BotDMM: Dual-channel Multi-Modal Learning for LLM-driven Bot Detection on Social Media

Jinglong Duan, Shiqing Wu, Weihua Li, Quan Bai, Minh Nguyen, Jianhua Jiang

PII: S1566-2535(25)00820-6
DOI: https://doi.org/10.1016/j.inffus.2025.103758
Reference: INFFUS 103758

To appear in: *Information Fusion*

Received date: 30 June 2025
Revised date: 27 August 2025
Accepted date: 16 September 2025

Please cite this article as: Jinglong Duan, Shiqing Wu, Weihua Li, Quan Bai, Minh Nguyen, Jianhua Jiang, BotDMM: Dual-channel Multi-Modal Learning for LLM-driven Bot Detection on Social Media, *Information Fusion* (2025), doi: https://doi.org/10.1016/j.inffus.2025.103758

Highlights

- We propose a temporal multi-modal framework that models how user evolve over time by by capturing sequential changes: tweets, AMR graphs, and social structures using LSTM and dynamic graph convolution.
- We design a dual-channel feature learner that separates content and structure with an orthogonality constraint to reduce redundancy and enhance generalization.
- Extensive experiments show BotDMM outperforms state-of-the-art baselines in detecting LLM-driven bots, traditional bots, and humans, with ablations and visualizations confirming its effectiveness.

BotDMM: Dual-channel Multi-Modal Learning for LLM-driven Bot Detection on Social Media

Jinglong Duan[a], Shiqing Wu[b,*], Weihua Li[a,**], Quan Bai[c], Minh Nguyen[a], Jianhua Jiang[d]

[a]School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand
[b]Faculty of Data Science, City University of Macau, Macau SAR, China
[c]School of Information and Communication Technology, University of Tasmania, Hobart, Australia
[d]Jilin Province Key Laboratory of Fintech, Jilin University of Finance and Economics, Changchun, China

Abstract

Social bot is becoming a growing concern due to their ability to spread misinformation and manipulate public discourse. The emergence of powerful large language models (LLMs), such as ChatGPT, has introduced a new generation of bots capable of producing fluent and human-like text while dynamically adapting their relational patterns over time. These LLM-driven bots seamlessly blend into online communities, making them significantly more challenging to detect. Most existing approaches rely on static features or simple behavioral patterns, which are not effective against bots that can evolve both their language and their network connections. To address these challenges, we propose a novel Dual-channel Multi-Modal learning (BotDMM) framework for LLM-driven bot detection. The proposed model captures discriminative information from two complementary sources: users' content features (including their profiles and temporal posting behavior) and structural features (reflecting local network topology). Furthermore, we employ a joint training approach that incorporates two carefully designed self-supervised learning paradigms alongside the primary prediction task to enhance discrimination between human users, traditional bots, and LLM-driven bots. Extensive experiments demonstrate the effectiveness and superiority of BotDMM compared to state-of-the-art baselines.

Keywords: BotDMM, Social bot detection, LLM-driven bot, Social network

1. Introduction

Nowadays, online social media platforms, such as Twitter, Facebook, and Instagram, have become a crucial part of human interaction, enabling users to communicate, browse commercial content, and share information [30]. However, due to the frequency of user activity and the vast amount of personal information shared on these platforms, malicious individuals and organizations increasingly exploit social media for their own purposes by creating fake automated accounts, known as social bot [20]. While many social bot is relatively harmless [28], others pose serious threats, including personal information theft [1], misinformation and rumors spread [25], and public opinion manipulation [52]. Consequently, concerns about the abuse of such harmful social bot is mounting.

To effectively detect social bot on social media, various approaches have emerged in recent years [6], which can be broadly divided into machine learning, deep learning, and graph-based approaches. Machine learning methods typically extract specific features from user profiles and activities, such as content features [9, 7], lexical and non-lexical analysis [26], and user behavioral patterns [27]. While deep learning approaches often generalize better on large datasets by employing advanced techniques, such as deep neural networks with active learning for scalable training [50] and attention mechanisms for extracting important features from multimodal data [15]. Nevertheless, these feature-oriented methods overlook the underlying social network structure, limiting their detection performance. Hence, graph-based approaches are proposed to address this gap by modeling user interactions. For instance, DA-SBCD proposes a graph-based deep autoencoder to detect social botnet communities of social bot with malicious behavioral similarity [35]. BotRGCN constructs a heterogeneous graph to represent activities on social media and applies relational graph convolutional networks for bot detection [17].

Although existing methods can effectively detect conventional social bot, they may be less effective against advanced bots powered by Large Language Models (LLMs). Social media bots guided by LLMs are found to more easily escape from existing methods [44]. By imitating human behavior patterns and embedding harmful viewpoints into generated content under carefully designed prompts, LLMs can facilitate the spread of misinformation and hateful content on social media [51]. As a result, users may be more inclined to trust such generated misinformation [33]. To address this challenge, researchers have proposed novel detection methods tailored to LLM-driven social bot, using text analysis and intent recognition specific to LLM outputs [39, 13, 18, 12]. However, these methods primar-

*Corresponding author
**Corresponding author
Email addresses: jinglong.duan@autuni.ac.nz (Jinglong Duan), sqwu@cityu.edu.mo (Shiqing Wu), weihua.li@aut.ac.nz (Weihua Li), quan.bai@utas.edu.au (Quan Bai), minh.nguyen@aut.ac.nz (Minh Nguyen), jjh@jlufe.edu.cn (Jianhua Jiang)

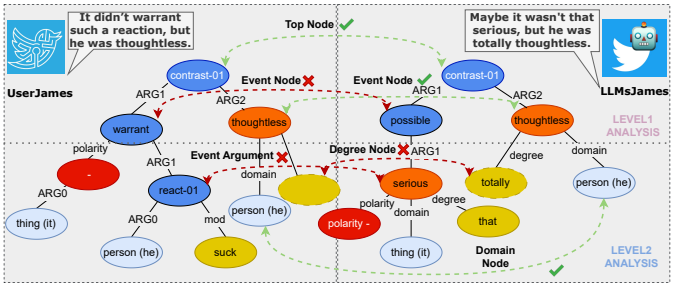


Figure 1: Comparison between human and LLM-driven bot on AMR

ily operate on static graphs and focus solely on the most recent user state, overlooking historical changes in social connections. This limitation makes it difficult to detect LLM-driven bots that strategically adapt their social connections over time to imitate human behaviors [21].

Moreover, few studies offer concrete frameworks for extracting structured semantics to uncover latent patterns, which limits the performance of detecting LLM-driven social bot. Hence, we leverage Abstract Meaning Representation (AMR) as a structured semantic modality and integrate it into our temporal pipeline as a novel semantic modality to construct more diverse user profiles on social media. Specifically, AMR is a structured semantic representation that models the meaning of a sentence as a directed graph, where concepts are encoded as nodes, and their relationships are expressed as labeled edges [5]. Different from surface-level text features, AMR abstracts away from lexical choices and focuses on the underlying meaning, making it well-suited for distinguishing stylistic and semantic differences between human-generated and LLM-generated content. As illustrated in Figure 1, even when the same sentence is rewritten by a large language model, its AMR graph often reveals subtle structural deviations from the human-authored version. These differences are often overlooked when simply reading the text, but become much more apparent when represented as semantic graphs. By analyzing the semantic gap between AMR graphs of different user posts, we can gain deeper insights into content regularity and meaning coherence, which is crucial for identifying bots that rely on artificial language generation [43].

To tackle the aforementioned challenges, we propose a novel **D**ual-channel **M**ulti-**M**odal learning for LLM-driven bot detection (BotDMM), which can effectively distinguish human users, traditional social bot, and LLM-driven social bot on social media. Specifically, the proposed BotDMM first employs LSTM networks to capture temporal changes in user content sequential features. Then, we devise a dual-channel learning (DCL) module consisting of a content learning module and a structure learning module, where the former focuses on capturing key information in users' content features and the latter aims to model relational behaviors among users in the social network. Lastly, we jointly train the proposed BotDMM by leveraging a self-supervised learning task in conjunction with the prediction task, which enables the separation of representations among human users, traditional social bot, and LLM-driven social bot, ultimately improving the prediction results of LLM-driven social bot. The main contributions of this paper can be summarized as follows:

- We propose a temporal multi-modal framework that explicitly models how users evolve over time by capturing sequential changes in both tweets and AMR graphs, as well as social structures, through LSTM and dynamic graph convolution.
- We introduce a novel dual-channel feature learning mechanism that separates content and structure representations and applies an decorrelation constraint to reduce feature redundancy. This enables the model to better isolate modality-specific patterns and improves generalization across diverse user types.
- Extensive experiments demonstrate that BotDMM outperforms state-of-the-art baselines in detecting LLM-driven bots, traditional bots, and human users. Ablation studies confirm the effectiveness of each proposed component, and visualization analysis validates the quality of the final embeddings.

2. Related Work

This section reviews recent studies on both traditional and LLM-driven social bot detection on social media. The conventional detection methods can be categorized into two types: machine learning-based methods and graph-based methods.

2.1. Traditional Social Media Bot Detection

Early approaches to social media bot detection mainly rely on traditional machine learning techniques. For instance, Ikram et al. developed a honeypot-based detection system using Support Vector Machines (SVM) and co-clustering methods. They extracted data from Facebook user timelines and categorized features into lexical and non-lexical types for classification [26]. Similarly, Andriotis and Takas applied AdaBoost to identify Twitter spam bots based on a combination of content, user metadata, and sentiment features [3]. Yang et al. leveraged an SVM model trained on behavioral features from 650,000 Sybil accounts on Renren, achieving strong performance [54]. Al-Qurishi et al. combined various classifiers, such as Random Forest (RF), Classification by Regression (ClsReg), and Ordinal Iterative Regression (OIR), leveraging content, profile, network, and activity-based features to detect Sybil accounts [2].

While effective in certain contexts, these traditional methods heavily depend on manually engineered features and often struggle to scale to large, complex datasets. Their reliance on fixed feature sets limits adaptability and reduces their capability to capture intricate behavioral patterns in real-time environments.

With the emergence of deep learning, more advanced techniques have been introduced to enhance detection accuracy. Models based on LSTM, CNN, and Bi-LSTM architectures show promising results [34]. For example, Daouadi et al. designed a feature augmentation strategy that encapsulated both

interaction volume and user engagement breadth, enabling deep learning models to detect Twitter bots more accurately [8]. Ellaky et al. proposed a hybrid approach combining BERT-based content feature extraction with a Random Forest classifier [14]. Martín-Gutiérrez and Hernández-Peñaloza introduced Bot-DenseNet, a multilingual detection architecture that utilized BERT and RoBERTa for tweet embeddings [38]. Kumar et al. applied CNN, LSTM, and BERT to extract content-based features from Twitter posts, yielding an effective deep learning model for detecting social media bots [31]. Zeng et al. developed MRLBot, a framework that integrates a Transformer-CNN hybrid model (DDTCN) for behavior analysis and a random walk-based module (IB2V) to model relational structures within communities [55].

Despite their strong performance, many of these deep learning approaches still overlook the structural information embedded in social networks, limiting their capacity to fully capture inter-node behavioral dynamics. To address this limitation, a growing body of research has explored graph-based methods that explicitly model network structure for improved bot detection.

2.2. Graph-Based Methods

By leveraging the topology of social networks, graph-based detection models demonstrate strong capabilities in identifying social media bots. Ng and Carley proposed BotBuster, a Mixture-of-Experts (MoE) framework in which each expert handles a distinct data modality (e.g., usernames, posts, metadata), allowing for effective bot detection even with incomplete inputs [40]. Feng et al. introduced BotMoE, a community-aware MoE model that integrates metadata, textual, and structural features for Twitter bot detection [37]. Wang et al. designed an unsupervised contrastive graph clustering method that jointly optimizes user embeddings and cluster assignments without requiring labeled data [48]. Huang et al. developed a compatibility-aware GNN to model heterogeneous user-bot interactions, significantly reducing training time compared to previous methods [24]. Zhou et al. presented LGB, a hybrid framework that fine-tuned a language model on unified account text and combined the resulting semantic embeddings with a pretrained GNN to detect sparsely linked social bot nodes [56]. He et al. proposed BotDGT, a framework that models social networks as dynamic graphs and incorporates both structural and temporal modules to capture evolving bot behaviors [22].

While many earlier graph-based methods effectively capture relational structures but often overlook temporal dynamics, advanced methods (e.g., BotDGT) model dynamics, yet they do not adequately model LLM-specific content patterns or the separation between textual and structural features. These limitations reduce their effectiveness in detecting more sophisticated, LLM-driven bots. To address these emerging challenges, recent research has turned its focus to detecting a new generation of social media bots powered by large language models, which exhibit more human-like behavior and content generation capabilities.

2.3. LLM-Driven Social Media Bots Detection

Recent advances in LLMs have led to the emergence of a new class of social media bots, prompting researchers to investigate their unique characteristics and detection challenges. Several studies have explored different aspects of this field. Ng and Carley pointed out that LLM-driven bots exhibit behavioral characteristics distinct from traditional bots and human accounts [41]. Mindner et al. analyzed linguistic distinctions between human and LLM-generated content, introducing novel feature sets, including text objectivity, repetition-based indicators, and grammatical error counts [39]. Li et al. conducted a large-scale behavioral study of LLM-driven social bot, revealing significant differences in activity patterns and toxicity levels compared to traditional bots in online social networks [33].

Other researchers have focused on detection methodologies. Duan et al. proposed Intent-Spectrum BotTracker (ISBT), a model that identifies LLM-driven bots by analyzing user intent and diversity of interactions [13]. Feng et al. developed a mixture-of-heterogeneous-experts framework, leveraging ChatGPT detectors to distinguish LLM-manipulated social media accounts [18]. Zhu et al. examined bots on the LLM-centric platform Chirper.ai, showing that these bots produce linguistically rich content, frequently engage in self-disclosure, and form distinct network structures, while zero-shot detection methods struggle to distinguish them from real users [57]. More recently, Duan et al. introduced LLM-BotGuard, a comprehensive detection framework combining pattern-informed feature extraction, a MoE module, and graph-based aggregation to capture the unique characteristics of LLM-generated content [12].

While these studies provide valuable insights into LLM-driven bot behavior, most focus primarily on static textual features or behavioral snapshots. Although Feng et al. [18] reconstructed the topological structure of their dataset, their model did not explicitly analyze the resulting network dynamics. In contrast, this research aims to explore dynamic behavioral patterns and develop models that specifically capture the structural evolution induced by LLMs within social networks.

3. Preliminaries

This section defines the LLM-based social media bot detection task and introduces the key notations and components used throughout the paper. We begin by formalizing the classification setting, followed by a description of the feature space and network structure associated with user accounts.

Problem description. Let $\mathcal{U} = \{u_1, u_2, ..., u_n\}$ denote the set of users on social media, where each user u_i is associated with a label y_i representing the user type. The label can indicate a genuine human user, a traditional social bot, or an LLM-driven bot. In this case, the bot detection task is defined as a three-class classification problem. The goal is to accurately predict the label y_i for each user u_i based on the input features, such as their behavior patterns and post content.

Features. Each user is associated with a set of multi-modal features that capture their numerical metadata, categorical attributes, profile descriptions, linguistic features, AMR representations, tweets, and social relationships. The numerical metadata, categorical features, and linguistic features are listed in Table 1.

Table 1: List of features

Linguistic Feature	Numerical Feature	Categorical Feature
similarity	followers_count	protected
words_per_tweet	friends_count	geo_enabled
readability	screen_name_length	verified
perplexity	favourites_count	contributors_enabled
lexical_diversity	active_days	is_translator
special_punctuation	statuses_count	is_translation_enabled
STDEV		profile_background_tile
punctuation_ratio		profile_use_background_image
word_count		has_extended_profile
sentiment_polarity		default_profile
sentence_count		default_profile_image

- **Numerical metadata** captures quantitative aspects of user behavior. These include statistics such as follower count, following count, and the number of active days. These values reflect the user's social influence, connectivity, and activity level on the platform. Prior research has shown that such metrics can offer strong discriminative signals when distinguishing between different users [17].

- **Categorical features** represent user properties that are inherently discrete. We utilize binary indicators such as account verification status, geo-tagging enablement, and default profile image usage. These features often correlate with account credibility and identity transparency, thereby contributing to robust user characterization.

- **Profile description** serves as a concise summary of their self-presentation, often including affiliations, roles, or interests that are typically authored manually. These descriptions provide complementary, high-quality signals for user identity characterization.

- **Linguistic features** are used to assess whether textual content is likely generated by LLMs. We extract specialized linguistic features, such as perplexity scores (measuring language model confidence), lexical diversity (measuring vocabulary richness), and sentiment polarity. These features are particularly effective for detecting LLM-generated text [39].

- **Abstract Meaning Representation (AMR)** provides a structured, graph-based representation of sentence meaning that encodes entities, relations, and events. To capture the semantic structure of user-generated content, we extract AMR graphs from user tweets over time. By modeling AMRs, the system gains access to abstract semantics beyond surface-level text, which is particularly valuable for identifying subtle manipulative behaviors or generated content.

Table 2: Notations used in BotDMM

Notation	Description
$\mathcal{G}_t$	Temporal graph snapshot at time step t
$\mathcal{U}$	Set of users
u_i	A user
$\mathcal{E}_t$	Set of edge at time step t
$\mathbf{f}^{\text{des}}$	User profile description
$\mathbf{f}^{\text{num}}$	Numerical metadata features
$\mathbf{f}^{\text{cat}}$	Categorical features of user
$\mathbf{f}^{\text{lin}}$	linguistic features (e.g., perplexity, sentiment)
$\mathbf{f}_t^{\text{tweet}}$	Tweet embedding at time step t
$\mathbf{f}_t^{\text{amr}}$	AMR-based embedding at time t
$\mathbf{h}_i^t$	Combined multi-modal feature vector of user u_i at time step t
$\tilde{\mathbf{c}}_i^t$	Content representation at time step t
$\tilde{\mathbf{s}}_i^t$	Structure representation at time step t
$\bar{\mathbf{c}}_i^t$	Orthogonalized content embedding
$\bar{\mathbf{s}}_i^t$	Orthogonalized structure embedding
$\bar{\mathbf{c}}_i$	Average content representation across time steps
$\bar{\mathbf{s}}_i$	Average structure representation across time steps
$\mathbf{q}_i$	Final fused representation of user u_i
$\mathcal{L}^{\text{ce}}$	Cross-entropy loss for classification
$\mathcal{L}^{\text{intra}}$	Intra-user contrastive loss (content vs structure)
$\mathcal{L}^{\text{inter}}$	Inter-user contrastive loss (same-class separation)
λ_1	Weight of intra-user contrastive loss
λ_2	Weight of inter-user contrastive loss
β	decorrelation constraint strength coefficient
k	Time span

- **Tweets** are used to model user behavior over time by capturing the sequence of posts they publish across multiple temporal steps. Each tweet is encoded using a pre-trained language model to obtain contextual embeddings. This dynamic information enables the model to detect behavioral trends, shifts in tone or intent, and anomalies indicative of automated or coordinated activities.

- **Social relationships** between users are modeled using a sequence of temporal graphs $\mathcal{G} = \{\mathcal{G}_1, \cdots, \mathcal{G}_t\}$, where each snapshot $\mathcal{G}_t = (\mathcal{U}, \mathcal{E}_t)$ captures follow and unfollow interactions at time t. The node set $\mathcal{U}$ represents the set of users, while the edge set $\mathcal{E}_t$ varies over time to reflect the dynamic nature of user relationships. This temporal graph formulation enables the model to capture subtle behavioral drift, such as an LLM-based bot adjusting its social connections to appear more human-like over time.

Our proposed framework leverages both content-level and structure-level features for bot detection. Instead of fusing all features into a single embedding, our approach explicitly separates semantic content from social structure, enabling more interpretable and targeted learning. The model is designed to track how users write, behave, and evolve over time, making it particularly well-suited for identifying accounts whose behavior is influenced by LLMs. Table 2 lists key notations and their definitions.

4. Methodology

In this section, we introduce the proposed BotDMM, a framework comprising three main modules. First, we extract

key information from the input features and align them to the same dimensionality. Next, we employ a dual-channel structure to learn the content and structure representations separately. To mitigate redundancy between the content and structure representations, we impose a decorrelation constraint on them, ensuring complementary feature learning. Finally, we perform predictions using a joint training approach that combines prediction loss and contrastive learning loss to optimize BotDMM. The overall framework is illustrated in Figure 2.

4.1. Feature Extraction

Our model processes six types of input features for each user. The first three features represent semantic information extracted by RoBERTa encoder [36], including $\mathbf{f}^{\text{des}}$ that contains user description representations, $\mathbf{f}^{\text{tweet}}$ that captures temporal tweet content representations, and $\mathbf{f}^{\text{amr}}$ that represents temporal AMR graph representations. The other three features capture different aspects of user behavior and characteristics, including $\mathbf{f}^{\text{num}}$ that represents numerical behavioral metrics, $\mathbf{f}^{\text{cat}}$ that contains categorical attributes, and $\mathbf{f}^{\text{lin}}$ that contains additional signals such as sentiment scores. Our model processes six types of input features for each user. The first three features represent semantic information extracted by the RoBERTa encoder [36], including $\mathbf{f}^{\text{des}}$ that contains user description representations, $\mathbf{f}^{\text{tweet}}$ that captures temporal tweet content representations, and $\mathbf{f}^{\text{amr}}$ that represents temporal AMR graph representations. The other three features capture different aspects of user behavior and characteristics, including $\mathbf{f}^{\text{num}}$ that represents numerical behavioral metrics, $\mathbf{f}^{\text{cat}}$ that contains categorical attributes, and $\mathbf{f}^{\text{lin}}$ that contains additional signals such as sentiment scores.

To ensure compatibility across different feature types, we map each feature into a unified embedding dimension. For each non-temporal feature, we employ a dedicated linear projection layer to transform, as follows:

$$\mathbf{h}_i^* = \mathbf{W}^*\mathbf{f}_i + \mathbf{b}^*, \tag{1}$$

where $\mathbf{W}^* \in \mathbb{R}^{\frac{d}{2}\times d^*}$ and $\mathbf{b}^* \in \mathbb{R}^{\frac{d}{2}}$ are learnable parameters, and d^* denotes the dimension of the input feature.

We use a lightweight sequence encoder (LSTM) to extract time-sliced tweet and AMR embeddings into feature sequences. The sequence encoder is not the key part of novelty; temporal features primarily arise from temporal graph snapshots and are leveraged by our dual-channel learning. The LSTM processing is defined as:

$$\mathbf{H}_i = \text{LSTM}([\mathbf{f}_i^{t-k}, \ldots, \mathbf{f}_i^t]) \tag{2}$$

From the resulting matrix $\mathbf{H}_i \in \mathbb{R}^{k\times\frac{d}{2}}$, we extract the representation at time step t for each temporal feature type. For tweet features, this extraction yields $\mathbf{h}_{i,t}^{\text{tweet}} = \mathbf{H}_i^{\text{tweet}}[t,:]$, while for AMR features, we obtain $\mathbf{h}_{i,t}^{\text{amr}} = \mathbf{H}_i^{\text{amr}}[t,:]$.

After that, we combine all features, including static features and temporal features, through concatenation to create the unified feature vector $\mathbf{h}_i^t \in \mathbb{R}^{3d}$ at time step t, as follows:

$$\mathbf{h}_\mathbf{i}^\mathbf{t} = [\mathbf{h}_i^{\text{des}} || \mathbf{h}_i^{\text{num}} || \mathbf{h}_i^{\text{cat}} || \mathbf{h}_i^{\text{lin}} || \mathbf{h}_{i,t}^{\text{tweet}} || \mathbf{h}_{i,t}^{\text{amr}}] \tag{3}$$

4.2. Dual-channel Feature Learning

4.2.1. Content Learning

In the feature extraction, we incorporate a set of features to capture the underlying user behavior patterns. To enhance the expressiveness of these features, the **Content Learning** module is employed to extract high-level semantic representations from $\mathbf{h}_i^t$, enabling the model to capture discriminative latent information for downstream tasks better. Specifically, we first transform the input feature $\mathbf{h}_i^t$ into a low-dimensional latent space, as below:

$$\mathbf{c}_i^t = \sigma\Big(\text{BN}(\mathbf{W}_1\mathbf{h}_i^t + \mathbf{b}_1)\Big), \tag{4}$$

where σ represents the ReLU activation function, $\mathbf{W}_1 \in \mathbb{R}^{d\times 3d}$ and $\mathbf{b}_1 \in \mathbb{R}^d$ are learnable weight matrix and bias. BN stands for Batch Normalization, which can help stabilize and accelerate the training process.

Subsequently, to further refine and enrich the latent representation, we employ a second transformation layer to learn more complex and abstract patterns from $\mathbf{c}_i^t$ as follows:

$$\tilde{\mathbf{c}}_i^t = \sigma\Big(\text{BN}(\mathbf{W}_2\mathbf{c}_i^t + \mathbf{b}_2)\Big), \tag{5}$$

where $\mathbf{W}_2 \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_2 \in \mathbb{R}^d$ are learnable parameters. σ and BN refer to the ReLU activation function and Batch Normalization, respectively. By employing this two-layer structure, we can learn robust feature representations that capture essential semantic patterns while filtering out irrelevant information, without introducing too much overhead.

4.2.2. Structure Learning

Most existing detection methods overlook the importance of network structure, which reveals users' friendships and connections [34, 38]. However, network structure is crucial for bot detection because bots often exhibit distinctive patterns in their social relationships, such as forming clusters with other automated accounts, following predictable connection strategies, or displaying abnormal ratios of followers to followees that differ significantly from organic human behavior [10]. To better utilize structure information for discriminating social bot, the **Structure Learning** module is employed to capture topological features of the network. Specifically, we employ a graph neural network to perform aggregation of neighbor information, capturing local graph structure information for better user (bot) representation.

Before conducting aggregation, similarly, we first reduce the dimensionality of the input feature $\mathbf{h}_i^t$ by mapping it into a low-dimensional latent space, as below:

$$\mathbf{s}_i^t = \sigma\Big(\text{BN}(\mathbf{W}_3\mathbf{h}_i^t + \mathbf{b}_3)\Big), \tag{6}$$

where σ represents the ReLU activation function, $\mathbf{W}_3 \in \mathbb{R}^{d\times 3d}$ and $\mathbf{b}_3 \in \mathbb{R}^d$ are learnable weight matrix and bias.

Given a temporal graph G_t, it is necessary to determine the weight of each edge, as the strengths of relationships among

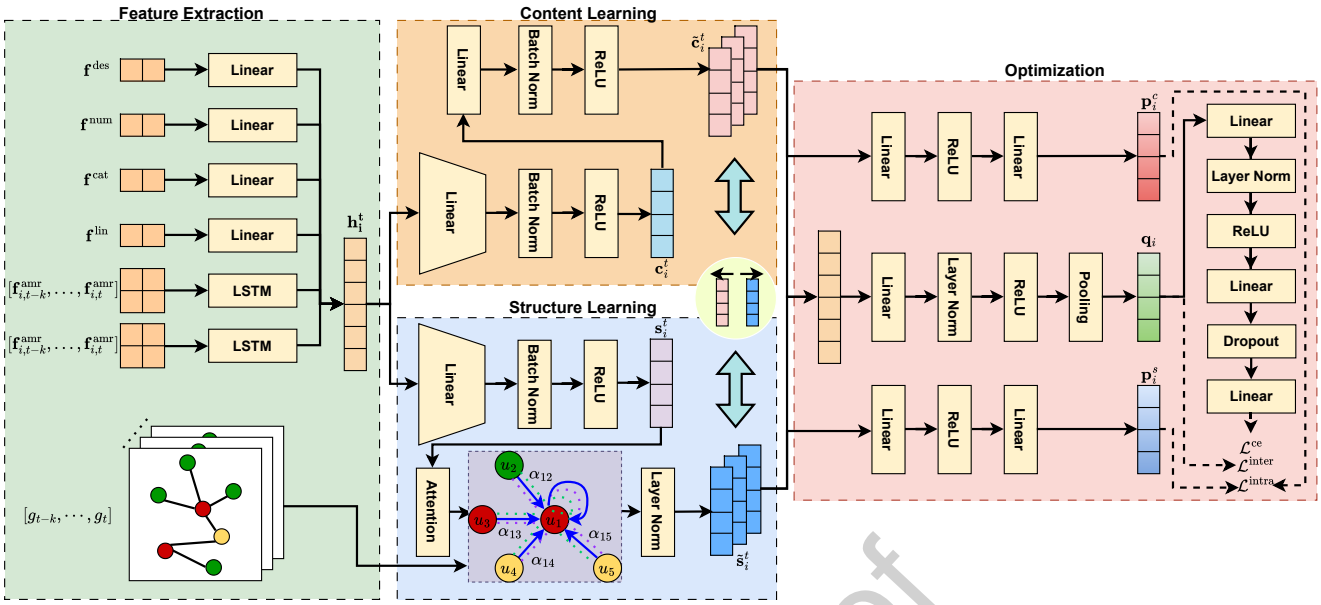


Figure 2: Framework of the proposed BotDMM

users could be very different. To capture such weights, we employ an attention mechanism [47, 46] that assigns higher importance to more relevant neighbors, thereby enabling more informative and effective aggregation. As an initial step, we apply two linear transformation layers to the target user node u_i and its neighbor u_j to obtain sufficient expressive information, i.e., $\hat{\mathbf{s}}_i^t = \mathbf{W_4}\mathbf{s}_i^t$ and $\hat{\mathbf{s}}_j^t = \mathbf{W_5}\mathbf{s}_j^t$, where $\mathbf{W_4} \in \mathbb{R}^{d\times d}$ and $\mathbf{W_5} \in \mathbb{R}^{d\times d}$ are two learnable weight matrices. Subsequently, we compute the attention coefficient that indicates the importance of u_j's features to u_i using these transformed representations as follows:

$$e_{ij} = \frac{\hat{\mathbf{s}}_i^t \cdot \hat{\mathbf{s}}_j^t}{\sqrt{d}}, \tag{7}$$

where $\mathbf{s}_i^t \cdot \mathbf{s}_j^t$ means dot product, and $\sqrt{d}$ is the scaling factor used to avoid extremely small gradients in the attention mechanism.

Once we compute e_{ij} for all $u_j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is the neighbors of u_i in the graph, we normalize these coefficients with the softmax function, making them easily comparable across different users, as follows:

$$\alpha_{ij} = \frac{\exp\big(\mathrm{WC}(e_{ij})\big)}{\sum_{u_n\in\mathcal{N}_i}\exp\big(\mathrm{WC}(e_{in})\big)}, \tag{8}$$

where WC is a weight capping function to avoid extremely small or large values:

$$\mathrm{WC}(e_{ij}) = \min\{\max\{e_{ij}, r\}, r\}, \tag{9}$$

where r is a constant.

Afterwards, we aggregate the information of neighbors into the target user by using the normalized coefficients, which are used to perform a linear combination of the corresponding neighbor features. Equation (9) formulates the aggregation process, where $\mathbf{W}_6 \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_6 \in \mathbb{R}^{d\times d}$ are learnable matrices, a_{ij} is the normalized attention coefficient, and LN represents Layer Normalization. $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$ are features of u_i and u_j, respectively.

$$\tilde{\mathbf{s}}_i^t = \mathrm{LN}\Big(\mathbf{h}_i^t + \mathbf{W}_6 \sum_{u_j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}_7\mathbf{h}_j^t\Big) \tag{10}$$

4.2.3. Decorrelation Constraint

Since content learning and structure learning both utilize h_i^t as the input, we introduce a feature vector-based decorrelation constraint mechanism to reduce redundancy between $\tilde{\mathbf{s}}_i^t$ and $\tilde{\mathbf{c}}_i^t$. Specifically, given a pair of feature vectors, the method first computes their cosine similarity and then removes the shared component from each vector by subtracting a scaled projection of one vector onto the other. The resulting vectors are re-normalized to maintain unit length. This process effectively separates overlapping information between content features and structure features, leading to more independent and robust feature representations. Let $\mathrm{sim}(\cdot,\cdot)$ be the cosine similarity function, we have:

$$\bar{\mathbf{c}}_i^t = \tilde{\mathbf{c}}_i^t - \beta\frac{\tilde{\mathbf{s}}_i^t}{\|\tilde{\mathbf{s}}_i^t\|_2}\mathrm{sim}(\tilde{\mathbf{c}}_i^t, \tilde{\mathbf{s}}_i^t), \tag{11}$$

$$\bar{\mathbf{s}}_i^t = \tilde{\mathbf{s}}_i^t - \beta\frac{\tilde{\mathbf{c}}_i^t}{\|\tilde{\mathbf{c}}_i^t\|_2}\mathrm{sim}(\tilde{\mathbf{c}}_i^t, \tilde{\mathbf{s}}_i^t), \tag{12}$$

where β is a hyperparameter controlling the orthogonality process. To be specific, the choice of β could optimize performance by balancing feature independence with joint discriminative capacity. A moderate value ensures meaningful cross-modal separation without eliminating beneficial information, with $\beta = 0.5$ representing the theoretical sweet spot between redundancy reduction and information preservation. By separating the content and structure features, the model can better capture the key

information that makes LLM-driven bots unique in each dimension, whether they fail at generating convincing content, exhibit suspicious network behavior, or both. This separation is crucial because some advanced bots might evade one type of detection but still exhibit a unique feature when viewed from another angle.

4.3. Model Optimization

4.3.1. Intra-user Contrastive Learning

Maximizing the agreement between different views of the same entity encourages the model to capture the invariant features of the entity, ultimately improving its performance [23]. In this work, we design an intra-user contrastive learning paradigm where the self-supervised signals are constructed from both the content feature and the structure feature. While our decorrelation constraint reduces redundancy between content and structure modalities at the feature level, the intra-user contrastive learning works to ensure these decorrelated representations remain mutually predictive of consistent user characterization.

We first use the average pooling to incorporate features from the current time step t to the past k time steps, as described in Equations (13) and (14).

$$\bar{\mathbf{c}}_i = \frac{1}{k}\sum_{t-k}^{t} \bar{\mathbf{c}}_i^t \tag{13}$$

$$\bar{\mathbf{s}}_i = \frac{1}{k}\sum_{t-k}^{t} \bar{\mathbf{s}}_i^t \tag{14}$$

Subsequently, $\bar{\mathbf{c}}_i$ and $\bar{\mathbf{s}}_i$ are fed into a two-layer fully connected network, as follows:

$$\mathbf{p}_i^c = \mathbf{W_9}\sigma(\mathbf{W}_8\bar{\mathbf{c}}_i + \mathbf{b}_8) + \mathbf{b}_9, \tag{15}$$

$$\mathbf{p}_i^s = \mathbf{W_{11}}\sigma(\mathbf{W}_{10}\bar{\mathbf{s}}_i + \mathbf{b}_{10}) + \mathbf{b}_{11}, \tag{16}$$

where $\mathbf{W}_8, \mathbf{W}_9, \mathbf{W}_{10}, \mathbf{W}_{11} \in \mathbb{R}^{\times}$ are learnable matrices, and $\mathbf{b}_8, \mathbf{b}_9, \mathbf{b}_{10}, \mathbf{b}_{11} \in \mathbb{R}^{d\times d}$ are learnable bias. σ represents the ReLU activation function. We adopt InfoNCE loss [42] to maximize the agreement between the content and the structure views. Specifically, the InfoNCE loss encourages the content and structure representations of the same user i (i.e., p_i^c and p_i^s) to be similar, while pushing apart the content representation of user i from the structure representations of other users j (i.e., p_i^c and p_j^s where $j \neq i$). The formulation is as follows:

$$\mathcal{L}^{\text{intra}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\left(\text{sim}(\mathbf{p}_i^c, \mathbf{p}_i^s)/\tau\right)}{\sum_{j=1}^{N}\exp\left(\text{sim}(\mathbf{p}_i^c, \mathbf{p}_j^s)/\tau\right)}, \tag{17}$$

where $\text{sim}(\cdot,\cdot)$ is the cosine similarity function, and τ is the temperature coefficient, and N denotes the number of users.

4.3.2. Inter-user Contrastive Learning

To better perform in bot detection, we devise an inter-user contrastive learning paradigm to capture latent mutual information between different users who belong to the same category. Specifically, it encourages entities belonging to the same class to have similar representations while pushing apart entities from different classes, enabling the model to learn discriminative features for better prediction.

We first concatenate the orthogonalized content and structure features and transform the combined vector into a low-dimensional space:

$$\mathbf{q}_i^t = \sigma\Big(\text{LN}\Big(\mathbf{W}_{12}[\bar{\mathbf{c}}_i^t \| \bar{\mathbf{s}}_i^t] + \mathbf{b}_{12}\Big)\Big), \tag{18}$$

where $[\cdot\|\cdot]$ is the concatenation operation, $\mathbf{W}_{12} \in \mathbb{R}^{d\cdot 2d}$ and $b_{12} \in \mathbb{R}^d$ are learnable parameters, LN means the Layer Normalization, and σ means the ReLU activation function. Subsequently, the average pooling aggregating features across all time steps is applied:

$$\mathbf{q}_i = \frac{1}{k}\sum_{t-k}^{k} \mathbf{q}_i^t. \tag{19}$$

Finally, similar to Equation (17), we use InfoNCE loss to maximize the mutual information between users belonging to the same category, e.g., both are human. Equation (20) describes the inter-user contrastive learning loss:

$$\mathcal{L}^{\text{inter}} = -\frac{1}{|M_i|}\sum_{m\in M_i} \log \frac{\exp\left(\text{sim}(\mathbf{q}_i, \mathbf{q}_m)/\tau\right)}{\sum_{j=1}^{N}\exp\left(\text{sim}(\mathbf{q}_i, \mathbf{q}_j)/\tau\right)}, \tag{20}$$

where $M_i = \{u_m | y_m = y_i, m \neq i\}$ represents a set of positive samples for u_i.

4.3.3. Prediction

Given the final feature $\mathbf{q}_i$, we adopt a three-layer fully connected network to obtain the logits $\mathbf{q}_i'''$ for each category, as follows:

$$\mathbf{q}_i' = \sigma\Big(\text{LN}(\mathbf{W}_{13}\mathbf{q}_i + \mathbf{b}_{13})\Big), \tag{21}$$

$$\mathbf{q}_i'' = \mathbf{W}_{14}\Big(\text{Dropout}(\mathbf{q}_i')\Big) + \mathbf{b}_{14}, \tag{22}$$

$$\mathbf{q}_i''' = \mathbf{W}_{15}\mathbf{q}_i'' + \mathbf{b}_{15}, \tag{23}$$

where $\mathbf{W}_{13} \in \mathbb{R}^{\frac{d}{2}\times d}$, $\mathbf{W}_{14}, \mathbf{W}_{15} \in \mathbb{R}^{\frac{d}{2}\times\frac{d}{2}}$, $\mathbf{b}_{13}, \mathbf{b}_{14}, \mathbf{b}_{15} \in \mathbb{R}^{\frac{d}{2}}$ are learnable parameters, and LN denotes Layer Normalization.

After that, a softmax function is applied to compute the probabilities of each category being the predicted one:

$$\hat{\mathbf{y}}_i = \text{softmax}(\mathbf{q}_\mathbf{i}'''). \tag{24}$$

At last, we formulate the learning objective as a cross-entropy loss function, as follows:

$$\mathcal{L}^{\text{ce}} = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_i^\top \log(\hat{\mathbf{y}}_i), \tag{25}$$

where $\mathbf{y}$ is the one-hot encoded label vector and $\hat{\mathbf{y}}$ is the predicted probability vector.

The overall loss of the proposed BotDMM is:

$$\mathcal{L}^{\text{total}} = \mathcal{L}^{\text{ce}} + \lambda_1\mathcal{L}^{\text{intra}} + \lambda_2\mathcal{L}^{\text{inter}}, \tag{26}$$

where λ_1 and λ_2 are two hyperparameters to control the weight of the corresponding contrastive learning loss, respectively.

Algorithm 1: Training of BotDMM (one epoch)

```
Initialize all model parameters
for each user u_i do
    Input f^des, f_i^cat, f_i^num, f_i^lin, f_{i,t}^tweet, f_{i,t}^amr, and G;
    Transform features using Equations (1)-(2);
    for t ∈ [t − k, t] do
        Concatenate features into h_i^t using Equation (3);
        Obtain c̃_i^t using Equations (4)-(5);
        Obtain s̃_i^t using Equations (6)-(10);
        Apply decorrelation constraint using
          Equations (11)-(12);
    Compute L^intra using Equations (13)-(17);
    Compute L^inter using Equations (18)-(20);
    Compute L^ce using Equations (21)-(25);
    Obtain the overall loss using Equation (26);
```

4.4. Complexity Analysis

Algorithm 1 illustrates the training process for our proposed BotDMM. In this pipeline, each user's multi-modal information is first projected into a unified dimensional space and concatenated into a feature vector. Subsequently, the unified vector is transformed into the content feature and structure feature in the dual-channel learning (DCL) module. At last, these learned feature representations are utilized to compute both feature-level and class-level contrastive losses, enabling the model to learn discriminative patterns for bot detection.

The computational cost of BotDMM consists of three main components. Suppose the input is a minibatch, and let B be the batch size. The feature extraction module has a complexity of $O(Bd(d_1 + d_2 + kd)$, where k refers to the number of time steps, d_1 denotes the total dimensionality of all static features, d_2 represents the total dimensionality of all temporal features, and d is the dimensionality of hidden representations. The complexity for the content learning module is $O(kBd^2)$. The computational cost for the structure learning module is $O(kBd^2 + k|\mathcal{N}|)$, where $|\mathcal{N}|$ denotes the total number of neighbors of all users. For the prediction module, the complexity is $O(Bd^2)$. Therefore, the overall complexity is $O(kBd^2 + k|\mathcal{N}| + Bd(d_1 + d_2))$.

5. Experiments and Analysis

In this section, we present six experiments designed to evaluate the performance of the proposed BotDMM model. The first experiment compares BotDMM with several baseline models in distinguishing between LLM-driven bots, traditional social bot, and genuine users. The second experiment examines the significance of the AMR component in improving detection accuracy. The third experiment analyzes the contribution of each module in the proposed model to the overall performance. The fourth experiment examines the effectiveness of the feature decoupling mechanism through correlation analysis between structural and content representations across different account types. The fifth experiment investigates the impact of varying the decorrelation constraint strength by adjusting the hyperparameter α within the range $[0, 1]$. Finally, the sixth experiment visualizes the 2D t-SNE projections of user embeddings generated by four different models, providing an intuitive comparison of their learned representations.

5.1. Experimental Setup

Datasets. Two datasets are mainly used in our experiments, i.e., TwiBot-20 [16] and SDAR [33]. Specifically, TwiBot-20 served as our primary dataset due to its comprehensive and realistic representation of the contemporary Twitter ecosystem. Each user record included (i) numerical profile statistics (e.g., number of followers, followings, and favourites), (ii) categorical attributes such as protection status, verification status, and location, (iii) explicit follower/following graph edges, (iv) full tweet text, and (v) the associated domain of the account. These features collectively provided rich insights into both individual behaviors and the surrounding network structure. SDAR, our secondary dataset, was sourced from Chirper.ai, a social platform populated by synthetic identities driven by large language models (LLMs). While SDAR adopted the same metadata schema as TwiBot-20, all tweets are generated by a variety of LLMs, offering a high-fidelity corpus of LLM-generated content for evaluating bot detection in LLM-centric environments.

Additionally, we also evaluate the performance of BotDMM on BotSim-24 dataset [45], which contains 2,907 user accounts with 1,907 human and 1,000 LLM-driven bots. This dataset served as an additional validation benchmark to assess the generalizability of our approach across different data distributions and bot detection scenarios. Unlike the TwiBot-20 and SDAR datasets, BotSim-24 is constructed from Reddit platform data and contains 2,907 user accounts (1,907 humans and 1,000 bots). Bot behaviors were powered by AI agents, which simulate more sophisticated and realistic bot activities. This Reddit-based dataset with LLM-driven bot behaviors serves as an additional validation benchmark to assess the cross-platform generalizability of our approach across different social media ecosystems and bot sophistication levels.

Data Preprocessing. We began by applying standard text-cleaning procedures, including the removal of control characters, extraneous symbols, and URL artifacts. From the TwiBot-20 dataset, we sample 11,826 labelled accounts. To simulate LLM-driven behavior, we replaced the tweet content and profile descriptions of 3,000 bot accounts from this sample with LLM-generated tweets sourced from SDAR, relabeled these accounts as LLM-driven bots, while left the remaining accounts and their original annotations unchanged. This controlled substitution preserves label reliability and keeps text-centric features (e.g., tweets, AMR, linguistic cues) semantically coherent over time.

To incorporate temporal dynamics, we assume that each user's tweet timeline and neighbor list are chronologically ordered from oldest (bottom) to newest (top). Based on this assumption, both sequences are divided into T equal-length segments, each representing a distinct phase in the user's life cycle on the platform. For each temporal slice, we reconstruct

the user's local subgraph using the adaptive edge-rewiring strategy introduced in [18], conditioning the rewiring choice on the current user's and candidate user's profile descriptions used as contextual information, which modeled the evolving decision-making patterns of LLM-driven bots when forming social connections. This process produced a sequence of graph snapshots $\{\mathcal{G}_t\}_{t=1}^{T}$, where changes in content and structure were aligned over time.

These dynamic views enabled our model to learn the gradual behavioral shifts of LLM-driven bots in both messaging and connectivity, which are patterns often obscured in static graphs but critical for robust detection. The final dataset was partitioned into 70% training, 20% validation, and 10% testing sets. All processed data will be publicly released to ensure reproducibility and compliance with data governance standards.

Feature Preprocessing. To ensure consistency and high-quality representation across all feature modalities, we applied dedicated preprocessing strategies to the six feature types incorporated in our model.

For user profile descriptions, we tokenized and encoded the text using a pre-trained RoBERTa model to obtain semantic embeddings. The same RoBERTa encoder was applied to raw tweets collected at each time step, yielding contextualized representations that capture temporal variations in user language. To introduce higher-level semantic structure, we generated AMR graphs for each tweet using a neural parser based on AMR-BART [4]. These graphs were then linearized into textual sequences and re-encoded with RoBERTa to derive AMR-informed semantic embeddings. Numerical metadata (e.g., follower counts) and categorical attributes (e.g., verification status) were preprocessed via one-hot encoding, followed by a fully connected projection layer to embed them in a shared latent space. LLM-specific features, including perplexity scores and sentiment values, were likewise projected through a dedicated fully connected layer.

All resulting embeddings, including textual, structural, numerical, and semantic, were concatenated to form a unified multimodal feature vector for each user at each time step.

Evaluation Metrics. To assess the performance of our model, we employed *accuracy*, *F1 score*, and *Matthews Correlation Coefficient (MCC)*. These metrics provide a comprehensive evaluation of classification correctness, robustness to class imbalance, and correlation with true labels.

Hyperparameters. To ensure a fair comparison across all experiments, we adopted consistent hyperparameter settings. Since our framework simultaneously optimizes classification and two contrastive learning objectives, we analyzed dropout rates to understand how regularization affects the balance between these competing goals. Too many dropouts might affect essential patterns needed for LLM-driven bot detection, while too few might cause overfitting. The model was trained using the Adam optimizer with a learning rate of was tuned via grid search over $\{1e-3, 5e-4, 1e-4, 5e-5\}$, with $1e-3$ was selected based on validation performance and a dropout rate of 0.3 was chosen from $\{0.3, 0.5, 0.7\}$ to control model capacity and mitigate overfitting. Training proceeded for a maximum of 100 epochs, with early stopping based on validation loss to prevent overfitting. To balance the multi-part objective, the feature-level contrastive loss was weighted by $\lambda_1 = 0.1$, and the class-level contrastive loss by $\lambda_2 = 0.1$.

Baselines. We evaluate our model against 12 representative baselines spanning traditional machine learning, deep learning, and graph-based approaches. These baselines are outlined below:

- **Lee et al.** [32] employs a random forest classifier based on a set of user features, including profile metadata and user activity statistics. This early approach is effective for traditional bots but struggles with more sophisticated behaviors.
- **Yang et al.** [53] uses a lightweight random forest model that relies on minimal metadata and twelve derived features to enable large-scale bot detection.
- **Wei et al.** [49] applies an LSTM model to capture sequential patterns in users' tweet histories, exploiting temporal dependencies in text. However, it does not leverage user interaction or network structure.
- **GCN** [29] applies a graph convolutional network that aggregates feature information from a user's local neighborhood in the social graph. It captures structural patterns among users but treats all neighbors equally, which limits its effectiveness in heterogeneous interaction contexts.
- **GAT** [47] incorporates an attention mechanism into the graph convolutional framework to assign learnable weights to neighboring users. This allows the model to focus on more informative connections and better capture user-level relational dynamics.
- **BERT** [11] uses a transformer-based language model to encode user-generated content, capturing contextual and semantic patterns in text. Its main strength lies in understanding nuanced language, but it lacks awareness of networks.
- **RoBERTa** [36] is an improved version of BERT, trained on more data with dynamic masking, offering richer textual feature extraction.
- **BotRGCN** [17] combines user content and metadata within a relational graph convolutional network to capture both node attributes and social connections. This allows detection based on both text and user network patterns.
- **GraphSAGE** [19] aggregates neighborhood information in large social graphs to learn node embeddings representing relational and structural properties. It can capture complex network patterns but does not directly utilize textual content.

- **ISBT** [13] builds on BotRGCN by incorporating intent recognition and entropy features, specifically enhancing detection of LLM-driven bots. It targets the semantic diversity that LLM-driven bots often exhibit.
- **BotMoE** [37] proposes a mixture-of-experts framework, where different modules focus on metadata, text, or graph features, with integration guided by community information. This modular design increases robustness against varied bot tactics.
- **LLM-BotGuard** [12] employs type-specific expert modules within a GraphSage to model LLM features, metadata, and network relations for each social account.

5.2. Experiment 1: Model Performance

Performance on Twibot-20/SDAR. This experiment evaluates the performance of BotDMM in distinguishing among human users, traditional social bot, and LLM-driven bots. We compare BotDMM against 12 representative baseline models spanning traditional feature-based classifiers, graph-based neural networks, deep learning language models, and recent state-of-the-art bot detection frameworks. The results are presented in Table 3.

BotDMM achieves superior performance across all evaluation metrics, demonstrating its effectiveness in capturing both linguistic and relational patterns critical for identifying sophisticated LLM-driven bots. The results reveal clear performance tiers among different model categories. Traditional machine learning approaches often perform poorly due to their reliance on static features and simple heuristics, which prove insufficient for detecting the complexity of modern bots. Graph-based models demonstrate improved performance by leveraging network structure. However, their limited ability to capture textual semantics restricts their effectiveness against LLM-driven bots. Deep learning language models demonstrate reasonable performance through contextual understanding, but their lack of relational modeling limits their capabilities.

Recent state-of-the-art models, particularly LLM-BotGuard, achieve competitive results. However, BotDMM consistently outperforms all baselines, indicating its superior ability to jointly model user-level relational structures and deep semantic features. The strong performance across multiple metrics confirms BotDMM's robustness in handling the complex tri-class classification challenge of distinguishing between human users, traditional bots, and LLM-driven bots.

Performance on BotSim-24. To evaluate the cross-platform generalizability of BotDMM, we conduct additional experiments on the BotSim-24 dataset. This experiment is a binary classification task, as BotSim-24 contains only human accounts and LLM-driven bots, without traditional bots. We employ the same baseline methods used in Section 5.2 to ensure fair comparison and maintain experimental consistency. To ensure experimental fairness and eliminate potential confounding factors, we maintain consistent experimental parameters across all methods.

Table 3: Performance comparison with baselines on Twibot-20/SDAR

Models	Accuracy	F1 score	MCC
Lee et al. [32]	0.4996	0.4622	0.3298
Yang et al. [53]	0.5748	0.5716	0.3817
Wei et al. [49]	0.6720	0.7353	0.3791
BERT [11]	0.7236	0.7213	0.5845
RoBERTa [36]	0.7684	0.7685	0.6070
GCN [29]	0.7030	0.6912	0.5560
GAT [47]	0.7402	0.7309	0.6166
GraphSage [19]	0.8014	0.8005	0.7108
BotRGCN [17]	0.7802	0.7749	0.6798
ISBT [13]	0.7828	0.7820	0.6805
BotMoE [37]	0.8402	0.8423	0.7517
LLM-BotGuard [12]	0.8563	0.8533	0.7894
BotDMM	**0.8833**	**0.8821**	**0.8255**

The experimental results are presented in Table 4.Our BotDMM model achieves superior performance across all evaluation metrics, with an accuracy of 95.42%, F1 score of 93.62%, and MCC of 90.25%. These results represent substantial improvements over the strongest baseline, LLM-BotGuard, with gains of 4.58% in accuracy, 5.38% in F1 score, and 7.83% in MCC.

The performance analysis reveals several key insights across different approaches. Traditional feature-based methods, such as those by Lee et al. and Yang et al., demonstrate limited effectiveness on this Reddit-based dataset, suggesting significant platform-specific behavioral differences that challenge the transferability of handcrafted features. Text-based approaches, including BERT and RoBERTa, achieve moderate performance but fail to capture the complex social dynamics present in LLM-driven bot behaviors, indicating that content analysis alone is insufficient for detecting sophisticated LLM-driven bots. Graph-based methods demonstrate notably improved performance, with GraphSage achieving competitive results at 86.64% accuracy, which highlights the critical importance of network structure in modern bot detection tasks. Recent state-of-the-art methods, such as BotMoE and LLM-BotGuard, demonstrate strong performance, confirming the significant challenge posed by sophisticated LLM-driven bots while establishing a high baseline for comparison.

The performance achieved by BotDMM demonstrates the effectiveness of our approach in generalizing across different platforms and bot sophistication levels.

5.3. Experiment 2: Ablation Study

To assess the contribution of each component in our proposed model, we conducted a comprehensive ablation study.

- **w/o AMR** does not have abstract meaning representaion.
- **w/o DCL** does not have dual-channel learning module.
- **w/o SLM** only has content learning module.

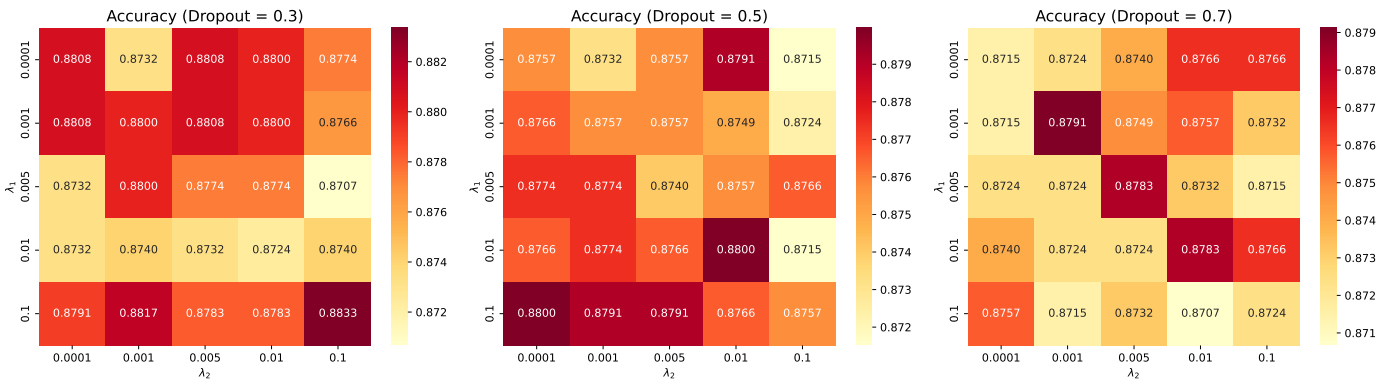


Figure 3: The analysis of different dropout and contrastive learning weight on Twibot20/SDAR

Table 4: Performance comparison with baselines on BotSim-24

Models	Accuracy	F1 score	MCC
Lee et al. [32]	0.7557	0.6484	0.4613
Yang et al. [53]	0.7137	0.6305	0.4087
Wei et al. [49]	0.8511	0.7251	0.6780
BERT [11]	0.7824	0.5366	0.5248
RoBERTa [36]	0.8053	0.6047	0.5781
GCN [29]	0.7061	0.4211	0.2871
GAT [47]	0.8168	0.7778	0.6550
GraphSage [19]	0.8664	0.8372	0.7573
BotRGCN [17]	0.8435	0.8145	0.7234
ISBT [13]	0.8702	0.8411	0.7631
BotMoE [37]	0.8969	0.8696	0.8053
LLM-BotGuard [12]	0.9084	0.8824	0.8242
BotDMM	**0.9542**	**0.9362**	**0.9025**

Table 5: Performance of BotDMM with different modules on Twibot20/SDAR

Models	Accuracy	F1	MCC
w/o AMR	0.8648	0.8628	0.7986
w/o DCL	0.8681	0.8653	0.8035
w/o SLM	0.8732	0.8704	0.8111
w/o CLM	0.8715	0.8711	0.8071
w/o OC	0.8800	0.8784	0.8199
w/o CL	0.8808	0.8790	0.8216
BotDMM	**0.8833**	**0.8821**	**0.8255**

- **w/o CLM** only has structure learning module.
- **w/o OC** has no decorrelation constraint applied.
- **w/o CL** does not adopt contrastive learning.

Table 5 reports the results in terms of Accuracy, F1 Score, and MCC across various ablated model configurations.

Abstract Meaning Representation (AMR) emerges as a crucial component for model performance. When AMR features are removed, the model experiences significant performance degradation, with accuracy dropping from 88.33% to 86.48%, F1 score decreasing to 86.28%, and MCC falling to 79.86%. This decline demonstrates that AMR captures essential semantic patterns that distinguish between different user types, particularly in identifying the structured linguistic behaviors characteristic of LLM-driven bots. The semantic abstraction provided by AMR appears to be especially valuable for detecting the subtle but systematic differences in how LLM-generated content is structured compared to human-generated text.

The dual-channel learning module proves essential for optimal performance. When removed entirely, the model experiences substantial degradation across all metrics, demonstrating the critical importance of simultaneously capturing both content and structural features. Individual ablation of either component also yields noticeable performance drops, with the structure learning module showing slightly greater impact than the content learning module. These results confirm that both structural and textual information are indispensable for accurately distinguishing between humans, traditional bots, and LLM-driven bots.

The auxiliary learning strategies provide additional performance benefits. Removing the decorrelation constraint, which maintains feature separation within user categories, results in modest performance decreases. Similarly, eliminating the contrastive learning component, designed to enhance differentiation between user types, leads to comparable reductions in model effectiveness. These findings demonstrate that both mechanisms make a meaningful contribution to the robustness and quality of the learned representations.

In contrast, the full version of our model, BotDMM, which integrates all modules, including structure and content learning, decorrelation constraints, and contrastive learning, achieves the best overall performance. This ablation analysis confirms that each component plays a distinct and complementary role in enhancing the model's performance in identifying and distinguishing LLM-driven bots from human users and traditional bots.

5.4. Experiment 3: Hyperparameter Analysis

In this experiment, we analyze the impact of two contrastive learning weights, i.e., λ_2 (inter-user) and λ_1 (intra-user), on

Table 6: Correlation analysis between content and structure features by class on Twibot20/SDAR

Class	Average Correlation
Human	0.2972
Traditional Bot	0.3403
LLM-driven Bot	0.2826
Overall	0.3083

model performance. We also examine the influence of different dropout rates to evaluate their interaction with contrastive learning. Early stopping was applied, halting training if the validation loss failed to improve for 10 consecutive epochs. The model checkpoint with the lowest validation loss was then restored.

Figure 3 presents heatmaps showing accuracy across different parameter combinations. The heatmaps reveal that lower dropout rates consistently yield better performance. Specifically, a dropout rate of 0.3 resulted in the highest accuracy across most parameter settings. This suggests that retaining more active neurons helps the model preserve informative patterns critical for distinguishing LLM-driven bots. Under this setting, the highest accuracy was achieved when both λ_2 and λ_{feature} were set to 0.1, indicating that moderate contrastive weights strike an effective balance between enhancing representation quality and maintaining training stability. As dropout increases, performance gradually degrades. At a dropout rate of 0.5, the best accuracy, which was 0.8800, occurred when λ_1 was high, 0.1, and λ_{class} was low, 0.0001, suggesting that feature-level contrastive learning becomes more influential when neuron activations are more aggressively pruned. At the highest dropout rate of 0.7, performance further declined, with peak accuracy reaching only 0.8791 at minimal contrastive weights.

Overall, these results highlight the importance of tuning both regularisation and contrastive learning parameters. The most effective configuration involves a lower dropout rate of 0.3, combined with moderate contrastive weights ($\lambda_2 = \lambda_1 = 0.1$), which together yield the best trade-off between model generalization and feature discriminability.

5.5. Experiment 4: Feature Correlation Analysis

To further evaluate the effectiveness of the proposed dual-channel feature learning mechanism, we conducted a correlation analysis between structural and content representations across different user categories. The results, summarized in Table 6, show that the overall average correlation is relatively low (0.3083), suggesting that the model successfully learns to reduce the redundancy of structural and semantic information.

When broken down by class, LLM-driven bots and human users exhibit the lowest average correlation values, having 0.2826 and 0.2972, respectively. This indicates that their structural and content behaviors are more independent. This aligns with expectations, as LLM-driven bots are designed to mimic human-like textual behavior while maintaining bot-like structural patterns, and human users naturally display diverse and less correlated behaviors across modalities. In contrast, traditional bots demonstrate a higher correlation score of 0.3403, which can be attributed to their repetitive and often templated behavior, resulting in stronger alignment between structural and content features.

These findings validate the effectiveness of the decoupling strategy in separating heterogeneous feature types. By minimizing feature entanglement, the model is better equipped to capture the nuanced characteristics of different account types, thereby enhancing its ability to detect sophisticated LLM-driven bots.

5.6. Experiment 5: Decorrelation Constraint Weight Analysis

This experiment aims to evaluate how the strength of the decorrelation constraint affects the model's ability to reduce the redundancy between content and structure representations, as well as its performance in detecting LLM-driven bots. Specifically, we investigate the influence of varying the orthogonality weight α, which governs the extent to which the content and structure features are encouraged to be decorrelated during training.

To this end, we vary α within the range $[0, 1]$ and assess the model's performance using three evaluation metrics, i.e., accuracy, F1 score, and MCC. The results, presented in Figure 5, reveal that the model achieves the best performance when $\alpha = 0.5$. At this setting, all three metrics reach their highest values, indicating an optimal balance between feature separation and information retention. In contrast, both the absence of the constraint ($\alpha = 0$) and the imposition of a maximal constraint ($\alpha = 1$) result in performance drops. This suggests that while enforcing feature independence is beneficial, excessive separation can degrade the shared representational quality needed for accurate classification.

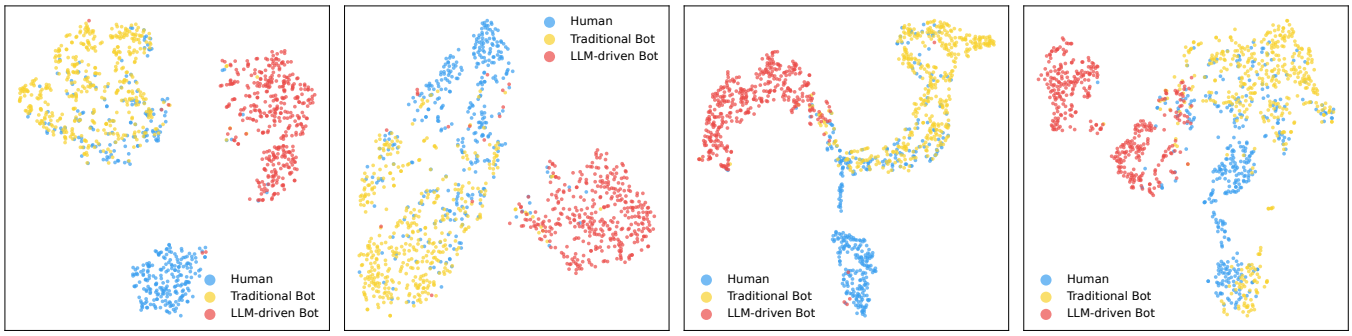
These findings underscore the importance of adjusting the decorrelation constraint. A moderate value of α effectively promotes complementary feature representations while preserving their joint discriminative capacity, turning out to be an essential factor in the success of the BotDMM framework.

5.7. Experiment 6: User Representation Visualization

The last experiment aims to assess the effectiveness of the learned user representations by visualizing their distributions in a two-dimensional space. We employ t-SNE to project the high-dimensional user embeddings generated by four different models into 2D for comparison. Each point in the plot represents a user account, colored according to its ground-truth label, i.e., blue for Humans, orange for Traditional Bots, and green for LLM-driven Bots.

As shown in Figure 4, the embeddings produced by BotDMM exhibit clear and well-separated clusters for each user category. In particular, BotDMM successfully differentiates LLM-driven bots from both human users and traditional bots. In contrast, embeddings learned by baseline models display noticeable overlaps between categories, making it more challenging to distinguish between them.

The superior cluster separation achieved by BotDMM can be attributed to its dual-channel feature learning mechanism



(a) BotDMM (b) LLM-BotGuard (c) BotMoE (d) BotRGCN

Figure 4: User representation visualization using t-SNE projection on Twibot20/SDAR. The horizontal and vertical axes represent the first and second principal components of the projected space. Colors indicate ground-truth labels.

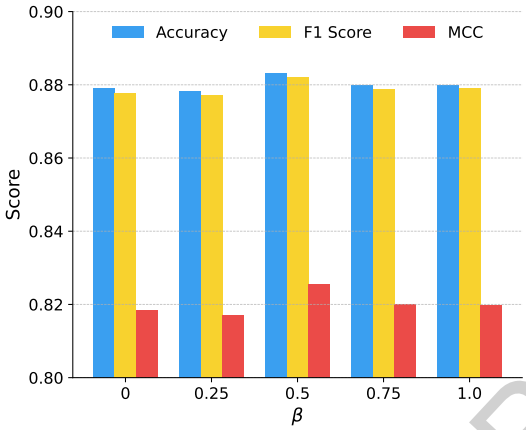


Figure 5: Performance of accuracy, F1 score, and MCC under different decorrelation constraint weights β on Twibot20/SDAR.

and the use of decorrelation constraints. These components enable the model to reduce the redundancy between structural and content representations more effectively, leading to more interpretable and discriminative embeddings.

Overall, this visualization not only reinforces the quantitative results observed in earlier experiments but also provides intuitive evidence of BotDMM's ability to learn semantically meaningful and well-structured user representations.

5.8. Discussions

The experimental results provide strong evidence for the effectiveness and robustness of our proposed BotDMM. The key insights drawn from the experiments are summarized below:

- *BotDMM excels in distinguishing LLM-driven bots from other user types.* BotDMM achieves the highest scores across all evaluation metrics compared to 12 baseline models. Its superior performance highlights the necessity of modeling both semantic and relational features to capture the complex behaviors exhibited by LLM-driven bots.

- *Reducing the redundancy between structure and content representations is critical for effective feature learning.* Through the ablation study and the correlation, it is evident that BotDMM effectively separates structural and content information. LLM-driven bots and humans exhibit lower correlation scores between modalities, indicating successful disentanglement. This separation reduces mutual interference, contributing to clearer class boundaries.

- *Decorrelation constraints improve representational clarity when properly tuned.* A moderate constraint strength can produce an optimal performance, while excessive or absent constraint weakens the model. This confirms the importance of promoting independence between modalities without enforcing complete separation.

- *Contrastive learning enhances inter-class separability and is sensitive to regularization.* Based on the ablation study and parameter analysis, contrastive learning enhances performance, particularly when combined with lower dropout rates. The best performance occurs when both λ_2 and λ_1 are set to 0.1 and dropout is low (0.3), suggesting that moderate contrastive strength and soft regularization yield optimal feature discriminability.

In summary, BotDMM achieves state-of-the-art performance in detecting LLM-driven bots by effectively combining disentangled feature learning with contrastive and orthogonality-based regularization. Moreover, we only used public data and did not identify individual users and keep the claims scoped to the studied domains.

6. Conclusion

In this paper, we propose a novel BotDMM framework for detecting LLM-driven social bot. Through learning the content of users' posts and users' neighboring relationships, the proposed BotDMM can effectively discriminate between humans, traditional bots, and LLM-driven bots. Meanwhile, the inter- and intra-user contrastive paradigms effectively improve the prediction performance. Extensive experiments on the dataset demonstrated state-of-the-art performance, while ablation clarified the contribution of each architectural component.

Nevertheless, BotDMM still has some limitations. The current framework does not capture the strategic decision-making

patterns that LLM-driven bots employ when selecting which users to connect with on social media platforms. In addition, our injection strategy assumes LLM-driven content under control, and SDAR is based on Chirper.ai. In future work, we will investigate these decision-making patterns to enable a deeper understanding of LLM-driven bot behavior and incorporate deeper user intent as key features estimated from temporal semantics.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Abulaish, M., Fazil, M., 2020. Socialbots: Impacts, threat-dimensions, and defense challenges. IEEE Technology and Society Magazine 39, 52–61.

[2] Al-Qurishi, M., Hossain, M.S., Alrubaian, M., Rahman, S.M.M., Alamri, A., 2017. Leveraging analysis of user behavior to identify malicious activities in large-scale social networks. IEEE Transactions on Industrial Informatics 14, 799–813.

[3] Andriotis, P., Takasu, A., 2018. Emotional bots: content-based spammer detection on social media, in: 2018 IEEE international workshop on information forensics and security (WIFS), IEEE. pp. 1–8.

[4] Bai, X., Chen, Y., Zhang, Y., 2022. Graph pre-training for amr parsing and generation. arXiv preprint arXiv:2203.07836 .

[5] Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., Schneider, N., 2013. Abstract meaning representation for sembanking, in: Proceedings of the 7th linguistic annotation workshop and interoperability with discourse, pp. 178–186.

[6] Chowdhury, S., Khanzadeh, M., Akula, R., Zhang, F., Zhang, S., Medal, H., Marufuzzaman, M., Bian, L., 2017. Botnet detection using graph-based feature clustering. Journal of Big Data 4, 1–23.

[7] Dan, J., Jieqi, T., 2017. Study of bot detection on sina-weibo based on machine learning, in: 2017 International Conference on Service Systems and Service Management, IEEE. pp. 1–5.

[8] Daouadi, K.E., Rebaï, R.Z., Amous, I., 2019. Bot detection on online social networks using deep forest, in: Artificial Intelligence Methods in Intelligent Algorithms: Proceedings of 8th Computer Science On-line Conference 2019, Vol. 2 8, Springer. pp. 307–315.

[9] Davis, C.A., Varol, O., Ferrara, E., Flammini, A., Menczer, F., 2016. Botornot: A system to evaluate social bots, in: Proceedings of the 25th international conference companion on world wide web, pp. 273–274.

[10] Dehghan, A., Siuta, K., Skorupka, A., Dubey, A., Betlen, A., Miller, D., Xu, W., Kamiński, B., Prałat, P., 2023. Detecting bots in social-networks using node and structural embeddings. Journal of Big Data 10, 119.

[11] Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. Bert: Pre-training of deep bidirectional transformers for language understanding, 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. 4171–4186.

[12] Duan, J., Li, W., Bai, Q., Nguyen, M., Wang, X., Jiang, J., 2025. Llm-botguard: A novel framework for detecting llm-driven bots with mixture of experts and graph neural networks. IEEE Transactions on Computational Social Systems .

[13] Duan, J., Li, Z., Wang, X., Li, W., Bai, Q., Nguyen, M., 2024. Intent-spectrum bottracker: Tackling llm-based social media bots through an enhanced botrgcn model with intention and entropy measurement, in: Principle and Practice of Data and Knowledge Acquisition Workshop, Springer. pp. 55–67.

[14] Ellaky, Z., Benabbou, F., Ouahabi, S., Sael, N., 2021. Word embedding for social bot detection systems, in: 2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS), IEEE. pp. 1–8.

[15] Fazil, M., Sah, A.K., Abulaish, M., 2021. Deepsbd: a deep neural network model with attention mechanism for socialbot detection. IEEE Transactions on Information Forensics and Security 16, 4211–4223.

[16] Feng, S., Wan, H., Wang, N., Li, J., Luo, M., 2021a. Twibot-20: A comprehensive twitter bot detection benchmark, in: Proceedings of the 30th ACM international conference on information & knowledge management, pp. 4485–4494.

[17] Feng, S., Wan, H., Wang, N., Luo, M., 2021b. 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.

[18] Feng, S., Wan, H., Wang, N., Tan, Z., Luo, M., Tsvetkov, Y., 2024. What does the bot say? opportunities and risks of large language models in social media bot detection. arXiv preprint arXiv:2402.00371 .

[19] Hamilton, W., Ying, Z., Leskovec, J., 2017. Inductive representation learning on large graphs. Advances in neural information processing systems 30.

[20] Hayawi, K., Saha, S., Masud, M.M., Mathew, S.S., Kaosar, M., 2023. Social media bot detection with deep learning methods: a systematic review. Neural Computing and Applications 35, 8903–8918.

[21] He, B., Yang, Y., Wu, Q., Liu, H., Yang, R., Peng, H., Wang, X., Liao, Y., Zhou, P., 2024a. Botdgt: Dynamicity-aware social bot detection with dynamic graph transformers. arXiv preprint arXiv:2404.15070 .

[22] He, B., Yang, Y., Wu, Q., Liu, H., Yang, R., Peng, H., Wang, X., Liao, Y., Zhou, P., 2024b. Dynamicity-aware social bot detection with dynamic graph transformers. IJCAI. ijcai. org , 5844–5852.

[23] He, K., Fan, H., Wu, Y., Xie, S., Girshick, R., 2020. Momentum contrast for unsupervised visual representation learning, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE. pp. 9726–9735.

[24] Huang, H., Tian, H., Zheng, X., Zhang, X., Zeng, D.D., Wang, F.Y., 2024. Cgnn: A compatibility-aware graph neural network for social media bot detection. IEEE Transactions on Computational Social Systems .

[25] Huang, Z., Lv, Z., Han, X., Li, B., Lu, M., Li, D., 2022. Social bot-aware graph neural network for early rumor detection, in: proceedings of the 29th international conference on computational linguistics, pp. 6680–6690.

[26] Ikram, M., Onwuzurike, L., Farooqi, S., Cristofaro, E.D., Friedman, A., Jourjon, G., Kaafar, M.A., Shafiq, M.Z., 2017. Measuring, characterizing, and detecting facebook like farms. ACM Transactions on Privacy and Security (TOPS) 20, 1–28.

[27] Kantepe, M., Ganiz, M.C., 2017. Preprocessing framework for twitter bot detection, in: 2017 International conference on computer science and engineering (ubmk), IEEE. pp. 630–634.

[28] Kenny, R., Fischhoff, B., Davis, A., Carley, K.M., Canfield, C., 2024. Duped by bots: why some are better than others at detecting fake social media personas. Human factors 66, 88–102.

[29] Kipf, T.N., Welling, M., 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 .

[30] Kudugunta, S., Ferrara, E., 2018. Deep neural networks for bot detection. Information Sciences 467, 312–322.

[31] Kumar, S., Garg, S., Vats, Y., Parihar, A.S., 2021. Content based bot detection using bot language model and bert embeddings, in: 2021 5th International Conference on Computer, Communication and Signal Processing (ICCCSP), IEEE. pp. 285–289.

[32] Lee, K., Eoff, B., Caverlee, J., 2011. Seven months with the devils: A long-term study of content polluters on twitter, in: Proceedings of the international AAAI conference on web and social media, pp. 185–192.

[33] Li, S., Yang, J., Zhao, K., 2023a. Are you in a masquerade? exploring the behavior and impact of large language model driven social bots in online social networks. arXiv preprint arXiv:2307.10337 .

[34] Li, S., Zhao, C., Li, Q., Huang, J., Zhao, D., Zhu, P., 2023b. Botfinder: a novel framework for social bots detection in online social networks based on graph embedding and community detection. World Wide Web 26, 1793–1809.

[35] Lingam, G., Rout, R.R., Somayajulu, D.V., Das, S.K., 2020. Social botnet community detection: a novel approach based on behavioral similarity in twitter network using deep learning, in: Proceedings of the 15th ACM Asia conference on computer and communications security, pp. 708–718.

[36] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. Roberta: A robustly optimized

bert pretraining approach. arXiv preprint arXiv:1907.11692 .
[37] Liu, Y., Tan, Z., Wang, H., Feng, S., Zheng, Q., Luo, M., 2023. 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.
[38] Martín-Gutiérrez, D., Hernández-Peñaloza, G., Hernández, A.B., Lozano-Diez, A., Álvarez, F., 2021. A deep learning approach for robust detection of bots in twitter using transformers. IEEE Access 9, 54591–54601.
[39] Mindner, L., Schlippe, T., Schaaff, K., 2023. Classification of human- and ai-generated texts: Investigating features for chatgpt, in: International conference on artificial intelligence in education technology, Springer. pp. 152–170.
[40] Ng, L.H.X., Carley, K.M., 2023. Botbuster: Multi-platform bot detection using a mixture of experts, in: Proceedings of the international AAAI conference on web and social media, pp. 686–697.
[41] Ng, L.H.X., Carley, K.M., 2025. Are llm-powered social media bots realistic? arXiv e-prints , arXiv–2508.
[42] Oord, A.v.d., Li, Y., Vinyals, O., 2018. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 .
[43] Park, H., Kim, B., Kim, B., 2024. Dart: An aigt detector using amr of rephrased text. arXiv preprint arXiv:2412.11517 .
[44] Piao, J., Lu, Z., Gao, C., Li, Y., 2025. Social bots meet large language model: Political bias and social learning inspired mitigation strategies, in: Proceedings of the ACM on Web Conference 2025, pp. 5202–5211.
[45] Qiao, B., Li, K., Zhou, W., Li, S., Lu, Q., Hu, S., 2025. Botsim: Llm-powered malicious social botnet simulation, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 14377–14385.
[46] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems 30.
[47] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 .
[48] Wang, X., Wang, K., Chen, K., Wang, Z., Zheng, K., 2024. Unsupervised twitter social bot detection using deep contrastive graph clustering. Knowledge-Based Systems 293, 111690.
[49] Wei, F., Nguyen, U.T., 2019. Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings, in: 2019 First IEEE International conference on trust, privacy and security in intelligent systems and applications (TPS-ISA), IEEE. pp. 101–109.
[50] Wu, Y., Fang, Y., Shang, S., Jin, J., Wei, L., Wang, H., 2021. A novel framework for detecting social bots with deep neural networks and active learning. Knowledge-Based Systems 211, 106525.
[51] Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Zhong, S., Yin, B., Hu, X., 2024. Harnessing the power of llms in practice: A survey on chatgpt and beyond. ACM Transactions on Knowledge Discovery from Data 18, 1–32.
[52] Yang, K.C., Torres-Lugo, C., Menczer, F., 2020a. Prevalence of low-credibility information on twitter during the covid-19 outbreak. arXiv preprint arXiv:2004.14484 .
[53] Yang, K.C., Varol, O., Hui, P.M., Menczer, F., 2020b. Scalable and generalizable social bot detection through data selection, in: Proceedings of the AAAI conference on artificial intelligence, pp. 1096–1103.
[54] Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B.Y., Dai, Y., 2014. Uncovering social network sybils in the wild. ACM Transactions on Knowledge Discovery from Data (TKDD) 8, 1–29.
[55] Zeng, F., Sun, Y., Li, Y., 2023. Mrlbot: Multi-dimensional representation learning for social media bot detection. Electronics 12, 2298.
[56] Zhou, M., Zhang, D., Wang, Y., Geng, Y.a., Dong, Y., Tang, J., 2024. Lgb: Language model and graph neural network-driven social bot detection. arXiv preprint arXiv:2406.08762 .
[57] Zhu, Y., He, Y., Haq, E.U., Tyson, G., Hui, P., 2025. Characterizing llm-driven social network: The chirper. ai case. arXiv preprint arXiv:2504.10286 .