Twitter Bot Detection Using Graph Neural Networks

Büşra Zenbilci[#1], Şükran Şafak Barutçu[*2]

Computer Engineering Department, Akdeniz University
Antalya, Turkey

[1]20170808054@akdeniz.edu.tr

[2]20190808003@akdeniz.edu.tr

***Abstract*— The spread of misinformation on social media platforms like Twitter has made detecting bot accounts more important than ever. In this project, we use a Graph Neural Network (GNN) to identify suspicious accounts with the help of the Twibot-20 dataset. By analyzing user profiles and tweet content, our model distinguishes between bots and real users. This paper explains our methodology, experimental results, and evaluation metrics like accuracy, precision, recall, and F1-score. Our findings show that using a GNN significantly improves the detection of Twitter bots.**

***Keywords*— Twitter Bot Detection, Graph Neural Networks, Twibot-20 Dataset, Misinformation, Model Training, Accuracy, Precision, F1-score, Recall**

1. INTRODUCTION

With the rapid growth of social media, challenges like misinformation and bot accounts have become more prevalent. Bots can distort public opinion and spread harmful content. Detecting and mitigating these bots is essential to maintaining the integrity of social media platforms. This project explores how Graph Neural Networks (GNNs) can be applied to detect bots on Twitter, using the Twibot-20 dataset to build and evaluate a robust classification model. By looking at user profiles and tweet content within a graph structure, we aim to improve the accuracy and reliability of bot detection.

2. RELATED WORK

Many techniques have been used for bot detection in the past, including machine learning models like Random Forests, Support Vector Machines (SVMs), and neural networks. Recently, graph-based models that capture the relationships between users have shown promise. GNNs, in particular, are good for tasks involving relational data, making them suitable for bot detection. Ferrara et al. (2016) provided an overview of how social bots have evolved and the techniques used to detect them. Varol et al. (2017) developed a framework for bot detection using various features from user data and behavior patterns. More recently, Alhosseini and Lemnaru (2020) explored the use of deep learning models, including GNNs, for detecting bot accounts on social media.

3. METHOD

3.1 DATASET DESCRIPTION

The Twibot-20 dataset, used in our research for detecting Twitter bots, consists of several important files, each playing a unique role in building and utilizing our graph-based model. Here's a detailed breakdown of each component in the dataset:

- split.csv: This file organizes user accounts into different subsets specifically for training, validation, and testing phases. This separation is crucial for systematically training and evaluating our model.

- edge.csv: Essential for constructing the graph, this file outlines the edges between nodes, where each node represents an individual Twitter user. These edges indicate relationships like followers and followings, which are vital for reflecting the social network structure within our graph.

- node.json: This JSON file contains comprehensive metadata for each node (user) in the graph, including user profile data such as activities and tweet content. These details serve as features for the graph nodes, enhancing the model's ability to detect patterns that indicate bot-like behavior.

- user_info.pt: Stored in a format compatible with PyTorch, this file contains tensors of processed user information. These tensors are ready-to-use data structures that facilitate efficient data manipulation and model training within the PyTorch framework.

- test.ipynb: This Jupyter notebook provides code examples for data handling, including procedures for loading, preprocessing, and preliminary data analysis. It serves as a practical guide for researchers to replicate and extend the dataset's application in bot detection tasks.

- label.csv: Crucial for supervised learning, this file catalogs the labels for each user, identifying them as either 'bot' or 'human'. These labels are the targets used in training our Graph Neural Network (GNN),

allowing it to learn from known cases and predict the classification of unlabeled users.

3.2 GRAPH CONSTRUCTION

Using the data from the files mentioned above, we construct a graph where nodes represent Twitter users and edges represent follower-following relationships derived from edge.csv. Features extracted from node.json and user_info.pt are associated with each node, providing a rich dataset for our GNN to learn from. This setup enables our model to effectively identify and classify Twitter bots.

3.3 MODEL

We used a two-layer Graph Convolutional Network (GCN). The first layer captures initial feature interactions, and the second layer aggregates these interactions to produce final node embeddings used for classification.

- First Layer: Applies graph convolution to capture local feature interactions.
- Activation: Uses ReLU (Rectified Linear Unit) function to introduce non-linearity.
- Second Layer: Refines the node embeddings and produces the output for classification.

3.4 TRAINING AND EVALUATION

We trained the model using cross-entropy loss and evaluated its performance using accuracy, precision, recall, and F1-score. These metrics were computed for both the training and test sets to comprehensively assess model performance. The model was trained for 200 epochs using the Adam optimizer with a learning rate of 0.01.

Evaluation Metrics:

- Accuracy: Ratio of correctly predicted instances to the total instances.
- Precision: Ratio of correctly predicted positive observations to the total predicted positives.
- Recall: Ratio of correctly predicted positive observations to all observations in the actual class.
- F1-Score: Weighted average of Precision and Recall.

4. EXPERIMENTAL RESULTS

In our study, we tested both a Graph Neural Network (GNN) and a Random Forest (RF) model on the Twibot-20 dataset to compare their performance in detecting Twitter bots. Below are the evaluation metrics for the RF model:

Random Forest Training Set Evaluation:

	precision	recall	f1-score	support
0	0.960445	0.806036	0.876492	3645.000000
1	0.864533	0.973883	0.915956	4633.000000
accuracy	0.899976	0.899976	0.899976	0.899976
macro avg	0.912489	0.889959	0.896224	8278.000000
weighted avg	0.906765	0.899976	0.898579	8278.000000

Random Forest Test Set Evaluation:

	precision	recall	f1-score	support
0	0.911719	0.733040	0.812674	1592.000000
1	0.812610	0.942229	0.872633	1956.000000
accuracy	0.848365	0.848365	0.848365	0.848365
macro avg	0.862164	0.837635	0.842653	3548.000000
weighted avg	0.857081	0.848365	0.845729	3548.000000

And this is the evaluation metrics for the GNN model:

Training Set Evaluation:

	precision	recall	f1-score	support
0	0.632252	0.585077	0.607751	3632.00000
1	0.693512	0.733965	0.713165	4646.00000
accuracy	0.668640	0.668640	0.668640	0.66864
macro avg	0.662882	0.659521	0.660458	8278.00000
weighted avg	0.666634	0.668640	0.666914	8278.00000

Test Set Evaluation:

	precision	recall	f1-score	support
0	0.657084	0.589319	0.621359	543.00000
1	0.679598	0.739062	0.708084	640.00000
accuracy	0.670330	0.670330	0.670330	0.67033
macro avg	0.668341	0.664191	0.664722	1183.00000
weighted avg	0.669264	0.670330	0.668277	1183.00000

These results provide a benchmark for the RF model's performance and allow us to compare it with our GNN-based approach.

5. COMPARISON OF RF AND GNN MODELS

There are several reasons why the Random Forest model might outperform the Graph Neural Network model in terms of accuracy on our dataset. Let's delve into some of these reasons:

1. Nature of Features and Model Suitability:

   Random Forest: RFs are very effective at handling structured tabular data with a mix of numerical and categorical features. They are robust to overfitting, especially with a sufficient number of trees, and can manage non-linear relationships between features.

   Graph Neural Network: GNNs excel when the graph structure and the relationships between nodes (e.g., social network connections) are crucial for making predictions. However, if the relationships between nodes do not add predictive value beyond individual node features, GNNs might struggle.

2. Feature Engineering and Data Representation:

   Random Forest: The feature engineering process, such as extracting TF-IDF features from tweets and profile features, might have produced highly informative features that RFs can exploit effectively. RFs also handle large feature spaces well with many trees.

   Graph Neural Network: The effectiveness of GNNs heavily depends on the quality of the graph and the node features. If the graph is noisy or if the node features are not well-represented, GNNs may not perform optimally. The aggregation of information from neighboring nodes might not add significant value if the node features themselves are strong predictors.

3. Model Complexity and Training:

   Random Forest: RFs are relatively easy to train and less sensitive to hyperparameter settings compared to deep learning models like GNNs. They require less fine-tuning and are less prone to issues like vanishing gradients or overfitting when using a large number of trees.

   Graph Neural Network: GNNs are more complex and require careful tuning of hyperparameters (e.g., learning rate, number of layers, hidden units) and training procedures. Training deep learning models can be more challenging and may require more data and computational resources to achieve optimal performance.

4. Graph Construction and Information Flow:

Random Forest: RF models do not depend on the graph structure and only leverage the individual node features, which might be sufficiently informative for the task.

Graph Neural Network: The effectiveness of GNNs depends on how well the graph structure captures the relationships between nodes. If the constructed graph does not accurately reflect meaningful relationships (e.g., follower connections that do not indicate similarity in behavior), the GNN may not leverage the graph structure effectively. The message-passing mechanism in GNNs may dilute the information if the graph is too sparse or too dense.

6. CONCLUSION

This project demonstrates that Graph Neural Networks are effective in detecting bot accounts on Twitter. By integrating profile features and tweet content into a graph-based model, the GNN can distinguish between genuine users and bots effectively. Future work could explore more advanced GNN architectures and additional features to improve detection accuracy even further. Our findings suggest that graph-based models, especially GNNs, are well-suited for bot detection on social media platforms. However, the comparison with the Random Forest model highlights the importance of choosing the right model based on the nature of the dataset and the specific characteristics of the features and relationships within the data.

REFERENCES

[1] Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2016). The rise of social bots. Communications of the ACM, 59(7), 96-104. doi:10.1145/2818717

[2] Varol, O., Ferrara, E., Davis, C. A., Menczer, F., & Flammini, A. (2017). Online Human-Bot Interactions: Detection, Estimation, and Characterization. In Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2017). AAAI.

[3] Alhosseini, S., & Lemnaru, C. (2020). A Survey of Machine Learning Techniques for Detecting Social Media Bots. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), 3278-3284. IEEE.