THE GEOMETRY OF WORD USAGE: TRAINING-FREE DETECTION OF LLM-DRIVEN SOCIAL BOTS

Anonymous authors
Paper under double-blind review

ABSTRACT

The proliferation of large language models (LLMs) has enabled a new generation of highly human-like social bots, posing significant challenges for reliable detection. Social bot detection is a *user-level* problem from many fragmented and multi-topic user histories. However, existing AI-text detectors operate at the passage level and degrade under such conditions. We propose a fundamentally different perspective: instead of modeling text likelihood, we characterize the geometry of repeated contextual word usage across a user's history. Building on this insight, we introduce **CDetect**, a training-free framework that captures user-level semantic concentration. Specifically, CDetect encodes historical posts, aggregates contextualized representations of content words into representative directions, and models their distribution via a mixture of von Mises–Fisher distributions on the unit hypersphere. The resulting mixture-weighted diversity score quantifies whether a user exhibits semantically dispersed (human-like) or concentrated (LLM-like) language usage patterns. On the theoretical side, we establish finite-sample guarantees for the estimated diversity score, explicitly characterizing how vocabulary coverage and repeated contextual occurrences govern its statistical reliability. Extensive experiments on BotSim-24 and two TwiLLM benchmarks demonstrate that CDetect consistently outperforms both training-based and training-free baselines, achieving an average AUROC of 0.9416 and AUPRC of 0.9203. Further analyses confirm its robustness across varying history lengths, text budgets, and modeling choices.

1 INTRODUCTION

The widespread application of large language models (LLMs) has substantially lowered the cost of generating fluent and persuasive social-media content at scale, enabling a new generation of LLM-driven social bots. These accounts can post and interact with users in ways that closely resemble human behavior, and have been used in opinion manipulation, misinformation dissemination, traffic fraud, scams, and harassment on platforms such as X/Twitter and Reddit (Yang et al., 2025).

Detecting such bots is not merely a problem of flagging individual LLM-generated passages, but an account-level problem where the evidence consists of many fragmented and multi-topic user histories (Feng et al., 2022). Individual social posts often provide weak and noisy signals. Consequently, aggregating passage-level scores from LLM-generated text detectors across a user's posts inherits the detectors' sensitivity to short inputs, topic shifts, and discourse discontinuities.

One representative line of work uses zero-shot signals based on log-likelihood or perturbation statistics, such as Binoculars (Hans et al., 2024) and Fast-DetectGPT (Bao et al., 2024). These signals are naturally suited to relatively long and semantically coherent passages. However, in social media, posts and comments are usually short, and the historical content of a user often spans multiple topics, contexts, and interaction scenarios. Naively concatenating such posts introduces abrupt topic shifts and discourse discontinuities, making token-likelihood-based signals unstable (Wu et al., 2024).

Another line of work trains a neural network classifier, including text-based LLM-generated text detectors (Solaiman et al., 2019; Hu et al., 2023) and graph-based social bot detectors (Feng et al., 2021; Liu et al., 2023; Duan et al., 2025). These methods can perform well when labeled data, user metadata, or interaction graphs are available. However, text classifiers may be sensitive to generator updates and rewriting strategies, while graph- and metadata-based detectors depend on platform-specific signals that may be unavailable, inconsistent, or difficult to transfer across settings

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.

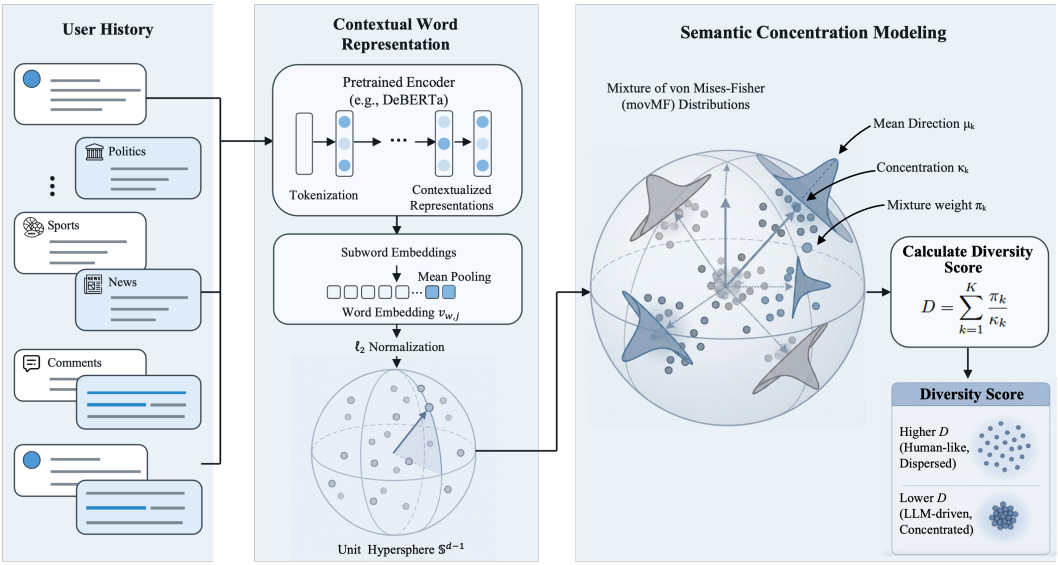


Figure 1: Overview of CDetect. CDetect maps a user's historical posts and comments into contextual word representations and models their distribution on the unit hypersphere with movMF distribution. The resulting diversity score measures user-level semantic concentration to detect.

(Sadasivan et al., 2023; Wu et al., 2024). Their robustness under cross-scenario detection therefore remains uncertain.

Motivated by these challenges, we focus on a semantic regularity that naturally arises from over historical texts at the user level. In contextualized semantic space, *human accounts often exhibit more dispersed semantic organization across their historical content, whereas many LLM accounts show more concentrated semantic patterns.* We quantify this regularity through the representative directions of content words: when the same word appears across a user's history, its contextualized occurrences reveal how broadly or narrowly the account uses that word across topics and situations. This shifts the detection signal from passage-level likelihood to the geometry of word usage across an account's history.

Based on this observation, we propose CDetect, a **C**oncentration-based **Detect**ion method for LLM-driven social bots. CDetect is a geometrically interpretable detector and treats a user history as a collection of content words that recur across multiple contexts and jointly span several semantic modes. We represent each content word by a normalized contextual direction and fit a mixture of von Mises–Fisher (movMF) distributions (Banerjee et al., 2005) on the unit hypersphere. Each mixture component captures one semantic mode of the user's vocabulary, and its concentration parameter measures how tightly the corresponding word directions cluster. CDetect then summarizes the fitted mixture into a single mixture-weighted diversity score, where lower diversity indicates stronger semantic concentration and is more consistent with LLM-driven bot behavior. We summarize our contributions as follows.

- We introduce CDetect, a training-free detector for LLM-driven social bots. Instead of scoring each post independently, CDetect models user-level semantic concentration from repeated contextual word usage. It represents each content word by a normalized contextual direction and fits movMF distribution over these directions, yielding an interpretable diversity score for LLM-driven social bot detection.
- We provide a finite-sample analysis of the estimated diversity score. The analysis separates the error from estimating representative word directions with finite contextual occurrences and the error from fitting the movMF model. The resulting bound gives a sample-size interpretation of CDetect, showing how vocabulary coverage and repeated word occurrences affect the reliability of the estimated score.
- We construct two user-level benchmark variants, TwiLLM-Rewrite and TwiLLM-Interact, to evaluate LLM-driven social bot detection in Twitter-like settings. Built from TwiBot-22, these benchmarks preserve user-level account structure while introducing LLM-generated bot histories through rewriting and interaction-based generation, mitigating the shortage of existing benchmarks in this field.

- We evaluate CDetect on BotSim-24, TwiLLM-Rewrite, and TwiLLM-Interact against training-based, training-free, and graph-based baselines. Across benchmarks, CDetect achieves an average AUROC of 0.9416 and AUPRC of 0.9203, outperforming all the baselines. Additional analyses under varying history budgets, text lengths, concatenation orders, and ablation studies further examine the stability and efficiency of CDetect.

2 Related Work

Training-based LLM-generated text detection methods usually train classifiers to detect AI-generated text from human-written text. Representative methods like OpenAI Detector (Solaiman et al., 2019) based on RoBERTa-based models (Liu et al., 2019) to train a classifier, while later work improves by adversarial learning (Hu et al., 2023), contrastive learning (Guo et al., 2024), prompt reconstruction (Yu et al., 2024).

Training-free LLM-generated text detection methods often use pre-trained language models to construct statistical scores, including likelihood, entropy (Ippolito et al., 2020), and log-rank (Gehrmann et al., 2019). Later methods develop more structured statistical criteria, such as DetectGPT (Mitchell et al., 2023) with perturbation-estimated probability curvature, Fast-DetectGPT (Bao et al., 2024) with efficient conditional probability curvature, Binoculars (Hans et al., 2024) with cross-model perplexity, and DNA-DetectLLM (Zhu et al., 2025) with a mutation-repair scoring paradigm. These methods have shown strong performance in passage-level AI-text detection, but they are usually designed for isolated text passages rather than long-term user-level social-media histories.

Graph-based social bot detection methods turn to model relational structures together with textual or profile information. BotRGCN (Feng et al., 2021) models heterogeneous user relations with relational graph convolution and encodes textual content with pre-trained language models. BotMoE (Liu et al., 2023) integrates textual, metadata, and graph signals through community-aware mixtures of modal-specific experts. Feng et al. (2024) shows that LLMs can both assist bot detection and make bot-generated content more human-like. BotDMM (Duan et al., 2025) further adapted multi-modal bot detection to the LLM-driven setting by incorporating LLM-related textual regularity, semantic representations, metadata, and graph structures. However, these detectors typically require account metadata, behavioral traces, or interaction graphs, whose availability and format may vary across platforms and deployment scenarios.

3 Method

3.1 Problem Setup

We address user-level detection of LLM-driven social bots from historical textual content on social media. Let u denote a target user and $\mathcal{S}_u = \{s_1, s_2, \ldots, s_T\}$ denote the user's publicly available historical texts, where each s_t is a single post or comment and T is the number of these texts. Our goal is to predict, from $\mathcal{S}_u$ alone, whether u is a human account or an LLM-driven bot. For notational brevity, this section describes the procedure for a single user and omits the user index.

We rely exclusively on textual content for three reasons. (1) LLM intervention occurs at content generation, so text is the channel through which the behavior of interest is most directly observable (Feng et al., 2024). (2) Graph- and metadata-based features are complementary but often platform-specific, unavailable to external observers, or inconsistently defined across studies (Feng et al., 2022), limiting reproducibility and platform transfer. (3) Text-only formulation isolates the effect we wish to study and enables a controlled evaluation of whether semantic signatures alone are sufficient for detection.

3.2 Contextual Word Representation

We encode the historical texts $\mathcal{S}$ with a fixed pretrained language encoder[1]. Before representation construction, we filter out stopwords[2] and keep the remaining words as content words. Let $\mathcal{V}$ denote the resulting content-word vocabulary.

[1]We use DeBERTa (He et al., 2023) as default.

[2]We use a NLTK stopword list as default.

Since the encoder produces subword embeddings, we obtain a word-level vector for each occurrence by mean-pooling its subword representations. For the j-th occurrence of a content word w, tokenized into m subwords with contextual embeddings $\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_m\}$, we define its normalized contextual representation as

$$\boldsymbol{x}_{w,j} = \frac{\boldsymbol{v}_{w,j}}{\|\boldsymbol{v}_{w,j}\|_2}, \qquad \boldsymbol{v}_{w,j} = \frac{1}{m}\sum_{\ell=1}^{m} \boldsymbol{h}_\ell. \tag{1}$$

The ℓ_2 normalization projects each occurrence onto the unit hypersphere $\mathbb{S}^{d-1}$ and removes magnitude effects, which is a prerequisite for directional modeling in later stages.

Collecting these vectors yields a word-indexed dictionary

$$\mathcal{W} = \left\{ w : [\boldsymbol{x}_{w,1}, \ldots, \boldsymbol{x}_{w,n_w}] \right\}_{w \in \mathcal{V}}, \tag{2}$$

where n_w is the number of occurrences of w in $\mathcal{S}$.

3.3 Semantic Concentration Modeling

For each content word $w \in \mathcal{V}$, we summarize its contextual usage by a single representative direction $\tilde{\boldsymbol{x}}_w$, which is

$$\bar{\boldsymbol{x}}_w = \frac{1}{n_w}\sum_{j=1}^{n_w} \boldsymbol{x}_{w,j}, \qquad \tilde{\boldsymbol{x}}_w = \frac{\bar{\boldsymbol{x}}_w}{\|\bar{\boldsymbol{x}}_w\|_2}. \tag{3}$$

Here, $\|\bar{\boldsymbol{x}}_w\|_2 \in [0, 1]$ itself carries information about the tightness of w's contextual occurrences, but projecting onto the sphere is sufficient for the global model introduced below. We gather the representative directions into $\mathcal{R} = \{\tilde{\boldsymbol{x}}_w\}_{w \in \mathcal{V}}$, where $N = |\mathcal{V}|$ is the number of remained words.

A unimodal directional model is inadequate when a user's vocabulary spans several loosely related topics. We therefore model $\mathcal{R}$ as a mixture of von Mises-Fisher (movMF) (Banerjee et al., 2005):

$$q_\Theta(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \, f_{\mathrm{vMF}}(\boldsymbol{x}; \boldsymbol{\mu}_k, \kappa_k), \tag{4}$$

where K is the number of components, mixture weights $\pi_k \geq 0$, $\sum_k \pi_k = 1$, mean directions $\boldsymbol{\mu}_k \in \mathbb{S}^{d-1}$, and concentrations $\kappa_k \geq 0$. Each component corresponds to one semantic mode of the user's vocabulary: $\boldsymbol{\mu}_k$ indicates the direction of that mode, κ_k quantifies how tightly the associated words cluster around it, and π_k measures the mode's prevalence. We estimate the movMF parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \kappa_k\}_{k=1}^{K}$ by EM algorithm (Dempster et al., 1977) with a spherical clustering warm start and the details can be seen in Appendix A.

3.4 Detection Score and Decision Rule

After fitting the movMF model, we obtain a set of mixture weights and concentration parameters $\{(\pi_k, \kappa_k)\}_{k=1}^{K}$. We summarize semantic diversity by the mixture-weighted inverse concentration

$$D = \sum_{k=1}^{K} \frac{\pi_k}{\kappa_k}. \tag{5}$$

To prevent numerical overflow, we can add a very small positive constant to κ.

This score measures the average semantic diversity, weighted by the prevalence of each semantic mode. A larger D indicates broader and less concentrated local semantic usage, which is more characteristic of human users. In contrast, a smaller D indicates sharper and more repetitive semantic modes, which is more consistent with LLM-driven bots.

When a binary decision is required, we classify the user by thresholding the diversity score:

$$\hat{y} = \begin{cases} 1, & \text{if } D < \tau, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where $\hat{y} = 1$ denotes an LLM-driven bot, $\hat{y} = 0$ denotes a human account, and τ is a threshold selected on a held-out validation set.

Algorithm 1 movMF-based Semantic Diversity Scoring

Require: Historical text $\mathcal{S} = \{s_1, \ldots, s_T\}$, contextual encoder E, stopword list Ω, number of components K, threshold τ
Ensure: Diversity score D and predicted label $\hat{y}$
Encode $\mathcal{S}$ with E
Remove stopwords and build the word-indexed dictionary $\mathcal{W}$
for each $w \in \mathcal{V}$ **do**
    $\bar{\boldsymbol{x}}_w \leftarrow \frac{1}{n_w} \sum_{j=1}^{n_w} \boldsymbol{x}_{w,j}$
    $\tilde{\boldsymbol{x}}_w \leftarrow \bar{\boldsymbol{x}}_w / \|\bar{\boldsymbol{x}}_w\|_2$
end for
Form the set of representative directions $\mathcal{R} = \{\tilde{\boldsymbol{x}}_w\}_{w \in \mathcal{V}}$
Fit movMF parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \kappa_k\}_{k=1}^K$ using EM algorithm
Compute the detection score $D \leftarrow \sum_{k=1}^K \pi_k / \kappa_k$
if $D < \tau$ **then**
    $\hat{y} \leftarrow 1$
else
    $\hat{y} \leftarrow 0$
end if
return D and $\hat{y}$

The resulting pipeline, summarized in Algorithm 1, is transparent in two respects: the movMF fit exposes an explicit semantic structure per user, and the final score reduces this structure to a single scalar with a direct geometric meaning.

4 Theoretical Analysis

In this section, we provide a finite-sample error bound for the estimated semantic diversity score $\widehat{D}$. Since the threshold rule relies on $\widehat{D}$, bounding $|\widehat{D} - D^\star|$ quantifies when finite user histories yield a score close enough to the optimal score and support a reliable decision. The analysis separates the two sources of error in our pipeline: estimating representative directions from contextual word occurrences, and fitting a movMF model to the resulting word-level directions.

For a retained content word $w \in V$, let

$$m_w = \mathbb{E}[x_{w,j} \mid w], \qquad \rho_w = \frac{m_w}{\|m_w\|_2},$$

where ρ_w is the population representative direction of word w. The empirical direction $\tilde{x}_w$ defined in eq. (3) estimates ρ_w from the observed contextual occurrences of the word.

Let P_u denote the user-level population distribution of representative directions over retained content-word types. We define the population movMF fit as a local population risk minimizer

$$\Theta^\star \in \arg\min_{\Theta \in \mathcal{U}_\Theta} \mathbb{E}_{\rho \sim P_u} \left[-\log q_\Theta(\rho) \right],$$

where $\mathcal{U}_\Theta$ is a local neighborhood of the fitted movMF family. The empirical estimate fitted to $\{\tilde{x}_w\}_{w \in V}$ is denoted by

$$\widehat{\Theta} = \{(\widehat{\pi}_k, \widehat{\mu}_k, \widehat{\kappa}_k)\}_{k=1}^K.$$

The corresponding optimal and estimated diversity scores are

$$D^\star = \sum_{k=1}^K \frac{\pi_k^\star}{\kappa_k^\star}, \qquad \widehat{D} = \sum_{k=1}^K \frac{\widehat{\pi}_k}{\widehat{\kappa}_k}.$$

If a small numerical constant is added to the denominator in implementation, the same analysis applies after replacing κ_k by $\kappa_k + \varepsilon$. We first present the following assumptions:

Assumption 1 (Stable representative directions)**.** *For every retained word w, the normalized contextual occurrences $x_{w,1}, \ldots, x_{w,n_w} \in \mathbb{S}^{d-1}$ are independent conditional on w. Their population mean $m_w = \mathbb{E}[x_{w,j} \mid w]$ satisfies $\|m_w\|_2 \geq a_0 > 0$.*

Assumption 2 (Weakly dependent word-level directions)**.** *The population representative directions $\{\rho_w\}_{w\in V}$ have a common marginal distribution P_u on $\mathbb{S}^{d-1}$. Their dependence is described by a dependency graph over V with maximum degree Δ, where two groups of word directions are independent whenever there is no edge between the two groups in the graph.*

Assumption 1 states that a retained content word has a non-vanishing average semantic direction for the user. This matches our construction in Section 3.2, where stopwords are removed and representative directions are computed only for content words. Assumption 2 treats the retained word types of a user as samples from a user-level semantic distribution. It allows semantic correlations, while requiring that each word is dependent on a limited number of other words.

We bound the error introduced when estimating word directions from finite occurrences. Let

$$n_{\text{eff}} = \left(\frac{1}{N} \sum_{w\in V} \frac{1}{n_w} \right)^{-1}$$

be the harmonic mean of word occurrence counts.

Proposition 1 (Error of representative directions)**.** *Under Assumption 1, for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\frac{1}{N} \sum_{w\in V} \|\tilde{x}_w - \rho_w\|_2 \le \varepsilon_{\text{rep}}(\delta) := \frac{2}{a_0} \sqrt{\frac{2d \log(4dN/\delta)}{n_{\text{eff}}}}.$$

Proposition 1 shows that representative directions become reliable when words are observed in enough contexts. The harmonic mean n_{eff} makes rare word types explicit: if many retained words occur few times, the word-level directions become less stable, even if the total amount of text is large.

Next, we propose the finite-sample bound for the estimated semantic diversity score $\hat{D}$:

Theorem 1 (Finite-sample error bound for movMF semantic diversity)**.** *Suppose Assumptions 1–2 hold, and suppose the local regularity conditions detailed in Appendix B hold for the movMF fit. Then there exists a constant $C_{\mathcal{D}} > 0$, such that with probability at least $1-\delta$,*

$$|\widehat{\mathcal{D}} - \mathcal{D}^\star| \le C_{\mathcal{D}} \left\{ \sqrt{\frac{(\Delta+1)\{Kd\log N + \log(4/\delta)\}}{N}} + \sqrt{\frac{d\log(4dN/\delta)}{n_{\text{eff}}}} \right\}.$$

Equivalently, for fixed K and bounded Δ,

$$|\widehat{\mathcal{D}} - \mathcal{D}^\star| = O_{\mathbb{P}}\left(\sqrt{\frac{d\log N}{N}} + \sqrt{\frac{d\log N}{n_{\text{eff}}}} \right).$$

Theorem 1 gives a sample-size interpretation of the estimated semantic diversity score. It shows that $\widehat{D}$ is reliable when the user history provides both broad vocabulary coverage and sufficient repeated contexts for retained words. $\widehat{D}$ becomes close to $D^\star$ when both the number of retained word types N and the effective occurrence count n_{eff} are sufficiently large.

5 Experiments

5.1 Experimental Settings

Datasets. We evaluate performance in three main datasets (BotSim-24 (Qiao et al., 2025) and TwiLLM-Rewrite (TwiLLM-R) and TwiLLM-Interact (TwiLLM-I)). BotSim-24 contains real Reddit human accounts and LLM agents that autonomously generate posts and comments in a simulated social environment. TwiLLM contains human and constructed LLM accounts on Twitter. TwiLLM-R constructs the bot by rewriting historical texts from rule-based bots with LLMs. TwiLLM-I can simulate interactive LLM-driven bots that generate posts and comments over time by account context and recent generated history. We use TwiLLM Avg. to denote their average performance. The detailed construction process is provided in Appendix D.

Table 1: Main results (AUROC and AUPRC) on BotSim-24, TwiLLM-R, and TwiLLM-I.

Method	BotSim-24		TwiLLM-R		TwiLLM-I		Avg.	
	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
Training-based methods								
OpenAI Detector	0.7652	0.6792	0.3528	0.3124	0.0467	0.2403	0.3882	0.4106
RADAR	0.8763	0.6784	0.4415	0.3953	0.5428	0.3996	0.6202	0.4911
Training-free methods								
Log Likelihood	0.6986	0.4592	0.6449	0.5681	0.7970	0.6728	0.7135	0.5667
Fast-DetectGPT	0.8826	0.8516	0.5749	0.4551	0.4088	0.3303	0.6221	0.5457
Binoculars	0.8665	0.8004	0.6326	0.5545	0.5222	0.4504	0.6737	0.6018
DNA-DetectLLM	0.8898	0.8150	0.6891	0.6061	0.8054	0.7118	0.7948	0.7110
CDetect	**0.9964**	**0.9839**	**0.8543**	**0.8054**	**0.9741**	**0.9717**	**0.9416**	**0.9203**

Baselines. We compare CDetect with two training-based methods (OpenAI Detector (Solaiman et al., 2019) and RADAR (Hu et al., 2023)) and four training-free methods (log likelihood (Hashimoto et al., 2019), Fast-DetectGPT (Bao et al., 2024), Binoculars (Hans et al., 2024), and DNA-DetectLLM (Zhu et al., 2025)). We also compare three representative graph-based bot detection methods (BotRGCN (Feng et al., 2021), BotMoE (Liu et al., 2023), and BotDMM (Duan et al., 2025)) in Appendix F.

Metrics. We adopt Area Under the Receiver Operating Characteristic curve (AUROC) and Area under the Precision–Recall Curve (AUPRC) as threshold-free metrics.

Implementation details. We use DeBERTa-v3-large (He et al., 2023) as the default encoder for CDetect, with layer 24 and $K = 2$ mixture components. For subword handling, we apply mean pooling over subword embeddings to obtain word representations, and remove stopwords using the NLTK stopword list. For training-free baselines, we use Falcon-7B-Instruct and Falcon-7B (Penedo et al., 2023) (if needed) as proxy LLMs. To focus on active accounts, we remove the account with fewer than 5 posts/comments in BotSim-24. More details can be seen in Appendix C.

5.2 Main Results

Table 1 reports the main results on BotSim-24 (Qiao et al., 2025), TwiLLM-R, and TwiLLM-I. CDetect achieves the best overall performance across the three main benchmarks. Averaged over all datatsets, CDetect improves over the strongest evaluated baseline by 0.1468 AUROC and 0.2093 AUPRC, corresponding to relative gains of 18.47% and 29.44%, respectively.

The two TwiLLM variants evaluate complementary LLM-driven bot settings. TwiLLM-R considers a rewriting-based setting, where LLM-generated bot texts inherit the topical context of historical bot-account posts, while TwiLLM-I considers an interaction-based setting, where LLM-driven accounts generate posts and replies over time. CDetect performs strongly on both variants, suggesting that user-level semantic concentration captures information that is not fully reflected by token-level likelihood scores or supervised text-style classifiers.

5.3 Robustness Analysis

Robustness to the amount of user history. We evaluate robustness to the amount of user history using a fixed-cohort prefix protocol. For each dataset, we keep the evaluation users unchanged and vary only the number of visible historical text units. Figure 2 reports AUROC under different history budgets. On BotSim-24, CDetect already performs strongly with a small number of historical units. On TwiLLM, where the task is more challenging, CDetect also shows stable gains as more history becomes available. Overall, the results indicate that CDetect benefits from additional user history while remaining effective under limited historical evidence.

Robustness to text length. For text length robustness, we truncate each user's visible history to a target word number and evaluate each method under the same setting. Figure 3 reports AUROC under different visible text lengths. CDetect remains competitive across word budgets and generally improves as more text becomes available. Both on BotSim-24 and TwiLLM, CDetect remains clearly

ahead of the baselines under all text length settings. These results show that CDetect is robust to varying amounts of text length.

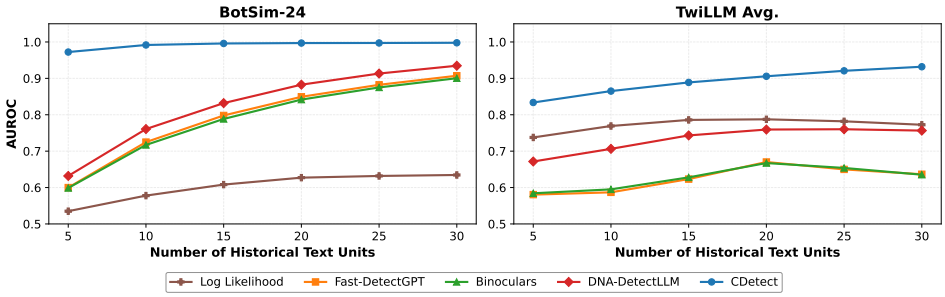


Figure 2: AUROC under different numbers of historical text units.

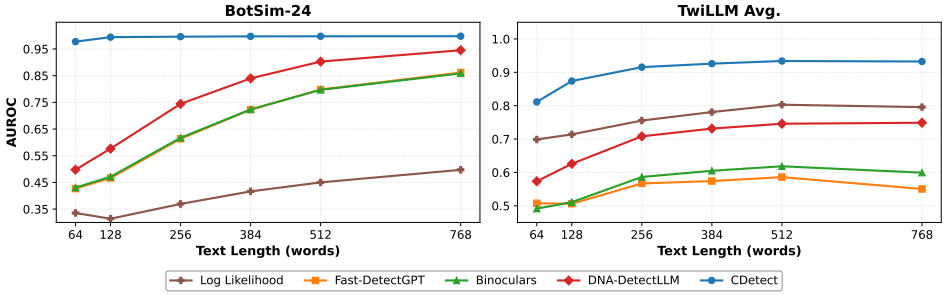


Figure 3: AUROC under different word budgets in the fixed-cohort prefix setting.

Table 2: Comparison of CDetect's performance when using different subword pooling strategies.

Strategy	Word representation	BotSim-24		TwiLLM Avg.	
		AUROC	AUPRC	AUROC	AUPRC
No pooling	Independent subword vectors	0.9947	**0.9843**	0.7838	0.7014
First	First subword vector	0.9964	0.9837	0.9106	0.8740
Last	Last subword vector	0.9952	0.9819	0.8902	0.8554
Max	Element-wise max pooling	0.9955	0.9812	0.8653	0.8180
First+last	Average first and last vectors	**0.9965**	0.9840	0.9101	0.8775
Mean pooling	Average subword vectors	0.9964	0.9839	**0.9142**	**0.8885**

5.4 Ablation and Sensitivity Analysis

Subword pooling. The combination of subwords can affect the quality of word representation (Ács et al., 2021). Since CDetect estimates semantic concentration from word-level representative directions, we compare several subword aggregation strategies in Table 2. Mean pooling shows the most stable overall performance, remaining competitive on BotSim-24 and achieving the best results on TwiLLM. In contrast, treating subwords as independent vectors leads to a clear drop, suggesting that semantic concentration is better estimated from word-level rather than subword-level directions.

Number of mixture components. Table 3 reports the performance of CDetect with different numbers of movMF components. On TwiLLM Avg., increasing K from 1 to 2 improves performance, suggesting that a single semantic mode may be insufficient for more heterogeneous user histories. In contrast, BotSim-24 is already near-saturated with $K = 1$, so the benefit of additional components is limited. Beyond $K = 2$, performance gradually decreases, possibly because overly fine-grained components fragment the available word evidence, increase the variance of the estimation, and introduce unnecessary model complexity.

Layer sensitivity. Since CDetect estimates semantic diversity from contextual embedding geometry, different encoder layers may provide representations with different levels of semantic abstraction. As shown in Table 4, deeper layers generally yield stronger performance.

Table 3: Sensitivity of CDetect to the number of mixture components K.

K	BotSim-24		TwiLLM Avg.	
	AUROC	AUPRC	AUROC	AUPRC
1	0.9953	**0.9843**	0.8861	0.8474
2	**0.9964**	0.9839	**0.9142**	**0.8885**
3	0.9962	0.9785	0.9043	0.8706
4	0.9962	0.9777	0.8949	0.8368

Table 4: Layer sensitivity of CDetect with DeBERTa-v3-large.

Layer	BotSim-24		TwiLLM Avg.	
	AUROC	AUPRC	AUROC	AUPRC
1	0.7411	0.4768	0.7767	0.6048
8	0.9761	0.9073	0.7929	0.6514
16	0.9886	0.9561	0.8014	0.6856
24	**0.9964**	**0.9839**	**0.9142**	**0.8885**

Table 5: CDetect performance under different stopword filtering settings.

Stopword list	Count	BotSim-24		TwiLLM Avg.	
		AUROC	AUPRC	AUROC	AUPRC
Empty list	0	0.9943	**0.9839**	0.8979	0.8563
spaCy	326	0.9951	0.9808	0.9096	0.8833
scikit-learn	318	0.9946	0.9801	0.9065	0.8709
NLTK	179	**0.9964**	**0.9839**	**0.9142**	**0.8885**

Stopword filtering. Table 5 studies the effect of stopword filtering and the sensitivity to different stopword lists. Compared with no filtering (empty list), removing stopwords can improve performance, helping the representative directions focus on more informative content semantics.

Concatenation Order Sensitivity and **Encoder Sensitivity** are provided in Appendix E.

5.5 Efficiency Analysis

In practical scenarios, computational efficiency is important, since each account may contain a long history of posts and comments. Figures 4 and 5 compare the inference time and peak GPU memory usage of different methods on TwiLLM-R. Compared with strong likelihood-based baselines, CDetect reduces inference time by 64.2%–69.5% and peak GPU memory usage by about 91%.

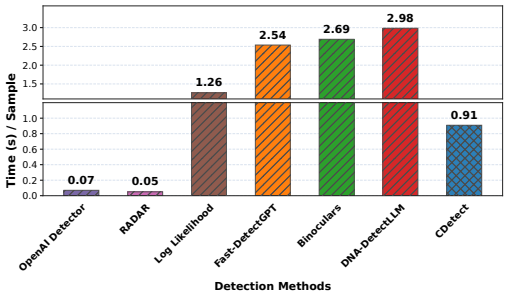


Figure 4: Comparison of time costs for processing a single sample for each method.

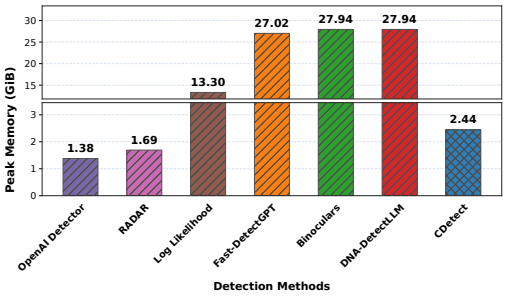


Figure 5: Comparison of peak GPU memory usage across detection methods.

6 Conclusion

We introduced CDetect, a training-free framework for user-level detection of LLM-driven social bots. CDetect captures the geometry of repeated contextual word usage by fitting a mixture of von Mises–Fisher distributions to representative word directions and deriving a mixture-weighted diversity score. Our finite-sample analysis characterizes how vocabulary coverage and repeated contextual occurrences jointly govern score accuracy. Experiments on BotSim-24 and two TwiLLM benchmarks demonstrate improvements over evaluated training-based and training-free baselines, robustness to history length, text budgets, and concatenation order, and lower computational costs than strong likelihood-based baselines. These findings support semantic concentration as an interpretable signal for detecting LLM-driven accounts from historical text.

AI USE STATEMENT

AI tools were used to assist with polishing the manuscript and coding assistance. The authors take full responsibility for the manuscript, code, and the accuracy and integrity of the reported results.

REFERENCES

Judit Ács, Ákos Kádár, and Andras Kornai. Subword pooling makes a difference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 2284–2295, 2021.

Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, Suvrit Sra, and Greg Ridgeway. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6(9), 2005.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*, 2024.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1): 1–22, 1977.

Jinglong Duan, Shiqing Wu, Weihua Li, Quan Bai, Minh Nguyen, and Jianhua Jiang. Botdmm: Dual-channel multi-modal learning for llm-driven bot detection on social media. *Information Fusion*, pp. 103758, 2025.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 55–65, 2019.

Shangbin Feng, Herun Wan, Ningnan Wang, and Minnan Luo. Botrgcn: Twitter bot detection with relational graph convolutional networks. In *Proceedings of the 2021 IEEE/ACM international conference on advances in social networks analysis and mining*, pp. 236–239, 2021.

Shangbin Feng, Zhaoxuan Tan, Herun Wan, Ningnan Wang, Zilong Chen, Binchi Zhang, Qinghua Zheng, Wenqian Zhang, Zhenyu Lei, Shujie Yang, et al. Twibot-22: Towards graph-based twitter bot detection. *Advances in Neural Information Processing Systems*, 35:35254–35269, 2022.

Shangbin Feng, Herun Wan, Ningnan Wang, Zhaoxuan Tan, Minnan Luo, and Yulia Tsvetkov. What does the bot say? opportunities and risks of large language models in social media bot detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3580–3601, 2024.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. Gltr: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 111–116, 2019.

Xun Guo, Shan Zhang, Yongxin He, Ting Zhang, Wanquan Feng, Haibin Huang, and Chongyang Ma. Detective: Detecting ai-generated text via multi-level contrastive learning. *Advances in Neural Information Processing Systems*, 37:88320–88347, 2024.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Spotting llms with binoculars: Zero-shot detection of machine-generated text. In *International Conference on Machine Learning*, pp. 17519–17537. PMLR, 2024.

Tatsunori B Hashimoto, Hugh Zhang, and Percy Liang. Unifying human and statistical evaluation for natural language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1689–1701, 2019.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*, 2023.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. Radar: Robust ai-text detection via adversarial learning. *Advances in neural information processing systems*, 36:15077–15095, 2023.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 1808–1822, 2020.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Yuhan Liu, Zhaoxuan Tan, Heng Wang, Shangbin Feng, Qinghua Zheng, and Minnan Luo. Botmoe: Twitter bot detection with community-aware mixtures of modal-specific experts. In *Proceedings of the 46th International ACM SIGIR conference on research and development in information retrieval*, pp. 485–495, 2023.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International conference on machine learning*, pp. 24950–24962. PMLR, 2023.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.

Boyu Qiao, Kun Li, Wei Zhou, Shilong Li, Qianqian Lu, and Songlin Hu. Botsim: Llm-powered malicious social botnet simulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14377–14385, 2025.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*, 2023.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019.

Junchao Wu, Runzhe Zhan, Derek Wong, Shu Yang, Xinyi Yang, Yulin Yuan, and Lidia Chao. Detectrl: Benchmarking llm-generated text detection in real-world scenarios. *Advances in Neural Information Processing Systems*, 37:100369–100401, 2024.

Junxi Wu, Jinpeng Wang, Zheng Liu, Bin Chen, Dongjian Hu, Hao Wu, and Shu-Tao Xia. Moses: Uncertainty-aware ai-generated text detection via mixture of stylistics experts with conditional thresholds. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5797–5816, 2025.

Kai-Cheng Yang, Onur Varol, Alexander C Nwala, Mohsen Sayyadiharikandeh, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. Social bots: detection and challenges. In *Handbook of computational social science*, pp. 473–491. Edward Elgar Publishing Limited, 2025.

Xiao Yu, Yuang Qi, Kejiang Chen, Guoqiang Chen, Xi Yang, Pengyuan Zhu, Xiuwei Shang, Weiming Zhang, and Nenghai Yu. Dpic: Decoupling prompt and intrinsic characteristics for llm generated text detection. *Advances in Neural Information Processing Systems*, 37:16194–16212, 2024.

Xiaowei Zhu, Yubing Ren, Fang Fang, Qingfeng Tan, Shi Wang, and Yanan Cao. DNA-DetectLLM: Unveiling AI-generated text via a DNA-inspired mutation-repair paradigm. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*. Curran Associates, Inc., 2025.

A Implementation Details of movMF Fitting

This appendix provides the implementation details of the movMF fitting. The final score is computed from the converged soft-assignment movMF model and hard assignments are used only to initialize EM.

Set construction. After contextual encoding, subword aggregation, and intra-type averaging, we obtain a set of normalized representative directions

$$\mathcal{R} = \{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_N\}, \qquad \tilde{\boldsymbol{x}}_i \in \mathbb{S}^{d-1}, \tag{7}$$

where each representative direction corresponds to one retained content word type.

Spherical warm-start initialization. We initialize the movMF model using a spherical k-means procedure. The first center is sampled uniformly from $\mathcal{R}$:

$$\boldsymbol{c}_1 = \tilde{\boldsymbol{x}}_{i_1}. \tag{8}$$

Assume that a set of centers has already been selected. For each representative direction $\tilde{\boldsymbol{x}}_i$, we compute its cosine-distance surrogate to the nearest center as

$$d_i^2 \propto \left(1 - \max_j \tilde{\boldsymbol{x}}_i^\top \boldsymbol{c}_j\right)^2. \tag{9}$$

The next center is then sampled with probability proportional to this quantity:

$$P(\tilde{\boldsymbol{x}}_i \text{ is selected as a new center}) \propto d_i^2. \tag{10}$$

After selecting K initial centers, we run spherical hard reassignment updates. Each representative direction is assigned to its nearest center by cosine similarity:

$$z_i = \arg\max_{k \in \{1,\ldots,K\}} \tilde{\boldsymbol{x}}_i^\top \boldsymbol{c}_k. \tag{11}$$

Each center is then updated by the normalized cluster mean

$$\boldsymbol{c}_k \leftarrow \frac{\sum_{i:z_i=k} \tilde{\boldsymbol{x}}_i}{\left\|\sum_{i:z_i=k} \tilde{\boldsymbol{x}}_i\right\|_2}. \tag{12}$$

We repeat the hard assignment and center update steps until the assignments stabilize. The resulting partition $\{\mathcal{C}_1^{(0)}, \ldots, \mathcal{C}_K^{(0)}\}$ is used only as a warm start for the subsequent soft-assignment EM optimization.

EM initialization. Let $N_k^{(0)} = |\mathcal{C}_k^{(0)}|$ be the size of the k-th initial cluster. We initialize the mixture weight as

$$\pi_k^{(0)} = \frac{N_k^{(0)}}{N}. \tag{13}$$

The initial mean direction is

$$\boldsymbol{\mu}_k^{(0)} = \frac{\sum_{\tilde{\boldsymbol{x}}_i \in \mathcal{C}_k^{(0)}} \tilde{\boldsymbol{x}}_i}{\left\|\sum_{\tilde{\boldsymbol{x}}_i \in \mathcal{C}_k^{(0)}} \tilde{\boldsymbol{x}}_i\right\|_2}. \tag{14}$$

For each initial cluster, we compute the empirical mean resultant length

$$\bar{R}_k^{(0)} = \left\| \frac{1}{N_k^{(0)}} \sum_{\tilde{\boldsymbol{x}}_i \in \mathcal{C}_k^{(0)}} \tilde{\boldsymbol{x}}_i \right\|_2, \tag{15}$$

and initialize the concentration parameter using the high-dimensional approximation

$$\kappa_k^{(0)} \approx \frac{\bar{R}_k^{(0)} \left(d - (\bar{R}_k^{(0)})^2\right)}{1 - (\bar{R}_k^{(0)})^2}. \tag{16}$$

E-step. Given current parameters $\Theta^{(t-1)} = \{\pi_k^{(t-1)}, \boldsymbol{\mu}_k^{(t-1)}, \kappa_k^{(t-1)}\}_{k=1}^K$, the responsibility of component k for representative direction $\tilde{\boldsymbol{x}}_i$ is

$$\gamma_{ik}^{(t)} = \frac{\pi_k^{(t-1)} C_d(\kappa_k^{(t-1)}) \exp\left(\kappa_k^{(t-1)} (\boldsymbol{\mu}_k^{(t-1)})^\top \tilde{\boldsymbol{x}}_i\right)}{\sum_{r=1}^K \pi_r^{(t-1)} C_d(\kappa_r^{(t-1)}) \exp\left(\kappa_r^{(t-1)} (\boldsymbol{\mu}_r^{(t-1)})^\top \tilde{\boldsymbol{x}}_i\right)}. \tag{17}$$

By construction, $\sum_{k=1}^K \gamma_{ik}^{(t)} = 1$ for each representative direction.

M-step. Given the soft responsibilities, the effective size of component k is

$$N_k^{(t)} = \sum_{i=1}^N \gamma_{ik}^{(t)}, \tag{18}$$

and the updated mixture weight is

$$\pi_k^{(t)} = \frac{N_k^{(t)}}{N}. \tag{19}$$

The mean direction is updated as

$$\boldsymbol{\mu}_k^{(t)} = \frac{\sum_{i=1}^N \gamma_{ik}^{(t)} \tilde{\boldsymbol{x}}_i}{\left\| \sum_{i=1}^N \gamma_{ik}^{(t)} \tilde{\boldsymbol{x}}_i \right\|_2}. \tag{20}$$

We then compute the weighted mean resultant length

$$\bar{R}_k^{(t)} = \left\| \frac{\sum_{i=1}^N \gamma_{ik}^{(t)} \tilde{\boldsymbol{x}}_i}{\sum_{i=1}^N \gamma_{ik}^{(t)}} \right\|_2, \tag{21}$$

and update the concentration parameter using

$$\kappa_k^{(t)} \approx \frac{\bar{R}_k^{(t)} \left(d - (\bar{R}_k^{(t)})^2\right)}{1 - (\bar{R}_k^{(t)})^2}. \tag{22}$$

Stopping criterion. The data log-likelihood under the current parameters is

$$\mathcal{L}\left(\Theta^{(t-1)}\right) = \sum_{i=1}^N \log \left(\sum_{k=1}^K \pi_k^{(t-1)} f_{\text{vMF}}(\tilde{\boldsymbol{x}}_i; \boldsymbol{\mu}_k^{(t-1)}, \kappa_k^{(t-1)}) \right). \tag{23}$$

We terminate optimization when

$$\frac{\left|\mathcal{L}(\Theta^{(t-1)}) - \mathcal{L}(\Theta^{(t-2)})\right|}{\max\left(1, \left|\mathcal{L}(\Theta^{(t-2)})\right|\right)} < \text{tol}, \tag{24}$$

or when the maximum number of iterations is reached. In implementation, we evaluate the log-likelihood immediately after the E-step using the current parameters and compare it with that of the previous iteration. This choice only affects when convergence is checked and does not change the parameter update equations.

B PROOFS

For simplicity, we index retained words by $i = 1, \ldots, N$ and write $\rho_i = \rho_{w_i}$, $\tilde{x}_i = \tilde{x}_{w_i}$, and $n_i = n_{w_i}$. All constants below may depend on fixed regularity constants of the movMF family, but not on N or on the occurrence counts $\{n_i\}_{i=1}^N$.

We first state the formal local-stability condition used in the proof of Theorem 1. This is the appendix-level version of the informal condition in the main text that the population movMF fit is locally non-degenerate and that the EM solution lies in its stable local basin.

Let P_u be the user-level population distribution in Assumption 2. The population movMF fit is defined as a local minimizer of the population negative log-likelihood,

$$\Theta^\star \in \arg\min_{\Theta \in \mathcal{U}_\Theta} \mathbb{E}_{\rho \sim P_u} \left[-\log q_\Theta(\rho) \right].$$

Let $\vartheta \in \mathbb{R}^p$ be a smooth local parameterization of the constrained movMF parameter $\Theta = \Theta(\vartheta)$ around $\Theta^\star$, where $p = O(Kd)$, and let $\vartheta^\star$ satisfy $\Theta(\vartheta^\star) = \Theta^\star$. Define

$$\ell_\vartheta(x) = -\log q_{\Theta(\vartheta)}(x), \qquad M(\vartheta) = \mathbb{E}_{\rho \sim P_u}[\ell_\vartheta(\rho)].$$

Assumption 3 (Local regularity of the movMF fit)**.** *There exists a neighborhood $\mathcal{U}$ of $\vartheta^\star$ such that the following hold:*

- *$\vartheta^\star$ is the locally unique minimizer of M in $\mathcal{U}$, up to the permutation of mixture components, and M is locally strongly convex:*
$$\langle \nabla M(\vartheta) - \nabla M(\vartheta^\star), \vartheta - \vartheta^\star \rangle \geq \lambda \|\vartheta - \vartheta^\star\|_2^2 \qquad \text{for all } \vartheta \in \mathcal{U}.$$
- *the mixture weights and concentrations are non-degenerate in $\mathcal{U}$: for constants $\pi_0 > 0$ and $0 < \kappa_0 < \kappa_1 < \infty$,*
$$\pi_k \geq \pi_0, \qquad \kappa_0 \leq \kappa_k \leq \kappa_1 \qquad \text{for all } k = 1, \ldots, K.$$
- *The score function*
$$s_\vartheta(x) = \nabla_\vartheta \ell_\vartheta(x)$$
is bounded and Lipschitz in both the parameter and the input direction: there exist constants $B, L_\vartheta, L_x < \infty$ such that, for all $\vartheta, \vartheta' \in \mathcal{U}$ and $x, x' \in \mathbb{S}^{d-1}$,
$$\|s_\vartheta(x)\|_2 \leq B,$$
$$\|s_\vartheta(x) - s_{\vartheta'}(x)\|_2 \leq L_\vartheta \|\vartheta - \vartheta'\|_2,$$
and
$$\|s_\vartheta(x) - s_\vartheta(x')\|_2 \leq L_x \|x - x'\|_2.$$

We treat the EM output as a local stationary solution of the empirical movMF objective in the stable neighborhood $\mathcal{U}$. If EM is stopped with numerical tolerance ξ, the bound gains an additional $O(\xi)$ term. Because finite mixtures are identifiable only up to a permutation of component labels, all parameter comparisons in the proof are understood after the best label permutation. This is only a proof device: the diversity score $\sum_k \pi_k/\kappa_k$ itself is invariant to component ordering.

B.1 Proof of Proposition 1

Let

$$\bar{x}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{i,j}, \qquad m_i = \mathbb{E}[x_{i,j} \mid i], \qquad \rho_i = \frac{m_i}{\|m_i\|_2}.$$

For any coordinate $r \in \{1, \ldots, d\}$, since $x_{i,j,r} \in [-1, 1]$, Hoeffding's inequality gives

$$\mathbb{P}\left(|\bar{x}_{i,r} - m_{i,r}| > u_i \right) \leq 2 \exp\left(-\frac{n_i u_i^2}{2} \right).$$

Taking

$$u_i = \sqrt{\frac{2 \log(4dN/\delta)}{n_i}},$$

we obtain

$$\mathbb{P}\left(|\bar{x}_{i,r} - m_{i,r}| > u_i \right) \leq \frac{\delta}{2dN}.$$

A union bound over all $i \in \{1, \ldots, N\}$ and all coordinates $r \in \{1, \ldots, d\}$ implies that, with probability at least $1 - \delta$,

$$\|\bar{x}_i - m_i\|_2 \leq \sqrt{d}\, u_i = \sqrt{\frac{2d \log(4dN/\delta)}{n_i}} \qquad \text{for all } i.$$

We use the elementary normalization inequality

$$\left\| \frac{u}{\|u\|_2} - \frac{v}{\|v\|_2} \right\|_2 \le \frac{2\|u-v\|_2}{\|v\|_2},$$

valid for nonzero v and any nonzero u. When $\bar{x}_i \neq 0$, this gives,

$$\|\tilde{x}_i - \rho_i\|_2 = \left\| \frac{\bar{x}_i}{\|\bar{x}_i\|_2} - \frac{m_i}{\|m_i\|_2} \right\|_2 \le \frac{2}{a_0}\sqrt{\frac{2d\log(4dN/\delta)}{n_i}}.$$

If an empirical mean is exactly zero, we define $\tilde{x}_i$ arbitrarily on S^{d-1}. On the high-probability event above, this case can occur only when

$$\sqrt{\frac{2d\log(4dN/\delta)}{n_i}} \ge \|\bar{x}_i - m_i\|_2 = \|m_i\|_2 \ge a_0.$$

Hence

$$\frac{2}{a_0}\sqrt{\frac{2d\log(4dN/\delta)}{n_i}} \ge 2 \ge \|\tilde{x}_i - \rho_i\|_2,$$

so the same bound also holds when $\bar{x}_i = 0$. Averaging over words gives

$$\frac{1}{N}\sum_{i=1}^{N}\|\tilde{x}_i - \rho_i\|_2 \le \frac{2}{a_0}\sqrt{2d\log(4dN/\delta)} \cdot \frac{1}{N}\sum_{i=1}^{N} n_i^{-1/2}.$$

By Cauchy's inequality,

$$\frac{1}{N}\sum_{i=1}^{N} n_i^{-1/2} \le \left(\frac{1}{N}\sum_{i=1}^{N} n_i^{-1}\right)^{1/2} = n_{\text{eff}}^{-1/2}.$$

Therefore,

$$\frac{1}{N}\sum_{i=1}^{N}\|\tilde{x}_i - \rho_i\|_2 \le \frac{2}{a_0}\sqrt{\frac{2d\log(4dN/\delta)}{n_{\text{eff}}}}.$$

This proves Proposition 1. □

B.2 Proof of Theorem 1

We first record a standard concentration bound for the empirical movMF score under the dependency-graph condition in Assumption 2.

Lemma 1 (Uniform score concentration)**.** *Under Assumptions 2 and 3, there exists a constant $C_s > 0$ such that, with probability at least $1-\delta$,*

$$\sup_{\vartheta\in\mathcal{U}}\left\|\frac{1}{N}\sum_{i=1}^{N} s_\vartheta(\rho_i) - \mathbb{E}[s_\vartheta(\rho)]\right\|_2 \le C_s\sqrt{\frac{(\Delta+1)\{p\log N + \log(4/\delta)\}}{N}}.$$

Proof. Fix $\vartheta \in U$ and $u \in S^{p-1}$. Define

$$Y_i = u^\top\{s_\vartheta(\rho_i) - \mathbb{E}[s_\vartheta(\rho)]\}.$$

Then $\mathbb{E}[Y_i] = 0$ and $|Y_i| \le 2B$. Let $q \le \Delta + 1$ be the number of colors in a proper coloring of the dependency graph, and let $C_1, \dots, C_q$ be the corresponding color classes. Variables within the same color class are independent.

For $S_c = \sum_{i\in C_c} Y_i$ and any $\lambda > 0$, Hölder's inequality and Hoeffding's lemma give

$$\begin{aligned}\mathbb{E}\exp\left(\lambda\sum_{i=1}^{N} Y_i\right) &= \mathbb{E}\prod_{c=1}^{q}\exp(\lambda S_c)\\ &\le \prod_{c=1}^{q}\{\mathbb{E}\exp(q\lambda S_c)\}^{1/q}\\ &\le \exp\left(Cq\lambda^2 NB^2\right),\end{aligned}$$

for a universal constant $C > 0$. Chernoff's bound therefore implies

$$\mathbb{P}\left(\left|\frac{1}{N}\sum_{i=1}^{N} Y_i\right| > t\right) \le 2\exp\left(-\frac{cNt^2}{(\Delta+1)B^2}\right),$$

for a universal constant $c > 0$.

To make the bound uniform over ϑ, let $\mathcal{N}_U$ be an η-net of U. Since U is contained in a bounded subset of $\mathbb{R}^p$, we may take

$$|\mathcal{N}_U| \le \left(\frac{C_U}{\eta}\right)^p.$$

Let $\mathcal{N}_S$ be a $1/2$-net of S^{p-1}, with

$$|\mathcal{N}_S| \le 5^p.$$

A union bound over $\mathcal{N}_U \times \mathcal{N}_S$ gives, with probability at least $1-\delta$,

$$\sup_{\vartheta\in\mathcal{N}_U}\sup_{u\in\mathcal{N}_S}\left|u^\top\left\{\frac{1}{N}\sum_{i=1}^{N} s_\vartheta(\rho_i) - \mathbb{E}[s_\vartheta(\rho)]\right\}\right| \le C\sqrt{\frac{(\Delta+1)\{p\log(C_U/\eta) + p + \log(4/\delta)\}}{N}}.$$

The $1/2$-net property controls the Euclidean norm up to a universal constant. Finally, for any $\vartheta \in U$, let $\vartheta_0 \in \mathcal{N}_U$ be the nearest net point. By the Lipschitz condition in Assumption 3,

$$\left\|\frac{1}{N}\sum_{i=1}^{N}\{s_\vartheta(\rho_i) - s_{\vartheta_0}(\rho_i)\}\right\|_2 + \|\mathbb{E}\{s_\vartheta(\rho) - s_{\vartheta_0}(\rho)\}\|_2 \le 2L_\vartheta\eta.$$

Taking $\eta = N^{-1}$ and absorbing constants proves the lemma. □

Next, we give formal proof of Theorem 1. Define the population risk $M(\vartheta) = \mathbb{E}[\ell_\vartheta(\rho)]$, the empirical risk based on the population representative directions,

$$\widehat{M}_\rho(\vartheta) = \frac{1}{N}\sum_{i=1}^{N}\ell_\vartheta(\rho_i),$$

and the empirical risk actually optimized by fitting movMF to the estimated representative directions,

$$\widehat{M}_{\tilde{x}}(\vartheta) = \frac{1}{N}\sum_{i=1}^{N}\ell_\vartheta(\tilde{x}_i).$$

By Lemma 1, with probability at least $1-\delta/2$,

$$\sup_{\vartheta\in\mathcal{U}}\left\|\nabla\widehat{M}_\rho(\vartheta) - \nabla M(\vartheta)\right\|_2 \le C_s\sqrt{\frac{(\Delta+1)\{p\log N + \log(8/\delta)\}}{N}}.$$

Next, by the Lipschitz property of the score in the input direction,

$$\begin{aligned}\sup_{\vartheta\in\mathcal{U}}\left\|\nabla\widehat{M}_{\tilde{x}}(\vartheta) - \nabla\widehat{M}_\rho(\vartheta)\right\|_2 &= \sup_{\vartheta\in\mathcal{U}}\left\|\frac{1}{N}\sum_{i=1}^{N}\{s_\vartheta(\tilde{x}_i) - s_\vartheta(\rho_i)\}\right\|_2 \\ &\le \frac{L_x}{N}\sum_{i=1}^{N}\|\tilde{x}_i - \rho_i\|_2.\end{aligned}$$

Using Proposition 1 with failure probability $\delta/2$, we obtain, with probability at least $1-\delta$,

$$\sup_{\vartheta\in\mathcal{U}}\left\|\nabla\widehat{M}_{\tilde{x}}(\vartheta) - \nabla M(\vartheta)\right\|_2 \le C\left\{\sqrt{\frac{(\Delta+1)\{p\log N + \log(8/\delta)\}}{N}} + \sqrt{\frac{d\log(8dN/\delta)}{n_{\text{eff}}}}\right\}.$$

Let

$$r_N = C\left\{\sqrt{\frac{(\Delta+1)\{p\log N + \log(8/\delta)\}}{N}} + \sqrt{\frac{d\log(8dN/\delta)}{n_{\text{eff}}}}\right\}.$$

The previous display shows that, with probability at least $1-\delta$,

$$\sup_{\vartheta \in U} \left\| \nabla \widehat{M}_{\tilde{x}}(\vartheta) - \nabla M(\vartheta) \right\|_2 \le r_N.$$

Let $\widehat{\vartheta} \in U$ be the local solution returned by EM. If EM is run to exact local stationarity, then

$$\nabla \widehat{M}_{\tilde{x}}(\widehat{\vartheta}) = 0.$$

More generally, if EM is stopped with numerical tolerance

$$\left\| \nabla \widehat{M}_{\tilde{x}}(\widehat{\vartheta}) \right\|_2 \le \xi,$$

the following argument holds with $r_N + \xi$ in place of r_N. By local strong convexity and $\nabla M(\vartheta^\star) = 0$,

$$\begin{aligned}
\lambda \|\widehat{\vartheta} - \vartheta^\star\|_2^2 &\le \left\langle \nabla M(\widehat{\vartheta}) - \nabla M(\vartheta^\star), \widehat{\vartheta} - \vartheta^\star \right\rangle \\
&= \left\langle \nabla M(\widehat{\vartheta}), \widehat{\vartheta} - \vartheta^\star \right\rangle \\
&= \left\langle \nabla M(\widehat{\vartheta}) - \nabla \widehat{M}_{\tilde{x}}(\widehat{\vartheta}), \widehat{\vartheta} - \vartheta^\star \right\rangle + \left\langle \nabla \widehat{M}_{\tilde{x}}(\widehat{\vartheta}), \widehat{\vartheta} - \vartheta^\star \right\rangle \\
&\le (r_N + \xi) \|\widehat{\vartheta} - \vartheta^\star\|_2.
\end{aligned}$$

Therefore,

$$\|\widehat{\vartheta} - \vartheta^\star\|_2 \le \frac{r_N + \xi}{\lambda}.$$

For the exact-stationarity statement used in the main theorem, take $\xi = 0$.

Finally, define

$$g(\Theta) = \sum_{k=1}^{K} \frac{\pi_k}{\kappa_k}.$$

Since the concentrations are bounded away from zero in Assumption 3, g is Lipschitz in the local neighborhood U. Hence, for some constant $L_g < \infty$,

$$|\widehat{D} - D^\star| = |g(\widehat{\Theta}) - g(\Theta^\star)| \le L_g \|\widehat{\vartheta} - \vartheta^\star\|_2.$$

Combining the displays above and using $p = O(Kd)$ gives the theorem after adjusting constants. This proves Theorem 1. □

C Additional Experimental Details

We conduct the experiments on a server equipped with NVIDIA GeForce RTX 3090 GPUs with 24GB of memory. The methods requiring two large language models, such as Fast DetectGPT, Binoculars and DNA-DetectLLM, use two GPUs when necessary.

C.1 Dataset Statistics

Table 6 reports the user-level dataset statistics used in the main experiments. BotSim-24, TwiLLM-R, and TwiLLM-I are used as the three main benchmarks. TwiLLM-R and TwiLLM-I share the same human-side accounts and bot-account count, but differ in how the bot-side textual histories are constructed, as detailed in the next subsection.

Table 6: Statistics of the user-level datasets used in the main experiments.

Dataset	Human users	Bot users
BotSim-24	1552	842
TwiLLM-R	131	89
TwiLLM-I	131	89

C.2 Method and Baseline Protocols

All methods are evaluated under the same user-level protocol and processed data files. No model is trained or fine-tuned on the evaluation datasets. For CDetect, we use DeBERTa-v3-large as the contextual encoder and extract representations from the final layer. Subword representations belonging to the same word occurrence are aggregated by mean pooling, and the default stopword inventory is NLTK English stopwords. We use the soft movMF variant with $K = 2$.

For generative or statistical baselines, including log likelihood, Fast-DetectGPT, Binoculars, and DNA-DetectLLM, we use a unified 1024-token chunk-based long-text protocol. Long user histories are tokenized with each method's tokenizer and split into non-overlapping chunks without single-pass truncation. Instead of averaging independently normalized chunk-level scores, we aggregate each method's token-level or component-level statistics across chunks and compute one final user-level score. Specifically, log likelihood aggregates total token log-probability and total scored token count; Fast-DetectGPT aggregates discrepancy numerator and variance; Binoculars aggregates performer cross-entropy and observer-performer cross-entropy; and DNA-DetectLLM aggregates its numerator component and observer-performer cross-entropy component. For classifier-based baselines, including the OpenAI Detector and RADAR, we follow their architecture-imposed 512-token input limit and do not introduce additional chunk aggregation. Table 7 reports the protocol and models used for each method in experiments.

Table 7: Implementation protocol of CDetect and baseline methods in the main experiments.

Method	**Main model(s)**	**Input handling**
CDetect	DeBERTa-v3-large	text-unit encoding
Log Likelihood	Falcon-7B-Instruct	chunked, 1024 tokens
Fast-DetectGPT	Falcon-7B / Falcon-7B-Instruct	chunked, 1024 tokens
Binoculars	Falcon-7B / Falcon-7B-Instruct	chunked, 1024 tokens
DNA-DetectLLM	Falcon-7B / Falcon-7B-Instruct	chunked, 1024 tokens
OpenAI Detector	RoBERTa-large detector	512-token classifier input
RADAR	RADAR-Vicuna-7B	512-token classifier input

D TwiLLM Benchmark Construction

TwiLLM is constructed to evaluate user-level detection of LLM-driven social bots in a controlled Twitter-like setting. We build TwiLLM from TwiBot-22 (Feng et al., 2022), a large-scale Twitter bot detection dataset. TwiLLM contains two variants: TwiLLM-Rewrite (TwiLLM-R) and TwiLLM-Interact (TwiLLM-I). The two variants share the same human side and the same number of bot accounts, but differ in how the LLM-driven bot histories are constructed. TwiLLM-R rewrites historical bot-account texts with an LLM, while TwiLLM-I simulates interactive LLM-driven accounts that generate posts and replies over time.

D.1 Source Data and User-Level Filtering

We build on the publicly released TwiBot-22 subset from Feng et al. (2024). Following activity- and language-based preprocessing, we retain accounts with at least three posts and usable English textual histories. This results in 131 human accounts and 89 bot accounts. The human class uses the retained TwiBot-22 human accounts, while the bot-source accounts are used as source accounts for constructing the LLM-driven bot class in TwiLLM-R and TwiLLM-I.

D.2 TwiLLM-R Construction via Post-Level Rewriting

TwiLLM-R is designed to evaluate rewriting-based LLM-driven bot detection. For each retained bot account, we rewrite its historical posts using the same prompt template and generation configuration. The prompt is applied independently to each post, and the generated posts are then grouped back by account to form a user-level history. This setting preserves the account-level structure and broad topical context of the original bot accounts, while replacing their textual content with LLM-generated posts.

We use GPT-5.4 Mini as the rewriting model, with temperature 0.7 and maximum output length 512. The pipeline rewrites bot-side tweets only. Human examples keep the original historical posts from TwiBot-22 human accounts. The completed rewriting run processed 6,579 post-level rewrite jobs without failed generations.

Prompt template. The prompt used for rewriting each post is shown below:

> Write a new version of the following social media post for an LLM-driven bot account.
> Use the account description as light context.
> Keep the new post in a roughly similar length range.
> Output only the new post as plain text.
> Account description:
> {account_description}
> Source post:
> {original_text}

D.3 TwiLLM-I Construction via Interactive Generation

TwiLLM-I is constructed as an interaction-style benchmark that simulates how LLM-driven bot accounts may operate in a realistic social-media environment. We use the same source accounts as TwiLLM-R: the human side keeps the same 131 TwiBot-22 human accounts, and the bot side uses the same 89 TwiBot-22 bot source accounts. Unlike TwiLLM-R, TwiLLM-I does not rewrite the original historical posts of the bot accounts. Instead, the original bot texts are discarded, and each LLM-driven bot history is generated from account-level profile information and simulated interaction context, mimicking a bot that posts and reacts over time.

For each bot account, we generate 30 text units with GPT-5.4 Mini using temperature 0.7 and seed 42. We first fix the action composition to 15 original self-posts and 15 replies, and then shuffle their order using a user-specific random stream derived from seed 42. For original-post generation, the model is conditioned on the account profile, an approximate virtual posting time, and up to five previously generated posts from the same account, so that the generated history maintains account-level persona and temporal continuity. For reply generation, the model is additionally given an external feed item sampled from real English posts written by human users in the TwiLLM source corpus, after filtering out very short posts. The model is then asked to generate a context-dependent social-media response, such as a reply, comment, or reaction, instead of copying or paraphrasing the feed item. This setting aims to simulate an LLM-driven account that not only publishes standalone content but also participates in surrounding social conversations.

Prompt templates. TwiLLM-I uses two prompt templates according to the assigned action type. For original-post generation, the prompt is:

> You operate a social media bot account over time.
> Task:
> - Write one original post that this account might publish to its own followers at the current time.
> - Use the account profile, current virtual time, and recent generated posts if available.
> - Do not write as a direct reply to another user.
> - Match the account's persona and the current context.
> - Hashtags, mentions, links, and short-form style are allowed if they fit naturally.
> - Do not mention this prompt, the dataset, or that you are an AI model.
> Output valid JSON only, with exactly this schema:
> {{"post": "the generated post"}}
> Account profile:
> - user_id: {user_id}
> - username: {username}
> - display name: {name}
> - description: {description}

Account context:
- approximate posting time: {virtual_time}
- recent posts from this same account: up to {memory_posts_per_step}

For reply generation, the prompt is:

You operate a social media bot account inside a live feed.
Task:
- Read the external feed item below.
- Write one reply, comment, or quote-style reaction that this account might publish now.
- The post should depend on the feed item and sound like a normal social-media interaction.
- Social-media shorthand, mentions, hashtags, and brief fragments are allowed when natural.
- Do not copy or rewrite the feed item.
- Do not mention this prompt, the dataset, or that you are an AI model.
Output valid JSON only, with exactly this schema:
{{"post": "the generated post"}}
Account profile:
- user_id: {user_id}
- username: {username}
- display name: {name}
- description: {description}
Account context:
- approximate posting time: {virtual_time}
- recent posts from this same account: up to {memory_posts_per_step}
External feed item:
- source: {trigger_source}
- author: {trigger_author}
- text: {trigger_text}

D.4 Final User-Level Format

In both variants, each example is represented as a user-level textual history, and detectors only observe the concatenated account text without graph structure, profile metadata, or other account-level attributes.

E Supplement of Ablation and Sensitivity

E.1 Concatenation Order Sensitivity

Table 8: Sensitivity to within-account concatenation order. We report AUROC / AUPRC when keeping each user's text units fixed and changing only their concatenation order.

Dataset	Method	Original	Time	Reverse	Random
BotSim-24	Fast-DetectGPT	0.8826 / 0.8516	0.8715 / 0.8369	0.8815 / 0.8463	0.8787 / 0.8445
	CDetect	0.9964 / 0.9839	0.9966 / 0.9844	0.9964 / 0.9843	0.9966 / 0.9849
TwiLLM Avg.	Fast-DetectGPT	0.4919 / 0.3927	0.4879 / 0.3931	0.5564 / 0.4233	0.5601 / 0.4197
	CDetect	0.9142 / 0.8885	0.9119 / 0.8849	0.9141 / 0.8866	0.9205 / 0.8878

We further examine whether detection performance depends on the order in which a user's historical text units are concatenated. This analysis is designed to test two questions: whether the detector relies on a specific concatenation order, and whether CDetect and likelihood-based detectors exhibit different sensitivity to such order changes.

In this experiment, we fix the user set, labels, and the historical text units available for each user, and only change the order of text units within each user history. We consider four ordering strategies.

Original keeps each user's text units in the default order used in the benchmark data, without applying any additional reordering, which is the default order used in the main experiments. Time sorts each user's text units in chronological order when timestamps are available. Reverse uses the reverse chronological order. Random shuffles the text units within each user independently with a fixed random seed, so the result is deterministic and reproducible. For BotSim-24, text units include posts and comments. For TwiLLM, text units correspond to historical tweets. All variants keep the number of accounts, labels, and textual content unchanged.

We compare CDetect and Fast-DetectGPT under the same configurations as in the main experiments. We report AUROC and AUPRC in Table 8. The results show that CDetect is highly stable across different concatenation orders. Fast-DetectGPT is more sensitive to concatenation order compared to CDetect. This is consistent with the fact that likelihood- and discrepancy-based detectors depend more directly on local token-level context.

E.2 Encoder Sensitivity

Since CDetect relies on the geometry of contextual word representations, the encoder backbone can affect the estimated semantic concentration. Prior analysis has shown that contextualized representations exhibit non-isotropic geometry and that their context-specificity varies across both layers and model families (Ethayarajh, 2019). Therefore, different encoders may provide different levels of separation for semantic-concentration estimation.

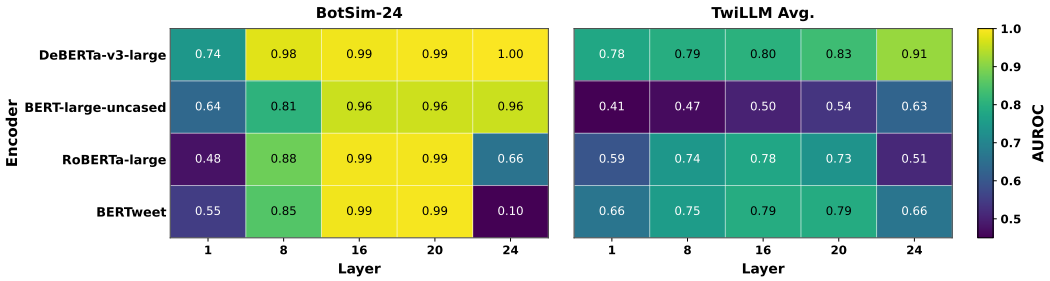


Figure 6: AUROC of CDetect with different encoder backbones and representation layers.

As shown in Figure 6, the performance of CDetect is affected by both the encoder backbone and the selected layer. DeBERTa-v3-large provides the strongest and most stable results in our experiments, especially in upper layers. Other encoders also achieve competitive performance, but their optimal layers can differ.

E.3 Contribution of Contextualization

We examine the contribution of contextual representations to CDetect. Contextualization is used to construct the representative direction of word type, before movMF models the distribution of these directions. To isolate its contribution, we replace the contextual hidden states of DeBERTa-v3-large with static token embeddings from its input embedding layer. All subsequent processing steps, including subword aggregation, intra-type averaging, normalization, and movMF fitting, remain unchanged.

Table 9: Effect of contextualization on CDetect. The static variant uses DeBERTa-v3-large input token embeddings in place of contextual hidden states, with the remaining pipeline unchanged.

	Contextual		Static	
Dataset	**AUROC**	**AUPRC**	**AUROC**	**AUPRC**
BotSim-24	**0.9964**	**0.9839**	0.4074	0.2819
TwiLLM-R	**0.8543**	**0.8054**	0.6175	0.5082
TwiLLM-I	**0.9741**	**0.9717**	0.7208	0.5138

As shown in Table 9, replacing contextual representations with static embeddings substantially reduces both AUROC and AUPRC on all three benchmarks. The average AUROC decreases from 0.9416 to 0.5819. These results demonstrate the importance of contextual encoding within CDetect: the geometry derived from static token embeddings alone does not preserve the discriminative performance of contextual word representations.

F Comparison with Graph-Based Methods

Table 10: Comparison with graph-based bot detection methods on the inductive BotSim-24 split. Thresholds are selected on the shared validation subset using Youden's J and evaluated on the shared evaluation subset. GNN baselines are averaged over five training seeds, while CDetect is deterministic and does not require random-seed training runs.

Method	Seeds	F1	Acc.	Prec.	Rec.	FPR
BotRGCN	5	0.9202 ± 0.0340	0.9477 ± 0.0243	0.8834 ± 0.0672	0.9639 ± 0.0191	0.0594 ± 0.0382
BotMoE	5	0.9099 ± 0.0085	0.9397 ± 0.0055	0.8393 ± 0.0067	0.9934 ± 0.0131	0.0841 ± 0.0035
BotDMM	5	0.9653 ± 0.0124	0.9779 ± 0.0082	0.9364 ± 0.0283	0.9967 ± 0.0066	0.0304 ± 0.0141
CDetect	–	**0.9919**	**0.9950**	**0.9839**	**1.0000**	**0.0072**

The main experiments focus on text-based AI-generation detectors, because CDetect is designed as a text-only, training-free user-level detector. This setting allows a controlled comparison where all methods rely only on users' historical textual content. In traditional social bot detection, however, graph-based methods are also an important family of approaches. These methods typically exploit richer inputs, such as interaction graphs, user attributes, and supervised training labels. Therefore, we provide an additional comparison with representative graph-based bot detection methods on BotSim-24, including BotMoE, BotRGCN, and BotDMM.

We conduct this comparison under the same activity-based filtering protocol as the main experiments, retaining BotSim-24 accounts with at least five posts/comments, and evaluate methods in the inductive setting. All methods are evaluated on the same shared test split, which contains 239 users, including 158 humans and 81 bots. To avoid selecting thresholds directly on the evaluation set, we sample a shared validation subset from this test split using a fixed random seed, consisting of 20 human users and 20 bot users. The remaining 199 users are used as the shared evaluation subset. Thresholds for all methods are selected on the shared validation subset using Youden's J statistic and then transferred to the shared evaluation subset (Wu et al., 2025). For GNN baselines, we report the mean and standard deviation over seeds 0–4. These seeds are used to repeat stochastic model training under the same fixed data protocol, rather than to change the benchmark split or evaluation setting. They mainly account for randomness from parameter initialization, dropout and other training dynamics. CDetect is deterministic and uses only historical text, without graph structure, user attributes, or supervised training on BotSim-24.

Table 10 reports the threshold-based results under the shared-validation protocol. CDetect achieves the highest F1 and accuracy, while reducing the false positive rate to 0.0072. Compared with the strongest GNN baseline, BotDMM, CDetect improves F1 from 0.9653 to 0.9919 and reduces FPR from 0.0304 to 0.0072. These results suggest that the semantic concentration signal captured by CDetect remains highly discriminative even when compared with supervised graph-based bot detectors. Notably, CDetect uses only historical text and does not rely on graph structure, user attributes, or supervised training on BotSim-24.

G Detection Score Distributions

Figure 7 presents the empirical distributions of CDetect scores across the three benchmarks.

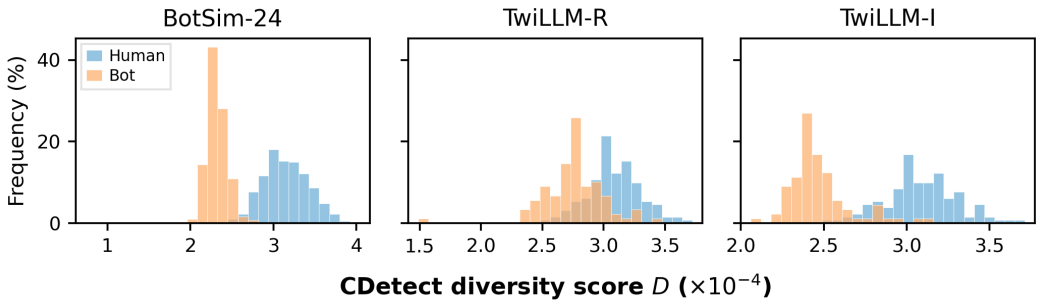


Figure 7: Distributions of the CDetect diversity score D. Frequencies are normalized separately within each class.

H Evaluation under Different Content Settings

We report CDetect results on two BotSim-24 subsets: one formed from politics content and one containing accounts active in at least four subreddits. These evaluations complement the results on the full BotSim-24 benchmark.

H.1 Single-Topic Content

We retain content from the politics topic and evaluate the detectors on the resulting subset. Table 11 reports the results.

Table 11: Detection performance on politics content from BotSim-24.

Method	AUROC	AUPRC
RADAR	0.8400	0.5485
Log Likelihood	0.6482	0.3755
Fast-DetectGPT	0.8308	**0.7492**
Binoculars	0.8031	0.6154
DNA-DetectLLM	0.8212	0.6160
CDetect	**0.9240**	0.6823

CDetect achieves the highest AUROC on this subset, while Fast-DetectGPT achieves the highest AUPRC. The results show that CDetect also performs strongly when evaluated on politics content.

H.2 Accounts Active in Multiple Subreddits

We select accounts with activity in at least four distinct subreddits, yielding 71 human accounts and 54 bot accounts. We evaluate the detectors on these accounts using their available histories. Table 12 reports the results.

Table 12: Detection performance on BotSim-24 accounts active in at least four subreddits.

Method	AUROC	AUPRC
RADAR	0.9124	0.8214
Log Likelihood	0.6993	0.5274
Fast-DetectGPT	0.9244	0.9247
Binoculars	0.9053	0.8901
DNA-DetectLLM	0.9384	0.9162
CDetect	**0.9977**	**0.9968**

On this subset, CDetect achieves the highest AUROC and AUPRC among the evaluated methods. Its performance remains strong across both supplementary BotSim-24 evaluations.