TIDEmark: Joint Token Watermarking for Diffusion Language Models via Multi-Marginal Optimal Transport

Anonymous authors
Paper under double-blind review

Abstract

Watermarks for autoregressive language models typically operate one token at a time. Diffusion language models (DLMs) instead generate text through successive steps that update multiple token positions in parallel. Many existing methods adapt side-information generation to DLMs while leaving the other watermarking components similar to their autoregressive counterparts. In this work, we exploit parallel token generation to watermark multiple tokens jointly, aiming to strengthen detection without changing their individual distributions. To construct a watermark over several token distributions at once, we leverage multi-marginal optimal transport and introduce TIDEmark (**T**ransport-based **I**nteraction for **D**istortion-free **E**mbedding of water**mark**s). TIDEmark coordinates token choices using token-pair watermark scores, and we prove an ideal distortion-free guarantee for the recorded token marginals. We make joint watermarking computationally tractable through a tree-structured interaction graph, combining belief propagation (BP) with the Sinkhorn algorithm for entropy-regularized transport. For fixed candidate-set sizes and a fixed number of side-information states, each bidirectional BP sweep scales linearly with the number of jointly watermarked tokens, avoiding exponential enumeration of token combinations. Experiments on summarization, question answering, and code generation show that TIDEmark can strengthen detection. TIDEmark results suggest that when the watermark is applied can affect the balance between generation quality and watermark detection.

1 Introduction

Text watermarking embeds a detectable signal in model-generated text (Kirchenbauer et al., 2023; Kuditipudi et al., 2023). Text watermarking is gaining practical importance for large-scale language-model services, as illustrated by SynthID-Text in Google's Gemini (Dathathri et al., 2024) and its announced adoption for future versions of Anthropic's Claude (Anthropic, 2026). A detector uses the watermark key to check for that signal in the output. Embedding a stronger signal can change which tokens the model selects and therefore change the content it generates. In code generation, for example, a token change can alter what a program does even when the code remains syntactically valid. The goal is to make watermarked text recognizable to a detector with pre-shared side information, while remaining indistinguishable from unwatermarked text to observers without it. This should be achieved without reducing text quality or task performance.

Diffusion language models can update several token positions in a single inference step, rather than generating text strictly from left to right (Sahoo et al., 2024; Shi et al., 2024; Nie et al., 2026; Sahoo et al., 2025). The model predicts token values, while a planner determines which positions to update (Kim et al., 2025; Ben-Hamu et al., 2026; Luxembourg et al., 2026). Standard masked samplers keep committed tokens fixed, whereas remasking and token-editing procedures can revise earlier choices (Wang et al., 2026; Bie et al., 2026; Team et al., 2026). Different sequences of model predictions and planner decisions can produce the same final text, so a detector cannot uniquely recover the generation trajectory from that text alone. A watermark must therefore leave a signal that remains detectable after revisions and can be checked without access to the intermediate generation steps. Figure 1 outlines the joint watermarking and detection procedure of TIDEmark.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

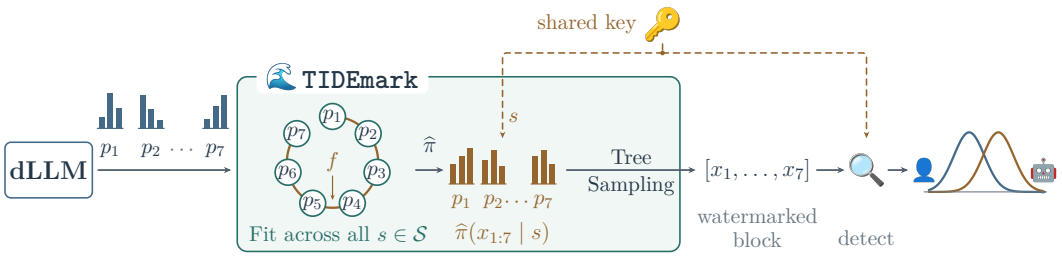

Figure 1: **`TIDEmark`: joint watermarking and detection.** The dLLM supplies recorded token distributions $p_1, \ldots, p_7$. `TIDEmark` uses Sinkhorn with BP to fit a joint coupling across all watermark states, favoring high watermark scores. A key-derived state s selects the conditional distribution used for joint tree sampling. The detector reuses the key and the block's graph structure. The blue and gold curves illustrate unwatermarked and watermarked score distributions, respectively.

Token-level optimal-transport watermarks, including HeavyWater (HW) and SimplexWater, couple each token separately with watermark information (Tsur et al., 2025b; 2026). These couplings can be applied after a block is complete, but delaying their application does not coordinate the token choices. Independent OT already exploits the choices available at each position. Within a block, some positions may admit several plausible tokens, whereas others are tightly constrained. This raises the question of whether joint watermarking can use the available flexibility across token positions more effectively to strengthen detection without sacrificing text quality and task performance. Computational efficiency must also be considered for reasonable generation times.

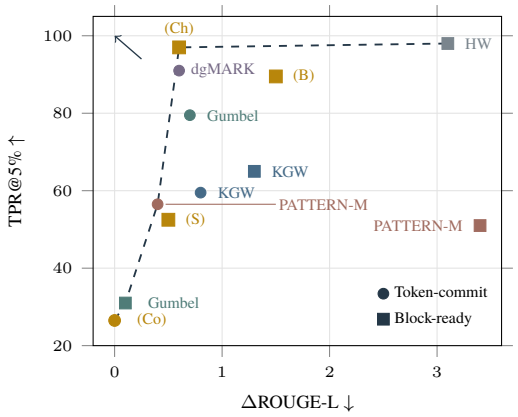

Figure 2: **TPR@5% versus ΔROUGE-L on Multi-News (LLaDA-Instruct).** ΔROUGE-L = clean − watermarked, so upper left is better. Gold denotes `TIDEmark` (ours). The dashed line is the frontier of the included displayed point estimates.

Figure 3 compares joint and separate OT on two fixed token distributions, with per-token scores derived by averaging a common joint score. In this example, joint OT achieves higher expected score gain at comparable conditional KL divergence from the original distributions.

`TIDEmark` watermarks a completed block using a secret key and the token probabilities recorded during generation. It uses multi-marginal optimal transport (MOT) to select the block's tokens jointly rather than applying a separate watermark at each position. In the ideal coupling, averaging over the watermark randomness recovers each recorded token distribution. A detector checks the resulting signal using the key, and final generated tokens. A dense MOT coupling represents every combination of candidate tokens across the block, which is exponentially complex in block size. `TIDEmark` therefore uses a tree-structured interaction graph, allowing message passing and Sinkhorn algorithm to fit the coupling without constructing the full transport tensor (Haasler et al., 2021; Benamou et al., 2015).

This paper makes three contributions:

1. **Joint watermarking for diffusion language models.** We introduce `TIDEmark`, extending token-level optimal transport to jointly watermark completed blocks or token groups selected during denoising, including token revisions.
2. **A theoretical distortion-free guarantee.** We prove that `TIDEmark`'s ideal coupling preserves the recorded token distributions after marginalizing over side information.

3. **Efficient graph-structured computation.** We integrate graph-structured optimal transport into `TIDEmark`, avoiding the full transport tensor. For tree graphs, each bidirectional BP sweep scales linearly in block length for fixed candidate-set sizes and a fixed number of side-information states.

Section 2 reviews token-level watermarking, optimal transport, and diffusion generation. Section 3 introduces joint block-level watermarking and its ideal marginal-preservation guarantee. Section 4 develops the graph-structured formulation, efficient sampling, and detection procedure. Section 5 presents the experimental setup and benchmark results. Section 6 discusses the design trade-offs and concludes the paper. The appendices provide related work, proofs, algorithmic details, reproducibility information, additional results, and limitations.

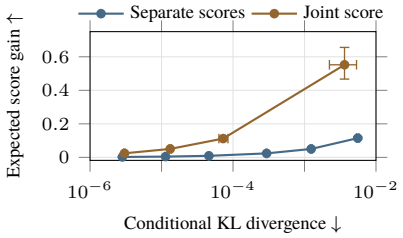

Figure 3: **Joint versus separate scores.** Multimarginal OT uses the joint score, while pairwise OT uses its separate per-token projections. Mean conditional KL divergence (↓, lower is better) against expected score gain over unchanged sampling (↑, higher is better).

2 Background

2.1 Token-level watermarking and optimal transport

In autoregressive language models, a watermark embedder often reweights each token distribution to embed a detectable statistical signal. This signal is based on a given key, allowing a detector with the same key to detect it from a sequence of watermarked tokens (Kirchenbauer et al., 2023; Kuditipudi et al., 2023; Tsur et al., 2026). We consider a next-token distribution p over a vocabulary $\mathcal{X}$.

A watermarking procedure has three components (Tsur et al., 2026): a score function, a side-information generator, and a watermark transformation. We take the side information S to be uniformly distributed over a predetermined finite set $\mathcal{S}$, with s denoting its realized value. In practice, key-derived seeds allow the generator and detector to reproduce this information. The watermark transformation uses s to modify p before sampling a token $x \in \mathcal{X}$. The score function $f(x, s)$ measures the agreement between the token and the side information, and the detector averages these scores over the generated sequence.

In token-level watermarks, a common proxy for output quality preservation is *distortion*, which measures the change in the token distribution after averaging over the side information (Tsur et al., 2026). Let X denote the watermarked token and $P_{X|S}$ its conditional distribution given S. A watermark is distortion-free when $\mathbb{E}_S[P_{X|S}] = p$, with the original next-token distribution p held fixed.

Within the previous framing (Tsur et al., 2026), given a score function $f : \mathcal{X} \times \mathcal{S} \to \mathbb{R}$ and token distribution $P_X = p$, finding a distortion-free watermark amounts to solving an optimal transport problem. We denote by $\Pi_{X,S}$ the set of joint distributions P_{XS} whose marginals are P_X and P_S, where P_S is uniform over $\mathcal{S}$. The OT problem is given by

$$P^{\star}_{XS} \in \underset{P_{XS} \in \Pi_{X,S}}{\arg\max} \left\{ \mathbb{E}_{P_{XS}}[f(X,S)] - \mathbb{E}_{P_X P_S}[f(X,S)] \right\}. \tag{1}$$

Here $P_X P_S$ denotes the independent product of the two marginals. Given side information s, the watermarked token is sampled from $P_{X|S=s}(x) = |\mathcal{S}| P^{\star}_{XS}(x, s)$. In practice, Sinkhorn's algorithm solves an entropy-regularized version of this problem (Cuturi, 2013; Gabriel & Marco, 2019).

While Tsur et al. (2026) explore several score functions, we consider an HW-based watermark, with score values $f(x, s)$ constructed from heavy-tailed random draws.[1]

2.2 Diffusion generation and watermarking

Diffusion generation starts from masked positions in absorbing-state models or uniformly random vocabulary tokens in uniform-state models (Lou et al., 2023; Sahoo et al., 2024; 2025). A block

[1] Following Tsur et al. (2026), we use draws from a LogNormal$(0, 1)$ distribution.

groups token positions processed over successive inference steps, with the denoiser predicting token values and the planner selecting positions to update (Nie et al., 2026; Kim et al., 2025; Ben-Hamu et al., 2026; Luxembourg et al., 2026). Standard unmasking keeps committed tokens fixed, whereas uniform-state sampling and remasking or token-editing procedures allow further revisions (Sahoo et al., 2025; Wang et al., 2026; Bie et al., 2026; Team et al., 2026).

For positionwise watermarking, the watermark transformation acts separately on each selected token distribution, even when several positions are processed in parallel. The detector computes the score $f(x, s)$ from each final token x and its corresponding side information s, reproduced through the specified seed-generation procedure.

Section 3 extends this token-level formulation to completed diffusion blocks, coupling their token distributions with the side information through multi-marginal optimal transport.

3 Jointly Watermarking Diffusion Language Models

A straightforward approach to watermark the output of a diffusion model is to apply a token-level watermarking scheme, with an adaptation to the seed-generation procedure (Wu et al., 2025; Gloaguen et al., 2026; Raban et al., 2026; Hong & No, 2026). One such adaptation is to record each token's distribution at its last commitment during denoising. After the block is generated, the token-level watermarking transform is applied to each recorded distribution *independently*. Detection follows the same score-based approach as in autoregressive models: using the specified seed-generation procedure and required detector inputs, the detector reproduces the side information, aggregates the token scores, and compares the result with a threshold.

However, a block generated in parallel offers the opportunity to coordinate token choices rather than watermark each position independently. We propose a joint block-level watermark based on a natural extension of OT. The joint formulation scores combinations of tokens against the side information, allowing choices at different positions to contribute together to the watermark signal. We investigate whether this additional flexibility strengthens detection while preserving each token's specified distribution after averaging over the side information.

3.1 A joint watermark through multi-marginal optimal transport

Consider a completed token block $(t_1, \dots, t_b)$ before watermarking, generated by the denoising and planning procedure described in Section 2.2, with corresponding token distributions $(p_1, \dots, p_b)$ over $\mathcal{X}$. Our method aims to watermark these distributions jointly.

Our problem naturally falls within the scope of MOT, which is the extension of OT to several marginal distributions. Given b marginal distributions $(p_1, \dots, p_b)$, it seeks a joint distribution over $\mathcal{X}^b$ that preserves these marginals and minimizes a given cost function in expectation. In our setting, the MOT problem is defined on $\mathcal{X}^b \times \mathcal{S}$. That is, it considers $b + 1$ marginal distributions: the b token distributions and the uniform side-information distribution P_S. Each token distribution p_i corresponds to a random variable X_i, with $X^b = (X_1, \dots, X_b)$.

For a given score function $f : \mathcal{X}^b \times \mathcal{S} \to \mathbb{R}$, the MOT problem is given by the following optimization over joint probability distributions π of (X^b, S):

$$\begin{aligned} \pi^\star \in \arg\min_{\pi} \mathbb{E}_\pi[-f(X^b, S)], \\ \text{subject to} \quad \pi_{X_i} = p_i \quad (i = 1, \dots, b), \qquad \pi_S = P_S. \end{aligned} \tag{2}$$

Optimizing this term maximizes the expected value of the score function f. For heavy-tailed scores such as those used in HW (Tsur et al., 2026), restricting f to $[0, 1]$ would require explicit normalization of the scores defining the transport cost $-f$.

The output of the optimization in equation 2 is a joint distribution $\pi^\star$ over $\mathcal{X}^b \times \mathcal{S}$. For a given side-information value s, the watermarked block is sampled jointly from the conditional distribution

$$P_{X^b|S=s}(x^b) = \frac{\pi^\star(x^b, s)}{P_S(s)} = |\mathcal{S}|\, \pi^\star(x^b, s). \tag{3}$$

The corresponding watermarked token distributions can be obtained by marginalizing this conditional distribution over the other token positions.

We refer to the resulting watermarking procedure as `TIDEmark`. `TIDEmark` follows a simple set of steps. Generation and detection use the same score function and seed-generation procedure. Upon watermarking a block, we collect the recorded token distributions of the generated block, solve the corresponding MOT problem, and sample jointly from the corresponding conditional block distribution. The detection procedure follows the same score-based approach: the detector aggregates the block scores and compares the result with a threshold. In practice, an entropy-regularized version of the MOT problem is solved with Sinkhorn's algorithm (see Section 4 for more details).

`TIDEmark`'s optimal joint coupling preserves each recorded token distribution after averaging over shared side information. This distortion-free guarantee applies to recorded token marginals, not the full joint text distribution, whose explicit representation is intractable.

Proposition (`TIDEmark` is distortion free). Fix the recorded token distributions $p_1, \ldots, p_b$ and a coupling $\pi^\star$ satisfying the token and side-information marginal constraints in equation 2. Draw one shared state $S \sim P_S$, then sample X^b from the corresponding conditional distribution in equation 3. Averaging the position-i marginal of the sampled block distribution over the shared side information recovers the recorded token distribution:

$$\mathbb{E}_{S\sim P_S}\left[\sum_{\substack{x^b\in\mathcal{X}^b\\ x_i=x}} \frac{\pi^\star(x^b,S)}{P_S(S)}\right] = p_i(x), \qquad i=1,\ldots,b, \quad x\in\mathcal{X}. \tag{4}$$

Proof in Appendix B.2.

4 Efficient block-level watermarking

4.1 The cost of block-level watermarks

While offering a strong watermarking alternative that leverages block-level information, a straightforward MOT does not scale well with the block size. A dense representation of the joint token-state coupling contains $|\mathcal{S}|\,|\mathcal{X}|^b$ entries. Sinkhorn updates that explicitly process this tensor therefore incur a cost exponential in the block size. While top-p or top-k sampling reduces the number of candidate tokens at each position Tsur et al. (2026), the exponential dependence on b limits us to very small block sizes. In what follows, we propose a modification to Sinkhorn's algorithm that reduces the exponential dependence, which enables block-level watermarks at large block sizes.

4.2 Complexity reduction through graph decompositions

To address the exponential dependence in b, we utilize a simplification of Sinkhorn's algorithm that builds upon sparse graph structure Altschuler & Boix-Adsera (2023). To that end, we introduce graph representations. We focus on *pairwise-decomposable* score functions that decompose into pairwise token score factors:

$$f(X^b,S) = \sum_{(i,j)\in\mathcal{I}} f(X_i,X_j,S), \tag{5}$$

where $\mathcal{I}$ is a subset of the pairs of distinct indices in $\{1,\ldots,b\}$, each included once. A pairwise-decomposable score can be equivalently represented with a computational graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$.[2]

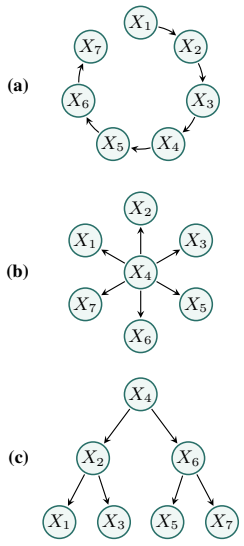

Figure 4: Illustrative token-interaction graphs for a fixed shared watermark state: (a) chain, (b) middle-star, and (c) balanced tree. All panels use the same seven positions. Arrows indicate parent-to-child sampling from the conditional joint block distribution.

[2]The BP-based Sinkhorn implementation for MOT requires a tree-structured token interaction graph for each fixed side-information value, so BP can efficiently compute the required marginals Haasler et al. (2021).

Each node represents a token position, and each pair $(i, j) \in \mathcal{I}$ corresponds to an edge $\{i, j\} \in \mathcal{E}$. The pair ordering specifies the arguments of $f(X_i, X_j, S)$, while BP passes messages in both directions along each edge.

Under this representation, including a cost term for every token pair yields a dense graph in which every pair of nodes is connected.

To construct $f(x_i, x_j, s)$, we extend the HW score construction from single tokens to token pairs (Tsur et al., 2026). Reproducible heavy-tailed draws assign a score to each token pair and side-information value.

As noted in previous work (Altschuler & Boix-Adsera, 2023; Tsur et al., 2025a), the complexity of the corresponding Sinkhorn solver is directly connected to the graph's structure. Specific graph structures introduce sparsity in the interactions represented by the cost tensor, allowing the required marginalizations to be computed without constructing the full tensor. For tree structures, the cost per Sinkhorn iteration consequently becomes polynomial in the block size (see Appendix B.7 for an operation-count analysis).

Algorithm 1 Joint block generation

Inputs: Distributions $p_1, \ldots, p_b$, secret key κ
Outputs: Watermarked block $x_1, \ldots, x_b$

$\widehat{\pi} \leftarrow$ SINKHORN-BP$(p_1, \ldots, p_b)$
$s \leftarrow$ SIDEINFORMATION(κ)
$(x_1, \ldots, x_b) \leftarrow$ TREESAMPLE$(\widehat{\pi}, s)$
return $x_1, \ldots, x_b$

Algorithm 2 Watermark detection

Inputs: Observed generated tokens x, secret key κ
Outputs: Score z_{eq} and detection decision

$G \leftarrow |x|, \quad L \leftarrow \lceil G/b \rceil$
$\{x^{(k)}\}_{k=1}^{L} \leftarrow$ PARTITION(x, b)
for $k = 1, \ldots, L$ **do**
  $s_k \leftarrow$ SIDEINFORMATION(κ)
  $r_k \leftarrow \sum_{(c,p)\in\mathcal{E}_k} f(x_{k,c}, x_{k,p}, s_k)$
end for
$T_{\text{obs}} \leftarrow G^{-1} \sum_{k=1}^{L} r_k$
$p \leftarrow p_0(T_{\text{obs}}; x)$ †
$z_{\text{eq}} \leftarrow \Phi^{-1}(1 - p)$
return z_{eq}, $(p < \alpha)$

† $p_0(t; x)$ is the analytical upper-tail probability under the declared null hypothesis. Details in Appendix B.5.

In this work, we consider three tree structures on a given block: a *chain*, a *middle-star*, and a *balanced tree* (Figure 4). The chain connects consecutive token positions, with arrows pointing from the first position toward the last. The middle-star directs edges from the middle position to every other position. The balanced tree directs edges from the middle position to the middle positions of the left and right portions, repeating this construction within each portion. When there are two middle positions, the earlier one is used. Each structure connects all b token positions with $b - 1$ edges. The structures change which token pairs interact directly while keeping the token and shared-state marginal constraints fixed.

4.3 Fitting and sampling the coupling

Our optimization solves an entropy-regularized MOT problem while retaining the prescribed token and side-information marginal constraints. The full objective is given in Appendix B.3. For regularization strength $\varepsilon > 0$, the optimal coupling on the prescribed supports admits the factorized form (Haasler et al., 2021)

$$\pi_\varepsilon^\star(x^b, s) = v(s) \prod_{i=1}^{b} u_i(x_i) \prod_{(i,j)\in\mathcal{I}} \exp\left(\frac{f(x_i, x_j, s)}{\varepsilon}\right). \tag{6}$$

The scaling factors u_i and v enforce the prescribed marginals, with normalization absorbed into these factors.

BP and Sinkhorn. To determine the scaling factors u_i and v, Sinkhorn alternates marginal computation with scaling updates. For each fixed side-information value s, belief propagation (BP; Yedidia et al., 2003) passes messages in both directions along the tree, computing the conditional token marginals and the total weight associated with that state. Combining these quantities across s gives the current token and side-information marginals without enumerating all token combinations. Sinkhorn then uses these marginals to update the scaling factors toward the prescribed distributions (Haasler et al., 2021; Benamou et al., 2015).

The graph and pair factors remain fixed during fitting. BP evaluates the marginals using the current scaling factors, and each scaling update targets one prescribed marginal. Fitting stops when the marginal error is within a prescribed tolerance or the maximum number of iterations is reached.

Tree sampling. Given side information s, the sampler draws a block from the conditional distribution $\widehat{\pi}(X^b \mid S = s)$. It samples the starting token from its conditional marginal, then each remaining token conditional on its sampled parent. BP supplies these conditional probabilities.

In Algorithm 1, p_i is the recorded distribution at token position $i = 1, \ldots, b$. In Algorithms 1 and 2, κ denotes the secret key. The side-information generator uses this key with the text context specified by the seeding procedure. Appendix B.4 details Sinkhorn-BP, BP-Marginals, and tree sampling in Algorithms 3, 4, and 5, respectively.

4.4 Detecting the watermark

The detector uses the same pair-score construction, block graphs, and seed-generation procedure as the generator. Following the score-based detection approach of HW (Tsur et al., 2026), it sums pair scores over all observed edges and normalizes by the generated-token count.

Let $G > 0$ be the observed generated-token count, b the block length, and $L = \lceil G/b \rceil$ the number of blocks. Token i in block k is denoted by $x_{k,i}$, $\mathcal{E}_k$ is that block's edge set, and s_k is its reproduced watermark state. Assume $b > 1$. The score is

$$T_{\text{obs}} = \frac{1}{G} \sum_{k=1}^{L} \underbrace{\sum_{(c,p)\in\mathcal{E}_k} f(x_{k,c}, x_{k,p}, s_k)}_{\text{sum of pair scores in block } k} . \tag{7}$$

The analytical null model assumes that block states are uniform, mutually independent, and independent of the observed tokens, with the score construction held fixed. The detector rejects the null when $p_0(T_{\text{obs}}; x) < \alpha$, where p_0 is the upper-tail probability under this null model and α is the chosen significance level. The analytical tail and its Gaussian-equivalent reporting scale are detailed in Appendix B.5, with detection pseudocode in Algorithm 2.

4.5 Beyond block-ready watermarking

Block-ready watermarking changes token choices after denoising has finished for that block, leaving no further within-block updates to restore generation coherence after watermarking. We therefore also apply `TIDEmark` within the denoising loop, updating only the k_t positions selected at step t, including newly unmasked positions and revisions. Thus, subsequent denoising steps can condition on the watermarked tokens. The detector does not observe the commit-time graph and instead aggregates scores over all earlier-to-later token pairs within each block. Scoring pairs that were not directly coupled during generation can dilute the watermark signal. Appendix B.6 provides further details.

5 Results

5.1 Experimental setup

The evaluation examines watermark detection, generation quality, task performance, and computational cost using LLaDA 1.0-Instruct (Nie et al., 2026) and LLaDA 2.1-mini (Bie et al., 2026).

The baselines include position-seeded KGW and Gumbel (Kirchenbauer et al., 2023; Gloaguen et al., 2026; Wu et al., 2025; Bagchi et al., 2025). Additional baselines are dgMARK (Hong & No, 2026), PATTERN-MARK (Chen et al., 2025), and HW (Tsur et al., 2026), alongside unwatermarked controls. The tables report token-commit and block-ready results for KGW, Gumbel, PATTERN-MARK, and HW, with coverage varying by model and task. dgMARK is evaluated at token commitment.

Token-commit watermarking is applied during denoising, before selected tokens are committed, while block-ready watermarking is applied after native block decoding is complete. Unwatermarked generations provide a reference for generation quality. Denoiser and planner parameters are aligned, subject to method-specific differences. The benchmarks include the commit-time configuration described in Section 4.5, which applies `TIDEmark` to the positions selected at each denoising step.

Table 1: **LCC detection and code quality.** `TIDEmark` variants are (Co) commit, (Ch) chain, (S) middle-star, and (B) balanced tree. Best and second-best watermarked values are bold and underlined.

Method	TPR@1 (%)	TPR@5 (%)	TPR@10 (%)	AUC	Mean $-\log_{10} p$	CE	Code sim.
Clean reference	–	–	–	–	–	0.706	58.0
Token-commit							
KGW	5.0	15.0	28.5	0.652	0.814	0.976	55.8
PATTERN-M	16.0	27.5	38.0	0.693	0.853	0.852	**58.5**
dgMARK	13.5	29.0	40.5	0.770	0.975	**0.757**	55.5
Gumbel	12.0	27.0	39.0	0.761	1.260	0.842	56.6
HW	13.0	19.5	29.0	0.705	1.014	0.894	57.2
`TIDEmark` (Co)	4.8	13.8	22.9	0.653	0.685	–	56.5
Block-ready							
KGW	30.3	41.0	56.9	0.858	1.262	1.529	53.4
Gumbel	6.4	13.3	30.9	0.641	0.702	0.916	57.3
PATTERN-M	11.2	29.8	40.4	0.741	0.875	2.241	51.7
HW	17.0	37.0	50.5	0.766	1.319	–	54.7
`TIDEmark` (Ch)	37.8	69.1	**79.8**	0.917	1.621	1.439	55.1
`TIDEmark` (S)	32.4	60.6	69.7	0.871	1.481	1.213	56.6
`TIDEmark` (B)	**57.4**	**76.1**	**79.8**	**0.925**	**1.740**	1.469	54.1

(a) LLaDA-Instruct

Method	TPR@1 (%)	TPR@5 (%)	TPR@10 (%)	AUC	Mean $-\log_{10} p$	CE	Code sim.
Clean reference	–	–	–	–	–	0.627	40.5
Token-commit							
KGW	51.1	85.6	88.8	0.960	1.849	1.123	40.9
Gumbel	2.1	21.3	30.9	0.676	0.762	**0.648**	**42.3**
HW	**98.4**	**98.4**	**98.4**	**0.993**	**2.155**	4.875	28.1
`TIDEmark` (Co)	34.6	45.7	48.9	0.758	1.179	1.094	38.4
Block-ready							
KGW	82.4	93.6	94.7	0.980	2.039	2.479	31.3
Gumbel	12.8	36.2	47.3	0.766	1.022	0.994	37.1
HW	96.3	**98.4**	**98.4**	**0.993**	2.146	5.086	26.5
`TIDEmark` (Ch)	11.7	38.8	55.9	0.789	1.074	0.919	39.1
`TIDEmark` (S)	38.3	50.5	56.9	0.848	1.262	0.871	38.9
`TIDEmark` (B)	21.8	40.4	53.7	0.804	1.136	0.916	39.5

(b) LLaDA 2.1

The evaluation examines the detection-quality tradeoff of `TIDEmark` across models and tasks, including its token-commit configuration and its chain, middle-star, and balanced-tree block-ready configurations. Code-generation benchmarks comprise LCC (Guo et al., 2023) and HumanEval (Chen et al., 2021). Question answering and summarization use WaterBench's (Tu et al., 2024) Finance QA and Multi-News (Fabbri et al., 2019) tasks, respectively. Tables 1 and 2 present the LCC and Multi-News results.

Detection is reported using AUROC and true-positive rates at empirical thresholds targeting 1%, 5%, and 10% false-positive rates (TPR@1, TPR@5, and TPR@10), while thresholds are selected and evaluated on the same clean set. We also report $-\log_{10}(p)$ averaged over texts, where p is the empirical p-value for each text. Higher values indicate stronger evidence of watermarking.

Generation quality is measured by completion-only cross-entropy (CE), scored by quantized LLaMA-3.3-70B-Instruct and averaged in nats per token, where lower is better. Task performance uses HumanEval pass@1, LCC code similarity, and ROUGE-L F1 for Finance QA and Multi-News.

Appendix B.6 details commit-time generation and detection. Appendix C provides experimental configurations (C.1), robustness results (C.2), the regularization tradeoff (C.3), runtime measurements (C.4), additional HumanEval and Finance QA results (C.5), and limitations (C.6).

5.2 Code-generation benchmarks

The code-generation benchmarks test joint block watermarking in a constrained domain, where low-entropy token predictions can make watermarking more challenging than in open-ended text generation. Table 1 reports LCC detection and code quality for LLaDA 1.0 and 2.1; HumanEval results appear in Appendix Table 7.

On LCC with LLaDA-Instruct, `TIDEmark` Chain improves the detection-quality balance relative to block-ready KGW at the selected operating points (Table 1): it reaches 69.1% rather than 41.0% TPR at a target 5% FPR, with lower CE and a smaller code-similarity decrease from the clean reference, 2.9 rather than 4.6 points. Balanced tree raises TPR further to 76.1%, but sacrifices some of Chain's code similarity. On the other hand, middle-star achieves the highest TPR among the selected `TIDEmark` configurations on LLaDA 2.1 at this threshold, reaching 50.5% with a code-similarity decrease of 1.6 points. Its detection rate remains below those of block-ready KGW and HW, whose stronger detection accompanies larger code-quality losses. Within this block-ready comparison, `TIDEmark` therefore offers operating points that retain more code similarity while maintaining a detectable watermark signal.

HumanEval complements the code-similarity evaluation with functional correctness measured by pass@1, with detailed results and analysis in Appendix C.5.

Table 2: **Multi-News detection and text quality.** TIDEmark variants are (Co) commit, (Ch) chain, (S) middle-star, and (B) balanced tree. Best and second-best watermarked values are bold and underlined.

Method	TPR@1 (%)	TPR@5 (%)	TPR@10 (%)	AUC	Mean $-\log_{10} p$	CE	ROUGE L
Clean reference	–	–	–	–	–	0.596	26.9
Token-commit							
KGW	39.5	59.5	65.5	0.864	1.387	0.948	26.1
Gumbel	63.5	79.5	83.0	0.931	1.770	0.858	26.2
dgMARK	83.0	91.0	95.0	0.974	1.998	0.988	26.3
PATTERN-M	14.5	56.5	60.5	0.834	1.235	0.882	26.5
TIDEmark (Co)	19.5	26.5	37.0	0.749	0.945	0.922	**26.9**
Block-ready							
KGW	47.0	65.0	77.5	0.905	1.531	1.003	25.6
Gumbel	20.5	31.0	40.5	0.740	0.948	**0.678**	26.8
PATTERN-M	20.5	43.0	59.5	0.820	1.165	1.717	23.5
HW	**97.0**	**98.0**	**98.0**	**0.994**	**2.162**	1.777	23.8
TIDEmark (Ch)	71.0	97.0	97.5	0.985	1.833	0.724	26.3
TIDEmark (S)	44.5	52.5	66.5	0.837	1.270	0.718	26.4
TIDEmark (B)	86.0	89.5	91.0	0.973	1.822	0.978	25.4

(a) LLaDA-Instruct

Method	TPR@1 (%)	TPR@5 (%)	TPR@10 (%)	AUC	Mean $-\log_{10} p$	CE	ROUGE L
Clean reference	–	–	–	–	–	1.418	24.7
Token-commit							
KGW	96.0	**99.5**	**100.0**	0.999	–	1.719	24.4
Gumbel	13.5	50.5	60.5	0.839	–	**1.436**	**24.8**
TIDEmark (Co)	36.5	56.5	63.0	0.864	1.354	1.793	24.7
Block-ready							
KGW	**99.5**	**99.5**	**100.0**	**1.000**	**2.200**	3.199	23.7
Gumbel	26.5	56.0	67.0	0.885	1.359	1.752	24.6
TIDEmark (Ch)	50.0	74.0	84.5	0.941	1.667	1.961	24.5
TIDEmark (S)	99.0	**99.5**	99.5	0.999	2.191	2.805	23.7
TIDEmark (B)	62.0	71.0	82.5	0.935	1.652	1.959	24.6

(b) LLaDA 2.1

5.3 Question answering and summarization

Finance QA and Multi-News evaluate question answering and multi-document summarization, respectively. Table 2 reports Multi-News detection and text quality for LLaDA 1.0 and 2.1; Finance QA results appear in Appendix Table 8.

On Multi-News, TIDEmark's selected block-ready configurations illustrate both the opportunity and the cost of joint watermarking (Table 2). With LLaDA-Instruct, Chain achieves 97.0% TPR at a target 5% FPR while reducing ROUGE-L by 0.6 points relative to the clean reference, with CE of 0.724. Strong detection therefore accompanies a limited change in summarization performance at this operating point. The balance is less favorable on LLaDA 2.1: middle-star reaches 99.5% TPR at the same target FPR, but ROUGE-L falls by 1.0 point and CE rises to 2.805. Chain and the balanced tree retain more of the clean reference's task performance, but detect fewer watermarked texts. On LLaDA 2.1, token-commit KGW matches block-ready TIDEmark middle-star's 99.5% TPR@5%, with lower CE and a smaller ROUGE-L decrease, highlighting the importance of model-specific configuration when balancing detection and generation quality.

Finance QA shows a similar detection-quality tradeoff across TIDEmark configurations, with detailed results and analysis in Appendix C.5.

6 Discussion and Conclusions

The results show how TIDEmark can use joint block watermarking to strengthen detection while retaining task performance at selected operating points. On LLaDA-Instruct, Chain combines strong Multi-News detection with a small ROUGE-L decrease and offers a better detection-quality balance than block-ready KGW on LCC. The other evaluated settings show why detection cannot be considered alone: configurations with stronger signals can also incur larger task-performance losses, particularly on code.

Block-ready TIDEmark applies the watermark once per completed block, whereas token-commit TIDEmark allows subsequent denoising to condition on watermarked tokens but requires repeated watermarking calls. Across the evaluated configurations, block-ready TIDEmark often achieves stronger detection than its token-commit counterpart at the cost of lower task performance, exposing a tradeoff between watermark detectability and generation quality.

TIDEmark extends optimal-transport watermarking from individual tokens to jointly sampled blocks. Its graph-structured formulation makes joint watermarking tractable while preserving recorded token marginals in the ideal distortion-free formulation. The results show that TIDEmark can strengthen detection, with a tradeoff between watermark detection and generation quality.

AI Use Statement

In this work, we used generative AI tools solely to assist with drafting and editing manuscript text. We did not use generative AI tools for any research task requiring disclosure under the ICLR 2027 policy, including ideation, theoretical development, formulation or proof of mathematical claims, experimental design, method implementation, data generation or preparation, analysis, or interpretation of results. All AI-assisted text was reviewed and revised by the authors. All scientific claims, technical content, and results were produced and verified by the authors. We take responsibility for the final content of this work.

Reproducibility Statement

The reproducibility materials are organized to separate theoretical and empirical details. Appendix B provides derivations, algorithms, and numerical procedures supporting the method. Appendix C records the watermark mechanisms, seeding rules, experimental configurations, additional ablations, full detection and quality results, and scope limitations.

References

Jason M Altschuler and Enric Boix-Adsera. Polynomial-time algorithms for multimarginal optimal transport problems with structure. *Mathematical Programming*, 199(1):1107–1178, 2023.

Anthropic. How Claude's text watermarking works, 2026. URL `https://www.anthropic.com/news/claude-text-watermark`.

Avi Bagchi, Akhil Bhimaraju, Moulik Choraria, Daniel Alabi, and Lav R Varshney. Watermarking discrete diffusion language models. *arXiv preprint arXiv:2511.02083*, 2025.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *Advances in Neural Information Processing Systems*, 38:55981–56007, 2026.

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015.

Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, et al. Llada2. 1: Speeding up text diffusion via token editing. *arXiv preprint arXiv:2602.08676*, 2026.

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.

Ruibo Chen, Yihan Wu, Yanshuo Chen, Chenxi Liu, Junfeng Guo, and Heng Huang. A watermark for order-agnostic language models. In *International Conference on Learning Representations*, volume 2025, pp. 4372–4388, 2025.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

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.

Alexander Richard Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1074–1084, 2019.

Jiaojiao Fan, Isabel Haasler, Johan Karlsson, and Yongxin Chen. On the complexity of the optimal transport problem with graph-structured cost. In *International conference on artificial intelligence and statistics*, pp. 9147–9165. PMLR, 2022.

Peyré Gabriel and Cuturi Marco. Computational optimal transport with applications to data sciences. *Foundations and trends® in machine learning*, 11(5-6):355–607, 2019.

Eva Giboulot and Teddy Furon. Watermax: breaking the llm watermark detectability-robustness-quality trade-off. *Advances in neural information processing systems*, 37:18848–18881, 2024.

Thibaud Gloaguen, Robin Staab, Nikola Jovanović, and Martin Vechev. Watermarking diffusion language models. In *International Conference on Learning Representations*, volume 2026, pp. 100104–100149, 2026.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. Longcoder: A long-range pre-trained language model for code completion. In *International Conference on Machine Learning*, pp. 12098–12107. PMLR, 2023.

Isabel Haasler, Axel Ringh, Yongxin Chen, and Johan Karlsson. Multimarginal optimal transport with a tree-structured cost and the schrodinger bridge problem. *SIAM Journal on Control and Optimization*, 59(4):2428–2453, 2021.

Pyo Min Hong and Albert No. dgmark: Decoding-guided watermarking for diffusion language models. *arXiv preprint arXiv:2601.22985*, 2026.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*, 2025.

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

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the reliability of watermarks for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 49660–49704, 2024.

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

Baixin Li and Haiyun He. Sac-copula: Quality-preserving watermarking for diffusion language models via smooth correlated gumbel fields. *arXiv preprint arXiv:2608.20839*, 2026.

Yukang Lin, Jiahao Shao, Shuoran Jiang, Wentao Zhu, Bingjie Lu, Xiangping Wu, Joanna Siebert, and Qingcai Chen. Watersearch: Exploring seed pooling for improving the quality-detectability trade-off in llm watermarking. *arXiv preprint arXiv:2512.00837*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Omer Luxembourg, Haim H. Permuter, and Eliya Nachmani. Plan for speed: Dilated scheduling for masked diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL `https://openreview.net/forum?id=DbzAsMRNhy`.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

Ofek Raban, Gal Chechik, and Ethan Fetaya. Lr-dwm: Efficient watermarking for diffusion language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 43510–43517, 2026.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. *Proceedings of machine learning research*, 267: 52584, 2025.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167, 2024.

DiffusionGemma Team, Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, et al. Diffusiongemma technical report. *arXiv preprint arXiv:2608.00146*, 2026.

Dor Tsur, Ziv Goldfeld, Kristjan Greenewald, and Haim Permuter. Neural estimation for scaling entropic multimarginal optimal transport. *arXiv preprint arXiv:2506.00573*, 2025a.

Dor Tsur, Carol Xuan Long, Claudio Mayrink Verdun, Hsiang Hsu, Haim Permuter, and Flavio P Calmon. Optimized couplings for watermarking large language models. *arXiv preprint arXiv:2505.08878*, 2025b.

Dor Tsur, Carol Long, Claudio Mayrink Verdun, Sajani Vithana, Hsiang Hsu, Richard Chen, Flavio Calmon, et al. Heavywater and simplexwater: Distortion-free llm watermarks for low-entropy distributions. *Advances in Neural Information Processing Systems*, 38:107204–107256, 2026.

Shangqing Tu, Yuliang Sun, Yushi Bai, Jifan Yu, Lei Hou, and Juanzi Li. Waterbench: Towards holistic evaluation of watermarks for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1517–1542, 2024.

Guanghan Wang, Yair Schiff, Subham Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. *Advances in Neural Information Processing Systems*, 38: 147282–147339, 2026.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *International Conference on Learning Representations*, volume 2026, pp. 57027–57051, 2026.

Linyu Wu, Linhao Zhong, Wenjie Qu, Yuexin Li, Yue Liu, Shengfang Zhai, Chunhua Shen, and Jiaheng Zhang. Dmark: Order-agnostic watermarking for diffusion large language models. *arXiv preprint arXiv:2510.02902*, 2025.

Jonathan S Yedidia, William T Freeman, Yair Weiss, et al. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8(236-239):0018–9448, 2003.

A Related Work

Coupling-based watermarking. Watermarking can be formulated as a coupling between a generated token and secret side information, with prescribed marginals controlling the token distribution (Tsur et al., 2025b). HeavyWater and SimplexWater develop this approach for low-entropy next-token distributions through score-based optimal transport (Tsur et al., 2026). They use different watermark score constructions to fit token-side-information couplings and detect the resulting dependence between the generated text and the secret. Both methods apply these couplings token by token during autoregressive generation.

Context-aware watermarking. Several diffusion watermarks adapt KGW-style green-list biasing to partially revealed contexts (Kirchenbauer et al., 2023). DMark constructs keyed token-pair preferences from revealed neighbors, with variants that also use predictions for masked neighbors (Wu et al., 2025). LR-DWM derives separate green-list biases from revealed left and right neighbors, using distinct keys for the two directions (Raban et al., 2026). WM-DLM accounts for unresolved context by averaging context-dependent watermark biases over context uncertainty (Gloaguen et al., 2026).

Watermark signal construction. Watermark signals can take the form of vocabulary-color patterns or pseudorandom sampling perturbations. PATTERN-MARK uses a key-generated sequence of vocabulary colors to assign watermark preferences to output positions, allowing tokens to be generated in arbitrary order (Chen et al., 2025). Pseudorandom Gumbel perturbations indexed by token position guide token sampling during diffusion (Bagchi et al., 2025). SAC-Copula introduces dependence between these perturbations through a Gaussian copula, producing a spatially correlated Gumbel field rather than independent noise across positions (Li & He, 2026). Rather than modifying token logits, dgMARK embeds the signal through the commitment schedule, selecting positions using parity matching between token identities and positions (Hong & No, 2026).

Candidate-selection watermarking. Watermarking can also operate through selection among generated chunks. WaterMax retains candidates with strong keyed watermark scores (Giboulot & Furon, 2024), while WaterSearch generates continuations under different watermark seeds and selects among them using both fidelity and watermark strength (Lin et al., 2025).

Graph-based Optimal Transport. Graph-structured costs enable multimarginal optimal transport algorithms that combine marginal scaling with BP, including tree-based methods and junction-tree extensions (Haasler et al., 2021; Fan et al., 2022). Polynomial-time algorithms have also been established for structured classes of multimarginal transport problems (Altschuler & Boix-Adsera, 2023). A complementary approach, Neural Entropic MOT (NEMOT), uses neural networks trained on mini-batches to estimate entropic multimarginal transport, with non-asymptotic error bounds (Tsur et al., 2025a).

B Theoretical Details

B.1 Notation

We use uppercase letters for random variables and lowercase letters for their realizations. The following table summarizes the notation used in the formulation, solver, and detector.

B.2 Proof of marginal preservation

This section restates the marginal-preservation guarantee in Equation 4 and proves it for the conditional sampling procedure of `TIDEmark`.

Proposition (`TIDEmark` is distortion free). Fix the recorded token distributions $p_1, \ldots, p_b$ and a coupling $\pi^\star$ satisfying the token and side-information marginal constraints in equation 2. Draw one shared state $S \sim P_S$, then sample X^b from the corresponding conditional distribution in equation 3. Averaging the position-i marginal of the sampled block distribution over the shared side information

Table 3: **Notation.** Symbols used in the formulation, solver, and detector.

Symbol	Meaning
Tokens and watermark coupling	
$\mathcal{X}$	Token vocabulary.
b	Number of token positions in the block formulation; nominal block length when partitioning a sequence.
t_i	Token at position i before watermarking.
p_i	Recorded token distribution at position i.
$X_i,\ x_i$	Watermarked token random variable and its realization.
$X^b,\ x^b$	Tuple of b watermarked token variables and its realization.
$A_i = \mathrm{supp}(p_i)$	Candidate support at position i.
$\mathcal{S},\ S,\ s$	Finite side-information set, state random variable, and realized state.
P_S	Prescribed side-information distribution, uniform over $\mathcal{S}$.
κ	Secret watermark key.
$\pi,\ \pi^\star$	Joint token-state coupling and an optimizer of the unregularized MOT objective.
$\pi^\star_\varepsilon$	Optimal entropy-regularized coupling.
$\widehat{\pi}$	Current or fitted numerical coupling.
ε	Entropy-regularization strength.
$H(\pi)$	Entropy of the joint coupling.
Graph and solver	
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Token interaction graph, its nodes, and its edges.
$\mathcal{I}$	Index pairs included in the score decomposition.
$f(x^b, s)$	Block watermark score.
$f(x_i, x_j, s)$	Token-pair score contributing to the block score.
$\psi_{ij}(x_i, x_j, s)$	Pair kernel $\exp(f(x_i, x_j, s)/\varepsilon)$.
$u_i,\ v$	Token and side-information scaling factors.
$q_i,\ q_S$	Token and side-information marginals of the current coupling.
$\mathrm{ne}(i)$	Neighbors of node i.
$m_{i\to j}(x_j, s)$	BP message from node i to node j.
$\delta,\ \tau$	Current maximum marginal error and stopping tolerance.
Detection	
$x,\ G = \lvert x\rvert$	Observed generated sequence and its token count.
$L = \lceil G/b \rceil$	Number of observed blocks.
b_k	Observed token count in block k.
$\mathcal{E}_k$	Scored edges in block k.
s_k	Reproduced watermark state for block k.
$h_k(s; x)$	Block-k pair-score sum evaluated at state s.
r_k	Block-k pair-score sum evaluated at the reproduced state.
T_{obs}	Observed sequence statistic, $G^{-1}\sum_k r_k$.
T_0	Corresponding random statistic under the analytical null.
$p_0(t; x)$	Inclusive analytical upper-tail probability at threshold t, conditional on x.
α	Detection significance threshold.
$\Phi,\ z_{\mathrm{eq}}$	Standard normal CDF and Gaussian-equivalent score $z_{\mathrm{eq}} = \Phi^{-1}(1 - p_0)$.

recovers the recorded token distribution:

$$\mathbb{E}_{S\sim P_S}\left[\sum_{\substack{x^b\in\mathcal{X}^b\\ x_i=x}} \frac{\pi^\star(x^b, S)}{P_S(S)}\right] = p_i(x), \qquad i = 1, \ldots, b, \quad x \in \mathcal{X}.$$

Proof. Hold the recorded distributions $p_1, \ldots, p_b$ and the feasible coupling $\pi^\star$ fixed before drawing the shared state $S \sim P_S$. Since P_S is uniform on finite nonempty $\mathcal{S}$, $P_S(s) = 1/|\mathcal{S}| > 0$.

The conditional in Equation 3 is

$$P_{X^b|S=s}(x^b) = \frac{\pi^\star(x^b, s)}{P_S(s)}.$$

The state-marginal constraint ensures normalization:

$$\sum_{x^b \in \mathcal{X}^b} \frac{\pi^\star(x^b, s)}{P_S(s)} = \frac{P_S(s)}{P_S(s)} = 1.$$

For any position i and token x:

$$\begin{aligned}
\Pr(X_i = x) &= \mathbb{E}_{S\sim P_S}\left[\sum_{\substack{x^b \in \mathcal{X}^b \\ x_i = x}} \frac{\pi^\star(x^b, S)}{P_S(S)}\right] \\
&= \sum_{s\in\mathcal{S}} P_S(s) \sum_{\substack{x^b \in \mathcal{X}^b \\ x_i = x}} \frac{\pi^\star(x^b, s)}{P_S(s)} \\
&= \sum_{s\in\mathcal{S}} \sum_{\substack{x^b \in \mathcal{X}^b \\ x_i = x}} \pi^\star(x^b, s) \\
&= p_i(x).
\end{aligned}$$

The factors $P_S(s)$ cancel, and the final equality follows from the token-marginal constraint in Equation 2. Thus, averaging over the shared side information recovers each recorded token distribution:

$$\Pr(X_i = x) = p_i(x), \qquad i = 1, \dots, b, \quad x \in \mathcal{X}.$$

B.3 Entropy-regularized multi-marginal optimal transport

For a general score $f : \mathcal{X}^b \times \mathcal{S} \to \mathbb{R}$ and $\varepsilon > 0$, the entropy-regularized MOT problem is (Benamou et al., 2015; Tsur et al., 2026)

$$\begin{aligned}
\pi^\star_\varepsilon &= \arg\min_\pi \left\{\mathbb{E}_\pi[-f(X^b, S)] - \varepsilon H(\pi)\right\}, \\
\text{subject to} \quad \pi_{X_i} &= p_i \quad (i = 1, \dots, b), \qquad \pi_S = P_S,
\end{aligned} \tag{8}$$

where the optimization is over joint probability distributions and

$$H(\pi) = -\sum_{x^b \in \mathcal{X}^b} \sum_{s\in\mathcal{S}} \pi(x^b, s) \log \pi(x^b, s). \tag{9}$$

Regularization changes the objective, not the marginal constraints.

B.4 Sinkhorn-BP and conditional tree sampling

Algorithm 3 fits the graph-based OT coupling: BP computes the current token and side-information marginals, and Sinkhorn updates the scaling factors to bring them toward their targets, $p_1, \dots, p_b$ and P_S (Haasler et al., 2021; Benamou et al., 2015). Algorithm 4 implements the BP computation: for the current scaling factors, it passes messages along the tree for each side-information value and combines the results to obtain those marginals (Yedidia et al., 2003; Haasler et al., 2021).

Factorized coupling. Work on the finite supports $A_i = \text{supp}(p_i)$, with $p_i(x_i) > 0$ for $x_i \in A_i$, and the uniform state distribution P_S. Let $\varepsilon > 0$, assume finite pair scores, and define positive, fixed pair kernels

$$\psi_{ij}(x_i, x_j, s) = \exp\left(\frac{f(x_i, x_j, s)}{\varepsilon}\right), \qquad \{i, j\} \in \mathcal{E}. \tag{10}$$

The token graph is a connected tree for each fixed state. The lookup $\psi_{ji}(x_j, x_i, s) = \psi_{ij}(x_i, x_j, s)$ reads the same stored table with its axes exchanged.

For current positive scalings u_i and v, write

$$W_{u,v}(x^b, s) = v(s) \prod_{i=1}^{b} u_i(x_i) \prod_{\{i,j\} \in \mathcal{E}} \psi_{ij}(x_i, x_j, s), \qquad \widehat{\pi}_{u,v} = \frac{W_{u,v}}{\sum_{x^b, s} W_{u,v}(x^b, s)}, \tag{11}$$

Normalization can equivalently be absorbed into v. The scaling vectors enforce marginal constraints, they are not additional watermark rewards.

Sinkhorn updates. For the current normalized coupling $\widehat{\pi} = \widehat{\pi}_{u,v}$, let $q_i = \widehat{\pi}_{X_i}$ and $q_S = \widehat{\pi}_S$ denote its token and side-information marginals. The update $u_i(a) \leftarrow u_i(a) p_i(a) / q_i(a)$ gives

$$\widehat{\pi}'(x^b, s) = \widehat{\pi}(x^b, s) \frac{p_i(x_i)}{q_i(x_i)}. \tag{12}$$

The updated distribution is normalized and its position-i marginal is p_i. The state update is analogous. BP recomputes the marginals after each coordinate update. Full BP calls make this ordering explicit; unchanged messages can be reused without substituting stale marginals. A maximum iteration count limits fitting latency, even if the prescribed tolerance has not been reached. The stopping test uses the current marginal error

$$\delta = \max \left\{ \max_i \|q_i - p_i\|_\infty,\ \|q_S - P_S\|_\infty \right\}.$$

Algorithm 3 adapts tree-structured Sinkhorn scaling to the prescribed token and side-information marginals (Haasler et al., 2021; Benamou et al., 2015). In the algorithm, $q_{1:b} = (q_1, \ldots, q_b)$ denotes the current token marginals.

Algorithm 3 Sinkhorn-BP fitting

Inputs: Token marginals $p_1, \ldots, p_b$, state marginal P_S
Outputs: Fitted coupling $\widehat{\pi}$

Initialize $u_i \leftarrow 1$ for each i, and $v \leftarrow 1$
repeat
  for $i = 1, \ldots, b$ **do**
    $(q_{1:b}, q_S) \leftarrow$ BP-Marginals(u, v)
    $u_i \leftarrow u_i \odot (p_i / q_i)$
  end for
  $(q_{1:b}, q_S) \leftarrow$ BP-Marginals(u, v)
  $v \leftarrow v \odot (P_S / q_S)$
until current marginal error $\leq \tau$ or the iteration budget is reached
return normalized coupling $\widehat{\pi}$ defined by the fitted factors

BP marginal computation. Let $\mathrm{ne}(i)$ be the neighbors of token node i. The index $\ell \in \mathrm{ne}(i)$ runs over these neighboring nodes. In the message equation, $x_i \in A_i$ and $x_j \in A_j$ are candidate token values at the sending and receiving nodes, respectively. For each fixed state s, the message from i to j is

$$m_{i \to j}(x_j, s) = \sum_{x_i \in A_i} u_i(x_i) \psi_{ij}(x_i, x_j, s) \prod_{\ell \in \mathrm{ne}(i) \setminus \{j\}} m_{\ell \to i}(x_i, s). \tag{13}$$

An empty product equals one. Computing messages toward a chosen root and then away from it yields all directed messages (Haasler et al., 2021). The resulting node weights are $\beta_i(x_i, s) = u_i(x_i) \prod_{\ell \in \mathrm{ne}(i)} m_{\ell \to i}(x_i, s)$. They are unnormalized weights, not the final token marginals. BP computes the token weights separately for each side-information value. Algorithm 4 applies standard sum-product belief propagation to the state-conditioned trees (Yedidia et al., 2003; Haasler et al., 2021). It combines these results using each state's current weight to obtain the token and side-information marginals (the target side-information distribution P_S is uniform).

Algorithm 4 BP-Marginals

Fixed: Tree $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, supports $A_1, \dots, A_b$, kernels ψ, root r
Inputs: Current scaling factors u, v
Outputs: Current token marginals $q_{1:b}$ and side-information marginal q_S

Root the tree at r; define parent(i) and child(i)
for $s \in \mathcal{S}$ **do**
  for each child node i in leaves-to-root order **do**
    Compute $m_{i\to\text{parent}(i)}(\cdot, s)$ by Equation 13 ▷ Subtree to parent
  end for
  for each parent node i in root-to-leaves order **do**
    Compute $m_{i\to j}(\cdot, s)$, $j \in$ child(i), by Equation 13 ▷ Parent to children
  end for
  For every $i \in \mathcal{V}$ and $x_i \in A_i$, set
  $\beta_i(x_i, s) \leftarrow u_i(x_i) \prod_{\ell \in \text{ne}(i)} m_{\ell\to i}(x_i, s)$ ▷ Unnormalized token weights
  $Z_s \leftarrow \sum_{x_r \in A_r} \beta_r(x_r, s)$ ▷ Total weight at fixed s
end for
$Z \leftarrow \sum_{s\in\mathcal{S}} v(s) Z_s$
For every $i \in \mathcal{V}$ and $x_i \in A_i$, set
  $q_i(x_i) \leftarrow Z^{-1} \sum_{s\in\mathcal{S}} v(s)\beta_i(x_i, s)$
$q_S(s) \leftarrow v(s)Z_s/Z$, for every s
return $q_{1:b}, q_S$

Sampling the fitted conditional. Given a side-information value s, the final tree factors define the conditional distribution $\widehat{\pi}(X^b \mid S = s)$. The factor $v(s)$ cancels when conditioning on this state. Rewards are attached to child-to-parent edges, so the root's weights combine its token scaling with the incoming BP messages without an additional root-only reward. Each child is then sampled conditional on its parent's token, with weights combining its token scaling, the connecting pair factor and messages from its own children - normalizing these weights gives the categorical probabilities used to sample the output tokens. Algorithm 5 summarizes this conditional tree-sampling procedure.

Algorithm 5 Conditional tree sampling

Fixed: Traversal root r
Inputs: Fitted factors of $\widehat{\pi}$, side-information value s
Outputs: A sample $x_1, \dots, x_b$ from $\widehat{\pi}(X^b \mid S = s)$

Root the tree at r; define parent(i) and child(i)
Compute leaves-to-root messages for state s from the fitted factors using Equation 13
$w_r(x_r) \leftarrow u_r(x_r) \prod_{\ell\in\text{child}(r)} m_{\ell\to r}(x_r, s)$, for every $x_r \in A_r$
Draw $x_r \sim w_r / \sum_{x_r\in A_r} w_r(x_r)$
for each child node j in root-to-leaves order, with $i =$ parent(j) **do**
  $w_j(x_j) \leftarrow u_j(x_j)\psi_{ij}(x_i, x_j, s) \prod_{\ell\in\text{child}(j)} m_{\ell\to j}(x_j, s)$, for every $x_j \in A_j$
  Draw $x_j \sim w_j / \sum_{x_j\in A_j} w_j(x_j)$
end for
return $x_1, \dots, x_b$

B.5 Analytical detection and numerical evaluation

The finite-state null. Fix the observed generated tokens x, block graphs, and pair-score construction. Let $G = |x| > 0$, $b > 1$ be the block length, and $L = \lceil G/b \rceil$ the number of blocks. Define the pair-score sum for block k as

$$h_k(s; x) = \sum_{(c,p)\in\mathcal{E}_k} f(x_{k,c}, x_{k,p}, s). \tag{14}$$

Under the analytical null, the block states are conditionally independent and uniform over $\mathcal{S}$. For the L observed blocks,

$$T_0 = \frac{1}{G} \sum_{k=1}^{L} h_k(S_k; x), \qquad S_k \overset{\text{iid}}{\sim} \text{Unif}(\mathcal{S}). \tag{15}$$

The observed statistic T_{obs} is defined in the main detection section. Its p-value is the inclusive upper tail

$$p_0(T_{\text{obs}}; x) = \frac{1}{|\mathcal{S}|^L} \sum_{s_1,\ldots,s_L \in \mathcal{S}} \mathbf{1}\left\{ \frac{1}{G} \sum_{k=1}^{L} h_k(s_k; x) \geq T_{\text{obs}} \right\}. \tag{16}$$

The shared state retains dependence among edge scores within each block; independence is assumed only across block states.

Numerical evaluation. Each block contributes the finite distribution obtained by evaluating $h_k(s; x)$ over all $s \in \mathcal{S}$ with equal probability. Convolving these distributions and scaling their sum by $1/G$ gives the null distribution of T_0.

Let b_k be the observed token count in block k, so that $\sum_k b_k = G$. Its per-token score is $h_k(s_k; x)/b_k$, with $b_k = b$ for a full block. The sequence score weights these block scores by their token counts:

$$T_{\text{obs}} = \sum_{k=1}^{L} \frac{b_k}{G} \frac{h_k(s_k; x)}{b_k} = \frac{1}{G} \sum_{k=1}^{L} h_k(s_k; x). \tag{17}$$

The observed statistic and null use the same $1/G$ scaling, without dividing by the number of edges or applying another block-count average.

Gaussian-equivalent significance. For $0 < p_0 < 1$, the equivalent-normal score is

$$z_{\text{eq}} = \Phi^{-1}(1 - p_0), \tag{18}$$

where Φ is the standard-normal cumulative distribution function. This transformation reports the finite-state tail on a normal scale.

B.6 Commit-time generation and detection

Commit-time `TIDEmark` applies the coupling during denoising rather than after block completion. In a parallel unmasking update, newly revealed tokens receive predictions from the same pre-update context (Sahoo et al., 2024). No newly sampled token in that update has yet served as context for another selected token's prediction. Replacing these proposals therefore does not invalidate within-update conditioning on their newly sampled values. Adaptive schedules, including entropy-budget unmasking, vary which positions are updated together. Subsequent model updates can condition on the watermarked tokens, without implying independence in the target text distribution or guaranteeing unchanged generation quality (Ben-Hamu et al., 2026; Luxembourg et al., 2026; Wu et al., 2026).

Commit-time coupling. The native sampler selects the positions to update. For a multi-token update within a block, `TIDEmark` orders the selected positions by text position and connects consecutive selected positions in a rightward chain, including across gaps of unselected positions. The pair coupling uses the block's shared state and the current token distributions. Fitting and conditional sampling follow Appendix B.4; rightward chain orientation does not change the bidirectional BP computation. Only native-selected positions are written back. LLaDA 1.0 updates previously masked positions, whereas LLaDA 2.1 also permits revisions of already visible tokens. The same-context unmasking argument above does not assert that revised tokens were absent from the pre-update context.

Singleton updates. A single selected position has no pair reward by itself. Under the preview-and-discard policy, an auxiliary position participates in the coupling, but its sampled token is discarded and only the selected token is written back. Without an eligible partner, the singleton is sampled without a pair reward. The rightward orientation specifies the ordering of a pair's endpoints, not a requirement that the auxiliary position lie to the right of the selected token.

Public detection. The final text does not reveal which tokens were coupled during generation. Using the key, the detector therefore scores all earlier-to-later token pairs within each block. When the update chains partition b eligible tokens into groups of at least two, generation couples between $\lceil b/2 \rceil$ and $b - 1$ pairs, whereas detection scores all $b(b-1)/2$ pairs. The watermark signal is thus spread across a linear number of directly coupled pairs within a quadratic number of scored pairs, which can dilute detection. Eligible tokens exclude the first terminal token and all subsequent positions.

Cost. The per-sweep bound in Appendix B.7 applies to each update's fitted chain, including any auxiliary partner. Total fitting work sums these costs over updates and solver sweeps. Public detec-

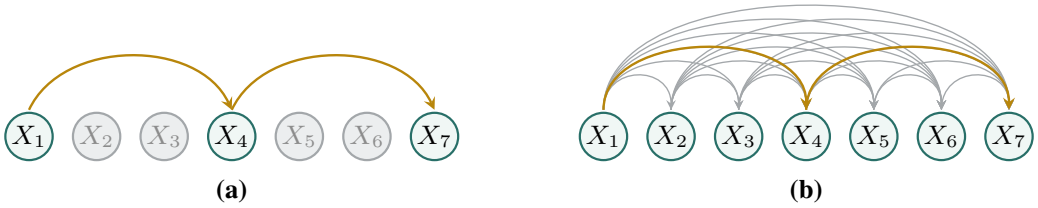

Figure 5: **Commit-time coupling and public detection.** (a) Selected positions $1, 4, 7$ form a chain across gray unselected positions. (b) The detector scores all earlier-to-later pairs. Gold highlights the same two pairs in both panels.

tion scores quadratically many pairs in each block, rather than only the edges used in an individual generation update.

B.7 Operation counts for tree-structured MOT

Consider the coupling on fixed, nonempty candidate supports A_i. Assume that pair factors are already available and that BP reuses intermediate products of incoming messages. Count scalar arithmetic and table accesses at unit cost.

Proposition. One complete bidirectional BP sweep over all side-information states computes the token and state marginals in

$$O\left(|\mathcal{S}|\left[\sum_{i=1}^{b}|A_i| + \sum_{(i,j)\in\mathcal{E}}|A_i||A_j|\right]\right) \tag{19}$$

operations (Haasler et al., 2021). Explicit storage of pair factors, scaling vectors, messages, and beliefs has the same asymptotic bound.

Proof. For a fixed state, a message from position i to position j sums over $|A_i|$ candidates for each of its $|A_j|$ outputs. Processing both directions of every edge over all states therefore gives the pair-product term. Reusing incoming-message products keeps message preparation and storage within the same asymptotic bound as the pairwise contractions. Pair-factor storage contributes the pair-product term, while beliefs and scaling vectors are covered by the node term.

A tree has $b - 1$ edges, giving the uniform upper bound

$$O\left(|\mathcal{S}|\, b\left(\max_{1\le i\le b}|A_i|\right)^2\right). \tag{20}$$

Thus, for bounded candidate and state counts, a sweep is linear in block size. Chains, middle-stars, and balanced trees share this bound, although unequal support sizes can produce different edge costs. In contrast, explicitly storing the dense coupling requires $|\mathcal{S}|\prod_i |A_i|$ entries.

Total fitting work depends on the number of BP sweeps required by the update schedule and monitoring procedure. Factor construction, model inference, token sampling, and detection are extra boilerplate operations.

C Experimental Setup, Ablations, and Limitations

C.1 Experimental configurations

Models and generation settings. The experiments use LLaDA 1.0-Instruct and LLaDA 2.1-mini. Current block-ready `TIDEmark` configurations process text in blocks of 32 tokens. For code generation, the maximum output length is 256 tokens for LLaDA 1.0 and 4,096 tokens for LLaDA 2.1; question answering and summarization use a 512-token limit for both models. LLaDA 1.0 uses confidence-based unmasking for block-ready generation and entropy-budget sampling for commit-time generation. LLaDA 1.0 fixed-schedule code-generation configurations use two tokens per commitment and 128 denoising steps. LLaDA 2.1 uses its native adaptive sampler, which permits token

revisions. Model-specific prompt templates and sampling settings are retained, with watermark parameters and clean controls specified below.

Watermark configurations. Current block-ready `TIDEmark` configurations use greedy native generation and an input temperature of 1 when constructing the token marginals. The output sampling temperature is 0.1 for code generation and 0.3 for question answering and summarization. Coupling fits use score normalization and at most 300 iterations. The entropy regularization parameter ϵ is on the order of 0.1. Commit-time configurations retain their separate sampling settings, so comparisons with block-ready generation do not isolate intervention timing alone.

Baseline settings. For LLaDA 1.0 code generation, KGW, Gumbel, and token-level HW use an input temperature of 0.5. PATTERN-MARK and dgMARK use greedy selection on LCC and beam-$k = 3$ on HumanEval. LLaDA 2.1 retains its native adaptive planner and separate sampling and revision temperatures. LLaDA 1.0 block-ready HW on HumanEval uses $\epsilon = 0.01$ and at most 2,000 fitting iterations with early stopping. LLaDA 2.1 HW uses $\epsilon = 0.05$, with 300 iterations on LCC and at most 2,000 iterations with early stopping on HumanEval.

Clean controls. Each configuration uses its designated unwatermarked reference. Empirical detection thresholds are selected and evaluated on the same clean cohort, without held-out FPR guarantees.

C.2 Robustness to attacks

The robustness evaluation uses LLaDA 1.0 with block-ready Chain watermarking. The original Finance QA and Multi-News cohorts construct side information from a MinHash of the previous block. This choice aims to limit seed disruption under local edits (Kirchenbauer et al., 2024). Specifically, for the previous block B_{k-1},

$$x^\star_{k-1} = \operatorname*{arg\,min}_{x \in B_{k-1}} h(x), \qquad s_k = g(x^\star_{k-1}),$$

where h assigns hash values to token identities and g maps the selected token to the current block's watermark state. Both functions are determined by the watermark key.

Watermarked outputs undergo random deletion, insertion, and substitution at rates of 10%, 20%, and 30% (Hong & No, 2026; Raban et al., 2026). Detection uses only the edited text, without access to the original block alignment or a search over possible boundaries.

Tables 4 and 5 jointly report detection and reconstruction after corruption for the original shared-MinHash cohorts. State recovery measures whether a received block recovers the original block's watermark state. Edge recall measures what fraction of the original graph's token-pair connections also appear in the graph reconstructed from corrupted text.

Detection persists despite incomplete reconstruction. Under 20% insertion on Finance QA, edge recall is 41.1%, while TPR@5% reaches 82.5%. However, robustness differs across datasets: at 30% edits, TPR@5% ranges from 60.0% to 67.0% on Finance QA, compared with 39.5% to 45.0% on Multi-News. These results show that complete graph reconstruction is not necessary for detection, while the retained detection rate depends on the dataset and attack.

State recovery also declines alongside token-pair edge recall as edit rates increase, indicating that edits disrupt both the watermark state and the scored connections. This motivates reconstruction-aware detection for structures tied to specific positions, such as middle-star and balanced tree, where edits can change the designated center or hierarchical connections. Such structures may require mechanisms to recover the original connections or avoid explicit reconstruction, as the all-pairs detector does in commit-time `TIDEmark`.

C.3 Regularization tradeoff

Figure 6 illustrates how entropy regularization affects conditional token distributions and detector scores on one fixed block. Among converged fits, increasing ϵ from 0.02 to 0.5 reduces mean conditional token KL from 0.433 to 0.0034 nats and mean detector score z from 3.649 to 0.315. Stronger regularization therefore keeps the state-conditioned token distributions closer to the input marginals, but weakens the detection signal on this block. The figure distinguishes the converged $\epsilon = 0.1$ fit, marked as an intermediate tradeoff rather than a validated optimum, from the $\epsilon = 0.01$ fit, which did

Table 4: **Robustness and reconstruction under attacks: Finance QA.** `TIDEmark` Chain on LLaDA 1.0, using the previous-block shared-MinHash seed from the original attack run. Detection and recovery values are in percent, except AUC and mean $-\log_{10} p$.

Attack	Edit (%)	AUC	Mean $-\log_{10} p$	TPR@1%	TPR@5%	TPR@10%	State Recon. (%)	Edges Recon. (%)
Unedited	–	0.983	5.103	64.5	91.9	94.9	93.7	94.2
Deletion	10	0.946	3.426	77.0	85.5	86.5	56.3	55.0
	20	0.902	2.652	60.5	71.0	78.5	39.4	34.3
	30	0.856	2.168	21.0	60.0	65.5	32.3	21.7
Insertion	10	0.954	3.589	77.0	84.5	88.0	57.5	59.8
	20	0.932	3.039	75.5	82.5	84.5	42.2	41.1
	30	0.884	2.506	48.5	67.0	77.5	32.3	30.4
Substitution	10	0.951	4.218	82.0	86.0	89.5	73.7	76.5
	20	0.876	2.996	45.5	69.0	73.5	54.7	59.7
	30	0.860	2.289	51.0	62.0	70.0	45.8	45.8

Table 5: **Robustness and reconstruction under attacks: Multi-News.** `TIDEmark` Chain on LLaDA 1.0, using the previous-block shared-MinHash seed from the original attack run. Detection and recovery values are in percent, except AUC and mean $-\log_{10} p$.

Attack	Edit (%)	AUC	Mean $-\log_{10} p$	TPR@1%	TPR@5%	TPR@10%	State Recon. (%)	Edges Recon. (%)
Unedited	–	0.963	4.693	66.2	84.3	89.9	98.3	98.5
Deletion	10	0.889	2.322	34.5	59.0	71.5	41.9	34.2
	20	0.837	1.783	26.5	49.0	57.5	25.9	18.3
	30	0.802	1.429	23.5	39.5	59.5	20.7	11.3
Insertion	10	0.895	2.513	34.0	61.5	72.0	40.5	37.1
	20	0.827	1.971	27.0	48.5	56.5	27.5	21.3
	30	0.799	1.631	34.5	42.5	52.5	20.6	14.9
Substitution	10	0.924	3.352	59.5	73.0	77.0	76.8	78.6
	20	0.876	2.427	36.5	63.0	71.5	61.7	61.3
	30	0.824	1.636	36.5	45.0	57.5	48.6	46.1

not converge within the iteration budget and is shown separately. These measurements characterize a single selected block, rather than text quality or detection rates across a dataset.

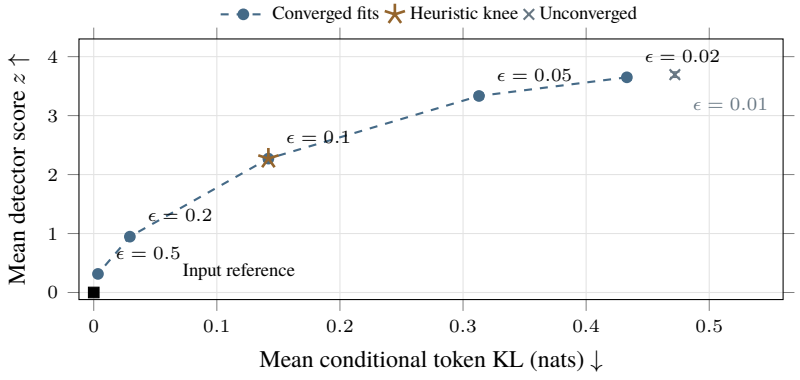

Figure 6: **Regularization, distribution shift, and detector score.** `TIDEmark` Chain on one outcome-selected 32-token LCC block from LLaDA 1.0-Instruct. Conditional token KL is averaged over states and positions. The star marks an intermediate tradeoff and the cross an unconverged fit.

Table 6: **Relative watermark runtime.** Mean percentage of recorded request time spent inside watermark calls. Values are not clean-relative slowdown. Dashes are unmeasured, not zero. TIDEmark uses the Joint $\epsilon = 0.1$ diagnostic recipe.

	Relative watermark runtime (%)			
	LLaDA 1.0		LLaDA 2.1	
Method	LCC	HumanEval	LCC	HumanEval
Token-commit				
KGW	–	–	6.9	9.8
DMark	1.2	9.6	–	–
Gumbel	0.6	3.3	7.5	13.8
HW	8.6	51.0	10.5	12.6
dgMARK	0.1	0.7	–	–
TIDEmark	29.0	74.4	64.5	66.1
Block-ready				
KGW	–	–	4.2	5.8
DMark	–	–	5.4	5.1
Gumbel	–	–	5.5	10.2
HW	6.8	32.3	8.6	9.5
TIDEmark Chain	5.4	27.9	19.7	36.1
TIDEmark Middle-star	6.1	29.0	20.2	38.0
TIDEmark Balanced	5.2	26.9	21.7	37.2

C.4 Runtime cost

Table 6 reports the share of recorded request time spent inside watermark calls, including preparation, fitting, and sampling. Block-ready watermarks run once per completed block rather than repeatedly during denoising of the same block. On HumanEval with LLaDA 2.1, watermark calls account for 36.1% of recorded request time for block-ready TIDEmark Chain, compared with 66.1% for token-commit TIDEmark and 9.5% for block-ready HW. Token revisions in LLaDA 2.1 can trigger additional watermarking calls, which block-ready TIDEmark avoids by waiting until block completion.

C.5 Additional benchmark results

Tables 7 and 8 report HumanEval and Finance QA results for both evaluated models, complementing the LCC and Multi-News results from Section 5. Each table presents detection performance, CE, and task performance, measured by pass@1 for HumanEval and ROUGE-L for Finance QA. Results are divided into token-commit and block-ready watermarking.

On HumanEval, stronger detection among TIDEmark's selected block-ready configurations accompanies lower functional correctness (Table 7). With LLaDA-Instruct, balanced tree reaches 57.3% TPR at a target 5% FPR, but pass@1 falls by 11.6 percentage points relative to the clean reference. Chain retains more correct solutions, reducing this loss to 8.0 points, while reaching 25.6% TPR. On LLaDA 2.1, middle-star provides TIDEmark's strongest detection at the same threshold, 36.6%, but incurs a 21.4-point pass@1 decrease. The token-commit configuration retains more functional correctness, with a 7.4-point decrease, although its TPR is lower at 22.0%. On both models, TIDEmark's configuration with the largest pass@1 decrease also achieves its highest detection rate.

On Finance QA, TIDEmark's selected block-ready configurations achieve stronger detection than its token-commit configuration, with a corresponding task-quality cost (Table 8). At a target 5% FPR, Chain reaches 99.0% TPR on LLaDA-Instruct and 100.0% on LLaDA 2.1, compared with 36.0% and 39.0% for token-commit. Chain's ROUGE-L decreases by 1.8 and 1.2 points relative to the clean references, whereas token-commit scores 1.1 points above the clean reference on LLaDA-Instruct and 0.1 point below it on LLaDA 2.1. Middle-star offers a different balance on LLaDA 2.1: it reaches 99.5% TPR while retaining higher ROUGE-L and lower CE than Chain. The selected configurations thus offer different balances between detection and task performance.

Table 7: **HumanEval detection and code quality.** TIDEmark variants are (Co) commit, (Ch) chain, (S) middle-star, and (B) balanced tree. Best and second-best watermarked values are bold and underlined.

Method	TPR@1 (%)	TPR@5 (%)	TPR@10 (%)	AUC	Mean $-\log_{10} p$	CE	pass@1 (%)
Clean reference	–	–	–	–	–	1.089	35.4
Token-commit							
KGW	12.8	43.3	51.8	0.803	1.024	–	19.5
PATTERN-M	7.3	27.4	46.3	0.758	0.758	–	23.8
dgMARK	65.2	81.1	88.4	0.962	1.729	–	22.0
Gumbel	37.2	55.5	62.8	0.869	1.327	–	24.4
HW	2.4	7.3	9.1	0.505	0.445	–	**34.8**
TIDEmark (Co)	3.0	12.2	27.4	0.668	0.702	**0.907**	31.7
Block-ready							
KGW	**71.3**	**89.6**	**95.1**	**0.979**	**1.899**	2.238	4.9
Gumbel	9.8	15.2	23.8	0.725	0.797	1.294	20.1
PATTERN-M	22.0	39.0	62.2	0.851	1.156	2.742	1.2
HW	0.6	12.8	22.0	0.689	0.720	1.505	24.4
TIDEmark (Ch)	2.4	25.6	33.5	0.727	0.800	1.249	27.4
TIDEmark (S)	9.8	28.0	49.4	0.760	0.919	1.188	26.8
TIDEmark (B)	40.2	57.3	65.2	0.872	1.273	1.418	23.8

(a) LLaDA-Instruct

Method	TPR@1 (%)	TPR@5 (%)	TPR@10 (%)	AUC	Mean $-\log_{10} p$	CE	pass@1 (%)
Clean reference	–	–	–	–	–	0.635	84.2
Token-commit							
KGW	24.4	47.0	64.0	0.852	1.274	1.045	65.9
Gumbel	5.5	12.8	19.5	0.573	0.590	**0.652**	**79.9**
HW	**94.5**	**95.7**	**97.0**	**0.988**	**2.056**	4.250	21.3
TIDEmark (Co)	14.6	22.0	28.7	0.640	0.763	0.961	76.8
Block-ready							
KGW	84.2	90.9	92.7	0.971	1.948	2.663	12.8
Gumbel	9.2	15.9	21.3	0.631	0.706	0.880	67.1
HW (input 0.7)	6.1	12.2	24.4	0.617	0.641	0.666	77.4
TIDEmark (Ch)	8.5	19.5	27.4	0.678	0.750	0.788	65.2
TIDEmark (S)	30.5	36.6	43.3	0.742	0.962	0.827	62.8
TIDEmark (B)	7.9	16.5	25.0	0.676	0.769	0.841	66.5

(b) LLaDA 2.1

Table 8: **Finance QA detection and text quality.** TIDEmark variants are (Co) commit, (Ch) chain, (S) middle-star, and (B) balanced tree. Best and second-best watermarked values are bold and underlined.

Method	TPR@1 (%)	TPR@5 (%)	TPR@10 (%)	AUC	Mean $-\log_{10} p$	CE	ROUGE L
Clean reference	–	–	–	–	–	1.460	17.4
Token-commit							
KGW	70.0	85.0	91.0	0.967	1.906	1.987	17.8
Gumbel	97.0	97.5	97.5	0.995	2.167	1.754	15.0
dgMARK	98.0	98.5	99.5	0.998	2.182	1.665	16.6
PATTERN-M	70.0	82.5	86.0	0.949	1.812	1.589	16.6
TIDEmark (Co)	16.0	36.0	45.0	0.831	1.092	**1.556**	**18.5**
Block-ready							
KGW	95.0	97.5	98.0	0.994	2.155	2.778	15.6
Gumbel	43.5	64.0	75.0	0.917	1.519	1.750	17.1
PATTERN-M	55.0	76.0	89.0	0.950	1.635	3.516	14.9
HW	**99.0**	**99.5**	**100.0**	**1.000**	**2.200**	4.369	14.2
TIDEmark (Ch)	95.5	99.0	99.0	0.994	2.178	2.798	15.6
TIDEmark (S)	49.0	71.0	78.0	0.906	1.467	1.876	16.7
TIDEmark (B)	98.5	99.0	**100.0**	0.999	2.178	2.728	15.3

(a) LLaDA-Instruct

Method	TPR@1 (%)	TPR@5 (%)	TPR@10 (%)	AUC	Mean $-\log_{10} p$	CE	ROUGE L
Clean reference	–	–	–	–	–	1.616	20.4
Token-commit							
KGW	98.0	**100.0**	**100.0**	**1.000**	–	1.881	20.3
Gumbel	15.5	30.5	39.5	0.802	–	**1.627**	**20.5**
TIDEmark (Co)	18.0	39.0	51.0	0.779	1.082	1.911	20.3
Block-ready							
KGW	**100.0**	**100.0**	**100.0**	**1.000**	**2.207**	3.384	19.3
Gumbel	27.5	41.0	49.0	0.835	1.203	2.003	20.1
TIDEmark (Ch)	**100.0**	**100.0**	**100.0**	**1.000**	**2.207**	3.551	19.2
TIDEmark (S)	99.5	99.5	**100.0**	**1.000**	2.201	3.015	19.4
TIDEmark (B)	**100.0**	**100.0**	**100.0**	**1.000**	**2.207**	3.538	19.3

(b) LLaDA 2.1

C.6 Limitations and Scope

The distortion-free guarantee applies to each recorded token distribution under the ideal coupling, after averaging over the secret side information. It does not guarantee preservation of the full text distribution or unchanged task performance. The practical implementation uses restricted candidate sets and a finite fitting budget, so its numerical accuracy must be assessed separately.

Block-ready watermarking waits for the entire block to finish decoding before applying the watermark and releasing its tokens. In revision-based diffusion models, tokens can still change during decoding, so waiting for the block to stabilize is part of the generation process rather than a requirement introduced by watermarking. Watermarking nevertheless adds processing time after block completion and before release.