Twitter Bot Identification: An Anomaly Detection Approach

Lulwah Alkulaib* ‡, Lei Zhang*, Yanshen Sun*, and Chang-Tien Lu*
* Department of Computer Science, Virginia Tech, Falls Church, VA 22043 USA
‡ Department of Computer Science, Kuwait University, Kuwait
{lalkulaib, zhanglei, yansh93, ctlu}@vt.edu

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10020919

***Abstract*—The vast presence of bots on Twitter requires reliable and accurate bot detection methods that differentiate legitimate bots from malicious ones. Despite the success of those methods, they fail to address the following challenges: (1) the huge size of datasets required to train a model to detect bots, (2) the constant evolution in bot accounts to evade automatic detection leads to scarcity in ground truth real-world datasets, and (3) the complexity in learning representations of a heterogeneous attributed network like Twitter. In this paper, we propose a novel framework, ADNET, to detect anomalies in Twitter-attributed networks using the least amount of labeled data. Specifically, we address the limitations of previous methods by proposing a topology-based active learning framework that uses a deep autoencoder to train the model and is able to handle large graphs better than previous methods. Our experimental results demonstrate that the proposed approach outperforms state-of-the-art methods in detecting anomalous bot accounts and reduces the annotation cost in Twitter attributed networks.**

***Index Terms*—Twitter bot detection, Automated accounts, Social Media**

I. Introduction

Twitter bots are accounts on Twitter that are fully or partially controlled by a computer program using the Twitter API. The unique ability of being controlled by a software allows those accounts to generate a large amount of content in a short period of time. Twitter's terms of service allow legitimate bots to operate on the social media platform as long as they clearly identify themselves as a bot in the user profile section [1]. This perk has proven useful to news agencies' accounts that can produce a high volume of news on a daily basis that is shared with their followers. Malicious bots, which do not identify themselves as bots on their profiles, have utilized this ability to spread fake news [1], [2], spam [3], and offensive content [4] on Twitter. Detecting malicious bots has been a challenging task due to the nature of those bots. They try to evade being automatically detected by turning the software controlling these accounts on and off intermittently and constantly changing their behavior patterns to mimic human behavior [5]. The vast presence of bots on Twitter requires reliable and accurate bot detection methods that differentiate legitimate bots from malicious ones.

[1]https://twitter.com/en/tos

Existing bot detection methods on Twitter focus on hand crafted features that identify bots using profile-related and tweet-related features using traditional classification methods [6]–[10]. Due to the evolving nature of bots on Twitter [11], those classification methods do not perform consistently well especially when new features are needed to detect the improved bot behavior. Deep learning methods like recurrent neural networks [12], graph neural networks [13], [14], and graph convolutional networks [15], [16] were proposed to correctly identify bot accounts. Despite the success of those methods, they fail to address the following challenges: (1) the huge size of datasets required to train a model to detect bots, (2) the constant evolution in bot accounts to evade automatic detection leads to scarcity in ground truth real-world datasets, (3) the complexity in learning representations of a heterogeneous attributed network like Twitter.

To tackle those challenges, we treat the bot detection problem as an anomaly detection problem. We propose a novel framework called ADNET, to detect anomalies in Twitter attributed networks using the least amount of labeled data. Specifically, ADNET partitions the network topologically, chooses the most informative nodes in each partition, and uses that subset of the network in training the learner as an input to the autoencoder to detect anomalies. The autoencoder learns graph representations using a graph transformer as an encoder, then reconstructs the topological structure and nodal attributes with corresponding decoders. The errors resulting from the autoencoders are used to score and rank nodes. The resulting nodes are then added to the labeled subset of the network until the stopping criterion is met. Our main contributions can be summarized as follows:

1) **Development of a novel attributed network topology-based active learning framework:** We propose a novel active learning querying method specifically for attributed networks that partitions the network topologically by utilizing community structural properties to select the most informative nodes to be labeled. Our method reduces the annotation cost in attributed networks. To the best of our knowledge, this is the first framework that utilizes a network partitioning active learning method to train a graph transformer for anomalous user detection in Twitter attributed networks.
2) **Design of an active learning algorithm for anomaly**

detection in attributed networks: Given an attributed network, a small subset of labeled nodes, the proposed algorithms train a graph transformer-based autoencoder to detect anomalies from a subset of the network chosen based on the most informative nodes in each partition.

3) **Extensive experimental evaluation and performance analysis:** Our method was extensively evaluated on three real-world Twitter datasets and three benchmark attributed network datasets. Comparisons with baselines and state-of-the-art methods demonstrated its effectiveness and efficiency.
4) **Extension of three existing real-world attributed networks for the anomaly detection task using Twitter data:** The proposed method was used to identify anomalous bot accounts on Twitter. In order to construct our attributed networks for each dataset, using existing datasets is insufficient. Three Twitter datasets, including bot accounts, were collected. These datasets were then extended by collecting their following and followers for each available user account. Then, ADNET was used to detect anomalous accounts (bots) in these Twitter attributed networks.

II. Related Work

A. Bot Detection in Twitter

Different methods have been presented for automatically detecting bots in Twitter datasets. Supervised approaches that extract features from user accounts and posts rely heavily on annotated datasets to learn the difference between bots and legitimate users accounts [17]. However, since bots constantly change their behavior to evade automatic detection [5], those datasets need to be continuously updated with new types of bots in order to be detected. More recently, works that adopt graph techniques have been proposed for Twitter bot detection. SATAR [18] leverages user information and the Twitter network graph structure features to identify bots. AlHosseini et. al. [19] proposed a GCN-based method that leverages node features and neighborhood features to detect bots. And BotRGCN [20] uses relational GCNs to represent the Twitter network and user features to detect bots. Although these methods have achieved comparative results, they heavily rely on the datasets they are trained on and types of bots to which they are exposed. We solve this issue by treating bot detection as an anomaly detection problem which helps in a more accurate bot detection that is not data or bot type dependent.

B. Anomaly Detection on Attributed Networks

Earlier work in graph anomaly detection used feature engineering to detect anomalies [21]–[23], which performs well on labeled datasets only. Although some statistical-based methods exist, they are computationally and time expensive [24], [25]. In recent years deep learning techniques like reinforcement learning (RL), graph attention networks (GATs), generative adversarial networks (GANs), and graph convolutional networks (GCNs) have been used in anomaly detection [26]–[31]. A recent survey [32], shows that graph anomaly detection using deep learning techniques generates better results when detecting anomalies. As the amount of attributed network datasets has increased, anomaly detection on attributed networks has gained more popularity among researchers. Recent existing methods use deep learning techniques in anomaly detection on attributed networks. AMEN [33] detects anomalous neighborhoods in attributed networks by leveraging each node's ego network. Radar [24] characterizes the residuals of attribute information and its coherence with network information for anomaly detection. ANOMALOUS [34] performs attribute selection and anomaly detection based on cut matrix decomposition as well as residual analysis. DOMINANT [27] utilizes a GCN autoencoder to reconstruct the attribute, and the adjacency matrix then ranks anomalies based on the aggregated error. Despite the success of these methods in identifying anomalies in attributed networks, they require a lot of training samples to learn, and their performance drops drastically on larger graphs. To the best of our knowledge, our proposed framework is the first to identify anomalies using topology-based active learning on attributed networks which performs significantly better, especially on large attributed networks.

C. Active Learning

Active learning algorithms incrementally choose which data points to annotate by querying an oracle to learn the correct prediction for a given problem [35]. An AL model's accuracy increases as it is exposed to more data samples. AL sampling methods can use one query at a time, which can lead to overfitting [36], batch mode where diverse instances are queried give better results for deep learning models [37], or a combination of both where the querying algorithm is assigned a budget that is used to choose a batch of nodes for labeling. For anomaly detection, AL has been used in medical datasets to identify anomalies with arrhythmia issues [38]. Russo et al. [39] adopt active learning to identify anomalies in environmental datasets using machine learning algorithms. These works all show that AL was used to resolve the label sparsity issue in their respective fields. Previous works applied AL to attributed networks [40], [41] query single nodes at a time which causes their models to overfit. In this work, the combined querying method was utilized by setting a budget to select a batch of nodes to prevent overfitting and minimize the use of training resources. While some studies have explored AL in anomaly detection, there aren't any studies that provide insight on using active learning on attributed network anomaly detection.

III. Proposed Framework

The motivation for this framework is that large attributed networks are becoming more popular as a datasource, and using them to train models is costly. We propose a framework that defines partitions in the attributed network graph, chooses the most informative nodes to query, and then uses them in model training. This proposed method reduces the cost of training models by not requiring as much labeled data and considers structural graph representations when detecting anomalies using the autoencoder.

A. Problem Statement

Active Learning on Attributed Networks Definition An attributed network is typically represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}\}$ with its adjacency matrix $A \in \mathbb{R}^{N \times N}$ and node labels Y, where $\mathcal{V}$ denotes the set of nodes, $\mathcal{E}$ denotes the set of edges, $\mathcal{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N]^T \in \mathcal{R}^{N \times M}$ represents the nodal attributes matrix, and each vector $\mathbf{t}_i \in \mathcal{R}^M$ represents the attributes for the node $v_i \in \mathcal{V}$. A binary adjacency matrix A is used to represent the attributed network's topological structure, where $A_{i,j} = 1$ if there is a link between nodes v_i and v_j, else $A_{i,j} = 0$. The nodes in the attributed network are divided into two sets: an initial labeled set $\mathcal{S}_0$ with node labels $Y_{\mathcal{S}_0}$, and the rest of the unlabeled nodes would belong to the unlabeled set $\mathcal{S}_1$ with node labels $Y_{\mathcal{S}_1}$.

Ranking Anomalies in Attributed Networks using Active Learning Problem: Given nodes from the attributed network V, and an initial labeled set of nodes $\mathcal{S}_0$, the goal is to determine which unlabeled nodes should be selected to label when given a fixed budget b that produces a model M with the lowest loss:

$$\begin{aligned} &\underset{\mathrm{S}_1 \subset V}{\arg\min} \mathcal{L}_\theta \left(Y_{\mathcal{S}_0} \cup S_1\right) \\ s.t. \quad & n(S_1) \leq b \end{aligned} \tag{1}$$

where $Y_{\mathcal{S}_0}$ is the set of existing pre-labeled nodes, S_1 is the set of unlabeled nodes we want to label, $n(S_1)$ is the cardinality of the set S_1, b is the budget for the labeling, θ is the parameter of the anomaly detection model learned from the labeled set (the union of S_1 and $Y_{\mathcal{S}_0}$), and $\mathcal{L}$ is the loss of the anomaly detection problem conditioned on model parameters θ and labeled data.

B. Preliminaries

Our ADNET framework shown in Figure 1, leverages a deep autoencoder with a graph self-attention encoder to enhance AL anomaly detection results in large attributed networks.

Self-Attention Based Anomaly Detection for Graph Data Unlike previous works that use GCNs [27] to encode graph data into representations, this method adopted a graph transformer [42] as a self-attention encoder that learns graph representations for anomaly detection in our AL framework. Due to the heterogeneous nature of the attributed network data, it is essential to preserve both node embeddings and graph structures to be able to identify anomalous nodes.

The $TransformerSelfAttention()$ in Equation 2 is utilized to learn vector representations of all nodes for the given graph G, then outputs $H'^{(k)}$ which is used in the GCN model to improve the vector representations of nodes by adding the structure of the graph G and produces a graph embedding as the output of the encoder.

The structure reconstruction decoder uses the representations to predict if a link exists between pairs of nodes by training a link prediction layer of the learned representations and its transpose shown in Equation 4.

$$H'^{(k)} = TransformerSelfAttention(H^k) \tag{2}$$

$$H^{k+1} = GCN(A, H'^{(k)}) \tag{3}$$

$$\widehat{\mathbf{A}} = \operatorname{sigmoid}\left(\mathbf{H}^{\mathbf{k+1}} \mathbf{H}'^{(\mathbf{k+1})}\right) \tag{4}$$

Nodal connectivity patterns are used as an indicator of the node being anomalous by calculating the reconstruction error $E_s = \mathbf{A} - \widehat{\mathbf{A}}$ where $\widehat{\mathbf{A}}$ is the estimated adjacency matrix. A higher norm value for E_S means that the node has a higher probability of being an anomaly with regard to the network structure. The attribute reconstruction decoder takes the learned representations H^{k+1} from the encoder to approximate nodal attributes information by computing reconstruction errors. To predict the original nodal attributes, we leverage a GCN denoted as in equation 5,

$$\widehat{\mathbf{X}} = f_{Relu}\left(\mathbf{H}^{\mathbf{k+1}}, \mathbf{A} \mid \mathbf{W}^{(m)}\right) \tag{5}$$

where $W^{(m)}$ is a trainable layer weight matrix needed to learn the network representation. The GCN is then used in computing the reconstruction errors $\mathbf{E}_\mathbf{A} = \mathbf{T} - \widehat{\mathbf{T}}$ to detect anomalies with regards to nodal attributes.

The reconstruction errors calculated in the structure reconstruction decoder and the attribute reconstruction decoder are used to detect anomalies in the attributed network. In order to account for both nodal attributes and graph structure in our attributed network, the model jointly learns the reconstruction errors by minimizing the deep autoencoder objective function:

$$\mathcal{L} = (1 - \alpha)\mathbf{E}_S + \alpha \mathbf{E}_A \tag{6}$$

where the controlling parameter α balances the reconstruction impacts. Consecutively, the approximation of the attributed network is iteratively calculated until the objective function converges and the reconstruction errors are calculated to rank nodal abnormalities. The anomaly score for each node can be computed as

$$\operatorname{ascore}(\mathbf{v}_i) = (1 - \alpha)\mathbf{e}_\mathbf{S} + \alpha \mathbf{e}_\mathbf{A} \tag{7}$$

Our work utilized the deep autoencoder structure and was developed with a graph-based self-attention encoder which allowed us to incorporate graph structures and nodal attributes in our model properly. The graph-based self-attention used in the encoder learns complex graph representations and preserves both node embeddings and graph structures better than using only GCNs or only graph transformers to identify anomalous nodes.

C. ADNET Framework Description

ADNET is an active learning anomaly detection framework for attributed networks. The architecture of our proposed framework is illustrated in Figure 1. The objective of ADNET is to select the most informative nodes from large attributed networks to be labeled such that the anomaly detection performance is improved with minimal labeling cost. Therefore, topology-based AL was incorporated with the deep autoencoder, which maps the attributed network into a latent low-dimensional feature space, and then recovers the original data based on latent representations to detect anomalies based on the computed reconstruction errors. The workflow of the ADNET framework shown in Fig. 1 and Algorithm 1 can be described as follows:

1) Given a graph $\mathcal{G}$, the initial labeled set L, the unlabeled set $\mathcal{U}$, and the budget η as input. The topology-based attributed network sampling in algorithm2 is performed.
2) After the graph partitions, cluster centroids, and the most informative nodes are defined, we annotate the selected nodes based on their partition and merge them with the labeled set of nodes L. We also subtract the selected nodes from the unlabeled set U.
3) Finally, we train the autoencoder model with the labeled set of nodes L that are queried by algorithm 2. The graph-based self-attention encoder takes the attributed network as an input, and learns the graph representations by learning both node embeddings and graph structures. The output vector is then passed on to the decoders, where the structure reconstruction decoder reconstructs the graph topology, and the attribute reconstruction decoder reconstructs the nodal attributes using the learned graph embeddings. The reconstruction errors would be used to rank the nodes based on their anomaly scores, where the top nodes are considered anomalous.

Using this approach, the strengths of active learning and deep autoencoders are combined to minimize the amount of labeling needed in large attributed networks and maximize the anomaly detection task performance.

Algorithm 1 Active Learning Anomaly Detection for Attributed Networks

```
function ACTIVEANOMALY($L, U, G$)
Given : the initial labeled set L, the unlabeled set U, the graph $G$, the partition number $K$, budget $\eta$, trade-off parameter $\alpha$ :
    $S \leftarrow$TopologyBasedANSampling$(L, U, G, K, \eta, \alpha)$
    $L \leftarrow L \cup S$
    $U \leftarrow U - S$
    $lambda$ $\leftarrow$train(L,G) //train model $M$ with labeled samples acquired from topology sampling
```

Topology-based Attributed Network Sampling A novel query strategy algorithm is shown in algorithm 2, which conducts a topology-based attributed network sampling. Inspired by previous works that consider community detection to partition well-defined networks [37], [43], the quality of each partition was measured as a function of modularity and purity. Existing community detection methods are applied to fully labeled graphs; this challenge was addressed by assigning unlabeled nodes to communities according to the average similarity between the unlabeled node and all nodes in a partition. The sampling strategy partitions the graph G into K-partitions following the method described below. Then, for each partition, topology-based community detection was conducted on labeled nodes. As for unlabeled nodes, they are assigned to their corresponding communities based on their similarity to a community calculated using equation 12. Finally, the unlabeled nodes closest to each community centroid were selected as the most informative nodes to use to train our model M in algorithm 1.

Algorithm 2 Topology-based Attributed Network Sampling

```
    **Input:** A graph G, the initial labeled set L, the unlabeled set U, budget $\eta$, partition number $K$, trade-off parameter $\alpha$
    **Output:** A subset of unlabeled nodes $S_1$ of size $\eta : S_1 \subseteq V \backslash S_0$
  $\mathcal{H}_K \leftarrow GraphPartition(K)$
  Set $S_1 = \emptyset$.          ▷ to hold the subset of unlabeled nodes
  for $H_k \in \mathcal{H}_K$ do
    $\eta_k \leftarrow \eta // K$
    $H_k \leftarrow H_k \backslash \{S_0 \cup S_1\}$
    $E_k \leftarrow \{g(v_i)\}_{i \in H_k}$
    $L_k \leftarrow L \cap H_k$
    $U_k \leftarrow U \cap H_k$
    $G_k = Generate(G, M_k)$          ▷ generating partitions
    $\mathcal{C} \leftarrow Initialize(G_k)$
    $Z \leftarrow \alpha P + (1-\alpha) Q$
    $Z_{\text{prev}} \leftarrow -\infty$
    while $Z > Z_{prev}$ do
      $\mathcal{C} \leftarrow PartitionNodes(G_k, \mathcal{C}, \alpha)$          ▷ greedily identifies partitions by maximizing modularity and purity
      $G \leftarrow Aggregate(G_k, \mathcal{C})$  ▷ Network reconstruction and moving nodes to their partitions
      Compute $P$ according to Eq. (10) and $Q$ according to Eq. (8)
      $Z_{\text{prev}} \leftarrow Z$
      $Z \leftarrow \alpha P + (1-\alpha) Q$
    end while
    for $v_i \in U_k$ do
      $v_i \leftarrow$ AssignCommunity$(v_i, C)$
      $C \leftarrow \{C \cup v_i\}$          ▷ Update the community
    end for
    end
    $CT \leftarrow$ FindCentroids$(C)$  ▷ Compute the centroids of the communities
    $S \leftarrow$ ClosestNodes$(CT, U_k, \eta_k)$  ▷ Find $\eta_k$ unlabeled nodes closest to the centroids.
    if $S \neq \emptyset$ then
      $S_1 = S_1 \cup S$
    end if
    return S
```

1) Topology-based Attributed Network Partition Method: In order to correctly identify the partitions in our attributed networks, the quality of the partition was calculated as a function of modularity and purity.

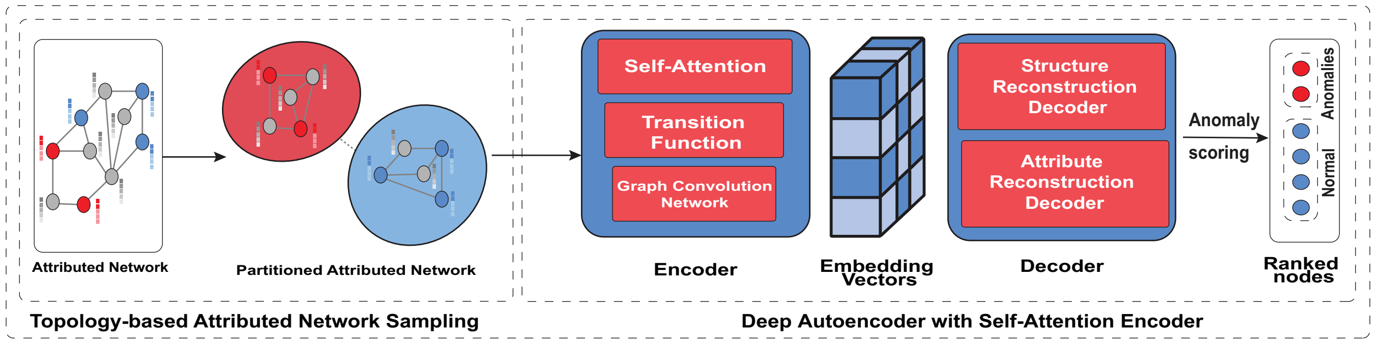

Attributed Network
Partitioned Attributed Network
Topology-based Attributed Network Sampling
Self-Attention
Transition Function
Graph Convolution Network
Encoder
Embedding Vectors
Structure Reconstruction Decoder
Attribute Reconstruction Decoder
Decoder
Anomaly scoring
Anomalies
Normal
Ranked nodes
Deep Autoencoder with Self-Attention Encoder

Figure 1: An overview of our proposed framework, ADNET, where a graph transformer-based autoencoder is trained to detect anomalies from a subset of the network chosen based on the most informative nodes in each partition

Modularity is a quality function that measures the degree to which connected nodes within a network can be decoupled into communities or partitions. The equation for modularity can be denoted as:

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta\left(c_v, c_w\right) \tag{8}$$

where m is the number of graph edges, $A_{v,w}$ is the adjacency matrix for $v, w \in V$, k_v, k_w is the degree of v, w, and $\delta\left(c_v, c_w\right)$ is the function determining if a node v, w belongs to the same partition with a value of 1 or 0 if the node doesn't belong to the partition.

Purity is a measure of the extent to which a partition contains a single class. The purity of a partition C is the average purity of all communities in the partition, computed as:

$$P = \frac{1}{|C|} \sum_{c \in C} P_c \tag{9}$$

where P is maximized when nodes in the same community share the same label and P_c is the frequency of the most common class in the network present in one partition. Purity for a given community c is denoted as:

$$P_c = \prod_{a \in A} \frac{\max\left(\sum_{v \in c} a(v)\right)}{|c|} \tag{10}$$

where A is the label set, $a \in A$ is a label, $a(v)$ is an indicator function that takes value 1 if $a \in A(v)$. The modularity and purity were combined linearly as in Eq. 11, by applying a trade-off parameter α, to tune the importance of each component and adapt the score according to different attributed networks.

$$Z = \alpha P + (1 - \alpha) Q \tag{11}$$

The basic idea is that nodes in the network try to traverse the community labels of all neighbors and select the community label that maximizes the modularity and purity. After maximizing the modularity and purity, each community is treated as a new node, and the process is repeated until the modularity no longer increases. This method is suitable for large-scale networks.

2) Most Informative Nodes Selection: Once the partitions are detected, unlabeled nodes are assigned to the communities based on their similarity to a partition (algorithm 2 line 20). Instead of measuring the similarities between all nodes v_j in a partition C_k, we pre-define c as a character vector of a given node and evaluate the cost of each node in the partition C_k. As c can be realized in many ways, we use the average of all the nodes in the partition C_k. As a consequence, the equation is formalized as follows:

$$\arg \min_{k=1,\ldots,K} f(g(v_i), c) \tag{12}$$

where $f(\cdot, \cdot)$ and $g(\cdot)$ are distance measure function, and aggregation function respectively. Jensen-Shannon divergence [44] for the f function. The g function is chosen according to our GCN model in the topology-based attributed network sampling ($g(v_i) = (A^2 F)$ where A is the normalized adjacency matrix and F is the feature matrix). The c is defined as follows:

$$c = \frac{1}{len(C_k)} \sum_{v_j \in C_k} g(v_j) \tag{13}$$

The centroids for each community are then computed using K-Means [45] by calculating the mean value for all nodes in the community and selecting η_k unlabeled nodes that are closest to the community centroids. These nodes are considered the most informative nodes due to their close proximity to the centroid.

IV. Experiments

We present our empirical evaluations using three real-world Twitter-attributed networks and three benchmark attributed network datasets to verify the effectiveness of our proposed framework ADNET. We evaluate each baseline with a labeling budget and report the AUC-ROC score for anomaly detection over the attributed network.

A. Datasets

We evaluate the proposed framework on three real-world datasets collected from Twitter [46], [47]. In addition, to compare our results with anomaly detection results, we evaluate our model on three widely used benchmark datasets for anomaly detection on attributed networks [24], [27]. In an attempt to present three new real-world attributed network datasets for anomaly detection, we use the Twitter API [2] to collect the tweets, user information, following, and followers networks for each user in the datasets found in [46], [47]. It is worth noting that since some of these datasets are old and Twitter removed some accounts, we experienced an average loss of 38% from the original Twitter dehydrated datasets. The statistical summary of all datasets is demonstrated in Table I. Moreover, since there is no ground truth for anomalies in the benchmark datasets (CiteSeer, Pubmed, and ACM), the anomalies are injected [27], [48]. Whereas in the Twitter datasets, we treat bots, the automated user accounts, as anomalies.

- **verified-2019 & botwiki-2019** We combine two datasets found in [47], one is a verified dataset of human accounts, and the other is a self-identified list of bots. We collect attributes related to their account, following and follower relations, and recent tweets for each user.
- **cresci-rtbust-2019** This dataset is composed of users who participated in retweeting Italian tweets over a two-week period in 2018 [46]. We collect attributes related to their account, following and follower relations, and recent tweets for each user.
- **gilani-17** This dataset is manually labeled 'bot' or 'human' based on hand crafted rules [49]. We collect attributes related to their account, following and follower relations, and recent tweets for each user.
- **CiteSeer** [48] is a public citation network. Each node is a published paper, while each edge denotes a citation relation between two papers. The textual contents of each paper are treated as its node features.
- **Pubmed** [48] is a public citation network. Each node is a published paper, while each edge denotes a citation relation between two papers. The textual contents of each paper are treated as its node features.
- **ACM** [50] is a citation network of papers published in nine areas before 2016. The dataset was turned into an undirected graph due to the sparsity of the original network.

B. Baseline Methods

We compare our proposed framework ADNET with the following baselines:

- **Botometer** [17]: A bot detection API service that uses a thousand user features in its analysis.
- **Alhosseini** [19]: A GCN approach to learn user representations for bot detection.
- **SATAR** [18]: A self-supervised representation learning framework that leverages user-related features for bot detection.
- **BotRGCN** [20]: A relational GCN approach for representation learning and bot detection.
- **DOMINANT** [27]: State-of-the-art deep model that explicitly models the topological structure and nodal attributes for node embedding learning using GCNs.
- **ANOMALOUS** [34]: A joint framework to conduct attribute selection and anomaly detection as a whole based on CUR decomposition and residual analysis.
- **Radar** [24]: An unsupervised learning framework used to characterize the residuals of attribute information and its coherence with the network information for anomaly detection in attributed networks.
- **Graph Transformer**: A variant of DOMINANT [27] that we create. It is a deep autoencoder model that captures the topological structure and nodal attributes for node embedding learning using a graph-based self-attention encoder to be used in attributed networks anomaly detection tasks.

C. Evaluation Metrics

We choose **AUC-ROC, Precision@N, and Recall@N** as our evaluation metrics since they are widely used in anomaly detection research methods [24], [27], [34].

- **Precision@N:** Precision at n is the proportion of anomalies in the top-n nodes in the ranked list.
- **Recall@N:**Recall at n is the proportion of true anomalies found in the total number of ground truth anomalies.
- **AUC-ROC:** The AUC-ROC curve is a classification performance measure at multiple thresholds. The probability curve, ROC, and AUC represent the capability of ranking an abnormal node higher than a normal node. This means that as the AUC value gets closer to 1, the model is better at ranking anomalies.

D. Parameter Setting

In the experiments on our different datasets, we used Adam [51] as an optimizer to minimize the loss function. We trained the proposed model with 300 epochs with a learning rate of 0.005. For the graph transformer encoder, we set the dropout to 0.1, the number of self-attention layers to 2, the number of GCN layers to 2, and the number of heads to 1. For our topology-based active learning sampling method, we choose a budget of 40 nodes from the unlabeled data for the citation datasets and a budget of 210 nodes for the Twitter datasets.

E. Experimental Results

In the experiments, we evaluate the performance of our proposed model in detecting anomalies by comparing it with the baseline methods. The precision and recall results for the benchmark datasets are presented in Table II for a budget of 210, and 40 nodes for the Twitter datasets and the citation datasets, respectively. Fig. 2 compares the AUC-ROC results of ADNET with the baselines. We present a sample of the results on two datasets due to page limitations; it is worth noting that all of our results exhibit similar trends. According to the results in tables II, Fig.2, and Fig. 3, we have the following observations:

- ADNET is able to detect bots (Twitter anomalies) better than the baseline methods. It significantly outperforms the

[2]https://developer.twitter.com/en/docs/twitter-api

Table I: Attributed networks datasets details

	verified-2019 & botwiki-2019	cresci-rtbust-2019	gilani-17	CiteSeer	ACM	PubMed
# Nodes	53,321	824,902	4,239	3,327	16,484	19,717
# Edges	671,907	824,272	16,956	4,732	71,980	44,338
# Attributes	17,509	42,051	400	3,703	8,337	500
# Anomalies	704	891	1,090	150	600	600

bot detection baselines. In addition, DOMINANT, Radar, ANOMALOUS, and the Graph Transformer models did not achieve satisfactory results on Twitter data.

- When using 210 and 40 nodes as a budget to label the datasets, we find that on all attributed network datasets, our proposed framework, ADNET, achieves the best anomaly detection performance in terms of Precision@N, Recall@N, and AUC-ROC. In particular, compared to the best results from the baselines, ADNET obtains a significant improvement on AUC-ROC. The main reason is that ADNET successfully learns from the most informative nodes queried and captures the nodal attributes and the graph structure, which enables the framework to achieve better performance when detecting anomalies.
- ADNET exhibits superior performance over the baselines confirming that training node selection from graph partitions enhances the active learning performance.
- Botometer, Radar, and Anomalous perform poorly compared to deep models. These shallow models are not able to capture the nodal and structural complexities in large attributed networks.
- The performance of the baseline methods deteriorates as the size of the attributed network grows. It is evident in the Twitter datasets where ADNET outperforms the baselines with a significant increase in performance.
- When graph-based self-attention is used as an encoder in the auto-encoder model (similar to Graph Transformer and ADNET), it outperforms the baselines. The reason is that it preserves both node embeddings and graph structures better than GCN-based encoders like DOMINANT.
- When comparing the performance of different methods with respect to increasing labeling budgets, all methods' performance increases as the labeling budget increases. Though ADNET offers the most significant improvement over other baselines.

V. Conclusion

In this paper, we propose a novel framework, ADNET, which uses active learning for anomaly detection in Twitter-attributed networks. Specifically, we address the limitations of previous methods and propose a topology-based active learning framework that uses a deep autoencoder to train the model and is able to handle large graphs better than previous methods. Our experimental results demonstrate that the proposed approach outperforms state-of-the-art methods in detecting anomalous bot accounts and reduces the annotation cost in Twitter-attributed networks.

References

[1] A. Al-Rawi, J. Groshek, and L. Zhang, "What the fake? assessing the extent of networked political spamming and bots in the propagation of# fakenews on twitter," *Online Information Review*, 2018.

[2] M. O. Jones, "The gulf information war— propaganda, fake news, and fake trends: The weaponization of twitter bots in the gulf crisis," *International journal of communication*, vol. 13, p. 27, 2019.

[3] A. H. Wang, "Detecting spam bots in online social networking sites: a machine learning approach," in *IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 335–342, Springer, 2010.

[4] O. Analytica, "Malicious uses of social media bots will rise," *Emerald Expert Briefings*, no. oxan-db, 2021.

[5] S. Qi, L. AlKulaib, and D. A. Broniatowski, "Detecting and characterizing bot-like behavior on twitter," in *International conference on social computing, behavioral-cultural modeling and prediction and behavior representation in modeling and simulation*, pp. 228–232, Springer, 2018.

[6] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *Eleventh international AAAI conference on web and social media*, 2017.

[7] O. Varol, C. A. Davis, F. Menczer, and A. Flammini, "Feature engineering for social bot detection," in *Feature engineering for machine learning and data analytics*, pp. 311–334, CRC Press, 2018.

[8] K.-C. Yang, O. Varol, C. A. Davis, E. Ferrara, A. Flammini, and F. Menczer, "Arming the public with artificial intelligence to counter social bots," *Human Behavior and Emerging Technologies*, vol. 1, no. 1, pp. 48–61, 2019.

[9] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Dna-inspired online behavioral modeling and its application to spambot detection," *IEEE Intelligent Systems*, vol. 31, no. 5, pp. 58–64, 2016.

[10] Z. Miller, B. Dickinson, W. Deitrick, W. Hu, and A. H. Wang, "Twitter spammer detection using data stream clustering," *Information Sciences*, vol. 260, pp. 64–73, 2014.

[11] D. A. Broniatowski, A. M. Jamison, S. Qi, L. AlKulaib, T. Chen, A. Benton, S. C. Quinn, and M. Dredze, "Weaponized health communication: Twitter bots and russian trolls amplify the vaccine debate," *American journal of public health*, vol. 108, no. 10, pp. 1378–1384, 2018.

[12] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," *Information Sciences*, vol. 467, pp. 312–322, 2018.

[13] Y. Yang, R. Yang, Y. Li, K. Cui, Z. Yang, Y. Wang, J. Xu, and H. Xie, "Rosgas: Adaptive social bot detection with reinforced self-supervised gnn architecture search," *arXiv preprint arXiv:2206.06757*, 2022.

[14] Y. Li, Y. Ji, S. Li, S. He, Y. Cao, Y. Liu, H. Liu, X. Li, J. Shi, and Y. Yang, "Relevance-aware anomalous users detection in social network via graph neural network," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2021.

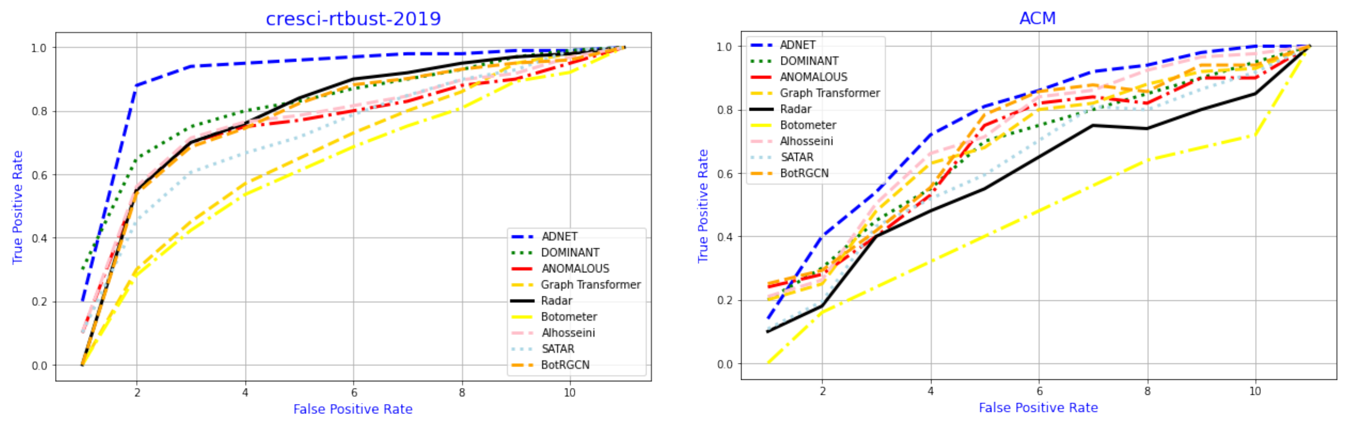
[15] S. Feng, H. Wan, N. Wang, and M. Luo, "Botrgcn: Twitter bot detection with relational graph convolutional networks," in *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 236–239, 2021.

[16] Q. Guo, H. Xie, Y. Li, W. Ma, and C. Zhang, "Social bots detection via fusing bert and graph convolutional networks," *Symmetry*, vol. 14, no. 1, p. 30, 2021.

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

[18] S. Feng, H. Wan, N. Wang, J. Li, and M. Luo, "Satar: A self-supervised approach to twitter account representation learning and its application in bot detection," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 3808–3817, 2021.

[19] S. Ali Alhosseini, R. Bin Tareaf, P. Najafi, and C. Meinel, "Detect me if you can: Spam bot detection using inductive representation learning,"



(a) (b)

Figure 2: ROC curves and AUC scores of all methods on different datasets

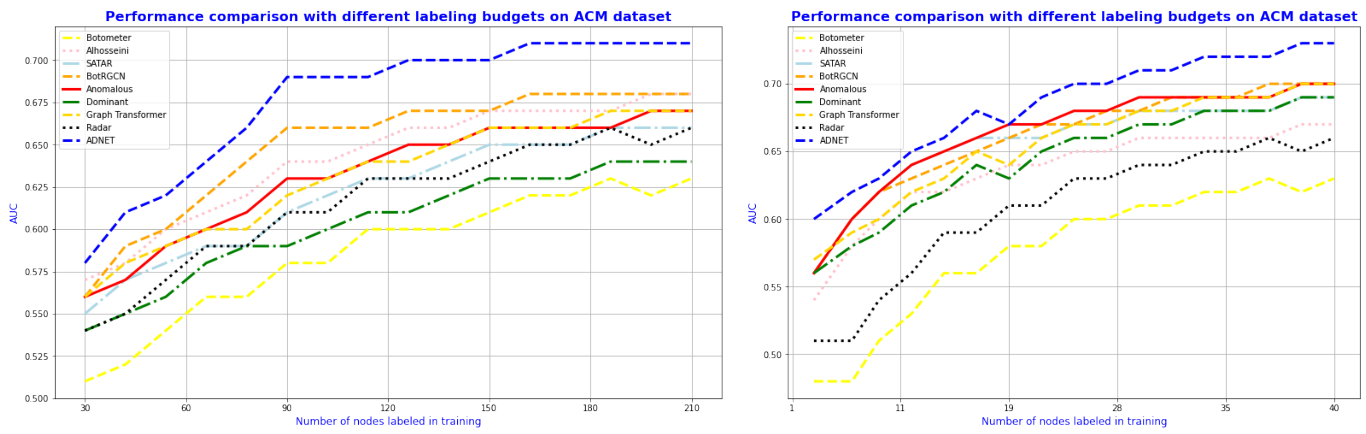


Figure 3: Performance comparison using different labeling budgets

in *Companion Proceedings of The 2019 World Wide Web Conference*, pp. 148–153, 2019.

[20] S. Feng, H. Wan, N. Wang, 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*, pp. 236–239, 2021.

[21] L. Akoglu, M. McGlohon, and C. Faloutsos, “oddball: Spotting anomalies in weighted graphs,” in *Advances in Knowledge Discovery and Data Mining*, pp. 410–421, Springer Berlin Heidelberg, 2010.

[22] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, “SpotLight,” 2018.

[23] N. Li, H. Sun, K. Chipman, J. George, and X. Yan, “A probabilistic approach to uncovering attributed graph anomalies,” in *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, Proceedings, pp. 82–90, Society for Industrial and Applied Mathematics, Apr. 2014.

[24] J. Li, H. Dani, X. Hu, and H. Liu, “Radar: Residual analysis for anomaly detection in attributed networks,” in *IJCAI*, pp. 2152–2158, 2017.

[25] S. Thudumu, P. Branch, J. Jin, and J. Singh, “A comprehensive survey of anomaly detection techniques for high dimensional big data,” 2020.

[26] K. Ding, J. Li, N. Agarwal, and H. Liu, “Inductive anomaly detection on attributed networks,” in *29th International Joint Conference on Artificial Intelligence, IJCAI 2020*, pp. 1288–1294, 2020.

[27] K. Ding, J. Li, R. Bhanushali, and H. Liu, “Deep anomaly detection on attributed networks,” in *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 594–602, Philadelphia, PA: Society for Industrial and Applied Mathematics, May 2019.

[28] K. Ding, J. Li, and H. Liu, “Interactive anomaly detection on attributed networks,” in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM ’19, (New York, NY, USA), pp. 357–365, Association for Computing Machinery, Jan. 2019.

[29] H. Fan, F. Zhang, and Z. Li, “Anomalydae: Dual autoencoder for anomaly detection on attributed networks,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5685–5689, May 2020.

[30] Y. Pei, T. Huang, W. van Ipenburg, and M. Pechenizkiy, “ResGCN: attention-based deep residual modeling for anomaly detection on attributed networks,” *Mach. Learn.*, Sept. 2021.

[31] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi, “A Semi-Supervised graph attentive network for financial fraud detection,” in *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 598–607, Nov. 2019.

[32] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, “A comprehensive survey on graph anomaly detection with deep learning,” June 2021.

[33] B. Perozzi and L. Akoglu, “Scalable anomaly ranking of attributed neighborhoods,” in *Proceedings of the 2016 SIAM International Conference on Data Mining (SDM)*, Proceedings, pp. 207–215, Society for Industrial and Applied Mathematics, June 2016.

[34] Z. Peng, M. Luo, J. Li, H. Liu, and Q. Zheng, “ANOMALOUS: A joint modeling approach for anomaly detection on attributed networks,” in *IJCAI*, pp. 3513–3519, 2018.

[35] C. C. Aggarwal, X. Kong, Q. Gu, J. Han, and S. Y. Philip, “Active learning: A survey,” in *Data Classification: Algorithms and Applications*, pp. 571–605, CRC Press, 2014.

Table II: Results of the evaluation of anomaly detection methods for Precision and Recall at the top n nodes using 210 nodes as the budget for Twitter datasets and 40 nodes as the budget for the citation networks.

(a) Benchmark Datasets (Twitter Data)

	verified-2019\ botwiki-2019				cresci-rtbust-2019				gilani-17			
N	50	100	200	300	50	100	200	300	50	100	200	300
Precision@N												
Botometer	0.148	0.129	0.324	0.343	0.176	0.231	0.353	0.390	0.188	0.248	0.377	0.418
Alhosseini	0.410	0.399	0.546	0.589	0.368	0.378	0.410	0.453	0.394	0.405	0.439	0.485
SATAR	0.399	0.495	0.536	0.600	0.347	0.411	0.485	0.496	0.371	0.440	0.519	0.530
BotRGCN	0.454	0.576	0.643	0.677	0.400	0.555	0.600	0.656	0.428	0.594	0.642	0.702
Radar	0.153	0.134	0.335	0.354	0.182	0.239	0.364	0.403	0.194	0.255	0.389	0.431
Anomalous	0.374	0.364	0.498	0.537	0.336	0.345	0.374	0.413	0.359	0.369	0.400	0.441
Dominant	0.363	0.45	0.488	0.546	0.316	0.374	0.441	0.451	0.338	0.400	0.471	0.482
Graph Transformer	0.392	0.497	0.555	0.584	0.345	0.479	0.518	0.566	0.369	0.512	0.554	0.605
ADNET	**0.535**	**0.678**	**0.755**	**0.817**	**0.469**	**0.651**	**0.721**	**0.867**	**0.501**	**0.696**	**0.771**	**0.927**
Recall@N												
Botometer	0.005	0.011	0.016	0.020	0.005	0.012	0.017	0.021	0.005	0.013	0.019	0.022
Alhosseini	0.055	0.115	0.205	0.256	0.060	0.122	0.219	0.273	0.064	0.131	0.234	0.292
SATAR	0.061	0.115	0.205	0.248	0.064	0.122	0.216	0.265	0.069	0.130	0.297	0.283
BotRGCN	0.062	0.119	0.219	0.302	0.067	0.127	0.234	0.323	0.072	0.135	0.251	0.345
Radar	0.005	0.012	0.015	0.018	0.005	0.011	0.016	0.019	0.005	0.011	0.017	0.02
Anomalous	0.047	0.098	0.174	0.217	0.051	0.104	0.186	0.232	0.054	0.111	0.199	0.248
Dominant	0.051	0.096	0.171	0.207	0.054	0.102	0.183	0.221	0.057	0.109	0.195	0.236
Graph Transformer	0.052	0.099	0.181	0.25	0.056	0.105	0.194	0.267	0.059	0.112	0.207	0.285
ADNET	**0.072**	**0.135**	**0.248**	**0.34**	**0.076**	**0.243**	**0.463**	**0.662**	**0.081**	**0.26**	**0.495**	**0.708**

(b) Benchmark Datasets (Citation Netwroks)

	CiteSeer				ACM				Pubmed			
N	50	100	200	300	50	100	200	300	50	100	200	300
Precision@N												
Radar	0.174	0.171	0.209	0.285	0.226	0.257	0.362	0.400	0.035	0.043	0.057	0.057
Anomalous	0.396	0.524	0.638	0.627	0.480	0.605	0.667	0.705	0.412	0.498	0.555	0.535
Dominant	0.397	0.490	0.618	0.609	0.486	0.619	0.676	0.752	0.392	0.487	0.544	0.572
Graph Transformer	0.447	0.561	0.675	0.722	0.565	0.652	0.695	0.783	0.474	0.515	0.563	0.601
ADNET	**0.616**	**0.774**	**0.832**	**0.920**	**0.777**	**0.897**	**0.957**	**0.962**	**0.649**	**0.705**	**0.771**	**0.810**
Recall@N												
Radar	0.048	0.069	0.114	0.174	0.045	0.080	0.114	0.151	0.005	0.010	0.014	0.017
Anomalous	0.105	0.212	0.349	0.396	0.078	0.149	0.269	0.321	0.045	0.093	0.165	0.206
Dominant	0.102	0.206	0.326	0.397	0.083	0.151	0.275	0.324	0.048	0.091	0.162	0.196
Graph Transformer	0.121	0.225	0.374	0.447	0.080	0.154	0.290	0.378	0.050	0.094	0.172	0.237
ADNET	**0.167**	**0.312**	**0.416**	**0.517**	**0.110**	**0.313**	**0.499**	**0.519**	**0.068**	**0.378**	**0.536**	**0.643**

[36] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang, "A survey of deep active learning," Aug. 2020.

[37] Z. Allen-Zhu, Y. Li, A. Singh, and Y. Wang, "Near-optimal discrete optimization for experimental design: a regret minimization approach," *Math. Program.*, vol. 186, pp. 439–478, Mar. 2021.

[38] T. Pimentel, M. Monteiro, J. Viana, A. Veloso, and N. Ziviani, "A generalized active learning approach for unsupervised anomaly detection," *Stat*, vol. 1050, p. 23, 2018.

[39] S. Russo, M. Lürig, W. Hao, B. Matthews, and K. Villez, "Active learning for anomaly detection in environmental data," *Environmental Modelling & Software*, vol. 134, p. 104869, Dec. 2020.

[40] F. Regol, S. Pal, Y. Zhang, and M. Coates, "Active learning on attributed graphs via graph cognizant logistic regression and preemptive query generation," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. Iii and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 8041–8050, PMLR, 2020.

[41] Y. Li, J. Yin, and L. Chen, "SEAL: Semisupervised adversarial active learning on attributed graphs," *IEEE Trans Neural Netw Learn Syst*, vol. 32, pp. 3136–3147, July 2021.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[43] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Sept. 2009.

[44] B. Fuglede and F. Topsoe, "Jensen-shannon divergence and hilbert space embedding," in *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, pp. 31–, 2004.

[45] Z. Zhang, J. Zhang, and H. Xue, "Improved K-Means clustering algorithm," in *2008 Congress on Image and Signal Processing*, vol. 5, pp. 169–172, ieeexplore.ieee.org, May 2008.

[46] M. Mazza, S. Cresci, M. Avvenuti, W. Quattrociocchi, and M. Tesconi, "RTbust: Exploiting temporal patterns for botnet detection on twitter," in *Proceedings of the 10th ACM Conference on Web Science*, pp. 183–192, New York, NY, USA: Association for Computing Machinery, June 2019.

[47] K.-C. Yang, O. Varol, P.-M. Hui, and F. Menczer, "Scalable and generalizable social bot detection through data selection," *AAAI*, vol. 34, pp. 1096–1103, Apr. 2020.

[48] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," 2008.

[49] Z. Gilani, R. Farahbakhsh, G. Tyson, L. Wang, and J. Crowcroft, "Of bots and humans (on twitter)," in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pp. 349–354, 2017.

[50] X. Huang, Q. Song, J. Li, and X. Hu, "Exploring expert cognition for attributed network embedding," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 270–278, 2018.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014.