AdaRelBot: Heterophily-Aware Graph Transformers for Twitter Bot Detection

Sunita Gaur
School of Computer Science & IT,
Devi Ahilya Vishwavidyalaya,
Indore, India
sunita9300@gmail.com

Dr. Yasmin Shaikh
International Institute of Professional Studies,
Devi Ahilya Vishwavidyalaya,
Indore, India
yasminshaikh01@gmail.com

Abstract
Bots use social media to disguise themselves among genuine users and form heterogeneous relationships contrary to the homophily principle, which forms the backbone of Graph Neural Networks. We present AdaRelBot – a graph transformer-based Twitter bot detection system, that takes into account quality of edges while reasoning. For each follow relation we compute cosine-similarity edge features based on profiles and recent tweets, providing the attention mechanism with explicit indication of reliability of the neighboring node. Dual head classifier combines classical multi-layer perceptron with a prototype-calibrated head by means of per-node learned gate allowing the model to rely on the similarity between class directions in cases of noisy embeddings. Training of the system is stabilized with Focal loss due to human-bot imbalance. In experiments on TwiBot-20 benchmark dataset, AdaRelBot reaches 86.37% accuracy and 88.10% F1-score. On Cresci-15 benchmark dataset, the same system shows 99.17% accuracy in standard train-test setting and 96.15% accuracy in leave-one-group-out setting, where text leakage across groups is not allowed. Prototype calibrated head appears to be a key component behind the system robustness, and analysis of the learned gate shows strong correlation with heterophily of the nodes, proving the gate to assign more MLP capacity to structurally heterogeneous neighborhoods.

Keywords: Twitter Bot Detection, Graph Transformers, Heterophily, Graph Neural Networks, Prototype Learning, Social Media

Introduction

Automated accounts, commonly referred to as social bots, constitute a significant proportion of user activity on social media platforms such as Twitter/X, Weibo, and Facebook. Although some bots have legal uses, malicious bots are frequently employed to spread disinformation, shape public opinion, boost trending topics unnaturally, masquerade as real users, and conduct organized harassment campaigns. The impact of even a small number of maliciously coordinated accounts can be highly disproportionate in terms of controlling conversations online. Therefore, bot detection has now become an important problem that helps keep the integrity of social networks on the Internet.

All social networks on the Internet can be modeled by means of a graph in which users represent nodes and follow relations between them – edges. This graph structure carries valuable structural information. A genuine human tends to form organic clusters with friends who share interests, while a bot farm has a distinctive wiring pattern. Graph Neural Networks have been widely used for social bot detection because they exploit the neighborhood information to generate more powerful representation [1][2]. They work by message passing, where each node uses information from its neighbor node to generate the structural representation. One basic assumption on which most GNNs work is the homophily assumption, wherein nodes having a connection share similar properties and are in the same class.

However, the homophily assumption is not true in real world social network. The malicious bot form edges with legitimate users to imitate normal social behavior and avoid being detected. Under standard message passing, the bot's features get averaged with genuine human features. . This leads to the situation when a bot is mainly connected to human nodes, hence causing the usual message passing procedures to collect erroneous information about neighborhoods. In turn, the node representations become similar to those of the legitimate users, thus the GNN-classifier can no longer tell them apart [3].

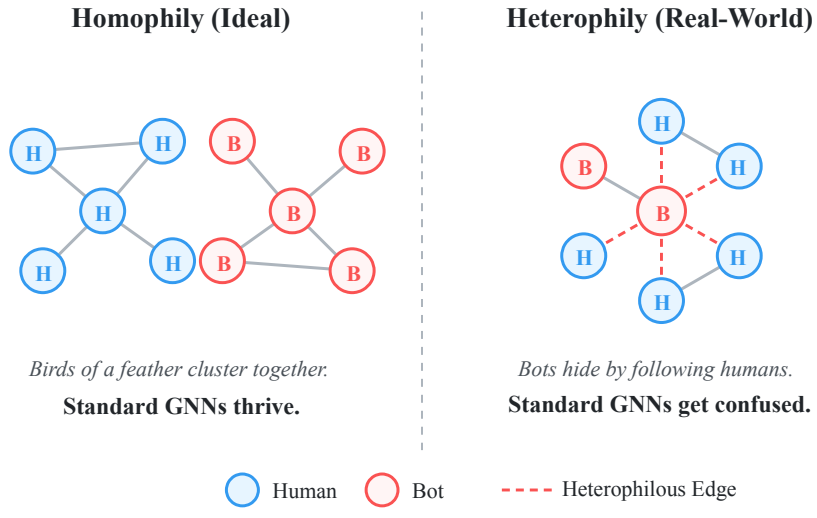


Figure 1 *Left: homophilous neighborhoods make message passing easy. Right: in the real Twitter graph, bots deliberately follow humans, producing heterophilous edges that pollute standard GNN representations.*

Figure 1 shows the basic distinction between homophilous and heterophilous graphs. In a homophilous graph, neighbors of a node tend to belong to the same category, thus making it easy for the message passing procedure to strengthen node representations. However, in reality, social networks are highly heterophilous, with malicious bots trying to make connections with legitimate users so as to blend in with the crowd. This makes GNN-based aggregation less effective and significantly increases the difficulty of accurate bot detection.

Proposed Approach

To address the challenges of hetrophily network, AdaRelBot, a heterophily-aware Graph Transformer framework for Twitter bot detection is proposed. The proposed approach takes into account the semantic compatibility of connected nodes for bot detection by using two complementary ideas. Firstly, before the model ever aggregates anything, the compatibility of two connected users is measured by computing the cosine similarity of their RoBERTa profile descriptions and recent tweets. These similarity scores become edge features and the attention mechanism uses them to down-weight heterophilous connections before they can corrupt the aggregation.

Secondly, rather than trusting a single classifier, two learned class prototypes are maintained, an ideal bot and an ideal human, in the shared embedding space. A tiny gating network decides, per node, how much to trust the standard MLP head versus the prototype head. When the MLP is reliable, the gate opens toward it. When the embedding is noisy, the prototype head takes over, because cosine similarity measures direction not magnitude, and direction is harder to corrupt.

AdaRelBot is trained with Focal Loss to counter the human-majority imbalance on TwiBot-20 [4]. This method focuses on the difficult to classify examples, while down-weighting the easy majority examples when learning from the data. The new architecture is trained in an end-to-end way using regular Graph Transformer modules [5].

Contributions

The major contributions of this paper can be stated as follows:

- An innovative Graph Transformer called AdaRelBot, which is a heterophily-aware model for bot detection on Twitter, is proposed. This model uses the concept of semantic edge features which can capture the compatibility between the linked nodes. Hence, the problem of the negative influence of heterophilious neighbors during message passing can be addressed.
- A prototype-calibrated dual head classifier with a learnable gating mechanism is presented for combining the parametric prediction with prototype-based prediction in a more robust manner.
- Experiments have been carried out on standard benchmark datasets such as TwiBot-20 and Cresci-15 datasets. A strict leave-one-group-out experiment has also been performed on Cresci-15 dataset.

Background and Related Work

Graph Neural Networks and Homophily

The GNN algorithm creates node embeddings by iteratively spreading node feature vectors through a pooling procedure involving information aggregation from nodes' neighbors [1][2].

On each level of propagation, a node gathers current embeddings of its neighbors, mixes them together, and then updates its own embedding using this mixed vector. This pooling procedure strengthens the node's true identity when neighbors have the same label. A human surrounded by humans becomes more recognizably human. Graph Attention Networks [6] replace uniform aggregation with a learned attention mechanism. The attention coefficient between two nodes is computed from their current embeddings.

While graph neural networks (GNNs) and graph attention networks (GATs) have shown great success in homophilous graphs, their effectiveness drastically drops in heterophilous social networks because they assume nodes are similar to their neighbors [3]. The embeddings of the nodes get polluted by the aggregation of the information from other nodes that are not in the same class as the node. As a result, the attention mechanism evaluates the importance score based on the polluted representation of the nodes, leading to inefficient message passing and poor classification results. In order to overcome this problem, AdaRelBot uses edge representations based on the raw description of users" profiles and tweets which are independent of evolving node embeddings.

There have been a number of works focusing on methods to increase the performance of GNNs on heterophilous graphs. The standard GNN models suffer from poor performance due to their inability to handle neighbors of other classes [3]. Geom-GCN tackled this problem by using the geometric relations between nodes in order to enhance message passing for heterophily [7]. In the domain of Twitter bots detection, RGT used relation-wise TransformerConv layers along with semantic attention in order to capture various edge types [8]. Unlike RGT, AdaRelBot uses Graph Transformers which take into account semantic compatibility as edge features.

Heterophily in Social Networks

Social networks tend to be heterophilous in nature since user connections are usually made for strategic rather than semantic purposes. In social media like Twitter, for instance, following does not imply the presence of shared interests or similar behavior of the same kind of account. Hacking bots are known to follow many other users in order to appear credible or escape detection. Therefore, local neighborhoods will consist of a combination of bots and human accounts, which violates the assumption of homophily made by all Graph Neural Networks. Given the heterophilous connections, making use of the connectivity of nodes alone in performing message passing could lead to misleading information within the nodes. Using semantic edge features helps to explicitly capture whether connections are meaningful or deceptive.

Bot Detection on Twitter

Early bot detection schemes for Twitter have largely been based on manually engineered features constructed from user profiles, network attributes, and behavior patterns. With the development of deep learning algorithms, recent progress has witnessed the success of language models like RoBERTa for the generation of semantic-rich representations of user-generated texts, thereby achieving breakthroughs in text-based bot detection [9]. Given the inherent nature of bots existing in intricate social networks, further researches have made use of graph learning paradigms. BotRGCN leveraged the power of Relational Graph Convolutional Networks to learn multiple relationship types in the Twitter graph, whereas RGT further advanced the field by introducing Graph Transformers with relation-aware attention mechanisms [8][10]. In contrast, AdaRelBot focuses on modeling semantic compatibility of connected users with edge features along with the adoption of prototype calibrated classification, achieving superior robustness in heterophilic graphs without multiple relation-specific graph convolution units.

Prototype Learning and Focal Loss

Prototypical learning has been found to be a useful way to make classification robust by representing each class using a number of learnable prototype vectors [11]. While parametric decision boundaries define the decision process of classical classifiers, prototype-based classifiers classify inputs according to their similarity to the prototype vectors that represent their respective classes. Such an approach is advantageous especially when node embeddings are noisily learned due to the heterophily in neighborhood aggregation since cosine similarity is insensitive to the magnitude differences in embeddings compared to standard linear classifiers.

Furthermore, Focal Loss mitigates the issue of data imbalance in bot detection problems by suppressing easy positive examples during training [4]. In this work, we leverage such merits of prototypical classification and Focal Loss and combine them together to design a robust classifier in heterophilous social networks.

Method

The structure of the proposed AdaRelBot framework is depicted in Figure 2. The framework comprises of four key phases, namely, (i) encoding heterogeneous features, (ii) edge feature construction, (iii) representation learning by utilizing two Graph Transformers, and (iv) node classification by means of a prototype-calibrated dual-head classifier. All of the mentioned components are developed with the aim to tackle heterophilous social networks while retaining semantic and structural features of Twitter users. Four modality encoders project heterogeneous user features into a shared 128-dimensional space. Cosine-similarity edge features are concatenated with a relation flag and fed into two TransformerConv layers. A learned gate blends MLP and prototype heads.

Figure 2 shows the full pipeline: feature encoding, edge-feature construction, two Graph Transformer layers, and a dual-head classifier.

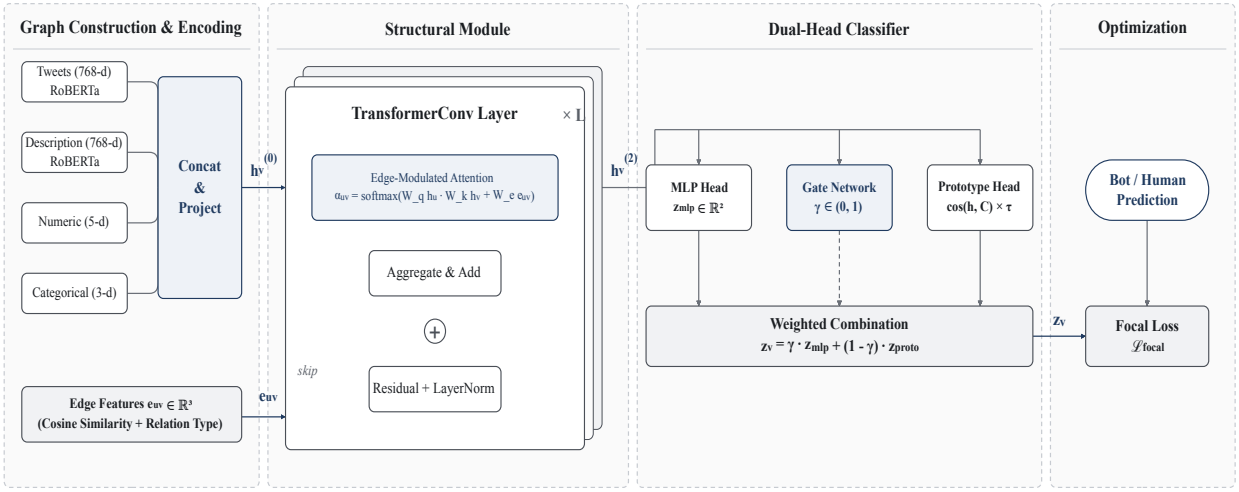


Figure 2 *AdaRelBot architecture. Four modality encoders project heterogeneous user features into a shared 128-dimensional space. Cosine-similarity edge features are concatenated with a relation flag and fed into two TransformerConv layers. A learned gate blends MLP and prototype heads.*

Input Data Representation

The framework will be used for detecting bots in attributed social graph representation of Twitter bot detection datasets. For the graph G=(V,E), nodes V are Twitter users, and edges E are social relations between Twitter users. Each Twitter user node has multiple heterogeneous attributes, including textual, numerical, and categorical data. Textual attributes include Twitter profile description and recent tweets, numerical attributes capture the activity of the user and profile statistics, and categorical attributes represent the categorical data about the user. These multiple types of attributes allow for capturing the required semantics, structure, and behavior of Twitter users for bot detection. The proposed framework is tested on two benchmark datasets - TwiBot-20 and Cresci-15 [12][13]

Feature Encoding

Our dataset consists of four input modalities: Description and Tweets (768-dimensional RoBERTa embeddings), Numeric (5-dimensional), and Categorical (3-dimensional). The properties of the four types of features differ greatly. The text embeddings are dense and semantically meaningful; numeric counts vary by orders of magnitude; categorical variables are sparse one-hot-like indicators. Every modality $m \in \{\text{des}, \text{tweet}, \text{num}, \text{cat}\}$ is projected to a 32-dimensional space using a linear layer followed by LeakyReLU activation. Numeric features go through BatchNorm normalization first to ensure consistent activations despite the wide distribution of follower and tweet counts. All four 32-dimensional vectors are concatenated into a single 128-dimensional vector and mapped through the final linear projection and LayerNorm and Dropout layers to get the node embedding $h_v^{(0)}$.

Edge Features: A Direct Compatibility Signal

In a vanilla TransformerConv, the attention coefficient $\alpha_{|uv|}$ is determined solely by the node features. When the embedding of a bot is corrupted with heterophilous propagation, the attention weights are meaningless. Bad embedding leads to bad attention, which allows even more dissimilar nodes to corrupt the representation further. AdaRelBot breaks this process by supplying an external signal on each edge: description cosine similarity, tweet cosine similarity, and a relation-type flag. The vector of edge features is:

$$e_{uv} = \left[\cos_{\text{des}(u,v)}, \cos_{\text{tweet}(u,v)}, r_{uv}\right] \in R^{3}.$$

Cosine similarity is the natural choice. RoBERTa embeddings are already L2-normalized, so the short bio of a bot does not negatively affect the embedding norm compared to the verbose humans'. Inner product and Euclidean distance led to noisier edge features and lower validation F1 score. Relation-type flag $r_{|uv|}$ is important because follow edges express the interest in other accounts whereas following edges reflect their popularity. It allows attention heads to develop relation-specific behavior without extra convolution weights.

Figure 3 illustrates the difference between aggregation with and without edge features.

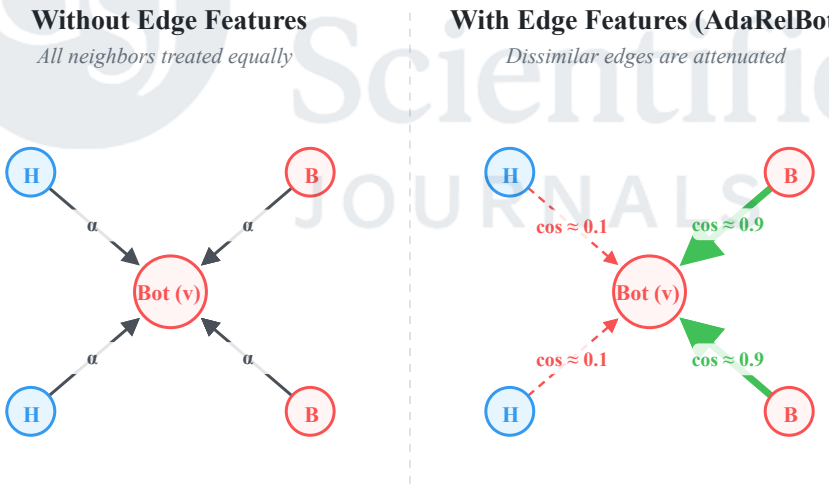


Figure 3 *Without edge features, a Graph Transformer treats every neighbor equally and cannot distinguish homophilous from heterophilous edges. AdaRelBot down-weights dissimilar edges before aggregation.*

In PyTorch Geometric's TransformerConv layer with edge_dim = 3 and beta = true, the edge features add additively to the equation:

$$h_{v'} = W_1 h_v + \sum_{u \in N(v)} \alpha_{uv} W_2 h_u + W_e e_{uv}.$$

The edge features shift each neighbor message depending on how compatible it is. High cosine similarity boosts the neighbor message; low cosine similarity suppresses it. Since edge features are pre-computed based on the content, they do not depend on the current state of the model. This is how we break the loop.

Implementation Efficiency

Naive computation of all pairwise cosine similarities for 229K edges is memory inefficient. We batch the computation in 50,000 edge chunks. We materialize once the entire edge attribute matrix of size $[\![E|,3]$ and pin it in GPU. On TwiBot-20 this takes additional 2.7 MB – insignificant amount compared to node feature tensors. Computation time is dominated by the TransformerConv layers and is not limited by the edge attributes.

Graph Convolution Layers

We use two TransformerConv layers. Two layers give the model enough receptive field to capture the community structure in the graph, but prevent the unstable training with deep graph networks. The output of each layer is added to the previous state of the embedding through the residual connection and normalized by LayerNorm. Residual connections retain the original signal and facilitate the gradient propagation. LayerNorm ensures consistent activations for nodes with varying degrees.

Dual-Head Classifier with Learned Gate

After the graph convolution, each node has a 128-dimensional embedding $h_v = h_v^2$. This vector is passed through a two-layer MLP to obtain the logits z_{mlp}. Simultaneously, we maintain two learned class prototypes $C \in R^{2\times 128}$ where the first row is the prototype of human and the second row is the prototype of bot. Both the node embedding and the prototypes are L2-normalized. Cosine-similarity logits are scaled by the learned temperature τ:

$$z_{\text{proto}} = \tau \cdot \left(\frac{h_v}{\|h_v\|_2}\right) \cdot \left(\frac{C}{\|C\|_2}\right)^{\top}$$

Cosine similarity only considers direction, therefore the prototype head is naturally resistant to embedding scale changes.

A gating network outputs the scalar $\gamma_v \in (0,1)$ for each node:

$$\gamma_v = \sigma\ \text{ReLU}\left(W_{g1} h_v\right) \cdot W_{g2}.$$

The resulting logits are computed as convex combination of heads:

$$z_v = \gamma_v z_{\text{mlp}} + \left(1 - \gamma_v\right) z_{\text{proto}}.$$

Large γ_v indicates that more importance should be given to the MLP head; small γ_v indicates that more importance should be given to the prototype head. The prototype head usually wins since its cosine-similarity decision is scale-invariant. However, the gate preserves capacity of the MLP head for special nodes.

Training Objective

We use Focal Loss [4] with focusing parameter of $\gamma = 2.0$ and class-balanced weights α_c. For an easy example where $p_{[i,y_i]} \approx 1$, the factor $\left(1 - p_{[i,y_i]}\right)^{\gamma}$ becomes almost zero and contributes little to the gradient. For a difficult example where the probability is predicted at $p \approx 0.4$, the factor remains large. Class-balancing weights α_c down-weight the human majority class and up-weight the bot minority class, so the model cannot easily rely on majority predictions. We apply label smoothing (0.1) to the labels to prevent the model from collapsing to the extreme 0/1 probabilities, which helps in the calibration plots of Section 3.2. The MLP head and the prototype head receive the auxiliary focal loss, which is weighted by 0.5 to keep both heads trainable and to prevent the gate from collapsing.

Experiments

Dataset and Metrics

Our experiments are run on two bot detection benchmarks. TwiBot-20 [12] consists of 229,580 nodes, 227,979 directed edges, and is divided into 8,278/2,365/1,183 train/validation/test nodes. Labeled nodes are sparsely distributed, the dataset is class imbalanced (approximately 70/30 human/bot), and bots attempt to emulate humans. Cresci-15 [13] is composed of 5,301 labeled accounts divided into five disjoint sets and is a smaller dataset but more densely labeled. Metrics used are accuracy, F1 score, and Matthews correlation coefficient (MCC). MCC is a number between -1 and 1, which can be considered high only if all four cells in confusion matrix are well-calibrated; it punishes models, which predict the class correctly, by predicting the majority.

Baselines

In the comparison, we consider algorithms that are described in the literature of TwiBot-20 and RGT [12][8]. Lee et al. [14] apply hand-crafted features from the social network graph. RoBERTa [9] uses a text-based transformer ignoring the graph. GAT [6] and SATAR [15] employ graph attention without an explicit mechanism of heterophily accounting. BotRGCN [10], BotMoE [16], and RGT [8] are the strongest structural baselines: they combine graph neural networks with user features and relations between nodes.

Implementation Details

Table 1 *Hyperparameters for AdaRelBot training.*

Setting	Value
Optimizer	AdamW
Learning rate	5×10^{-3}

Setting	Value
Weight decay	5×10^{-4}
Dropout	0.3
Embedding dimension	128
Attention heads	8
Focal γ	2.0
Label smoothing	0.1
Auxiliary loss weight	0.5
Epochs	50 per seed
Seeds	42, 123, 456, 2024, 9999
Selection criterion	Highest validation F1

Our implementation employs TransformerConv from PyTorch Geometric [5]. Attributes of edges are calculated once in batches of 50k edges and stored on the GPU. One seed is trained during 3-4 minutes on a modern GPU, 20 minutes on the CPU. The total number of parameters is about 520K.

The hyperparameters in table 1 balance capacity and stability. 128 dimensions of hidden space are sufficiently big for four modalities but not too big to overfit 8,000 labeled training nodes. Eight attention heads allow different heads to specialize on different semantic signals. Dropout of 0.3 works as a regularization. Validation F1 scores are used to select models since they take into account false positives and false negatives.

Main Results

Table 2 demonstrates test set results. AdaRelBot achieves better results than any baseline in all tests. On TwiBot-20, the improvement over RGT is 1.17 points in accuracy and 1.22 points in F1 score. With 1,183-node test set, this means that we achieve about 14 additional accounts classification. On Cresci-15 dataset, AdaRelBot achieves 99.17% accuracy and 99.35% F1 score in the standard split, outperforming RGT (96.89% / 97.58%). However, the standard split is overly optimistic since each of the five groups is internally text-homogeneous, and therefore random 80/10/10 split leaks information about group identity. Below we conduct the leave-one-group-out analysis. TwiBot-22 results are left for future work.

Table 2 *Benchmark comparison across Cresci-15, TwiBot-20, and TwiBot-22. Baseline numbers include mean ± std where available. AdaRelBot on TwiBot-22 is not yet implemented.*

	Cresci-15		TwiBot-20		TwiBot-22	
Model	Accuracy (%)	F1-score (%)	Accuracy (%)	F1-score (%)	Accuracy (%)	F1-score (%)
Lee et al. [14]	98.19 ± 0.07	98.52 ± 0.06	75.73 ± 0.19	79.37 ± 0.19		
GAT [6]	96.44 ± 0.19	97.22 ± 0.14	77.32 ± 0.73	80.51 ± 0.65	77.53 ± 0.08	53.47 ± 0.46
RoBERTa [9]	95.70 ± 0.15	94.06 ± 0.21	74.97 ± 0.23	72.80 ± 0.32	71.92 ± 0.64	16.15 ± 4.98
SATAR [15]	92.72 ± 0.59	93.84 ± 0.52	61.70 ± 1.75	71.95 ± 0.69		
BotRGCN [10]	96.37 ± 0.15	96.80 ± 0.27	83.21 ± 0.37	87.68 ± 0.32	76.75 ± 0.08	48.29 ± 0.66
BotMoE [16]	95.30 ± 0.16	96.39 ± 0.11	84.22 ± 0.34	86.89 ± 0.34	79.25 ± 0.00	**56.62 ± 0.40**
RGT [8]	96.89 ± 0.16	97.58 ± 0.12	85.20 ± 0.24	86.88 ± 0.22	**81.93 ± 0.19**	23.85 ± 0.20
AdaRelBot	**99.17 ± 0.54**	**99.35 ± 0.43**	**86.37 ± 0.27**	**88.10 ± 0.34**		

Training is consistent across seeds: AdaRelBot's TwiBot-20 has std 0.27% accuracy, 0.34% F1; Cresci-15 has std 0.54% accuracy, 0.43% F1.

Statistical Significance

In five runs with different seeds, AdaRelBot's TwiBot-20 accuracy ranges between 86.12%–86.76% and F1 between 87.56%–88.46%. Cresci-15 ranges between 98.31%–99.81% accuracy and 98.67%–99.85% F1. The margin to baseline is substantial compared to the seed variance.

Cresci-15 Robustness: Group-Leakage and Strict EvaluationThe close to perfect numbers on standard splits for Cresci-15 suggest an obvious question: Is the model learning how to detect bots, or just recognize group memberships? We have run several tests.

Text-only classification. Simple logistic regression on RoBERTa tweet embeddings alone gets 99.25% F1 on the test split. No graph. No GNN. The description embeddings alone can get 86.82%. The text almost solves the problem by itself.

Train-test text similarity. For each and every test example, there is a training example with a cosine similarity larger than 0.95 in description space. Larger than 0.99 in tweets. The group is text-homogeneous, and the stratified split contains the same members in the training and test sets. The classifier can achieve 99% by learning group style, not bot behavior.

Training curves. Both training and validation loss steadily decrease to almost zero, leaving a 0.004 gap between train and validation. There is no classical overfitting here – the model truly performs well on the test split because the test split contains text almost identical to what it was trained on.

Leave-one-group-out. We hold out a whole group of users for testing and train on the rest of the four groups, iterating over all five groups. Each leave-one-out experiment runs with five different seeds. See table 3 for the results.

Table 3 *Leave-one-group-out evaluation on Cresci-15. Each fold holds out an entire user group for testing. TWT bots (the most sophisticated group) are the hardest, with accuracy dropping to 86.27%.*

Held-Out Group	Type	Accuracy (%)	F1 (%)	ROC-AUC (%)
E13	Human	97.25 ± 0.20	–	–
TFP	Human	98.93 ± 1.36	–	–
FSF	Bot	96.94 ± 0.75	97.51 ± 0.59	99.87 ± 0.11
INT	Bot	99.54 ± 0.09	–	–
TWT	Bot	86.27 ± 3.63	92.59 ± 2.11	–
Weighted avg.		**96.15**	–	–

AdaRelBot performs well for most groups: classification of E13 humans (97.25%), FSF fake followers (96.94%), INT social bots (99.54%), and TFP humans (98.93%) achieves good results even in the absence of a target group in the training set. On the other hand, the most advanced group, TWT Twitterbot group, gets an accuracy of 86.27%. The average accuracy of 96.15% is just 3% lower than the accuracy in standard splits, which illustrates the significant influence of the text overlap in the random split setting.

In case when the group is represented by only one class, F1 and ROC-AUC measures are degenerate. They are reported only if there are representatives of both classes. TwiBot-20 dataset that uses a snapshot of Twitter users without any division into specific groups avoids this problem.

Detailed Metrics and Calibration

Table 4 presents the per-class precision, recall, ROC-AUC and PR-AUC values obtained using the five-seed ensemble. Confusion matrix gives 116 FP and 38 FN on TwiBot-20, just 4 total errors on Cresci-15.

Table 4 *Extended metrics for AdaRelBot on TwiBot-20 and Cresci-15. Ensemble metrics are computed from the average of five independently trained seeds.*

Metric	TwiBot-20	Cresci-15
Accuracy (%)	86.37 ± 0.27	99.17 ± 0.54
F1-score (%)	88.10 ± 0.34	99.35 ± 0.43
Precision (%)	83.48 ± 0.46	99.35 ± 0.60
Recall (%)	93.28 ± 1.27	99.35 ± 0.51
ROC-AUC (%)	93.09 ± 0.04	99.97 ± 0.03
PR-AUC (%)	93.24 ± 0.09	99.98 ± 0.02
MCC	0.7291 ± 0.0070	0.9822 ± 0.0116
ECE	0.1172	0.0660
Brier	0.1111	0.0101

The ECE of 0.117 for TwiBot-20 represents a model that operates on the decision boundary with hard nodes. The label smoothing of 0.1 helps ensure that middle probability bins are not left empty; the reliability diagram in decile-binned form reveals the real uncertainty as opposed to empty bins. For Cresci-15, ECE is 0.066 and Brier score 0.010.

Figure 4 contains the ROC curves. TwiBot-20 has an AUC of 0.93, while Cresci-15 AUC is essentially 1.0.

Figure 4 *ROC curves for AdaRelBot on TwiBot-20 (AUC = 0.93) and Cresci-15 (AUC > 0.99).*

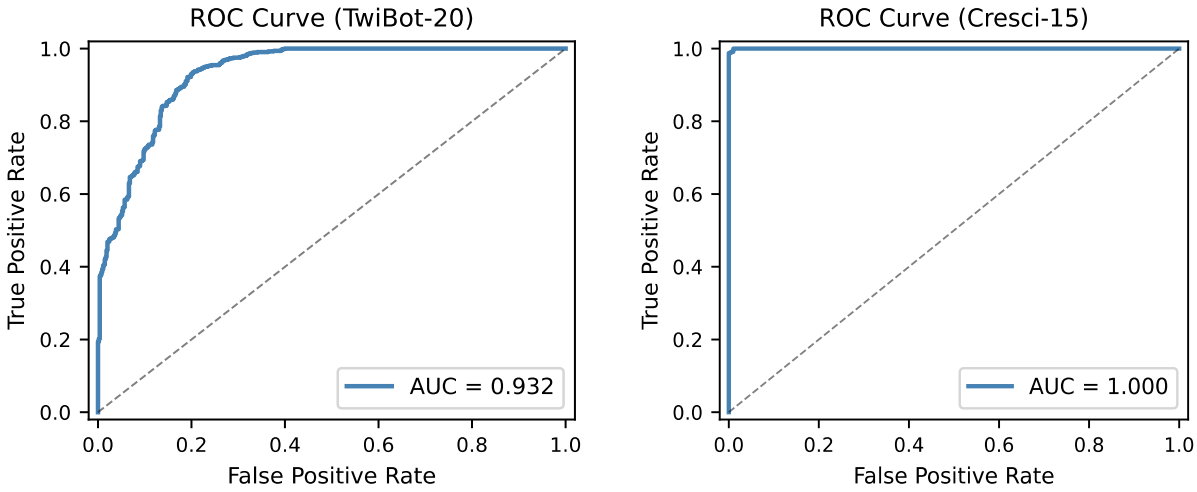


Figure 5 depicts the PR curves. In TwiBot-20, the precision declines from 1.0 to the baseline of 0.54 as recall increases to 1.0. An AP of 0.93 implies that most of the curve falls well above the baseline. In Cresci-15, precision is close to perfect through the entire range of recall values.

Figure 5 *Precision-Recall curves with baseline (dashed). TwiBot-20 AP = 0.93; Cresci-15 AP > 0.99.*

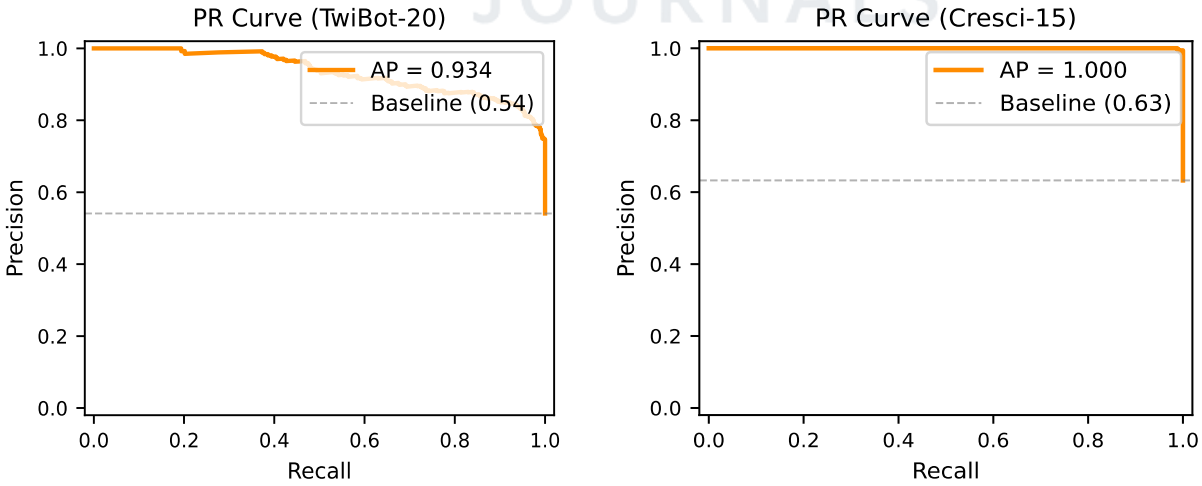
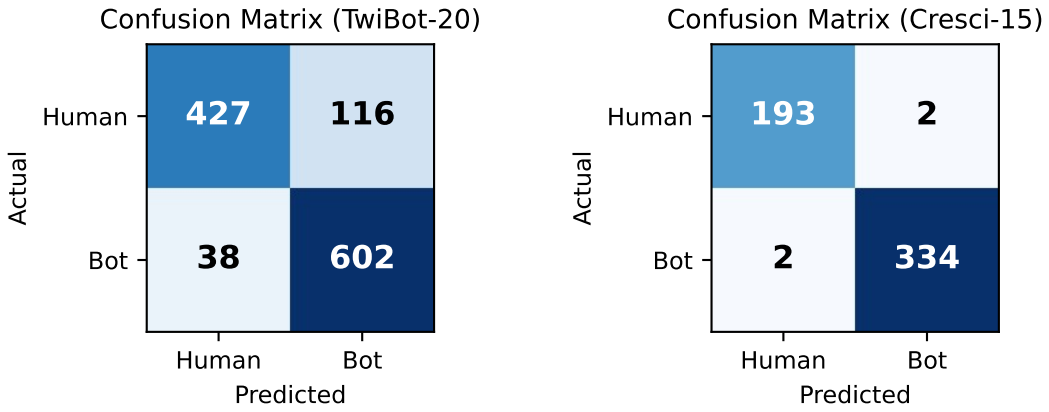


Figure 6 shows the reliability diagrams. On TwiBot-20, the model is under-confident on medium probability bins. It is understandable since with the conservative blending used by the gate, the

prototype head tends to draw the predictions towards less confident probabilities, especially when the MLP becomes overconfident on small evidence.

Figure 6 *Ensemble confusion matrices for TwiBot-20 and Cresci-15.*



Ablation Studies

Each AdaRelBot component is dropped individually and training (3 seeds, TwiBot-20) is repeated again. These ablations include no edge features (edge_dim=none), no prototype head ($\gamma_v = 1$), and class-weighted cross-entropy instead of Focal Loss.

Ablation study on TwiBot-20 (3 seeds). The prototype head provides the largest and most consistent gain; edge features and Focal Loss have smaller, seed-dependent effects.

Configuration	Accuracy (%)	F1 (%)	MCC
AdaRelBot (full)	86.45 ± 0.24	88.22 ± 0.09	0.7313 ± 0.0030
w/o edge features	86.67 ± 0.11	88.18 ± 0.03	0.7329 ± 0.0014
w/o prototype head	85.57 ± 1.64	87.85 ± 0.94	0.7230 ± 0.0212
w/o Focal Loss	86.53 ± 0.34	88.07 ± 0.35	0.7301 ± 0.0074

The prototype head is the dominant mechanism. Removing it drops F1 by 0.4 points and doubles the seed-to-seed variance. Without the prototype gate, the MLP alone becomes brittle; some seeds get it, some don't. Edge features and Focal Loss have smaller effects on TwiBot-20. The cosine-similarity prototypes appear to encode enough discriminative structure by themselves on this benchmark, leaving the edge features and focal reweighting to refine decisions at the margin. On datasets with more subtle text patterns, these components would likely matter more.

Analysis

AdaRelBot Explanation

Edge features disrupt the over-smoothing cycle. In standard GNNs there is a negative feedback loop: heterophilous aggregation contaminates the embeddings, deteriorating the attention weights, making room for more bad neighbors. A bot following many humans will see its representation pulled towards the human cluster by the end of layer 1. At layer 2, the attention compares the bot's contaminated embedding with neighbors and sees how similar they are, so averages them even more. Edge features circumvent this. They are extracted from the raw data, thus unaffected by the degradation of learned embeddings. The model can still distinguish between human and bot at the edge level despite several aggregation layers having blurred the node-level representations.

However, in what cases does the edge features become useful? We inspect the test nodes with high heterophily. For such nodes, the incoming neighborhood contains mostly elements of the other class. A standard Graph Transformer is bombarded with misleading information. Edge features reduce the attention on such edges. In TwiBot-20, the overall effect is small since node-only attention performs well on the easy nodes. Edge features are refining the hard minority cases.

Prototypes act as a regularizer. The cosine-similarity prototype head is bounded: the logits lie within the range of $[-\tau, +\tau]$ independently of embedding scaling. If the corrupted embedding pushes the MLP to predict extreme logit values, the prototype head can do nothing but assign the node somewhere in-between the two class prototypes. The gate in [eq]:gate combines the output of the prototype head with the MLP prediction, bringing the latter into play where it makes more sense. As seen in the ablation table, dropping the prototype head gives the most prominent reduction in the metrics and increasing the variance indicates the instability of the MLP alone across seeds.

Focal Loss deals with class imbalance. Absence of Focal Loss causes the optimizer to optimize for accuracy by relying on the human majority class. The $(1-p)^{\gamma}$ term guarantees that the node with probability of 0.99 has a negligible gradient, while the one with probability of 0.4 has a strong gradient. The model cannot rely on the majority class anymore.

Gate-Heterophily CorrelationIs the learned gate dependent on heterophily? We calculate the per-node heterophily as the ratio of incoming neighbors with different labels:

$$\mathrm{het}(v) = \frac{1}{|N(v)|} \sum_{u \in N(v)} 1[y_u \neq y_v].$$

Nodes of degree zero are pruned. There is a positive correlation between the gate and heterophily: Pearson $r = 0.095$ ($p < 0.005$), Spearman $\rho = 0.232$ ($p < 10^{-13}$). The Pearson correlation is relatively low because the gate responds to other things than only heterophily. The Spearman correlation is much higher because ordering of gate values, rather than their exact values, tracks structural ambiguity. Nodes with more opposite-label neighbors receive higher γ_v, giving more importance to the MLP head. The parametric transform is particularly valuable in heterophilous neighborhoods.

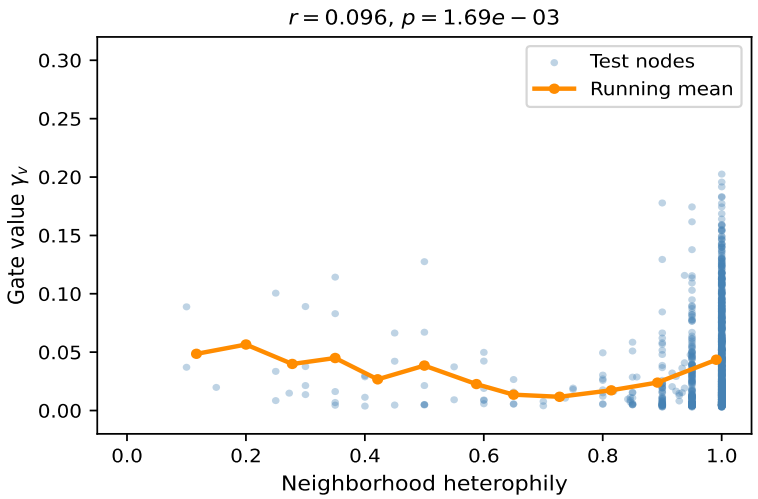


Figure 7 *Learned gate γ_v versus per-node heterophily on the TwiBot-20 test set. The orange curve is a running mean; higher heterophily is associated with larger γ_v, i.e., greater reliance on the MLP head.*

AdaRelBot vs. RGT

[8] learns per-relation TransformerConv layers for follow edges and for following edges separately, and fuses them via semantic attention. AdaRelBot fuses all the edges in a single graph, adding the relation type as another dimension of the edge feature vector. This means no duplicated convolutions, no additional attention over relation types, and an extra prototype head that RGT does not have. Both models tackle heterophily in their own ways. AdaRelBot does so with fewer learned components.

Limitations

TwiBot-20 and Cresci-15 are two data snapshots. Bots on different platforms or in different times may create different patterns in graphs. Edge features in AdaRelBot are static: once a bot switches its persona after having been followed, old edges will carry incorrect information. The gate-heterophily correlation exists but is relatively weak, because the gate responds to other sources of uncertainty that we cannot disentangle. Also, the group-homogeneous nature of text features in Cresci-15 dataset discussed earlier makes its results in the standard split less interpretable.

Broader Impact

Detection of bots reduces spread of disinformation and coordinated abuse. However, any classifier that discriminates between real and fake users may be abused. Governments or platforms may use such a classifier to repress criticism or incorrectly flag some automated accounts. AdaRelBot should be used for pre-screening only; human review and appeals should always be available.

ConclusionIn this paper we have shown that heterophily does not require complex per-relation architecture to be efficiently solved. Cosine similarity-based edge features, computed based on raw text from profiles and tweets, and then processed by a plain TransformerConv allow Graph

Transformers to distinguish between deceptive and real edges. A classifier with two heads allows to combine parametric and prototype-based predictions, with the prototype head providing an interpretable baseline and the MLP compensating when confident. On TwiBot-20, AdaRelBot gets 86.37% accuracy and 88.10% F1 score. On Cresci-15, leave-one-group-out evaluation shows 96.15% weighted accuracy, which is the right figure to consider. Standard split's 99% is biased due to text leakage.

The gate-heterophily correlation shows that the model uses the mechanism in the intended way: heterophilous neighborhoods receive higher weight of MLP. However, the correlation is relatively weak. Gate responds to multiple factors.

Several open problems. Dynamic edge features that are updated during training could help to account for bots that switch their persona after acquiring followers: our static features last for one training run but will become outdated in a live system. The gate γ_v itself is an interesting signal: those nodes with consistently low γ_v are the ones the model finds ambiguous and could be subject to human review. Fine-tuning the precomputed RoBERTa may reveal linguistic characteristics that general embeddings do not capture. Using temporal graphs could allow to track sudden increase of followers, one of typical patterns for bots. The main concept – making edge quality explicit and letting prototype calibrate the classification – is applicable beyond the area of bot detection, wherever heterophily hampers standard GNNs.

Reproducibility

All the code is self-contained. For TwiBot-20, running train.py reproduces the five-seed benchmark with the hyperparameters from table 4. For Cresci-15, preprocess_cresci15.py creates tensors and graph from raw CSV, train_cresci15.py runs the five-seed standard split, and train_cresci15_loocv.py reproduces the leave-one-group-out evaluation. ./train --dataset both runs both datasets. The ablation script disables one component at a time. Source code, preprocessing scripts and trained weights will be released upon publication.

References

[1] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in Proceedings of the 5th International Conference on Learning Representations (ICLR), 2017.

[2] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), 2017, pp. 1025–1035.

[3] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs," in Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS), 2020, pp. 7793–7804.

[4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980–2988.

[5] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification," arXiv preprint arXiv:2009.03509, 2020.

[6] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in Proceedings of the 6th International Conference on Learning Representations (ICLR), 2018.

[7] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-GCN: Geometric Graph Convolutional Networks," in Proceedings of the 8th International Conference on Learning Representations (ICLR), 2020.

[8] S. Feng, Z. Tan, R. Li, and M. Luo, "Heterogeneity Aware Twitter Bot Detection with Relational Graph Transformers," in Proceedings of the 36th AAAI Conference on Artificial Intelligence, 2022, pp. 6377–6385.

[9] Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv preprint arXiv:1907.11692, 2019.

[10] S. Feng, H. Wan, N. Wang, J. Li, and M. 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 (ASONAM), 2021, pp. 380–384. doi: 10.1109/ ASONAM52529.2021.9595513.

[11] J. Snell, K. Swersky, and R. Zemel, "Prototypical Networks for Few-Shot Learning," in Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), 2017, pp. 4080–4090.

[12] S. Feng, H. Wan, N. Wang, and M. Luo, "TwiBot-20: A Comprehensive Twitter Bot Detection Benchmark," in Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM), 2021, pp. 4481–4490. doi: 10.1145/3459637.3482450.

[13] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race," in Proceedings of the 24th International Conference on World Wide Web Companion (WWW Companion), 2015, pp. 963–972.

[14] K. Lee, B. D. Eoff, and J. Caverlee, "Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter," Proceedings of the 5th International AAAI Conference on Weblogs and Social Media, vol. 5, no. 1, pp. 185–192, 2011.

[15] S. Ali Alhosseini, B. Bin Makni, and J. Caverlee, "SATAR: A Self-Attention Network for Twitter Bot Detection via Adversarial Learning," arXiv preprint arXiv:1906.06987, 2019. [16] X. Zhang, J. Zhang, J. Zhu, Y. Chen, J. Chen, and D. Li, "BotMoE: A Mixture-of-Experts Model for Twitter Bot Detection," in Proceedings of the 2023 IEEE International Conference on Data Mining (ICDMW), 2023, pp. 1346–1351.