(\frac{(p_t)_\delta(v)}{p_t(v)}\right)\right) \\ &= a_t D_{\mathrm{KL}}((p_t)_\delta \| p_t). \end{aligned} \tag{112}$$

The inequality is strict when $0 < a_t < 1$ and $(p_t)_\delta \neq p_t$. Thus the write-dependent upper bound extends to this randomized gate, while the equality in equation (86) becomes an inequality.

D.4 Optimal Allocation under a Frozen-state Budget

Let $z_1, \ldots, z_m$ be a fixed finite collection of states, with weights $\nu_i > 0$ and $\sum_i \nu_i = 1$. At state i, let $e_i \in [0, \infty)$ and $c_i \in [0, \infty)$ be finite benefit and cost. The state weights, benefits, and costs do not depend on the activation policy. For a vector of pre-proposal activation probabilities $a \in [0, 1]^m$, set

$$B(a) := \sum_{i=1}^m \nu_i a_i e_i, \qquad C(a) := \sum_{i=1}^m \nu_i a_i c_i. \tag{113}$$

For a finite budget $L \ge 0$, consider

$$\max_{a \in [0,1]^m} B(a) \quad \text{subject to } C(a) \le L. \tag{114}$$

For example, setting $e_i = e_{p_i(G_i)}(\delta)$ makes the objective the average increase in green-match probability at the fixed states, with weights ν_i.

Theorem D.18 (Evidence-per-cost threshold allocation)**.** *Problem (114) has an optimal solution a^* with at most one coordinate strictly between zero and one. There is a finite $\lambda_{lagrange} \ge 0$ such that*

$$a_i^* = \begin{cases} 1, & e_i > \lambda_{lagrange} c_i, \\ 0, & e_i < \lambda_{lagrange} c_i, \\ a \text{ value in } [0,1], & e_i = \lambda_{lagrange} c_i, \end{cases} \qquad \lambda_{lagrange}(C(a^*) - L) = 0. \tag{115}$$

For each fixed $\lambda_{lagrange}$, any feasible a^* satisfying equation (115) is optimal. The optimal value also obeys

$$\max_{a : C(a) \le L} B(a) = \min_{\lambda_{lagrange} \ge 0} \left\{ \lambda_{lagrange} L + \sum_i \nu_i (e_i - \lambda_{lagrange} c_i)_+ \right\}. \tag{116}$$

We first prove a local exchange identity and an exact primal–dual gap formula. We will then construct a feasible threshold solution in every budget regime.

Lemma D.19 (Exchange of equal amounts of budget)**.** *Let a be feasible. Suppose $c_i, c_j > 0$, $e_i/c_i > e_j/c_j$, $a_i < 1$, and $a_j > 0$. Then a cannot be optimal for equation (114).*

Proof. Choose

$$0 < \xi \le \min\{\nu_i c_i (1 - a_i), \nu_j c_j a_j\}. \tag{117}$$

Define

$$a_i' := a_i + \frac{\xi}{\nu_i c_i}, \qquad a_j' := a_j - \frac{\xi}{\nu_j c_j}, \tag{118}$$

with all other coordinates unchanged. The upper bound on ξ guarantees $a' \in [0,1]^m$. The cost change is exactly

$$C(a') - C(a) = \nu_i c_i \frac{\xi}{\nu_i c_i} - \nu_j c_j \frac{\xi}{\nu_j c_j} = \xi - \xi = 0. \tag{119}$$

Thus feasibility is preserved, whereas the benefit change is

$$B(a') - B(a) = \nu_i e_i \frac{\xi}{\nu_i c_i} - \nu_j e_j \frac{\xi}{\nu_j c_j} = \xi \left(\frac{e_i}{c_i} - \frac{e_j}{c_j} \right) > 0. \tag{120}$$

This contradicts optimality. Notice that the weights cancel in the benefit-to-cost ratios, but remain in the feasible amount ξ that can be exchanged. □

Lemma D.20 (Exact gap certificate)**.** *For $\lambda_{lagrange} \geq 0$, define*

$$D(\lambda_{lagrange}) = \lambda_{lagrange} L + \sum_i \nu_i (e_i - \lambda_{lagrange} c_i)_+. \tag{121}$$

For every $a \in [0,1]^m$,

$$\begin{aligned} D(\lambda_{lagrange}) - B(a) = \lambda_{lagrange}(L - C(a)) \\ + \sum_i \nu_i \big[(1 - a_i)(e_i - \lambda_{lagrange} c_i)_+ + a_i (\lambda_{lagrange} c_i - e_i)_+ \big]. \end{aligned} \tag{122}$$

For feasible a, this difference is nonnegative. It vanishes exactly when the threshold conditions and complementary slackness in equation (115) hold.

Proof. Put $d_i = e_i - \lambda_{lagrange} c_i$. Then

$$\begin{aligned} D(\lambda_{lagrange}) - B(a) &= \lambda_{lagrange} L + \sum_i \nu_i (d_i)_+ - \sum_i \nu_i a_i (d_i + \lambda_{lagrange} c_i) \\ &= \lambda_{lagrange}(L - C(a)) + \sum_i \nu_i \big((d_i)_+ - a_i d_i \big) \\ &\overset{(a)}{=} \lambda_{lagrange}(L - C(a)) + \sum_i \nu_i \big((1 - a_i)(d_i)_+ + a_i (-d_i)_+ \big), \end{aligned} \tag{123}$$

where (a) uses $d_i = (d_i)_+ - (-d_i)_+$. For feasible a, every term in the last line is nonnegative: $L - C(a) \geq 0$, $\lambda_{lagrange} \geq 0$, $0 \leq a_i \leq 1$, and $\nu_i > 0$. Therefore the sum is zero if and only if all its terms are zero. The first term gives complementary slackness. If $d_i > 0$, its coordinate term vanishes exactly when $a_i = 1$; if $d_i < 0$, exactly when $a_i = 0$; if $d_i = 0$, it vanishes for every $a_i \in [0,1]$. These are precisely the asserted threshold conditions. □

Proof of Theorem D.18. The feasible set is nonempty (it contains $a = 0$), closed, and bounded in $\mathbb{R}^m$. Its continuous objective thus attains a maximum. We now give an explicit optimal vector and a finite multiplier.

Zero-cost and zero-benefit coordinates. For $c_i = 0 < e_i$, set $a_i^* = 1$: this increases benefit without consuming budget. For $e_i = 0$, set $a_i^* = 0$, including when $c_i = 0$. It remains to decide the states in

$$J := \{ i : c_i > 0,\ e_i > 0 \}, \qquad C_+ := \sum_{i \in J} \nu_i c_i. \tag{124}$$

The coordinates already fixed agree with any positive multiplier; when $\lambda_{lagrange} = 0$, all zero-benefit coordinates lie on the threshold boundary.

Case 1: $L \geq C_+$. Set $a_i^* = 1$ for every $i \in J$ and $\lambda_{lagrange} = 0$. Then $C(a^*) = C_+ \leq L$, every positive-benefit state is activated, and every other state is on the equality branch $e_i = \lambda_{lagrange} c_i = 0$. Complementary slackness holds because $\lambda_{lagrange} = 0$. This case includes $J = \varnothing$ and $L = 0 = C_+$.

Case 2: $L = 0 < C_+$**.** Set $a_i^* = 0$ for every positive-cost state. Since there are finitely many such states and their costs are strictly positive, the number

$$M := \max_{i:c_i>0} \frac{e_i}{c_i} \tag{125}$$

is finite. Choose $\lambda_{lagrange} = M + 1$. Then $e_i < \lambda_{lagrange} c_i$ at every positive-cost state, while the already chosen zero-cost states satisfy the threshold rule. Moreover, $C(a^*) = 0 = L$, proving complementary slackness.

Case 3: $0 < L < C_+$**.** Let $J = \{i_1, \dots, i_k\}$ be ordered so that

$$\rho_1 \geq \cdots \geq \rho_k > 0, \qquad \rho_j := \frac{e_{i_j}}{c_{i_j}}, \qquad w_j := \nu_{i_j} c_{i_j} > 0. \tag{126}$$

Ties can be ordered arbitrarily. Define $W_0 = 0$, $W_j = \sum_{\ell=1}^{j} w_\ell$, and

$$j_* := \min\{j : W_j \geq L\}. \tag{127}$$

This index exists because $W_k = C_+ > L$, and its minimality gives $W_{j_*-1} < L \leq W_{j_*}$. Set

$$a_{i_j}^* = \begin{cases} 1, & j < j_*, \\ \dfrac{L - W_{j_*-1}}{w_{j_*}}, & j = j_*, \\ 0, & j > j_*, \end{cases} \qquad \lambda_{lagrange} := \rho_{j_*}. \tag{128}$$

The boundary activation lies in $(0, 1]$, and all other coordinates are integral. Its cost is

$$\begin{aligned} C(a^*) &= \sum_{j<j_*} w_j + w_{j_*} \frac{L - W_{j_*-1}}{w_{j_*}} \\ &= W_{j_*-1} + L - W_{j_*-1} = L. \end{aligned} \tag{129}$$

If $e_{i_j} > \lambda_{lagrange} c_{i_j}$, then $\rho_j > \rho_{j_*}$, which forces $j < j_*$ and hence $a_{i_j}^* = 1$. If $e_{i_j} < \lambda_{lagrange} c_{i_j}$, then $j > j_*$ and $a_{i_j}^* = 0$. Coordinates tied with $\lambda_{lagrange}$ fall on the free boundary branch, including any tied coordinates selected fully or left inactive. The zero-cost and zero-benefit coordinates were already chosen consistently. Equation (129) supplies complementary slackness.

Optimality in all three cases. Each construction is feasible and satisfies equation (115). Lemma D.20 therefore gives

$$B(a^*) = D(\lambda_{lagrange}). \tag{130}$$

For every feasible a, the same lemma gives $B(a) \leq D(\lambda_{lagrange}) = B(a^*)$, proving optimality. All constructed solutions have at most one fractional coordinate. More generally, any feasible threshold vector with complementary slackness also has zero gap and is optimal.

Finally, for every $\widetilde{\lambda}_{lagrange} \geq 0$, Lemma D.20 applied to a^* implies $D(\widetilde{\lambda}_{lagrange}) \geq B(a^*) = D(\lambda_{lagrange})$. Thus the constructed multiplier minimizes D, and equation (116) follows. □

Corollary D.21 (Allocation with finite draft-time information)**.** *Suppose the observable information partitions the states into nonempty classes $J_1, \dots, J_k$. An admissible activation must be constant on each class: $a_i = \alpha_j$ for $i \in J_j$. Define*

$$\omega_j := \sum_{i \in J_j} \nu_i, \qquad \bar{e}_j := \frac{\sum_{i \in J_j} \nu_i e_i}{\omega_j}, \qquad \bar{c}_j := \frac{\sum_{i \in J_j} \nu_i c_i}{\omega_j}. \tag{131}$$

An optimal admissible rule has $\alpha_j = 1$ if $\bar{e}_j > \lambda_{lagrange} \bar{c}_j$, $\alpha_j = 0$ if $\bar{e}_j < \lambda_{lagrange} \bar{c}_j$, and boundary choices ensuring feasibility and complementary slackness.

Proof. For every admissible a, regrouping the finite sums by information class yields

$$\begin{aligned} B(a) &= \sum_{j=1}^{k} \sum_{i \in J_j} \nu_i \alpha_j e_i = \sum_{j=1}^{k} \omega_j \alpha_j \bar{e}_j, \\ C(a) &= \sum_{j=1}^{k} \sum_{i \in J_j} \nu_i \alpha_j c_i = \sum_{j=1}^{k} \omega_j \alpha_j \bar{c}_j. \end{aligned} \tag{132}$$

Every vector $\alpha \in [0,1]^k$ defines an admissible a, giving an exact reduction to the aggregated problem. The weights ω_j are positive and sum to one. Theorem D.18 applies to this problem and proves the claim. Equivalently, $\bar{e}_j = \mathbb{E}[e \mid \mathcal{I} = j]$ and $\bar{c}_j = \mathbb{E}[c \mid \mathcal{I} = j]$. For example, if two equally likely indistinguishable states have $(e, c) = (1, 1)$ and $(1, 3)$, then

$$\frac{\mathbb{E}[e \mid \mathcal{I}]}{\mathbb{E}[c \mid \mathcal{I}]} = \frac{1}{2}, \qquad \mathbb{E}\left[\frac{e}{c}\middle| \mathcal{I}\right] = \frac{1}{2}\left(1 + \frac{1}{3}\right) = \frac{2}{3}.$$

Thus replacing the ratio of conditional expectations by the conditional expectation of a ratio changes the rule in general. □

Remark D.22 (Scope of the allocation theorem)**.** The threshold rule uses an expected cost budget and allows a pre-proposal coin flip at a boundary state. The finite state space is important when the budget is zero. For example, with X uniform on $(0, 1)$, $e(X) = 1$, $c(X) = X$, and $L = 0$, feasibility forces $a = 0$ almost everywhere, whereas every finite $\lambda_{lagrange}$ activates a positive-measure set $\{x : 1 > \lambda_{lagrange} x\}$ and incurs positive cost. In the finite model, a threshold above all positive-cost benefit-to-cost ratios handles the zero-budget case.

A cost-only threshold gives the same ordering when all positive benefits are equal and the estimated costs rank states correctly, with suitable handling of boundary states. A cost-only rule may also be optimal at a particular budget even if the full orderings differ. These are conditions on the fixed-state problem. The theorem provides a reference for evaluating the draft and coverage gates, whose decisions use estimated write costs. Establishing their optimality would require checking the corresponding benefits, costs, and state weights.

D.5 A Homogeneous KL-Only Model Favors Distributed Evidence

We now allow bias strength to vary across states, in contrast to the fixed-bias activation problem above. Fix $N \geq 1$, distributions p_i and green lists G_i with a common mass $p_i(G_i) = s \in (0, 1)$, and a total evidence requirement $E \in (0, N(1 - s))$. For distributions w_i with $w_i(G_i) \geq s$, set $u_i = w_i(G_i) - s \in [0, 1 - s]$. Define

$$d_s(u) := d_{\mathrm{Ber}}(s + u \| s), \quad 0 \leq u \leq 1 - s, \tag{133}$$

with the value at $u = 1 - s$ defined by continuity.

Corollary D.23 (Uniform allocation of a prescribed evidence increment)**.** *Among all $w_1, \ldots, w_N$ satisfying $u_i \geq 0$ and $\sum_i u_i = E$,*

$$\sum_{i=1}^{N} D_{\mathrm{KL}}(w_i \| p_i) \geq \sum_{i=1}^{N} d_s(u_i) \geq N d_s(E/N). \tag{134}$$

Equality throughout holds precisely when $u_i = E/N$ for every i and each w_i is the corresponding KGW tilt, with common finite bias

$$\delta_* = \mathrm{logit}(s + E/N) - \mathrm{logit}(s). \tag{135}$$

In fact, the scalar gap has the explicit lower bound

$$\sum_{i=1}^{N} d_s(u_i) - N d_s(E/N) \geq 2 \sum_{i=1}^{N} (u_i - E/N)^2. \tag{136}$$

We prove the bound by convexity, including the endpoint cases.

Proof. First fix the increments u_i. If $0 < u_i < 1 - s$, Theorem D.4 gives the minimum conditional KL $d_s(u_i)$, achieved uniquely by the tilt with green mass $s + u_i$. At $u_i = 0$, KL nonnegativity gives the minimum zero, uniquely at $w_i = p_i$. At $u_i = 1 - s$, any admissible w_i is supported on G_i, and Lemma D.5 gives

$$D_{\mathrm{KL}}(w_i \| p_i) = \log(1/s) + D_{\mathrm{KL}}(w_i \| p_i(\cdot \mid G_i)) \geq \log(1/s) = d_s(1 - s). \tag{137}$$

This proves the first inequality in equation (134), including endpoints.

For $0 < u < 1 - s$, expand and differentiate:

$$d_s(u) = (s+u)\log\frac{s+u}{s} + (1-s-u)\log\frac{1-s-u}{1-s},$$
$$d_s'(u) = \log\frac{s+u}{s} + 1 - \log\frac{1-s-u}{1-s} - 1 = \log\frac{(s+u)(1-s)}{s(1-s-u)},$$
$$d_s''(u) = \frac{1}{s+u} + \frac{1}{1-s-u} = \frac{1}{(s+u)(1-s-u)} \geq 4. \tag{138}$$

The last inequality uses $x(1-x) \leq 1/4$ with $x = s+u \in (0,1)$. Put $\bar{u} = E/N \in (0, 1-s)$. For $u < 1-s$, twice integrating the second derivative along the segment from $\bar{u}$ to u gives

$$d_s(u) - d_s(\bar{u}) - d_s'(\bar{u})(u-\bar{u}) = (u-\bar{u})^2 \int_0^1 (1-t) d_s''(\bar{u} + t(u-\bar{u}))\, dt$$
$$\geq 4(u-\bar{u})^2 \int_0^1 (1-t)\, dt = 2(u-\bar{u})^2. \tag{139}$$

For clarity, the identity follows by setting $g(t) = d_s(\bar{u} + t(u-\bar{u}))$ and writing

$$g(1) - g(0) - g'(0) = \int_0^1 (g'(t) - g'(0))\, dt = \int_0^1 \int_0^t g''(r)\, dr\, dt = \int_0^1 (1-r) g''(r)\, dr.$$

The endpoint $u = 0$ is valid by the same derivatives or continuity. Letting $u \uparrow 1-s$ in the inequality part of equation (139) proves it also at the upper endpoint, since d_s is continuous there and $d_s'(\bar{u})$ is finite.

Sum equation (139) over $u = u_i$. Since $\sum_i (u_i - \bar{u}) = E - N\bar{u} = 0$, the linear terms cancel, yielding

$$\sum_i d_s(u_i) - N d_s(\bar{u}) \geq d_s'(\bar{u}) \underbrace{\sum_i (u_i - \bar{u})}_{=0} + 2\sum_i (u_i - \bar{u})^2$$
$$= 2\sum_i (u_i - \bar{u})^2 \geq 0. \tag{140}$$

This proves equation (134) and equation (136). Equality in equation (134) forces every squared deviation to vanish, so $u_i = \bar{u}$ for all i. Since $0 < \bar{u} < 1 - s$, the equality conditions in Theorem D.4 then force each w_i to be the unique finite KGW tilt at that mass. Conversely, these choices attain every bound with equality, proving the claimed optimum. □

The minimum is attained by giving every position the same green-list probability increase, E/N, with common bias $\delta_* = \mathrm{logit}(s + E/N) - \mathrm{logit}(s)$. The required bias therefore depends on the evidence assigned to each position. This result minimizes KL when all initial green-list masses are equal and bias strengths can vary. Appendix D.4 studies write selection when speculative costs vary across states.

E Experimental Protocols and Additional Results

E.1 Shared Evaluation Protocol and Architecture Coverage

Backbones, tasks, and decoder artifacts. Following MarkLLM [13], the main comparison uses Vicuna-7B, Vicuna-13B, and Vicuna-33B [46] with 400 C4 samples [47] and 80 first-turn MT-Bench prompts [48]. We evaluate token-level SpS, Medusa [19], and EAGLE-1/2/3 [20, 21, 22]. The token-level main comparison uses a Vicuna-v1.3-68M draft and lookahead 5. The exact 7B artifact names are recorded with the hyperparameters below.

Architecture coverage and generalizability. Our experiments use the Vicuna family because it is the only backbone family in our evaluation for which compatible public artifacts

are available across token-level SpS, Medusa, and multiple EAGLE variants. Specifically, the Qwen2.5 configurations are used only for token-level mechanism analysis and cross-model evaluation because compatible Medusa and EAGLE drafting artifacts are unavailable in our experimental setup. For EAGLE, we evaluate EAGLE-3 with Vicuna-13B using the corresponding public draft artifact, while the Vicuna-7B and Vicuna-33B configurations use the available EAGLE-1/2 artifacts. The reported gains may vary across backbones due to differences in draft–target alignment, proposal topology, and gate thresholds.

Sampling and random seeds. The main comparison uses multinomial sampling with temperature 1, top-$p = 1$, top-$k = 0$, up to 256 continuation tokens, and EOS stopping. Each of five independent generation seeds evaluates the same 80 MT-Bench and 400 C4 prompts; main-table standard deviations are over these seeds. Cross-model tests use the separate model pairs and shorter protocol in Appendix E.2.1. The held-out study uses disjoint development and test prompts and fixed-length generation, as specified in Appendix E.6.

Detection and efficiency metrics. We report TPR@1% FPR, acceptance, accepted tokens per target-model call, generation time or overhead, and WD20 detection. The ChiselMark detector uses the text and key with $M = 999$ reference keys and $\alpha = 0.01$. Following the EAGLE evaluation convention, acceptance is measured along the verified chain. Table 5 normalizes time by watermark-free token-level SpS for the same backbone and dataset. Table 1 reports overhead relative to the watermark-free decoder of the same architecture. Text quality is measured separately using paired judge scores, preferences, and perplexity; Appendix E.5 gives its protocol.

Baseline implementations and operating-point scope. The baseline-selection rationale and compatibility table are in Appendix A.3. Table 3 gives the exact KGW, SWEET, EWD, MorphMark, and ChiselMark settings. In particular, SWEET gates generation by draft entropy, whereas EWD uses the KGW generation configuration and entropy-weighted detection. ChiselMark's stronger bias with selective writes makes the main comparison neither equal-δ nor matched-TPR; the controlled study includes a development-selected matched-TPR comparison. For feature-level vanilla KGW, Medusa uses the last accepted token at the start of the round as an anchor, while EAGLE uses each draft node's parent token.

Verifier scope and computational resources. Theorem 4.1 covers the gated token-level sampler. Appendix C specifies the direct-draw feature-level verifier, and Appendix D.2 analyzes its conditional reuse probability. We evaluate the reported Medusa and EAGLE configurations empirically. All experiments used a server with four NVIDIA H20 GPUs; reproducing the complete set requires approximately 400 GPU hours.

E.2 Watermark Hyperparameters

Main-Comparison Configurations. The main comparisons and held-out controlled study use different watermark configurations, detailed in Table 3 and Table 7, respectively.

For the Vicuna-7B row of the main comparison, the token-level target is Vicuna-7B-v1.3 with a Vicuna-v1.3-68M draft; Medusa uses the Medusa-Vicuna-7B-v1.3 head, and EAGLE-1 uses the EAGLE-Vicuna-7B-v1.3 draft head. Sampling, prompt counts, lookahead, and seeds follow Appendix E.1.

Table 3 reports the watermark-specific operating points. For KGW-based methods, $\gamma = 0.5$ and detector context length $\ell = 1$. For ChiselMark, $B = 2$ gives an expected green fraction of $1/2$. The candidate density ρ denotes the fraction of positions eligible for intervention before gating; the realized write rate can therefore be lower when a gate vetoes eligible positions.

E.2.1 Cross-Model Acceptance Recovery

We evaluate the nine model–dataset configurations of Figure 1 under a separate fixed-bias protocol ($\delta = 5$). The evaluated target–draft pairs are Qwen2.5-7B-Instruct with Qwen2.5-1.5B-Instruct, Qwen2.5-Coder-7B-Instruct with Qwen2.5-Coder-1.5B-Instruct, and Vicuna-7B-v1.3 with a Vicuna-v1.3-68M draft model. Token-level speculative sampling uses lookahead 4. Each dataset uses 30 prompts under each of two generation/key seeds, giving

Table 3: **Watermark and decoder configurations for the main comparison (Table 5).**

Decoder	Method	Configuration
Token-level speculative sampling		
Token SpS	KGW	$\delta = 2$, $\gamma = 0.5$, $\ell = 1$
Token SpS	SWEET	$\delta = 2$, $\gamma = 0.5$, $\ell = 1$; watermarking is enabled when the unwatermarked draft entropy is at least 0.9 nats
Token SpS	EWD	$\delta = 2$, $\gamma = 0.5$, $\ell = 1$; entropy-weighted detection
Token SpS	MorphMark	Exponential variant with $k_{\text{exp}} = 1.3$, $p_0 = 0.15$, $\gamma = 0.5$, and prefix/context length 1
Token SpS	ChiselMark	$\delta = 5$, $\rho = 0.5$, $B = 2$, $\ell = 1$
Medusa verification		
Medusa	KGW	$\delta = 2$, $\gamma = 0.5$, $\ell = 1$
Medusa	MorphMark	Exponential variant with $k_{\text{exp}} = 1.3$, $p_0 = 0.15$, $\gamma = 0.5$, and prefix/context length 1
Medusa	ChiselMark	$\delta = 4$, $\rho = 1$, $B = 2$, $\ell = 1$
EAGLE-1 verification		
EAGLE-1	KGW	$\delta = 2$, $\gamma = 0.5$, $\ell = 1$
EAGLE-1	MorphMark	Exponential variant with $k_{\text{exp}} = 1.3$, $p_0 = 0.15$, $\gamma = 0.5$, and prefix/context length 1
EAGLE-1	ChiselMark	$\delta = 4$, $\rho = 1$, $B = 2$, $\ell = 1$

Target–draft pair	Dataset	Watermark-free	Dense KGW	ChiselMark
Qwen2.5-Instruct	HumanEval	0.885	0.787	0.884
	BigCodeBench	0.889	0.792	0.895
	TinyStories	0.742	0.704	0.758
Qwen2.5-Coder-Instruct	HumanEval	0.888	0.823	0.897
	BigCodeBench	0.922	0.823	0.923
	TinyStories	0.728	0.683	0.735
Vicuna-v1.3	HumanEval	0.434	0.398	0.437
	BigCodeBench	0.459	0.395	0.472
	TinyStories	0.445	0.440	0.466

Table 4: Acceptance recovery across the nine model–dataset configurations of Figure 1 at $\delta = 5$, under an independent generation protocol.

60 outputs per cell. Generation uses temperature 1, top-$p = 1$, top-$k = 0$, at most 128 continuation tokens, and EOS stopping.

Dense KGW uses $\delta = 5$, $\gamma = 0.5$, $\ell = 1$, $\rho = 1$, with no gate. ChiselMark uses $\delta = 5$, $B = 2$, $\ell = 1$, $\rho = 0.5$, and the draft green-mass gate $q_G \geq 0.35$.

Table 4 reports acceptance-rate point estimates at $\delta = 5$ for the same three target–draft pairs and three datasets, using the independent protocol specified above. Dense KGW reduces acceptance in all nine configurations, with the largest drops in the Qwen code-generation settings. ChiselMark brings acceptance close to the watermark-free level across the grid. This experiment examines acceptance recovery; detection power and end-to-end time were not measured under this protocol.

Some acceptance point estimates exceed the watermark-free level. Such increases are compatible with the fixed-state analysis in Appendix D.2: Proposition D.7 bounds the absolute overlap change and allows either sign.

E.3 Complete Main Comparison

The complete 24-configuration comparison covers Vicuna-7B, Vicuna-13B, and Vicuna-33B, including EAGLE-3 at its available 13B configuration. Across scales, ChiselMark recovers speculative acceptance, with smaller gains for EAGLE on C4.

We compute $(\mathrm{AR}_{\mathrm{ChiselMark}} - \mathrm{AR}_{\mathrm{KGW}})/(\mathrm{AR}_{\mathrm{NoWM}} - \mathrm{AR}_{\mathrm{KGW}})$ for each of the 24 configurations and take the unweighted mean, obtaining approximately 61.5% from the displayed means. Medusa uses the round-anchor seeding convention specified in Appendix E.1; it must be considered when interpreting detection.

Table 5: **Comprehensive experimental results** on MT-Bench and C4 with Vicuna-7B, Vicuna-13B, and Vicuna-33B. Results are reported as mean ± standard deviation. Standard deviations are over 5 random seeds. Generation time is normalized by watermark-free token-level SpS under the same backbone and dataset.

	MT-Bench				C4			
Method	TPR@1(↑)	TPR@1(↑) WD20	Normalize Time(↓)	AR.(↑)	TPR@1(↑)	TPR@1(↑) WD20	Normalize Time(↓)	AR.(↑)
Vicuna-7B								
Token-level SpS								
No watermark	—	—	1.000 ± 0.000	0.467 ± 0.006	—	—	1.000 ± 0.000	0.564 ± 0.009
+ KGW	0.5600 ± 0.0487	0.2900 ± 0.0324	1.118 ± 0.019	0.362 ± 0.006	0.8500 ± 0.0486	0.6465 ± 0.0308	1.125 ± 0.022	0.465 ± 0.005
+ SWEET	0.6200 ± 0.0360	0.3850 ± 0.0240	1.159 ± 0.015	0.362 ± 0.009	0.9000 ± 0.0425	0.7490 ± 0.0675	1.172 ± 0.024	0.462 ± 0.008
+ EWD	**0.7350 ± 0.0358**	0.5200 ± 0.0190	1.176 ± 0.021	0.363 ± 0.011	**0.9265 ± 0.0149**	0.7945 ± 0.0966	1.194 ± 0.028	0.448 ± 0.018
+ MorphMark	0.2450 ± 0.0420	0.1075 ± 0.0326	1.124 ± 0.024	0.370 ± 0.007	0.7665 ± 0.0373	0.4410 ± 0.0409	1.154 ± 0.020	0.461 ± 0.006
+ ChiselMark	0.6950 ± 0.0577	**0.5400 ± 0.1066**	**1.028 ± 0.016**	**0.427 ± 0.005**	0.9035 ± 0.0477	**0.8290 ± 0.0194**	**1.023 ± 0.040**	**0.520 ± 0.010**
Medusa								
No watermark	—	—	0.776 ± 0.031	0.675 ± 0.004	—	—	1.021 ± 0.024	0.433 ± 0.010
+ KGW	0.1150 ± 0.0163	0.0575 ± 0.0209	0.877 ± 0.014	0.543 ± 0.004	0.5265 ± 0.0418	0.3135 ± 0.0296	1.151 ± 0.068	0.390 ± 0.005
+ MorphMark	0.0375 ± 0.0125	0.0275 ± 0.0137	0.848 ± 0.021	0.547 ± 0.004	0.3565 ± 0.0917	0.1645 ± 0.0424	1.122 ± 0.028	0.412 ± 0.007
+ ChiselMark	**0.6825 ± 0.0447**	**0.5275 ± 0.0399**	**0.784 ± 0.028**	**0.625 ± 0.009**	**0.8865 ± 0.0938**	**0.7880 ± 0.0887**	**1.080 ± 0.036**	**0.428 ± 0.010**
EAGLE-1								
No watermark	—	—	0.860 ± 0.041	0.839 ± 0.008	—	—	0.881 ± 0.040	0.599 ± 0.014
+ KGW	0.5300 ± 0.0464	0.0775 ± 0.0137	0.981 ± 0.048	0.737 ± 0.009	0.8635 ± 0.0545	0.2610 ± 0.0111	0.956 ± 0.061	0.493 ± 0.011
+ MorphMark	0.2550 ± 0.0447	0.1275 ± 0.0163	1.017 ± 0.022	0.739 ± 0.003	0.7765 ± 0.0480	0.4755 ± 0.0384	1.042 ± 0.059	0.508 ± 0.005
+ ChiselMark	**0.5425 ± 0.0190**	**0.2850 ± 0.0428**	**0.908 ± 0.049**	**0.826 ± 0.005**	**0.8865 ± 0.0361**	**0.6590 ± 0.0487**	**0.906 ± 0.037**	**0.520 ± 0.009**
EAGLE-2								
No watermark	—	—	0.872 ± 0.038	0.835 ± 0.009	—	—	0.869 ± 0.043	0.604 ± 0.012
+ KGW	0.5425 ± 0.0411	0.0825 ± 0.0168	0.973 ± 0.044	0.731 ± 0.010	0.8785 ± 0.0508	0.2745 ± 0.0146	0.968 ± 0.057	0.489 ± 0.010
+ MorphMark	0.2675 ± 0.0401	0.1350 ± 0.0185	1.026 ± 0.026	0.744 ± 0.004	0.7890 ± 0.0446	0.4880 ± 0.0357	1.056 ± 0.063	0.511 ± 0.006
+ ChiselMark	**0.5550 ± 0.0209**	**0.2975 ± 0.0399**	**0.916 ± 0.046**	**0.821 ± 0.006**	**0.8985 ± 0.0329**	**0.6715 ± 0.0451**	**0.918 ± 0.041**	**0.527 ± 0.008**
Vicuna-13B								
Token-level SpS								
No watermark	—	—	1.000 ± 0.000	0.455 ± 0.010	—	—	1.000 ± 0.000	0.553 ± 0.011
+ KGW	0.5175 ± 0.0420	0.2750 ± 0.0250	1.147 ± 0.019	0.344 ± 0.009	0.8160 ± 0.0473	0.6020 ± 0.0361	1.154 ± 0.024	0.451 ± 0.008
+ SWEET	0.5975 ± 0.0347	0.3575 ± 0.0301	1.183 ± 0.020	0.351 ± 0.009	0.8720 ± 0.0426	0.7110 ± 0.0559	1.198 ± 0.029	0.447 ± 0.008
+ EWD	**0.7050 ± 0.0360**	0.4875 ± 0.0250	1.139 ± 0.021	0.347 ± 0.007	**0.9135 ± 0.0207**	0.7640 ± 0.0773	1.148 ± 0.023	0.452 ± 0.007
+ MorphMark	0.2900 ± 0.0437	0.1400 ± 0.0271	1.151 ± 0.025	0.361 ± 0.009	0.7365 ± 0.0449	0.4215 ± 0.0417	1.179 ± 0.022	0.448 ± 0.008
+ ChiselMark	0.6625 ± 0.0476	**0.5725 ± 0.0877**	**1.051 ± 0.019**	**0.418 ± 0.007**	0.8845 ± 0.0412	**0.8515 ± 0.0268**	**1.046 ± 0.033**	**0.509 ± 0.010**
Medusa								
No watermark	—	—	0.806 ± 0.031	0.653 ± 0.006	—	—	1.042 ± 0.028	0.421 ± 0.010
+ KGW	0.1575 ± 0.0209	0.0800 ± 0.0168	0.911 ± 0.018	0.519 ± 0.006	0.4790 ± 0.0434	0.2875 ± 0.0291	1.181 ± 0.057	0.376 ± 0.008
+ MorphMark	0.0700 ± 0.0168	0.0450 ± 0.0143	0.881 ± 0.022	0.533 ± 0.006	0.4095 ± 0.0763	0.1985 ± 0.0418	1.151 ± 0.035	0.394 ± 0.009
+ ChiselMark	**0.6375 ± 0.0451**	**0.5550 ± 0.0381**	**0.814 ± 0.029**	**0.603 ± 0.009**	**0.8470 ± 0.0732**	**0.8125 ± 0.0768**	**1.109 ± 0.036**	**0.414 ± 0.010**
EAGLE-1								
No watermark	—	—	0.883 ± 0.036	0.824 ± 0.008	—	—	0.902 ± 0.040	0.582 ± 0.012
+ KGW	0.5700 ± 0.0401	0.0975 ± 0.0163	1.017 ± 0.042	0.711 ± 0.009	0.8385 ± 0.0487	0.2425 ± 0.0158	0.992 ± 0.053	0.474 ± 0.011
+ MorphMark	0.3025 ± 0.0409	0.1625 ± 0.0198	1.061 ± 0.027	0.724 ± 0.005	0.7310 ± 0.0454	0.4500 ± 0.0349	1.085 ± 0.058	0.488 ± 0.007
+ ChiselMark	**0.5925 ± 0.0244**	**0.3300 ± 0.0381**	**0.941 ± 0.043**	**0.799 ± 0.007**	**0.8515 ± 0.0365**	**0.6985 ± 0.0431**	**0.943 ± 0.037**	**0.508 ± 0.009**
EAGLE-3								
No watermark	—	—	0.907 ± 0.033	0.812 ± 0.009	—	—	0.907 ± 0.039	0.578 ± 0.012
+ KGW	0.5850 ± 0.0369	0.1050 ± 0.0168	1.009 ± 0.039	0.704 ± 0.010	0.8565 ± 0.0453	0.2570 ± 0.0176	1.016 ± 0.051	0.469 ± 0.010
+ MorphMark	0.3200 ± 0.0401	0.1725 ± 0.0205	1.074 ± 0.030	0.731 ± 0.006	0.7495 ± 0.0431	0.4635 ± 0.0358	1.101 ± 0.056	0.493 ± 0.008
+ ChiselMark	**0.6075 ± 0.0259**	**0.3450 ± 0.0391**	**0.949 ± 0.041**	**0.795 ± 0.008**	**0.8665 ± 0.0336**	**0.7150 ± 0.0452**	**0.954 ± 0.036**	**0.516 ± 0.009**
Vicuna-33B								
Token-level SpS								
No watermark	—	—	1.000 ± 0.000	0.486 ± 0.009	—	—	1.000 ± 0.000	0.572 ± 0.012
+ KGW	0.4975 ± 0.0399	0.2525 ± 0.0271	1.143 ± 0.018	0.381 ± 0.009	0.8010 ± 0.0436	0.5815 ± 0.0385	1.158 ± 0.025	0.481 ± 0.008
+ SWEET	0.5850 ± 0.0409	0.3425 ± 0.0314	1.169 ± 0.022	0.372 ± 0.007	0.8615 ± 0.0375	0.6895 ± 0.0527	1.184 ± 0.028	0.474 ± 0.009
+ EWD	**0.6800 ± 0.0326**	0.4675 ± 0.0288	1.137 ± 0.020	0.378 ± 0.008	**0.9025 ± 0.0219**	0.7435 ± 0.0754	1.151 ± 0.021	0.479 ± 0.007
+ MorphMark	0.3175 ± 0.0349	0.1575 ± 0.0274	1.154 ± 0.025	0.388 ± 0.009	0.7110 ± 0.0487	0.4020 ± 0.0396	1.171 ± 0.024	0.477 ± 0.008
+ ChiselMark	0.6575 ± 0.0481	**0.5825 ± 0.0837**	**1.048 ± 0.019**	**0.439 ± 0.007**	0.8725 ± 0.0384	**0.8460 ± 0.0279**	**1.044 ± 0.031**	**0.529 ± 0.010**
Medusa								
No watermark	—	—	0.804 ± 0.026	0.689 ± 0.007	—	—	1.041 ± 0.031	0.452 ± 0.011
+ KGW	0.1650 ± 0.0224	0.0825 ± 0.0168	0.916 ± 0.019	0.557 ± 0.007	0.4875 ± 0.0441	0.2785 ± 0.0265	1.182 ± 0.052	0.409 ± 0.008
+ MorphMark	0.0725 ± 0.0163	0.0475 ± 0.0137	0.889 ± 0.025	0.569 ± 0.006	0.4210 ± 0.0712	0.2055 ± 0.0378	1.149 ± 0.036	0.421 ± 0.009
+ ChiselMark	**0.6250 ± 0.0405**	**0.5575 ± 0.0371**	**0.816 ± 0.029**	**0.641 ± 0.009**	**0.8425 ± 0.0674**	**0.8060 ± 0.0725**	**1.104 ± 0.032**	**0.439 ± 0.010**
EAGLE-1								
No watermark	—	—	0.879 ± 0.033	0.858 ± 0.009	—	—	0.902 ± 0.038	0.614 ± 0.011
+ KGW	0.5725 ± 0.0379	0.1025 ± 0.0163	1.012 ± 0.039	0.751 ± 0.008	0.8290 ± 0.0478	0.2360 ± 0.0179	0.981 ± 0.049	0.513 ± 0.010
+ MorphMark	0.2950 ± 0.0391	0.1675 ± 0.0227	1.058 ± 0.029	0.759 ± 0.006	0.7245 ± 0.0418	0.4375 ± 0.0324	1.071 ± 0.054	0.521 ± 0.008
+ ChiselMark	**0.6025 ± 0.0240**	**0.3375 ± 0.0354**	**0.938 ± 0.041**	**0.836 ± 0.007**	**0.8460 ± 0.0341**	**0.7030 ± 0.0415**	**0.938 ± 0.036**	**0.541 ± 0.009**
EAGLE-2								
No watermark	—	—	0.902 ± 0.032	0.847 ± 0.009	—	—	0.898 ± 0.037	0.619 ± 0.012
+ KGW	0.5800 ± 0.0349	0.1100 ± 0.0185	1.004 ± 0.037	0.744 ± 0.009	0.8485 ± 0.0437	0.2490 ± 0.0181	1.003 ± 0.047	0.509 ± 0.010
+ MorphMark	0.3075 ± 0.0371	0.1750 ± 0.0217	1.067 ± 0.031	0.765 ± 0.007	0.7380 ± 0.0405	0.4520 ± 0.0347	1.086 ± 0.052	0.524 ± 0.007
+ ChiselMark	**0.6175 ± 0.0259**	**0.3525 ± 0.0369**	**0.946 ± 0.039**	**0.832 ± 0.008**	**0.8615 ± 0.0319**	**0.7185 ± 0.0438**	**0.947 ± 0.035**	**0.548 ± 0.009**

E.4 Text-Edit Robustness Settings

Text-edit procedures. For word substitution (Word-S), we randomly replace 10%, 20%, or 30% of the words with WordNet synonyms [49]. For word deletion (Word-D), we randomly remove 10%, 20%, or 30% of the words. The 20% word-deletion setting corresponds to WD20 in the main-comparison tables. For paraphrasing, we use Dipper [50] with lexical diversity settings of 20 and 40, denoted as Dipper-20 and Dipper-40, respectively. We additionally use gpt-5.4-mini-2026-03-17 for LLM-based paraphrasing, denoted as GPT.

Evaluation protocol. The robustness evaluation in Figure 5 uses a separate robustness protocol with 80 watermarked texts per configuration, generated from 80 C4 prompts in a single run with 200 new tokens per prompt. Token-level SpS and EAGLE-1 use Vicuna-7B, while EAGLE-3 uses Vicuna-13B. Table 5 uses the main-comparison sample sizes and output lengths.

E.5 Text Quality

We evaluate response quality separately from Table 5 using Vicuna-13B on 100 instances sampled with replacement from 80 first-turn MT-Bench prompts. For each sampled instance and decoder, we generate KGW, ChiselMark, and watermark-free (No-WM) responses under matched decoding configurations. We compare the 1–10 judge score of each watermarked response with its paired No-WM response and report the mean difference with a 95% confidence interval from 10,000 paired bootstrap resamples. These intervals describe each method's difference from No-WM; equivalence and differences between watermark methods require separate tests. We also compare each response pair in randomized A/B order and report Win/Tie/Loss preferences. Perplexity is computed using the target model and normalized by No-WM.

Table 6 combines these quality results with TPR@1% FPR and relative acceptance loss from the main-comparison experiments in Table 5. Relative acceptance loss is $\frac{\mathrm{AR}_{\mathrm{NoWM}}-\mathrm{AR}_{\mathrm{method}}}{\mathrm{AR}_{\mathrm{NoWM}}}$. Across token-level SpS and the evaluated EAGLE decoders, ChiselMark has judge-score differences closer to zero and lower PPL ratios than dense KGW in this evaluation.

Table 6: **Text quality and speculative-efficiency impact under watermarking.** TPR@1% FPR and relative acceptance loss are taken from the independent main-comparison experiments in Table 5. Judge-score differences, pairwise preferences, and PPL ratios are evaluated separately on 100 instances sampled with replacement from the 80 MT-Bench first-turn prompts using Vicuna-13B. Δ Judge denotes the paired judge-score difference relative to No-WM, with brackets indicating 95% confidence intervals. Pairwise preference reports Win/Tie/Loss against No-WM. PPL is normalized by No-WM.

Decoder	Method	TPR@1% FPR ↑	Relative acceptance loss ↓	Δ Judge score vs. No-WM ↑	Pairwise preference vs. No-WM	PPL ratio ↓
Token SpS	KGW	0.52	24.4%	−0.24 [−0.36, −0.12]	19 / 46 / 35	1.18
	ChiselMark	**0.66**	**8.1%**	**−0.06** [**−0.16**, **+0.04**]	**25 / 51 / 24**	**1.12**
EAGLE-1	KGW	0.57	13.7%	−0.31 [−0.45, −0.17]	17 / 44 / 39	1.34
	ChiselMark	**0.59**	**3.0%**	**+0.02** [**−0.09**, **+0.13**]	**27 / 48 / 25**	**1.25**
EAGLE-3	KGW	0.58	13.3%	−0.27 [−0.40, −0.14]	19 / 43 / 38	1.31
	ChiselMark	**0.61**	**2.1%**	**−0.04** [**−0.15**, **+0.07**]	**24 / 52 / 24**	**1.23**

E.6 Ablations: Which Lever Matters?

Setup. We compare selected operating points to examine the roles of candidate sparsity and cost-aware gating at the token level, and verification-side coverage gating at the feature level. Unless stated otherwise, the controlled experiments use Vicuna-7B as the target model, Llama-2-68M as the token-level draft, and EAGLE-1 for feature-level analysis. **Data split and output length.** We use 32 C4 development prompts only to select operating points and 60 disjoint C4 test prompts for final evaluation. All reported held-out runs generate exactly 192 continuation tokens with EOS stopping disabled. This fixed-length protocol avoids a confound in which stronger watermark biases alter continuation length

and thereby affect both detection power and latency per output token. Token-level results average five independent key–seed runs; EAGLE-1 results average three. Confidence intervals are obtained by prompt-clustered bootstrap resampling.

Methods. At the token level, we compare watermark-free decoding, dense KGW [12], ungated keyed candidate sparsity, a draft-entropy gate, a draft green-mass gate, and full ChiselMark. Full ChiselMark combines candidate density $\rho < 1$ with the pre-proposal gate $q_G \geq a_Q$, where q_G is the draft probability mass assigned to the keyed green list.

Metrics. We evaluate detectability using TPR@1% FPR under the text-and-key detector. Efficiency is measured by accepted tokens per target-model call and generation overhead relative to the watermark-free decoder of the same architecture. The first measures proposal reuse; the second also reflects implementation and serving costs. Configurations were selected using detection and efficiency metrics. PPL ratios, computed with one fixed scoring model relative to watermark-free generation, are reported separately to assess text quality.

E.6.1 Held-Out Controlled-Study Hyperparameters

The controlled study uses $B = 2$ (expected green fraction 1/2) and $\ell = 1$. Its settings differ from those of the main comparison. The draft green-mass gate uses $q_G \geq a_Q$, equivalently $-\log q_G \leq \tau_Q$ with $\tau_Q = -\log a_Q$; the feature-level gate uses $C^{\text{tree}} \leq \tau_{\text{tree}}$. Table 7 summarizes the watermark configurations evaluated in the held-out controlled study, including the bias strength, candidate density, and gate settings.

Table 7: **Held-out controlled-study watermark configurations.** Braces denote separately evaluated settings; – denotes no gate. τ_H is the entropy threshold. These settings apply to Section 5.4 and are distinct from the main-comparison settings in Table 3.

Configuration	δ	ρ	Gate setting
Token-level speculative sampling			
Dense	$\{1, 1.5, 4\}$	1	–
Sparse KGW	5	$\{0.25, 0.5\}$	–
ChiselMark	4	0.25	$a_Q = 0.1$
ChiselMark	4	0.5	$a_Q = 0.35$
Cost gate only	4	1	$a_Q \in \{0.5, 0.65\}$
Entropy gate	4	1	$\tau_H \in \{3, 5\}$
EAGLE-1/2/3 verification-side intervention			
Dense verifier	$\{1, 2, 4\}$	1	–
Sparse Verification-Only KGW	4	0.25	–
Coverage gate	4	1	$\tau_{\text{tree}} = 0.01$

E.6.2 Candidate Density

We compare dense watermarking ($\rho = 1$), ungated keyed sparsity ($\rho \in \{0.25, 0.5\}$), and full ChiselMark using the configurations in Appendix E.6.1. Figure 9 (right) and Table 1 report the tested settings. At the selected points in Table 1, ungated sparse KGW has higher TPR and slightly lower overhead, while ChiselMark accepts more tokens per target call. Theorem D.10 relates sequence distortion to the number of writes, and Corollary D.8 bounds local distortion at admitted writes.

E.6.3 Gate-Based Write Selection

Token-level gates. The token-level comparisons include ungated keyed sparsity, a draft-entropy gate, a draft green-mass gate, and full ChiselMark. Candidate density sets eligibility across the sequence, while the draft green-mass gate decides which eligible writes to admit. Theorem D.10 bounds sequence distortion using the expected number of writes, and Corollary D.8 bounds local distortion at admitted writes. The experiments evaluate the detection and efficiency of the resulting configurations.

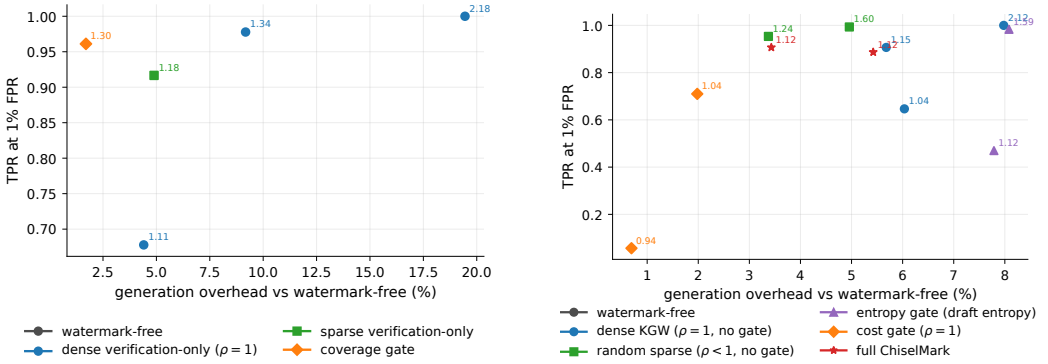


Figure 9: **Left: Held-out EAGLE-1 detectability–efficiency trade-off.** All watermarked configurations preserve the native draft tree. Dense verification-only reweighting reduces proposal reuse, while the coverage gate gives lower overhead at the reported detection rate. **Right: Held-out token-level detectability–efficiency trade-off.** TPR@1% FPR is plotted against overhead relative to watermark-free speculative sampling for dense, sparse, and gated configurations. Accepted tokens per target call are reported in Table 1. Numbers next to markers are PPL ratios. All points use 60 held-out C4 prompts and fixed 192-token continuations.

Feature-level gates. At the feature level, every watermarked method leaves EAGLE-1's native draft tree unchanged. We compare dense verification-only KGW, sparse verification-only KGW, and ChiselMark's verifier-side coverage gate. This comparison isolates whether protecting target-side proposal reuse is necessary even after direct perturbations to draft construction have been removed.

Preserving the tree removes direct topology amplification, but dense target-side reweighting can still reduce reuse of the already-constructed proposal set. The verifier-side coverage gate addresses this second effect. Corollary D.9 shows that, at a fixed verification state, the coverage loss of an admitted write is bounded by the gate threshold:

$$[p(\mathcal{C}) - p_w(\mathcal{C})]_+ \le U\tau_{\text{tree}}.$$

Thus, unlike the token-level gate, whose role is local distributional affordability, the feature-level gate directly protects verifier-side proposal reuse.

E.6.4 Candidate Sites versus Written Sites

The local evidence increment in Eq. (43) depends on the target green-list mass and is largest at intermediate values. Based on the draft green-list mass, our gate decides whether to write or veto each site. Detection then sums the scores over all eligible sites, including vetoed ones. Because the gate depends on the key, written and vetoed sites can exhibit different score distributions. We therefore calculate the evidence per candidate across the full eligible set, as detailed in Section 5.5. Finally, the random-key calibration in Eq. (13) separately controls false positives under its exchangeability assumptions.

E.7 Why the Draft Green-Mass Gate Works

The frozen-state results in Section 5.5 show that the draft green-mass gate selects writes with lower mean positive-part acceptance loss. Here we examine how writes also change the contexts reached during generation. In the tested model pair, every ungated configuration has a negative same-context contribution, and generation shifts toward entropy ranges with lower baseline acceptance. Within each entropy range, the change is positive and offsets the composition loss in all tested settings.

For example, dense watermarking at $\delta = 4$ has total overlap change -0.0089, made up of -0.0160 from same-context reweighting, -0.0174 from entropy composition, and $+0.0245$

from within-entropy changes. The overall effect therefore depends on the balance of these terms and can vary across model–dataset pairs. Candidate density controls how often a write is attempted across the sequence. This complements the gate's decision at each current context and agrees with Theorem D.10, which bounds sequence distortion using the expected number of realized writes. The preceding subsection explains how vetoed candidates affect detection evidence.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- watermarkspeculativecodes.zip