Rethinking LLM Watermark Detection in Black-Box Settings: A Non-Intrusive Third-Party Framework

Anonymous ACL submission

Abstract

While watermarking serves as a critical mechanism for LLM provenance, existing secret-key schemes tightly couple detection with injection, requiring access to keys or provider-side scheme-specific detectors for verification. This dependency creates a fundamental barrier for real-world governance, as independent auditing becomes impossible without compromising model security or relying on the opaque claims of service providers. To resolve this dilemma, we introduce TTP-Detect, a pioneering black-box framework designed for non-intrusive, third-party watermark verification. By decoupling detection from injection, TTP-Detect reframes verification as a relative hypothesis testing problem. It employs a proxy model to amplify watermark-relevant signals and a suite of complementary relative measurements to assess the alignment of the query text with watermarked distributions. Extensive experiments across representative watermarking schemes, datasets and models demonstrate that TTP-Detect achieves superior detection performance and robustness against diverse attacks.

1 Introduction

Large language models (LLMs) have rapidly shifted from research prototypes to widely deployed generative systems, producing fluent text at negligible marginal cost. This unprecedented scalability has amplified long-standing concerns around synthetic content, as LLMs can now be used to automate misinformation campaigns, fabricate credible documents, and violate intellectual property rights. In parallel with these risks, LLM watermarking has emerged as a practical mechanism for provenance: by embedding subtle statistical signals during generation, watermarks enable LLM service providers to verify whether a suspicious text is watermarked, offering a promising solution for the proactive forensics of AI-generated content.

Most LLM watermarking techniques, regardless of whether they modify token probabilities (Kirchenbauer et al., 2023) or influence the sampling procedure (Dathathri et al., 2024), ultimately function by introducing a key-dependent bias toward certain tokens during generation. Then the detector must reconstruct this bias to verify the watermark; consequently, watermark injection and detection necessarily share the same secret key. This coupling poses a fundamental obstacle in real-world attribution. A watermark cannot be independently verified, since courts or platform moderators lack access to the key and must accept the service provider's claim about whether a watermark is present. The process becomes opaque and trust-dependent. Allowing third parties to perform detection would require disclosing the key, which would compromise security by enabling adversaries to imitate or remove the watermark. Thus, current private-key schemes cannot simultaneously support independent verification and preserve the confidentiality on which their security depends.

This limitation makes a **Trusted Third Party (TTP)** essential for watermark governance: an entity that can verify the watermark without access to the secret key, preventing LLMs providers from misreporting results, blocking adversarial forgery, and enabling evidence suitable for regulatory or judicial use. Recent efforts toward third-party or publicly verifiable watermarking (Liu et al., 2024a; Fairoze et al., 2025; Duan et al., 2025) take steps in this direction, but they retain a critical structural assumption: watermark injection and detection remain tightly paired, with verification logic tailored to specific injection mechanisms. This one-to-one coupling conflicts with real-world governance, where watermark injection is the responsibility of model providers, while verification and oversight belong to independent regulators. As long as detection is bound to the injection design, it cannot function as a neutral, reusable auditing layer

across heterogeneous models and watermarking schemes. To resolve this mismatch, we introduce **TTP-Detect**, a key-agnostic black-box watermark detection framework that explicitly decouples detection from injection, allowing a TTP to assess watermark presence directly from output text alone.

TTP-Detect reframes absolute-threshold detection as a *relative hypothesis testing* problem: given a query text, the goal is to determine whether it aligns more closely with the statistical behavior of watermarked outputs than with ordinary text. Our key observation is that no single statistic can reliably characterize watermark presence across algorithms, since different watermarking schemes concentrate their effects on different aspects of the generation process. TTP-Detect operationalizes this insight by first mapping texts into a representation space that amplifies *watermark-relevant discrepancies*, and then evaluating the query through *complementary relative measurements* that capture local consistency, global distributional shifts, and generation-time likelihood patterns (captured by post-hoc re-scoring). These signals provide partially independent evidence and are robustly calibrated into a *unified decision score*. This design yields a detection framework that is robust across diverse watermarking schemes evaluated in our study and readily extensible via additional relative measurement modules.

Our main contributions are as follows: **(1)** We introduce a pioneering formulation for third-party, key-agnostic watermark verification in black-box settings. By decoupling verification from injection, we establish a key-agnostic paradigm that enables third-party auditing without compromising security. **(2)** We propose TTP-Detect, a unified framework that reframes verification as a relative hypothesis test. By leveraging proxy representations and complementary relative measurements, our approach captures subtle watermark traces without access to the service provider's model states or specific algorithms. **(3)** We demonstrate superior universality and robustness through extensive experiments. TTP-Detect reliably identifies a wide range of representative watermarking schemes and maintains robustness under attacks.

2 Related Work

Private-Key Based LLM Watermarking Existing LLM watermarking schemes generally fall into two primary categories: logits-based and sampling-based. The pioneering logits-based method, KGW (Kirchenbauer et al., 2023), partitions the vocabulary using a hash of preceding tokens seeded with a secret key, biasing generation toward green list tokens. To improve robustness against text editing, subsequent works propose global vocabulary partitioning (Zhao et al., 2024) or semantic-aware partitioning (Liu et al., 2024b; Ren et al., 2024a; He et al., 2024). To better preserve text quality, Hu et al. (2024); Chen et al. (2025) introduce unbiased reweighting, while Ren et al. (2024b); Huo et al. (2024); Liu and Bu (2024); Lee et al. (2024); Lu et al. (2024); Wang et al. (2025b) dynamically adjust the vocabulary partition or watermark strength. Sampling-based methods, in contrast, influence token selection without altering logits: Aaronson (2023); Christ et al. (2024); Fu et al. (2024); Kuditipudi et al. (2024); Dathathri et al. (2024); Mao et al. (2025) explore token-level sampling strategies, while Hou et al. (2024a,b); Dabiriaghdam and Wang (2025) focus on sentence-level schemes. Recent advances further optimize the robustness–quality trade-off by synergistically combining logits- and sampling-based modifications (Wang et al., 2025a).

Publicly Detectable Watermarking To mitigate the dependence on private keys during detection, some works explore publicly detectable or publicly verifiable watermarking schemes. UPV (Liu et al., 2024a) implements generation and detection using two neural networks that share common token embeddings. Fairoze et al. (2025) design a publicly detectable scheme that combines digital signatures, error-correcting codes and rejection sampling to embed a cryptographically verifiable signature into generated text. PVMark (Duan et al., 2025) further makes existing secret-key schemes publicly verifiable by wrapping the detectors with zero-knowledge proofs and re-engineering representative schemes into ZK-friendly circuits. However, their verification logic remains tightly coupled to the injection mechanism and typically depends on provider-controlled training or shared parameters, limiting portability across watermarking schemes and undermining detector neutrality.

3 Preliminary

3.1 LLM Generation

Consider an autoregressive LLM $\mathcal{M}$ with vocabulary $\mathcal{V}$. Given a prompt x and a sequence of preceding tokens $y_{<i}$, the model predicts the next token

y_i iteratively. At each step i, $\mathcal{M}$ maps the context to a logits vector $\boldsymbol{l}_i = \mathcal{M}(x, y_{<i}) \in \mathbb{R}^{|\mathcal{V}|}$. This vector is then normalized via softmax to yield a probability distribution $\boldsymbol{p}_i$. Finally, the token y_i is selected from $\boldsymbol{p}_i$ using a sampling strategy, denoted as $y_i \sim \text{Sample}(\boldsymbol{p}_i)$.

3.2 LLM Watermarking

A watermarking algorithm consists of a generator and a detector. The generator modifies the standard generation process to produce watermarked text using a private key ξ.

Logits-based Watermarking. These methods inject the watermark signal by modifying the logits $\boldsymbol{l}_i$ before the softmax layer. A representative method is the KGW (Kirchenbauer et al., 2023) scheme. At each step i, the vocabulary $\mathcal{V}$ is partitioned into a green list $\mathcal{G}_i$ and a red list using a hash function seeded with the previous context and the private key. A bias δ is added to the logits of tokens in $\mathcal{G}_i$:

$$\tilde{\boldsymbol{l}}_i[v] = \begin{cases} \boldsymbol{l}_i[v] + \delta, & \text{if } v \in \mathcal{G}_i \\ \boldsymbol{l}_i[v], & \text{otherwise} \end{cases}. \tag{1}$$

This bias encourages the model to select tokens from $\mathcal{G}_i$. During detection, the detector calculates a z-score based on the observed proportion of $\mathcal{G}_i$ tokens in the text. If the z-score exceeds a threshold, the text is deemed watermarked.

Sampling-based Watermarking. These methods embed the watermark during the sampling stage. For example, AAR (Aaronson, 2023) employs a pseudo-random sequence generated by the key ξ. The sampling process is modified to prioritize tokens that align with the sequence. The detection process verifies whether the generated tokens correlate significantly with the pseudo-random sequence derived from the key.

4 TTP-Detect: A Black-Box Third-Party Detection Framework

We first outline the TTP-Detect setting, then describe representation extraction, relative measurements and ensemble detection modules.

4.1 Threat Model and Problem Formulation

Threat Model. We consider a three-party setting involving a user (U), a model service provider (S), and a trusted third-party auditor (D). The provider S deploys a proprietary watermarking algorithm and exposes only a standard text generation API with a binary watermark control flag to the auditor D, without revealing the watermarking mechanism, secret keys, or internal model states. The user U submits a query text t_q for verification. D is assumed to be a regulated and compliance-certified entity (e.g., an accredited auditing or digital forensics organization) that performs watermark detection independently. Neither U nor S has access to the detection pipeline, and D has no access to the watermarking algorithm or secret keys. This defines a black-box watermark detection problem in which D rely solely on observable output behavior.

Problem Formulation. We formalize black-box watermark detection as a hypothesis testing problem. Given a query text t_q, D evaluates whether t_q is more consistent with S's watermarked output distribution than with its unwatermarked counterpart. Since D cannot model the universal distribution of unwatermarked text, we adopt S's unwatermarked output as a local reference anchor for the null hypothesis. Specifically, we test:

$$\mathcal{H}_0 : t_\text{q} \sim P_\text{o}, \quad \mathcal{H}_1 : t_\text{q} \sim P_\text{wm}, \tag{2}$$

where P_o and P_wm denote S's unwatermarked and watermarked output distributions, respectively.

TTP-Detect Framework Overview. To operationalize the above hypothesis test without access to likelihood functions, D queries S to construct two empirical *reference sets*, $\mathcal{T}_\text{o} \sim P_\text{o}$ and $\mathcal{T}_\text{wm} \sim P_\text{wm}$. Given a query text t_q, TTP-Detect first maps the query and reference texts into a proxy-induced representation space to amplify watermark-relevant discrepancies (Sec. 4.2). It then applies M complementary *relative measurement modules* to produce scores $\{A_m\}_{m=1}^{M}$, where each A_m measures the affinity of t_q to $\mathcal{T}_\text{wm}$ relative to $\mathcal{T}_\text{o}$ (Sec. 4.3). Finally, it aggregates and calibrates these signals into a single statistic $\text{Agg}(A_1, \ldots, A_M)$ for decision-making (Sec. 4.4). Accordingly, unwatermarked texts are expected to yield consistently low relative scores across modules, resulting in an aggregate statistic consistent with $\mathcal{H}_0$. We provide further discussion on practical deployment in Appendix H.

4.2 Proxy-Based Representation Extraction

Watermark signals are subtle and distributed, making direct detection from raw text difficult in black-box settings. We therefore extract representations with a *proxy model* adapted to amplify watermark-relevant differences, supported by prior evidence on *watermark learnability* (Gu et al., 2024).

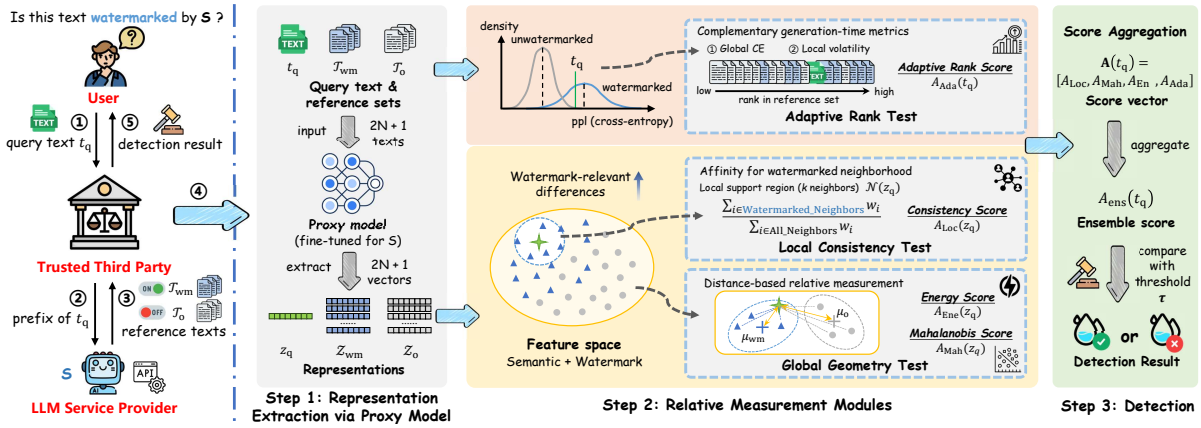


Figure 1: TTP-Detect framework overview.

Discriminative Instruction Tuning. To obtain representations that separate watermarked and unwatermarked outputs, we adapt a proxy model via a supervised discrimination objective. We construct a training set $\mathcal{D}_{\text{sft}}$ by sampling prompts from a corpus and querying S to produce paired completions, each consisting of an unwatermarked output t_{o} and its watermarked counterpart t_{wm}. The proxy model $\mathcal{M}_{\text{proxy}}$ is then tuned to predict the corresponding watermark label sequence from the completion text, optimized via conditional negative log-likelihood:

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}_{\text{sft}}|} \sum_{(t,y)\in\mathcal{D}_{\text{sft}}} \sum_{i=1}^{|y|} \log P_\theta(y_i|t, y_{<i}). \tag{3}$$

Representation Extraction. At detection time, D queries S to build N paired reference samples, yielding two reference sets $\mathcal{T}_{\text{o}} = \{t_{\text{o}}^{(j)}\}_{j=1}^{N} \sim P_{\text{o}}$ and $\mathcal{T}_{\text{wm}} = \{t_{\text{wm}}^{(j)}\}_{j=1}^{N} \sim P_{\text{wm}}$. Given any text t (either the query t_{q} or a reference sample), we encode it with $\mathcal{M}_{\text{proxy}}$ and extract the ℓ_2-normalized hidden state of the final token from the last layer, denoted as $z(t) \in \mathbb{R}^{d_{\text{proxy}}}$. This representation captures both contextual semantics and watermark-discriminative cues internalized during proxy training. Accordingly, we obtain feature sets $\mathcal{Z}_{\text{o}} = \{z(t_{\text{o}}^{(j)})\}_{j=1}^{N}$ and $\mathcal{Z}_{\text{wm}} = \{z(t_{\text{wm}}^{(j)})\}_{j=1}^{N}$, and represent the query as $z_{\text{q}} = z(t_{\text{q}})$.

4.3 Relative Measurement Modules

Watermarking induce bias that does not concentrate on a single statistic, but manifests across different statistical scales of the generated text. As a result, relying on a single detection criterion is brittle under algorithmic diversity and semantic variation. TTP-Detect employs multiple relative measurements, each probing watermark consistency from a complementary statistical perspective.

4.3.1 Local Consistency Test

Local consistency measures whether a query representation is locally surrounded by watermarked samples, reflecting the tendency of watermark-induced bias to preserve neighborhood-level coherence under similar semantics.

We estimate the local watermark likelihood by examining the neighborhood of the query representation z_{q} in feature space. Let $\mathcal{X}_{\text{ref}} = \mathcal{Z}_{\text{o}} \cup \mathcal{Z}_{\text{wm}}$ denote the combined reference set. Since all representations are ℓ_2-normalized, semantic affinity is measured by cosine similarity. We define the local support region $\mathcal{N}(z_{\text{q}})$ as the k nearest neighbors of z_{q} in $\mathcal{X}_{\text{ref}}$. To quantify local affinity, we employ a kernel-based density estimator over $\mathcal{N}(z_{\text{q}})$. Each neighbor $z_i \in \mathcal{N}(z_{\text{q}})$ contributes a weight

$$w_i = \exp\left(-\frac{1 - \text{sim}(z_{\text{q}}, z_i)}{\sigma^2}\right), \tag{4}$$

where σ is an adaptive bandwidth set to the average cosine distance within $\mathcal{N}(z_{\text{q}})$, normalizing for variations in local sample density.

The *local consistency score* is then defined as the normalized density mass of watermarked samples:

$$A_{\text{Loc}}(z_{\text{q}}) = \frac{\sum_{z_i\in\mathcal{N}(z_{\text{q}})\cap\mathcal{Z}_{\text{wm}}} w_i}{\sum_{z_i\in\mathcal{N}(z_{\text{q}})} w_i}. \tag{5}$$

A higher A_{Loc} indicates that z_{q} lies in a neighborhood dominated by watermarked references, providing evidence of local watermark consistency.

4.3.2 Global Geometry Test

While local measurements identify watermark traces on local manifolds, it may fail to capture

the systematic distributional shifts. Watermarking operates as a global intervention, often resulting in distinguishable geometric signatures across the entire feature space. We measure this global affinity using two complementary distance-based tests.

Mahalanobis Score. To account for correlated watermark perturbations, we evaluate the query using a Mahalanobis distance. Given the high dimensionality of representations, we project reference samples into a lower-dimensional subspace via PCA (details in Appendix A) and estimate class-specific regularized covariances. The *Mahalanobis score* is defined as

$$\Delta_{\text{Mah}}(z_{\text{q}}) = \delta^2_{\hat{\Sigma}_{\text{o}}}(\tilde{z}_{\text{q}}, \tilde{\mu}_{\text{o}}) - \delta^2_{\hat{\Sigma}_{\text{wm}}}(\tilde{z}_{\text{q}}, \tilde{\mu}_{\text{wm}}), \quad (6)$$

where δ^2_{Σ} denotes the squared Mahalanobis distance in the subspace. A positive Δ_{Mah} indicates closer alignment with the watermarked distribution.

Energy Score. Some complex watermarking schemes induce irregular distributions that violate the Gaussian assumption of the Mahalanobis test. To handle this, we additionally employ a non-parametric Energy distance. We define the *Energy score* as the contrast between the energy distances to the two reference sets:

$$\Delta_{\text{Ene}}(z_{\text{q}}) = \delta_{\text{Ene}}(z_{\text{q}}, \mathcal{Z}_{\text{o}}) - \delta_{\text{Ene}}(z_{\text{q}}, \mathcal{Z}_{\text{wm}}), \quad (7)$$

where δ_{Ene} measures the average pairwise distance adjusted by the internal dispersion of each reference set (see Appendix A for details).

Both statistics are mapped to normalized scores via a sigmoid function, yielding A_{Mah} and A_{Ene}.

4.3.3 Adaptive Rank Test

This test targets watermark signals manifested in the generation dynamics rather than in embedding geometry. Since some schemes perturb token selection probabilities with minimal semantic or representational shift, watermark traces can surface in token-level likelihood patterns even when representation-based measurements are inconclusive. We therefore compute negative log-likelihood (NLL) based statistics under $\mathcal{M}_{\text{score}}$ and apply a direction-adaptive rank test against paired reference sets.

Given a text $t{=}(x_1, \ldots, x_L)$, we compute the token-wise NLL sequence $\boldsymbol{\ell}{=}\{\ell_i\}_{i=1}^{L}$, where $\ell_i{=}-\log P(x_i|x_{<i})$. We then extract two complementary statistics: (1) global cross-entropy $E_{\text{GE}}{=}\frac{1}{L}\sum_{i=1}^{L}\ell_i$, capturing overall fluency; (2) local volatility $E_{\text{LV}}{=}\sqrt{\frac{1}{L}\sum_{i=1}^{L}\left(\ell_i - E_{\text{GE}}\right)^2}$, capturing generation confidence variability.

For a feature $f \in \{E_{\text{GE}}, E_{\text{LV}}\}$, the effect direction varies across watermarking schemes. To avoid assuming a fixed direction, we infer an adaptive direction from the reference sets $\mathcal{T}_{\text{o}}$ and $\mathcal{T}_{\text{wm}}$: $\rho_f = \text{sign}(\mathbb{E}_{t\in\mathcal{T}_{\text{wm}}}[f(t)] - \mathbb{E}_{t\in\mathcal{T}_{\text{o}}}[f(t)])$, so that larger $\tilde{f}(t) = \rho_f \cdot f(t)$ is aligned with the watermarked tendency for this feature.

We estimate the conformity ranks of t_{q} under both reference sets:

$$p^f_{\text{wm}} = \frac{1 + \sum_{t\in\mathcal{T}_{\text{wm}}} \mathbb{I}\left[\tilde{f}(t) \le \tilde{f}(t_{\text{q}})\right]}{|\mathcal{T}_{\text{wm}}| + 1}, \qquad p^f_{\text{o}} = \frac{1 + \sum_{t\in\mathcal{T}_{\text{o}}} \mathbb{I}\left[\tilde{f}(t) \ge \tilde{f}(t_{\text{q}})\right]}{|\mathcal{T}_{\text{o}}| + 1}. \quad (8)$$

After alignment, a larger $\tilde{f}(t_{\text{q}})$ implies larger p^f_{wm} and smaller p^f_{o}. We normalize them into a watermark tendency score (ϵ for numerical stability): $s_f(t_{\text{q}}){=}p^f_{\text{wm}}/(p^f_{\text{wm}} + p^f_{\text{o}} + \epsilon) \in [0, 1]$.

Finally, we fuse global fluency and local volatility into an *adaptive rank score*, where λ trades off the overall likelihood shift and local volatility:

$$A_{\text{Ada}}(t_{\text{q}}) = \lambda\, s_{E_{\text{GE}}}(t_{\text{q}}) + (1-\lambda)\, s_{E_{\text{LV}}}(t_{\text{q}}). \quad (9)$$

4.4 Ensemble and Detection

Each relative measurement module captures a complementary aspect of the watermark signal, including local consistency (A_{Loc}), global geometry (A_{Mah}, A_{Ene}), and adaptive rank (A_{Ada}). Given the diversity of watermarking mechanisms, no single score is universally reliable. We therefore combine these measurements through a lightweight ensemble. For a query text t_{q}, we form a score vector $\mathbf{A}(t_{\text{q}}){=}[A_{\text{Loc}}, A_{\text{Mah}}, A_{\text{Ene}}, A_{\text{Ada}}]^\top \in [0, 1]^4$ and compute the *ensemble score* as

$$A_{\text{ens}}(t_{\text{q}}) = \sigma\left(\mathbf{w}^\top \mathbf{A}(t_{\text{q}}) + b\right), \quad (10)$$

where $\mathbf{w} \in \mathbb{R}^4$ and $b \in \mathbb{R}$ are learnable parameters, and σ is the sigmoid function.

Robust Calibration Strategy. To estimate appropriate weights that remain effective even under potential evasion attacks, we construct an augmented validation set $\mathcal{D}_{\text{val}}$ that includes both standard reference pairs and their adversarially perturbed counterparts (e.g., via paraphrasing or editing). We model the ensemble weights using logistic regression by minimizing the binary cross-entropy loss on $\mathcal{D}_{\text{val}}$.

Decision Rule. To support third-party auditing, the final decision threshold τ is selected to explicitly control the false positive rate (FPR). We calibrate τ on a large held-out set of benign, unwatermarked texts:

$$\tau = \inf \left\{ \gamma \in (0,1) : \widehat{\mathrm{FPR}}(\gamma; \mathcal{D}_{\text{benign}}) \leq \alpha \right\}, \tag{11}$$

where α is the target FPR. The final verdict is given by $\hat{y} = \mathbb{I}(A_{\text{ens}}(t_{\text{q}}) \geq \tau)$, where $\widehat{\mathrm{FPR}}$ denotes the empirical false positive rate. The large sample size of $\mathcal{D}_{\text{benign}}$ ensures that the estimated threshold is statistically significant and reliable.

TTP-Detect offers significant extensibility. As new detection techniques emerge, they can be seamlessly integrated as additional feature channels in $\mathbf{A}$ without altering the core architecture.

5 Experiments

5.1 Experimental Setup

Models. We employ three families of LLMs to instantiate the different roles in our framework. On the service provider side, we use Llama-3.1-8B (Grattafiori et al., 2024) and OPT-6.7B (Zhang et al., 2022) as generation models. On the detector (TTP) side, we use Qwen2.5-3B (Qwen et al., 2025) as the proxy model for representation extraction, and use Qwen3-1.7B (Yang et al., 2025) as the scoring model in the adaptive rank test module.

Watermarking Schemes. We choose UPV (Liu et al., 2024a), which supports detection without access to a secret key, as the black-box watermark detection baseline. Under the TTP-Detect framework, we perform black-box detection for seven watermarking schemes, including KGW (Kirchenbauer et al., 2023), Unigram (Zhao et al., 2024), SynthID (Dathathri et al., 2024), SWEET (Lee et al., 2024), Unbiased (Hu et al., 2024), SymMark (Wang et al., 2025a), and MorphMark (Wang et al., 2025b). Details are provided in Appendix B.

Datasets and Evaluation Metrics. To stay aligned with standard evaluation practices in LLM watermark detection, we adopt the widely used news-like C4 (Raffel et al., 2020) and long-form OpenGen (Krishna et al., 2023) datasets to assess detection performance under both in-domain and distribution-shifted settings. We report TPR, TNR, best F1, and AUROC as the primary metrics. For robustness under different attack settings, we additionally plot ROC curves, which characterize the trade-off between FPR and TPR across varying decision thresholds.

Implementation Details. We treat each pair of (watermarking scheme, generation model) as an independent setting. To train the proxy model, we construct an instruction-tuning corpus from C4 by sampling 6,000 segments, taking the first 30 tokens as prompts, and generating matched watermarked and unwatermarked continuations that are then wrapped into an instruction template. Separately, we use additional C4-held-out data to tune module-specific hyperparameters and to estimate the ensemble weights from per-module scores. All reported numbers are computed on disjoint evaluation data, with OpenGen used exclusively to assess out-of-domain generalization. During detection, we use the first 50 tokens of the queried text as a prompt and sample $N{=}16$ reference texts under both watermarked and unwatermarked conditions. Further implementation details are provided in Appendix C, and hyperparameter analysis is provided in Appendix F.

5.2 Main Results

Table 1 shows that watermark detection performance under the TTP-Detect framework varies across watermark families. Specifically, although we include UPV as a baseline, its detector network is trained for a particular generator network, which is more lenient than TTP-Detect's settings.

For classic logits-based schemes, KGW_{TTP} achieves F1 and AUC on both C4 and OpenGen that exceed or closely match UPV, while $\text{Unigram}_{\text{TTP}}$ consistently attains AUC no lower than 0.999. These results indicate that the traditional red-green list paradigm induces a clear separation between watermarked and unwatermarked texts, and such signals can be effectively captured by TTP-Detect. For improved schemes SWEET and MorphMark, although they incorporate dynamic thresholding and adaptive watermark strength, they remain fundamentally rooted in the red-green list paradigm and therefore still exhibit salient, detectable signals. Notably, $\text{SWEET}_{\text{TTP}}$ incurs only a 0.38% average AUC drop compared to KGW_{TTP}.

For the synthetic watermark SymMark, TTP-Detect reaches perfect performance (F1 and AUC of **1.000**) on all datasets and models, demonstrating strong perceptibility to its synthesized watermark patterns. For distribution-preserving methods such as SynthID and Unbiased, we evaluate under their

Watermark	C4 Dataset								OpenGen Dataset							
	Llama-3.1-8B				OPT-6.7B				Llama-3.1-8B				OPT-6.7B			
	TPR	TNR	F1	AUC	TPR	TNR	F1	AUC	TPR	TNR	F1	AUC	TPR	TNR	F1	AUC
UPV	0.985	0.980	0.983	0.991	0.990	0.990	0.990	0.998	0.995	0.960	0.978	0.994	0.995	0.980	0.988	0.996
Logits-based Schemes																
KGW_{TTP}	0.980	0.980	0.980	0.998	1.000	0.990	0.995	0.999	0.980	0.950	0.966	0.993	0.980	1.000	0.990	0.999
$\text{Unigram}_{\text{TTP}}$	1.000	0.990	0.995	0.999	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
$\text{SWEET}_{\text{TTP}}$	0.985	0.965	0.975	0.997	0.995	0.995	0.995	0.999	0.980	0.920	0.951	0.981	0.980	0.970	0.975	0.997
$\text{MorphMark}_{\text{TTP}}$	0.945	0.965	0.955	0.981	0.990	0.940	0.966	0.993	0.925	0.835	0.885	0.942	0.925	0.965	0.944	0.989
Distribution-preserving Schemes																
$\text{Unbiased}_{\text{TTP}}$	0.870	0.845	0.859	0.911	0.865	0.870	0.867	0.920	0.795	0.775	0.787	0.838	0.775	0.760	0.769	0.826
$\text{SynthID}_{\text{TTP}}$	0.865	0.930	0.894	0.938	0.910	0.905	0.908	0.957	0.875	0.785	0.838	0.896	0.860	0.855	0.858	0.924
Synthetic Scheme																
$\text{SymMark}_{\text{TTP}}$	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 1: Watermark detection performance of various watermarking schemes under TTP-Detect framework.

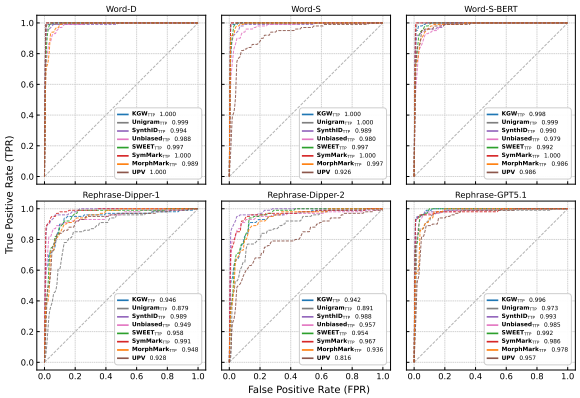
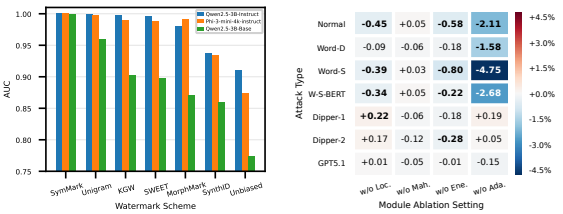


Figure 2: Robustness across various attack types.



(a) Proxy model ablation (b) Module ablation

Figure 3: Ablation study for (a) proxy model and (b) relative measurement modules.

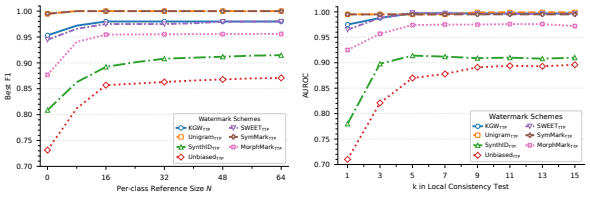
original non-distortion setting. Compared to biased watermarking, the induced distributional differences between watermarked and unwatermarked texts are substantially weaker, making detection more challenging; nonetheless, TTP-Detect is still able to capture measurable discrepancies.

5.3 Robustness to Attacks

To evaluate the robustness of TTP-Detect, we attack watermarked texts generated by Llama-3.1-8B on C4 with six transformations grouped into two categories: *editing*-based attacks (Word-D, Word-S, Word-S-BERT) and *paraphrasing*-based attacks (Dipper-1, Dipper-2, GPT-5.1). Figure 2 reports the main results, while Appendix D provides the full attack configurations. The ROC curves and AUC scores show that TTP-Detect remains highly robust across a broad range of attacks, e.g., achieving an average AUC of 0.980 for KGW_{TTP}. This robustness stems from the framework's relative measurement formulation and its integration of multiple complementary modules, which reduces reliance on any single feature or failure mode. Among all evaluated perturbations, Dipper-1 and Dipper-2 are the most disruptive. In particular, $\text{Unigram}_{\text{TTP}}$ experiences a more noticeable degradation, which we attribute to the proxy model's strong specialization to Unigram's fixed vocabulary partitioning; consequently, proxy-dependent modules become less reliable under paraphrases that introduce substantial lexical substitutions and sentence-level reorganization. Nevertheless, under most attack types and for the majority of watermarking schemes, TTP-Detect tends to outperform UPV, suggesting improved robustness under realistic post-generation edits.

5.4 Ablation Analysis

Impact of Proxy Model Choice. In our main experiments, we use LoRA-tuned Qwen2.5-3B-Instruct as the proxy model. To assess the sensitivity of this choice, we replace it with a comparably sized model from a different family, Phi-3-mini-4k-instruct (Abdin et al., 2024), while keeping the rest of the pipeline unchanged. As shown in Figure 3a, the two proxies yield very similar performance across watermarking schemes, with the average AUC differing by less than 1%, suggesting that TTP-Detect is not tied to any particular model family. We further test whether proxy fine-tuning is essential by using an untuned Qwen2.5-3B-Base model as the proxy, where we extract text repre-



(a) Size of reference set (b) Hyperparameter k

Figure 4: (a) Impact of the size of reference set; (b) hyperparameter analysis of k in Local Consistency Test.

Phase/Module	Model for Test	Time (s)
Service Provider Side		
Generation of Reference Set	Llama-3.1-8B	6.1374
TTP Side		
Representation Extraction	Qwen2.5-3B	0.8348
Local Consistency Test	–	0.0012
Relative Mahalanobis Score	–	0.0165
Relative Energy Score	–	0.0113
Adaptive Rank Test	Qwen3-1.7B	1.7328

Table 2: Efficiency Analysis.

sentations by mean-pooling the last-layer hidden states. Although performance degrades compared to instruction-tuned proxies, TTP-Detect remains effective without proxy training. In particular, under $\text{SymMark}_{\text{TTP}}$ and $\text{Unigram}_{\text{TTP}}$, the AUC drop relative to the instruction-tuned proxy is below 4%, indicating that proxy fine-tuning primarily amplifies watermark-sensitive cues, rather than being a prerequisite for the relative measurement.

Impact of Relative Measurement Modules. We conduct a leave-one-out ablation: for each watermarking scheme, we remove one of the four modules and measure the change in AUC under every attack scenario, while keeping all other settings fixed. Figure 3b visualizes the ablation results for $\text{SynthID}_{\text{TTP}}$, where each cell denotes the AUC change relative to the full system. The Adaptive Rank Test module is the most influential component: removing it yields the largest AUC drops in most scenarios, even though it can slightly improve performance under the two Dipper paraphrase settings. This pattern highlights the intended complementarity of our design: a module may be particularly valuable for broad robustness even if it is not uniformly beneficial under every specific attack. We provide the corresponding analyses for the remaining watermarking schemes in Appendix G.

5.5 Further Analysis

Impact of the Size of Reference Set. We vary the per-class reference set size as $N \in \{0, 8, 16, 32, 48, 64\}$ and report the resulting best F1 in Figure 4a. Here, N=0 corresponds to a reference-free setting, where we directly rely on the tuned proxy to classify the query based on its output without relative comparisons. Overall, incorporating a reference set consistently improves performance across all watermark schemes. For schemes with stronger signals (e.g., Unigram and SymMark), the proxy already exhibits substantial separability, and the gains from increasing N are modest. In contrast, for distribution-preserving schemes (e.g., Unbiased and SynthID), using N=8 and N=16 yields clear benefits over N=0, improving average best F1 by more than 5% and 10%, respectively. Beyond N=16, the improvements taper off, indicating diminishing returns. We therefore adopt N=16 as a practical trade-off between effectiveness and cost.

Efficiency Analysis. Table 2 reports the efficiency results under a reference set size of $N = 16$ and a generation length of 300 tokens. The dominant cost of TTP-Detect comes from invoking the service provider to generate the reference samples. With batched inference on Llama-3.1-8B, this step completes in 6.14s. On the TTP side, modules can be executed in parallel. The modules that rely on proxy-model representation extraction (Local Consistency Test and the Mahalanobis/Energy scores) finish within 1s in total, while the Adaptive Rank Test completes in 1.74s. Overall, the full TTP-side computation stays below 2s.

Hyperparameter Analysis. Figure 4b analyzes the hyperparameter k in the Local Consistency Test module; additional details and other hyperparameters are discussed in Appendix F.

6 Conclusion

This paper presents TTP-Detect, a non-intrusive black-box verification framework that decouples detection from watermark design. By casting verification as a relative hypothesis test using paired watermarked/unwatermarked reference samples, it avoids reliance on private keys and scheme-specific detectors. Experiments demonstrate strong detectability and robustness under realistic attacks across representative watermark families and generation models. These gains stem from a proxy representation space and complementary relative measurements, combined via a lightweight calibrated ensemble. Efficiency results show modest TTP-side overhead, supporting scalable third-party watermark audits.

Limitations

TTP-Detect decouples watermark verification from injection, providing a practical pathway toward neutral, third-party auditing in black-box settings. While the framework is agnostic to the watermarking algorithm, underlying model and secret key, its gains can be less pronounced for schemes explicitly engineered to minimize bias or distortion. This is because TTP-Detect's relative measurement modules primarily exploit observable distributional and generative discrepancies between watermarked and non-watermarked outputs; when such discrepancies are intentionally suppressed, the available signal becomes inherently weaker. A promising direction is to broaden the measurement suite with additional relative tests that capture subtler watermark-induced effects. Nonetheless, TTP-Detect constitutes an important step toward deployable black-box watermark verification as an auditing primitive, helping bridge the gap between watermark design and real-world accountability.

References

Scott Aaronson. 2023. Watermarking of large language models. Talk at the Large Language Models and Transformers Workshop, Simons Institute for the Theory of Computing.

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Ruibo Chen, Yihan Wu, Junfeng Guo, and Heng Huang. 2025. Improved unbiased watermark for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20587–20601, Vienna, Austria. Association for Computational Linguistics.

Miranda Christ, Sam Gunn, and Or Zamir. 2024. Undetectable watermarks for language models. In *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pages 1125–1139. PMLR.

Amirhossein Dabiriaghdam and Lele Wang. 2025. SimMark: A robust sentence-level similarity-based watermarking algorithm for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 30773–30794, Suzhou, China. Association for Computational Linguistics.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, and 1 others. 2024. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823.

Haohua Duan, Liyao Xiang, and Xin Zhang. 2025. Pvmark: Enabling public verifiability for llm watermarking schemes. *Preprint*, arXiv:2510.26274.

Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. 2025. Publicly-detectable watermarking for language models. *Preprint*, arXiv:2310.18491.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

Jiayi Fu, Xuandong Zhao, Ruihan Yang, Yuansen Zhang, Jiangjie Chen, and Yanghua Xiao. 2024. GumbelSoft: Diversified language model watermarking via the GumbelMax-trick. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5791–5808, Bangkok, Thailand. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. 2024. On the learnability of watermarks for language models. In *The Twelfth International Conference on Learning Representations*.

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. 2024. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4115–4129, Bangkok, Thailand. Association for Computational Linguistics.

Abe Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. 2024a. 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, Mexico City, Mexico. Association for Computational Linguistics.

Abe Hou, Jingyu Zhang, Yichen Wang, Daniel Khashabi, and Tianxing He. 2024b. k-SemStamp: A clustering-based semantic watermark for detection of machine-generated text. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1706–1715, Bangkok, Thailand. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2024. Unbiased watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Mingjia Huo, Sai Ashish Somayajula, Youwei Liang, Ruisi Zhang, Farinaz Koushanfar, and Pengtao Xie. 2024. Token-specific watermarking with enhanced detectability and semantic coherence for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Frederick Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2024. 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, Bangkok, Thailand. Association for Computational Linguistics.

Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip S. Yu. 2024a. An unforgeable publicly verifiable watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024b. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Yepeng Liu and Yuheng Bu. 2024. Adaptive text watermark for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. 2024. An entropy-based text watermarking detection method. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11724–11735, Bangkok, Thailand. Association for Computational Linguistics.

Minjia Mao, Dongjun Wei, Zeyu Chen, Xiao Fang, and Michael Chau. 2025. Watermarking large language models: An unbiased and low-risk method. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7939–7960, Vienna, Austria. Association for Computational Linguistics.

George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41.

OpenAI. 2025. GPT-5.1 Model (OpenAI API Documentation). `https://platform.openai.com/docs/models/gpt-5.1`.

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. 2024. MarkLLM: An open-source toolkit for LLM watermarking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 61–71, Miami, Florida, USA. Association for Computational Linguistics.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2024a. A robust semantics-based watermark for large language model against paraphrasing. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 613–625, Mexico City, Mexico. Association for Computational Linguistics.

Yubing Ren, Ping Guo, Yanan Cao, and Wei Ma. 2024b. Subtle signatures, strong shields: Advancing robust and imperceptible watermarking in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5508–5519, Bangkok, Thailand. Association for Computational Linguistics.

Yidan Wang, Yubing Ren, Yanan Cao, and Binxing Fang. 2025a. From trade-off to synergy: A versatile symbiotic watermarking framework for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10306–10322, Vienna, Austria. Association for Computational Linguistics.

Zongqi Wang, Tianle Gu, Baoyuan Wu, and Yujiu Yang. 2025b. MorphMark: Flexible adaptive watermarking for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4842–4860, Vienna, Austria. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Details of Global Geometry Test

Notation. Let $\mathcal{Z}_{\text{o}} = \{z_{\text{o}}^i\}_{i=1}^N$ and $\mathcal{Z}_{\text{wm}} = \{z_{\text{wm}}^i\}_{i=1}^N$ denote the ℓ_2-normalized representations of the non-watermarked and watermarked reference sets, respectively. We write the joint reference set as $\mathcal{X}_{\text{ref}} = \mathcal{Z}_{\text{o}} \cup \mathcal{Z}_{\text{wm}}$ with $|\mathcal{X}_{\text{ref}}| = 2N$.

PCA Projection. To mitigate covariance singularity and reduce noise in high-dimensional spaces, we fit PCA on the centered joint reference set. Let the empirical mean be

$$\mu_{\text{ref}} = \frac{1}{2N} \sum_{z \in \mathcal{X}_{\text{ref}}} z. \tag{12}$$

Let $W \in \mathbb{R}^{d \times d'}$ be the matrix of the top-d' principal directions computed from $\{z - \mu_{\text{ref}} : z \in \mathcal{X}_{\text{ref}}\}$. In practice d' is set to a small value satisfying $d' \leq \min(d, 2N-1)$ to respect the rank constraint after centering. For any representation z, we define its projected vector as

$$\tilde{z} = W^\top (z - \mu_{\text{ref}}) \in \mathbb{R}^{d'}. \tag{13}$$

We then compute class means in the subspace:

$$\tilde{\mu}_c = \frac{1}{N} \sum_{z \in \mathcal{Z}_c} \tilde{z}, \quad c \in \{\text{o}, \text{wm}\}. \tag{14}$$

Regularized Mahalanobis Distance. Let $\tilde{\Sigma}_c$ be the empirical covariance of projected vectors in class c:

$$\tilde{\Sigma}_c = \frac{1}{N-1} \sum_{z \in \mathcal{Z}_c} (\tilde{z} - \tilde{\mu}_c)(\tilde{z} - \tilde{\mu}_c)^\top. \tag{15}$$

We apply shrinkage regularization to ensure invertibility:

$$\hat{\Sigma}_c = \tilde{\Sigma}_c + \alpha \cdot \frac{\text{tr}(\tilde{\Sigma}_c)}{d'} \mathbf{I}_{d'}, \tag{16}$$

where α is a hyperparameter and $\mathbf{I}_{d'}$ is the identity matrix. The squared Mahalanobis distance used in the main text is

$$\delta^2_{\hat{\Sigma}_c}(\tilde{z}, \tilde{\mu}_c) = (\tilde{z} - \tilde{\mu}_c)^\top \hat{\Sigma}_c^{-1} (\tilde{z} - \tilde{\mu}_c). \tag{17}$$

Accordingly, the Mahalanobis contrast in the main text is

$$\Delta_{\text{Mah}}(z_{\text{q}}) = \delta^2_{\hat{\Sigma}_{\text{o}}}(\tilde{z}_{\text{q}}, \tilde{\mu}_{\text{o}}) - \delta^2_{\hat{\Sigma}_{\text{wm}}}(\tilde{z}_{\text{q}}, \tilde{\mu}_{\text{wm}}). \tag{18}$$

Energy Distance Definition. To avoid parametric assumptions, we additionally compute a non-parametric energy statistic. We use the following query-to-set energy distance:

$$\delta_{\text{Ene}}(z_{\text{q}}, \mathcal{Z}_c) = \frac{2}{N} \sum_{z \in \mathcal{Z}_c} \|z_{\text{q}} - z\|_2 - \frac{1}{N^2} \sum_{z, z' \in \mathcal{Z}_c} \|z - z'\|_2, \tag{19}$$

where the first term measures the average potential energy between the query and the set, and the second term subtracts the internal dispersion of the set itself. The energy contrast in the main text is then

$$\Delta_{\text{Ene}}(z_{\text{q}}) = \delta_{\text{Ene}}(z_{\text{q}}, \mathcal{Z}_{\text{o}}) - \delta_{\text{Ene}}(z_{\text{q}}, \mathcal{Z}_{\text{wm}}). \tag{20}$$

Score Normalization. As stated in Sec. 4.3.2, we map both geometry contrasts to $[0, 1]$ using a sigmoid:

$$\begin{aligned} A_{\text{Mah}}(z_{\text{q}}) &= \sigma\big(\beta_{\text{Mah}} \Delta_{\text{Mah}}(z_{\text{q}})\big), \\ A_{\text{Ene}}(z_{\text{q}}) &= \sigma\big(\beta_{\text{Ene}} \Delta_{\text{Ene}}(z_{\text{q}})\big), \end{aligned} \tag{21}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$. β_{Mah} and β_{Ene} are temperature (scale) parameters used to avoid saturation and make the two statistics numerically comparable before ensembling.

B Details of Watermarking Schemes

We take UPV (Liu et al., 2024a) as our main comparison baseline and evaluate the black-box detection of seven target watermarking schemes under our TTP-Detect Framework.

Baseline. We take UPV (Liu et al., 2024a) as our main comparison baseline. Since the original paper does not release pre-trained watermark generation/detection networks for our selected generation models and datasets, we re-train UPV based on the official implementation. Concretely, we train UPV's generator and detector networks on C4 and OpenGen for Llama-3.1-8B and OPT-6.7B, respectively. Following UPV's design, we set the `bit_number` to match the corresponding vocabulary size (17 for Llama-3.1-8B and 16 for OPT-6.7B), while keeping the remaining hyperparameters consistent across all settings.

We also noticed that there are two more works that are related to third-party or publicly verifiable watermarking, namely PVMark (Duan et al., 2025) and PDW (Fairoze et al., 2025). However, we didn't choose them as baseline for reasons.

- PVMark is not a standalone public detector, but a zero-knowledge-proof (ZKP) plugin that makes a

keyed watermark detector publicly verifiable: an owner/authorized party runs watermark detection without revealing the secret key, and third parties mainly verify the correctness proof rather than performing key-free detection themselves. Moreover, PVMark requires scheme-specific ZKP-friendly redesign of the original embedding/detection pipeline, which changes the implementation assumptions and is orthogonal to our black-box, parameter-free third-party detection setting. Finally, while PVMark reports implementations across multiple languages, the paper does not provide an officially released open-source repository for reproduction.

- PDW proposes a new publicly-detectable watermarking scheme that embeds a publicly verifiable cryptographic signature into model outputs via a rejection-sampling-based decoding procedure. This setting assumes the service provider is willing to (i) modify the generation pipeline to perform signature-conditioned sampling and (ii) publish the public verification key required for detection. As a result, PDW is not a key-free third-party verification baseline for existing private-key watermarks under black-box/API access; rather, it changes the watermarking design and deployment assumptions. Moreover, its signature-driven rejection sampling introduces non-trivial generation overhead and is difficult to align with our cost-aware evaluation protocol that treats the provider as a standard text-generation API.

Target Watermarking Schemes. We evaluate the black-box detectability of KGW (Kirchenbauer et al., 2023), Unigram (Zhao et al., 2024), SynthID (Dathathri et al., 2024), SWEET (Lee et al., 2024), Unbiased (Hu et al., 2024), SymMark (Wang et al., 2025a) and MorphMark (Wang et al., 2025b) under the TTP-Detect framework. To align with prior work, we adopt the same hyperparameter settings as the open-source MarkLLM repository (Pan et al., 2024) for all schemes. For SymMark, we follow the original paper and use UniEXP, a hybrid symbiotic configuration that combines Unigram and AAR (Aaronson, 2023), and we adopt the hyperparameters from the authors' official implementation. The complete configurations are summarized in Table 3.

C Further Implementation Details

System Configuration. All experiments are conducted on a CentOS Linux 7 (Core) server which equipped with dual-socket Intel Xeon Platinum 8375C CPUs (64 physical cores, 128 hardware threads in total) and 2 NVIDIA A100 80GB PCIe GPUs. Our software stack is based on Python 3.12.2, PyTorch 2.5.0 with CUDA 12.1, and Transformers 4.52.4.

Training of Proxy Model for Representation Extraction. We adopt Qwen2.5-3B-Instruct (Qwen et al., 2025) as the representation extractor for main experiments and apply LoRA (Hu et al., 2022) fine-tuning using the LLaMAFactory (Zheng et al., 2024) framework. For each (watermarking scheme, generation model) pair, we sample 6,000 segments from C4, use the first 30 tokens as prompts, and query the generator to produce paired watermarked and unwatermarked continuations; we then wrap each completion with an instruction-tuning template to form training instances. The goal is to endow the model with a preliminary ability to distinguish between watermarked and unwatermarked texts, while largely preserving its original semantic understanding capability. The construction of the instruction-tuning dataset is illustrated in Figure 5, and the configuration for fine-tuning is illustrated in Table 4.

Evaluation Details. For each pair of (watermarking scheme, generation model), we construct the test set by sampling held-out instances from C4 and OpenGen that are disjoint from the training and validation splits. Specifically, we take the first 30 tokens of each instance as the prompt and generate both a watermarked and an unwatermarked continuation, forming paired test samples. In the main experiments, for each query text we use its first 50 tokens to generate a reference set of $N{=}16$ watermarked/unwatermarked pairs. For the Local Consistency Test module, we set the number of neighbors in the local support region to $k{=}7$. For the Adaptive Rank Test, we set $\lambda{=}0.6$ to balance overall fluency and local volatility. We report the main results averaged over five test sets, each containing 200 paired watermarked/unwatermarked samples.

D Settings for Robustness Evaluation

We evaluate robustness under six attacks, grouped into two categories: *editing*-based transformations

Algorithm	Source	Parameter	Value
UPV	Liu et al. (2024a)	window_size	3
		layers	5
		use_sampling	"True"
		delta	2.0
		sampling_temp	0.7
		train_num_samples	10000
		bit_number	16(OPT-6.7B) / 17(Llama-3.1-8B)
KGW	Kirchenbauer et al. (2023)	γ (gamma)	0.5
		δ (delta)	2.0
		z_threshold	4.0
		prefix_length	1
		f_scheme	"time"
		window_scheme	"left"
		hash_key	15485863
Unigram	Zhao et al. (2024)	γ (gamma)	0.5
		δ (delta)	2.0
		z_threshold	4.0
		hash_key	15485863
SWEET	Lee et al. (2024)	γ (gamma)	0.5
		δ (delta)	2.0
		z_threshold	4.0
		hash_key	15485863
		prefix_length	1
		entropy_threshold	0.9
SynthID	Dathathri et al. (2024)	ngram_len	5
		sampling_table_size	65536
		sampling_table_seed	0
		watermark_mode	"non-distortionary"
		num_leaves	2
		context_history_size	"mean"
		threshold	0.52
Unbiased	Hu et al. (2024)	type	"gamma"
		n_grid	10
		key	42
		prefix_length	5
		p_threshold	0.0005
SymMark (UniEXP)	Wang et al. (2025a)	γ (gamma)	0.25
		δ (delta)	4.0
		unigram_hash_key	0
		z_threshold	4.0
		prefix_length	4
		exp_hash_key	15485863
		threshold	1e-4
		top_k	0
		token_entropy_threshold	2
		semantic_entropy_threshold	1
		k_means_top_k	64
		k_means_n_clusters	10
MorphMark	Wang et al. (2025b)	γ (gamma)	0.5
		type	"exp"
		k_linear	1.55
		k_exp	1.3
		k_log	2.15
		p_0	0.15
		hash_key	15485863
		prefix_legnth	1
		z_threshold	2.0
		f_scheme	"time"
		window_scheme	"left"

Table 3: Parameter configuration for chosen watermarking algorithms.

Instruction Template for the Training of Representation Extraction Proxy Models

```
System: "You are a helpful assistant that helps people detect whether the text is watermarked or not."
Instruction: "Please determine whether the following text is watermarked or not:"
Input: [Text content to be detected...]
Response_positive: "This text is watermarked."
Response_negative: "This text is unwatermarked."
```

Figure 5: Instruction template used to construct the LoRA fine-tuning data for representation extraction models.

Parameter	Value
finetuning_type	"lora"
lora_rank	8
lora_target	"all"
template	"qwen"
learning_rate	1.0e-4
num_train_epochs	3.0
lr_scheduler_type	"cosine"
warmup_ratio	0.1

Table 4: Parameter configuration for the fine-tuning of proxy model using LLaMAFactory.

that directly modify the surface form, and *paraphrasing*-based transformations that rewrite sentences more globally. All attacks are applied to the watermarked texts before running the detector.

Editing Attacks. These editing attacks perturb the token sequence locally via deletion or substitution while largely preserving the overall semantics, with attack strengths controlled by the corresponding edit ratios.

- **Word-D** randomly deletes words with a deletion ratio of 0.3, namely each word is independently removed with probability 0.3.
- **Word-S** performs synonym substitution based on WordNet (Miller, 1995) with a substitution ratio of 0.5: it first identifies word positions that have available WordNet synonyms, then uniformly samples a subset (up to 50% of all words) and replaces each selected word with a randomly chosen synonym.
- **Word-S-BERT** is a context-aware variant of synonym substitution. It samples word positions in the same manner as Word-S (ratio 0.5), masks each selected position, and uses a masked language model to predict a contextually plausible replacement token; positions where the mask token is truncated are skipped.

Rephrase-GPT5.1 Attack Prompt

```
Prompt: Please rewrite the input paragraph while preserving its original meaning. Output only the rewritten paragraph.\n\n
Input: {text}
```

Figure 6: Prompt used for the Rephrase-GPT5.1 paraphrasing attack.

Paraphrasing Attacks. These paraphrasing attacks rewrite the text more globally, inducing substantial lexical and/or syntactic variation while aiming to preserve the original semantics, thereby posing stronger distribution shifts than local edits.

- **Rephrase-Dipper-1** applies a restatement attack using the Dipper paraphraser (Krishna et al., 2023), emphasizing lexical variation while largely preserving sentence structure. We set `lex_diversity` $= 60$ and `order_diversity` $= 0$.
- **Rephrase-Dipper-2** uses Dipper paraphraser with stronger rewriting level by enabling both lexical and word-order variation, producing more diverse restatements. We set `lex_diversity` $= 60$ and `order_diversity` $= 60$.
- **Rephrase-GPT5.1** paraphrases the text with OpenAI's GPT5.1 API (OpenAI, 2025) (temperature 0.2) instructed to rewrite the input while preserving its original meaning. The prompt for paraphrasing is shown in Figure 6.

E Impact of Text Length

We further analyze how the generation length affects the detection performance of TTP-Detect. To

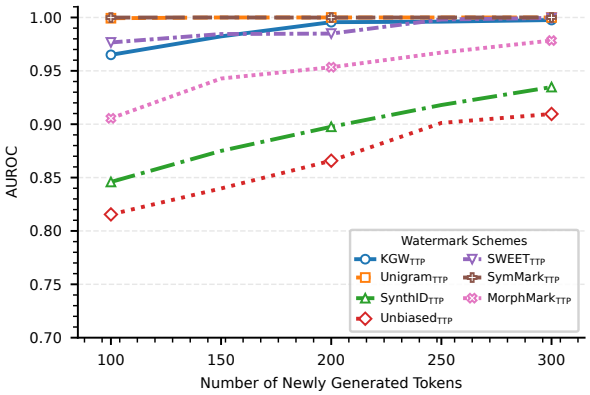


Figure 7: Impact of generated text length.

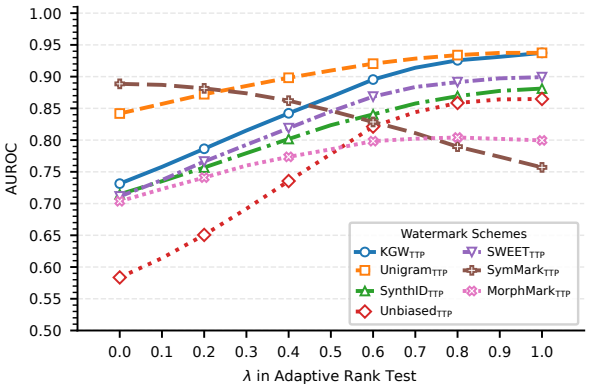


Figure 8: Hyperparameter analysis of λ in Adaptive Rank Test module.

eliminate length as a confounding factor, we always construct the reference set using completions whose length matches the query text, and then vary the number of newly generated tokens L from 100 to 300. Figure 7 reports the resulting AUROC scores.

Overall, longer texts consistently yield better detection across all watermark schemes, but the magnitude of improvement is highly scheme-dependent. For strong-signal schemes, performance saturates quickly: Unigram$_{TTP}$ and SymMark$_{TTP}$ remain near-perfect (≈ 1.0) even at $L = 100$, while KGW$_{TTP}$ and SWEET$_{TTP}$ already exceed 0.95 at $L = 100$ and essentially reach the ceiling by $L = 250$. In contrast, schemes designed to be less distortive or more unbiased exhibit a much stronger dependence on text length.

This trend is consistent with the accumulation effect of token-level watermark signals: as L increases, the expected watermark evidence grows while the variance of relative measurements (e.g., representation- or likelihood-based discrepancies against the reference set) decreases, making the watermarked and unwatermarked distributions more separable. Practically, these results suggest that moderate lengths (around 200 tokens) are sufficient for near-ceiling performance on strong schemes, whereas weaker-signal schemes may require longer generations to achieve comparable reliability—at the cost of increased generation and representation-extraction overhead.

F Hyperparameter Analysis

Impact of k in Local Consistency Test. In the Local Consistency Test, the hyperparameter k specifies the number of nearest neighbors used to define the local support region around the query text's representation. Figure 4b reports the detection performance under N=16 while varying k. Across all seven watermarking schemes, the AUROC score improves as k increases when $k < 7$. When $k \geq 7$, the curves largely saturate and remain stable for most schemes. We therefore set k=7 as the default choice for N=16. For other reference set sizes, we follow a consistent heuristic and set k to the largest odd number not exceeding $N/2$; for example, k=15 for N=32 and k=31 for N=64.

Impact of λ in Adaptive Rank Test. In the Adaptive Rank Test, we use a mixing coefficient $\lambda \in [0, 1]$ to balance two complementary signals: the overall likelihood shift and the local volatility. Figure 8 reports the resulting AUROC when sweeping λ from 0 to 1, where larger λ places more emphasis on the overall shift term.

Overall, most watermarking schemes benefit from increasing λ. In particular, KGW$_{TTP}$, SWEET$_{TTP}$, and SynthID$_{TTP}$ exhibit a largely monotonic improvement as λ increases, suggesting that their detectable signal is dominated by a consistent global bias rather than highly irregular local variations. The effect is most pronounced for Unbiased$_{TTP}$, whose AUROC rises sharply from a weak regime at $\lambda = 0$ to a competitive level at $\lambda \geq 0.6$, indicating that emphasizing the global shift is critical for revealing its more subtle watermark traces. By contrast, SymMark$_{TTP}$ shows the opposite trend: performance degrades steadily as λ increases, implying that its signal is better captured by the volatility component and can be attenuated when the detector over-weights the global shift. MorphMark$_{TTP}$ shows moderate gains up to mid-range λ and then saturates (or slightly drops), suggesting a weaker dependence on this trade-off.

Based on these observations, we set $\lambda = 0.6$

globally as a robust compromise: it achieves near-saturated performance for the majority of schemes while avoiding overly aggressive emphasis on the global-shift term that would substantially harm SymMark$_{\text{TTP}}$. This choice provides a stable default without requiring per-scheme hyperparameter tuning.

G Full Result of Module Ablation

We report the module-level ablation results for the remaining six watermarking schemes in Figure 9. For each scheme, we remove one of the four relative measurement modules at a time and re-evaluate the resulting detector under all attack settings. Each heatmap cell shows the change in AUROC relative to the full configuration, where darker colors indicate a larger impact.

The contribution of each module varies across watermarking schemes, reflecting the joint effect of the watermarking scheme and the specific attack type, as schemes inject watermark signals with different structures and strengths that are disrupted differently by each attack. For instance, on Unigram$_{\text{TTP}}$, removing the Adaptive Rank Test causes an average AUROC decrease of 9.57% under the two Dipper paraphrase attacks, whereas on Unbiased$_{\text{TTP}}$ the same ablation primarily affects editing-based attacks such as Word-D, Word-S, and Word-S-BERT.

While we occasionally observe slight performance gains after removing a module for a specific attack, the overall ensemble remains effective: our weighting strategy is designed to improve average robustness across heterogeneous scenarios, rather than to over-optimize for any single setting.

H Practical Deployment Considerations

H.1 Cooperation Among Parties and Minimal Trust Assumptions

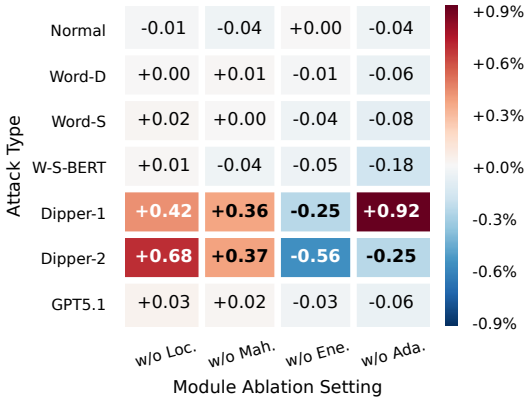
TTP-Detect is designed for a multi-party governance workflow, where watermark injection and verification are separated by design. A feasible deployment requires coordination among (i) a regulator or standard-setting body U, (ii) a compliance-certified auditor D (the trusted third party, TTP), and (iii) the model service provider S. Concretely, the provider exposes a standard text-generation interface with a binary watermark-control flag (watermarked vs. unwatermarked), while keeping the watermark algorithm and secret key confidential. The auditor only relies on observable outputs to (a) construct paired reference sets, (b) calibrate detection statistics, and (c) issue audit reports, without requiring access to the provider's internal watermarking logic. This separation of responsibilities enables neutral, third-party verification while minimizing trust assumptions: the provider controls injection, whereas verification and oversight are performed independently.

H.2 Improving Reference Feasibility via Prompt Reconstruction

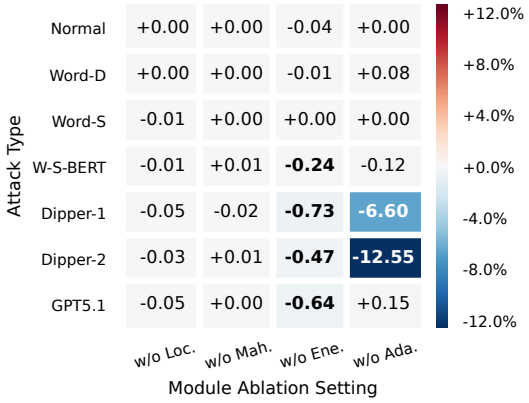
In our experiments, reference texts are generated by prompting the provider S with the prefix of the query text, producing paired watermarked/unwatermarked completions under the same prompt. Although TTP-Detect can remain effective even without reference sets (see Sec. 5.5), tighter semantic alignment between the query and its references is generally beneficial for relative measurements, as it reduces semantic-induced variance that may confound watermark-induced discrepancies.

In real-world settings, however, prefix-only prompting may still yield semantic drift for certain queries (e.g., long-form or instruction-following outputs), because the original user instruction is unavailable to the auditor D. As an optional enhancement, D can reconstruct a plausible instruction that could have produced the query text, using an auxiliary LLM. Given a received query text t_{q}, the auditor first infers an instruction $\hat{p}$ ("reverse-prompting"), and then queries the provider using a composite prompt formed by "$\hat{p}$ + prefix(t_{q})" to generate the paired reference set. This lightweight procedure aims to make reference completions closer to t_{q} in semantics and style, allowing relative measurement modules to focus on watermark differences rather than prompt mismatch.

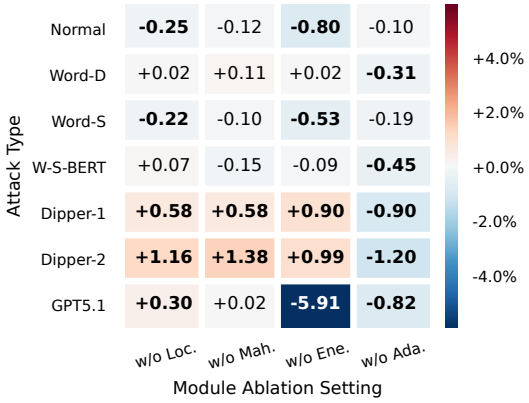
In a pilot study on 50 paired KGW watermarked/unwatermarked samples constructed from ELI5 (Fan et al., 2019) using Llama-3.1-8B, we use the GPT-5.1 API as the auxiliary LLM for prompt reconstruction, which improves AUROC to 0.975 (a +7.7% absolute gain over prefix-only prompting). We emphasize that this result is preliminary: prompt reconstruction can be ambiguous and introduces additional computation, but it provides a practical path to improve reference feasibility when the original prompt is unavailable. The instruction for prompt reconstruction is shown in Figure 10.



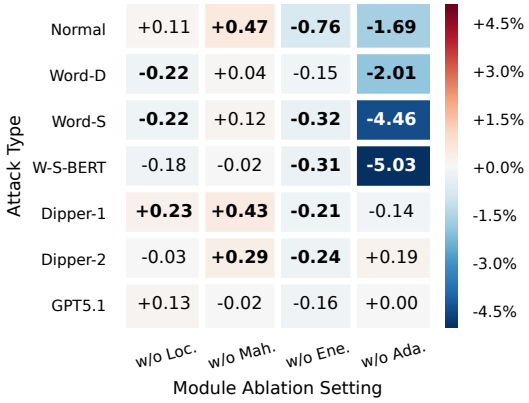
(a) KGW_{TTP}



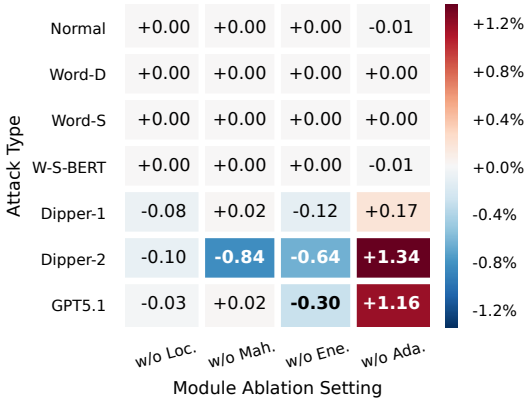
(b) $\text{Unigram}_{\text{TTP}}$



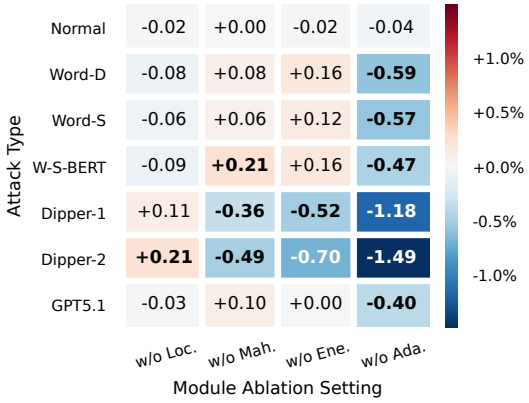
(c) $\text{MorphMark}_{\text{TTP}}$



(d) $\text{Unbiased}_{\text{TTP}}$



(e) $\text{SymMark}_{\text{TTP}}$



(f) $\text{SWEET}_{\text{TTP}}$

Figure 9: Module ablation results for various watermarking schemes.

Prompt Reconstruction Instruction

```
Prompt: You are given a response produced by a large language model, but the original user prompt
is unknown. Infer a plausible prompt that could have elicited the response. Output only the
reconstructed prompt, with no explanation.\n\n Response: {text}
```

Figure 10: Reverse-prompting instruction used to reconstruct an unknown user prompt from the query text.