Bot Detection in Social Media Using GraphSage and BERT

Abhishek Deshmukh
Dept. of Computer Science
San Jose State University
San Jose, CA, USA
abhishek.deshmukh@sjsu.edu

Melody Moh
Dept. of Computer Science
San Jose State University
San Jose, CA, USA
melody.moh@sjsu.edu

Teng-Sheng Moh
Dept. of Computer Science
San Jose State University
San Jose, CA, USA
teng.moh@sjsu.edu

***Abstract*—The rise of digital platforms has resulted in an influx of widespread misinformation and automated bot activities, all of which continues to post a significant threat to information integrity and societal discourse. Misinformation and disinformation are often disguised as credible information spread through bots that have been designed to manipulate societal discourse. Hence, detecting and combating this issue requires advanced detection strategies when compared with traditional Machine Learning (ML)-based approaches. This paper introduces a novel graph-based detection system to tackle the increasing challenge of identifying bot users on social media. The strength of GraphSage (Graph Sample and Aggregation) for network-based pattern recognition has been integrated with the strength of BERT (Bidirectional Encoder Representations from Transformers) for deep contextual analysis together to create a robust bot detection system. This system concatenates BERT embeddings with GraphSage embeddings into a comprehensive feature vector, it therefore captures a rich blend of textual and network characteristics. SVM (Support Vector Machine) is then used to process the embeddings and classify the accounts. The system has achieved an impressive accuracy of 98.68% on the Cresci-15 dataset, outperforming all the compared models. On the Twibot-22 dataset, the system has achieved an accuracy of 74.62%, placing it in the middle tier of the state-of-art models compared with. These results highlight the efficacy and scalability of the proposed system across large-scale datasets and complex network graphs. Concatenating the embeddings to generate a more diverse feature set may be widely applicable to other areas of bot detection and misinformation classification.**

***Index Terms*—BERT, GraphSage, SVM, GCN, Bot Detection, Misinformation, Social Network**

I. Introduction

Social media has grown at an accelerated rate which has facilitated the widespread dissemination of information. It however has inadvertently intensified the propagation of misinformation and bot-driven contents. The vast outreach of social media provides an ideal breeding ground for these deceptive practices. Misinformation and disinformation have been skillfully crafted to blur the lines between facts and fiction by using bots that are intricately programmed to mimic human behavior, which have posed significant challenges to discerning the truth. Sophisticated detection strategies are badly needed by these complexities. As the social media landscape continues to evolve, the methodologies employed to ensure the preservation of information integrity and the protection of genuine public discourse need to be in constant contention with malicious tactics.

Traditional Machine Learning (ML) detection methods primarily rely on content-based analysis or statistical models [1], which are becoming increasingly inadequate due to the evolving complexity of social platforms. Graph-based ML techniques have shown encouraging results [2] as these techniques leverage the structural dynamics of social media data, enabling a deeper understanding. They capture the inter-connectedness, thus providing a holistic view of the data landscape.

This paper implements a novel method that concatenates BERT embeddings with GraphSAGE embeddings to create a rich feature vector containing textual and network information of each user which is then classified by an SVM. The proposed method leverages the deep contextual insights provided by BERT and the neighborhood aggregation strengths of GraphSAGE, demonstrating enhanced capabilities in detecting subtle nuances and patterns indicative of bot activities. The results are promising, suggesting that the new method has substantial merit and encourages further studies to strengthen the findings.

The paper is organized as follows: Section II describes the related works that explore graph-based ML techniques with a particular emphasis placed on how these techniques have been adapted and refined to more accurately identify misinformation campaigns and automated bot activities within complex network structures. Section III provides the intricate details on the implementation of the methodology. Section IV illustrates the results of this methodology while comparing and contrasting with other current methodologies; the section also provides key insights into the results. Finally, Section V provides a summary of the project while also suggesting key areas of improvement and highlighting the possibility of further research.

II. Related Works

In the field of bot detection, leveraging sophisticated computational techniques to identify automated accounts has become increasingly critical as bots evolve to mimic human behaviors on digital platforms. The integration of diverse machine learning and natural language processing models has been a focal point of recent research efforts.

979-8-3503-7311-0/24/$31.00 ©2024 IEEE

The paper [3] presents a novel model for social bot detection named BGSRD, which synergistically combines the semantic text analysis capabilities of BERT [4] processing strengths of GCNs [5]. The authors aim to address the challenges in detecting sophisticated social bots on platforms like Twitter, where bots are becoming increasingly adept at mimicking human behavior. The BGSRD model operates by constructing a heterogeneous graph from the data and utilizing BERT to generate node representations. This method takes advantage of the pre-training power of BERT and the structural pattern recognition of GCNs, resulting in a model that can effectively identify bots by learning from both text content and user interaction patterns.

The synergy between BERT's linguistic prowess and GCN's structural analysis offers a potent tool for the identification and scrutiny of social bots, which are increasingly sophisticated in their mimicry of human online activity.

BotRGCN (Bot detection with Relational Graph Convolutional Networks) is introduced in [6]. It targets Twitter bot detection, addressing challenges like disguise and community behavior of bots. BotRGCN constructs a heterogeneous graph from Twitter's network, applying relational GCNs. It uses multi-modal user semantic and property information for enhanced detection. The methodology combines numerical and categorical user data with tweets and descriptions encoded by RoBERTa [7], providing a comprehensive feature set for the GCN.

In the BotRGCN model [6], a heterogeneous graph is constructed to represent Twitter's complex social network. This graph is built using various types of nodes and edges, reflecting different relationships and interactions on the platform.In this case, nodes represent Twitter users, and edges signify different kinds of relationships like followings, retweets, or mentions. This diverse graph structure allows BotRGCN to capture a wide range of relational data, offering a detailed view of the social interactions and connections that can be indicative of bot-like behavior. This rich relational representation forms the basis for the model's sophisticated bot detection capabilities. The methodology's application on the TwiBot dataset [8] demonstrates BotRGCN's effectiveness. The model outperforms baselines, showing its capability in using follow relationships and user information to detect bots. BotRGCN's design, incorporating a heterogeneous graph and multi-modal information, proves essential for robust performance in Twitter bot detection. This paved the way for the idea of incorporating BERT with GraphSage for this paper.

III. Proposed Method

A. Datasets

This sections highlights some of the key datasets used as benchmarks in the paradigm of detecting misinformation and bots.

1) TwiBot dataset: The TwiBot-20 and TwiBot-22 datasets [8], [9] are critical resources in the field of misinformation classification and bot detection, particularly for research involving graph-based machine learning techniques.

The TwiBot-20 dataset was created using a breadth-first search strategy starting from various seed users on Twitter to form a representative user graph. The data encompassed the latest 200 tweets from each user, all user properties accessible via the Twitter API, and follower-following relationships, ensuring a rich set of information for bot detection algorithms. For the annotation process, the authors used criteria based on previous research to identify bots, including tweet originality and activity patterns. They conducted a crowdsourced review where active Twitter users served as annotators, with ambiguous cases being further probed via direct messages or evaluated by the research team for consensus.

TwiBot-22 advances further, addressing the shortcomings of previous datasets like scale limitations and annotation quality. It is aimed at refining the detection of Twitter bots by offering a more diverse and detailed graph structure of the Twitter network. This dataset aids in deepening the understanding of user interactions and the overall network structure, which is crucial for developing more sophisticated graph-based bot detection methods. For this paper the Twibot-22 version of the dataset was used.

2) Cresci-15 Dataset: The Cresci-2015 dataset [10] is a well-established benchmark for research in social bot detection. The Cresci-2015 dataset comprises two distinct groups of Twitter accounts: genuine accounts and social bots. The genuine accounts were manually verified to ensure their authenticity, while the bot accounts were collected from publicly available sources of known bot operations. This clear division makes the dataset particularly useful for training and testing bot detection systems.

In total, the dataset includes 2,372 genuine accounts and 2,991 bot accounts, providing a rich and balanced environment for developing and evaluating machine learning models. Each account in the dataset is represented through a variety of features including tweet content, temporal tweeting patterns, account properties (like creation date and follower/following ratios), and social connections, which can all be leveraged to distinguish between human and bot-operated accounts.

The Cresci-2015 dataset has been widely used in academic and industrial research, serving as a cornerstone for many studies. It allows researchers to validate the effectiveness of their detection algorithms and to compare the performance of different approaches under consistent conditions. The dataset's structured composition and the realism of its account profiles make it a valuable resource for advancing the field of social bot detection.

B. Challenges

BERT is a large scale pre-trained model which makes generating embeddings for a large number of tweets quite computationally expensive. Hence, this paper also incorporates DistilBERT, a distilled version of BERT that retains most of the original model's predictive power but is smaller and faster. This adaptation is crucial for processing large-scale data efficiently, making it a practical choice for real-world applications where computational resources are limited.

In this paper, the substantial Twibot-22 dataset is leveraged, which includes approximately 1 million users and over 80 million tweets. This dataset presents a unique opportunity to analyze user behavior and identify potential bots using sophisticated machine learning techniques. A key part of the methodology involves generating BERT text embeddings for the tweets to capture nuanced language patterns that might indicate automated activity.

Method	Processing Time (hours:mins)
BERT	$\approx 320:23$
DistilBERT	$\approx 100:55$

TABLE I
TIME DURATION FOR PREPROCESSING ALL USER TWEETS IN TWIBOT 22

The above Table showcases the duration required to process the data to generate BERT embeddings for all user tweets. Originally, the plan was to process embeddings for all tweets available per user. However, the sheer volume of data in Twibot-22 posed significant computational challenges. Processing times using the full BERT model extended to around 300 hours, and even with the faster DistilBERT model, the time was around 100 hours. These durations were impractical for appropriate timeline and resource allocation.

To address this, the approach was adjusted by limiting the scope of the analysis to the 15 most recent tweets per user. This decision was driven by a few key considerations:

- Relevance and Recency: Recent tweets are more likely to reflect current behavior and patterns relevant to detection algorithms.
- Efficiency: By focusing on fewer tweets per user, it could significantly reduce computational demands without compromising the integrity of the results.
- Scalability: This approach allowed processing a larger number of users within a reasonable timeframe, enhancing the overall scalability of the methodology.

Method	Processing Time (hours:mins)
BERT	$\approx 75:12$
DistilBERT	$\approx 13:40$

TABLE II
TIME DURATION FOR PREPROCESSING 15 TWEETS PER USER IN TWIBOT 22

With this adjustment, the processing time for generating BERT embeddings was reduced to approximately 75 hours, and to about 13 hours using DistilBERT. This reduction ensures that a robust analysis pipeline is maintained while staying within practical limits for computation and time resources. This is highlighted in the table above.

C. Methodology

This section defines this papers sophisticated methodology for detecting bots, integrating both text-based and graph-based analytical techniques. The workflow, depicted in the accompanying diagram, outlines a comprehensive process beginning with dataset preprocessing and culminating in the evaluation of classification results.

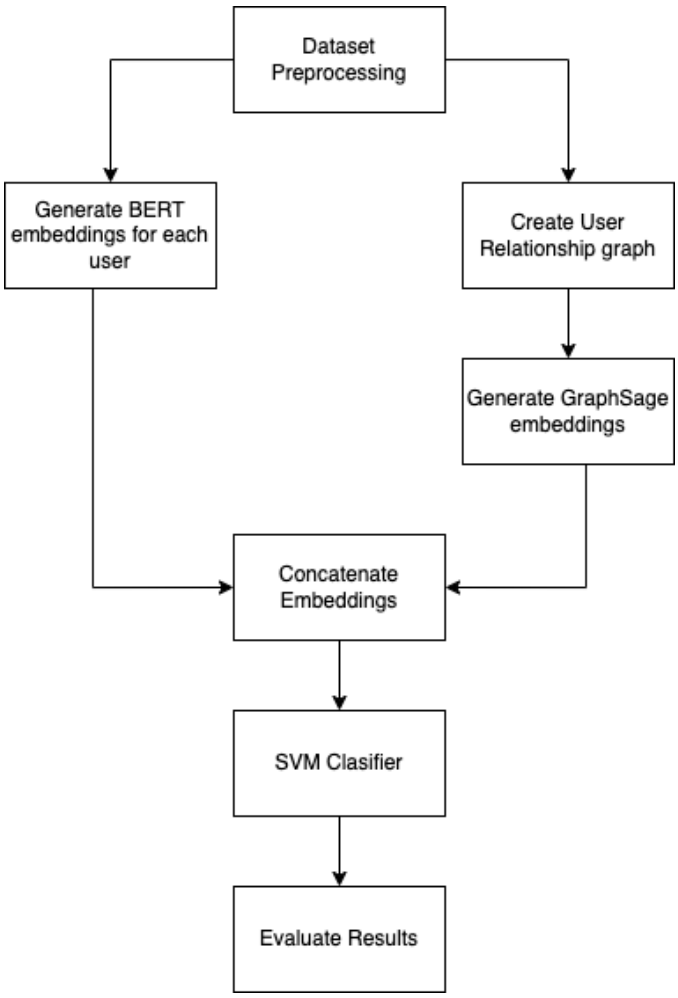

Fig. 1. Methodology workflow

The first stage involves preprocessing the dataset, where the dataset is cleaned and organized to ensure it is amenable for analysis. This foundational step is crucial for preparing the data for subsequent embedding generation processes. Then BERT embeddings are generated for each user, utilizing their tweet content to extract rich, contextual language features. These embeddings are instrumental in capturing the subtle linguistic patterns that may indicate automated bot activities.

Concurrently, a user relationship graph is constructed that visualizes the connections between users based on their interactions on the platform. This graph is essential for analyzing the social dynamics and influence patterns among the users, providing a structural perspective that complements the textual analysis.

Following the construction of the graph, the GraphSage algorithm is applied to generate embeddings from this user relationship graph. GraphSage is particularly adept at incorporating node feature information, which allows us to capture the nuanced relational data embedded within the graph structure.

The embeddings from GraphSage encapsulate these relational dynamics, offering another layer of features for bot detection.

These two sets of embeddings, derived from BERT and GraphSage, are then concatenated to form a comprehensive feature set that combines both textual and relational information. This concatenated feature set feeds into a Support Vector Machine (SVM) classifier, chosen for its robust performance in high-dimensional spaces. The SVM classifier utilizes these complex features to distinguish between bot and genuine users effectively.

The final stage of the workflow is the evaluation of the SVM classifier's results. This evaluation assesses the effectiveness of the combined textual and graph-based approach, providing insights into the accuracy of the bot detection methodology. Through this detailed implementation process, this paper aims to leverage the strengths of both text and graph-based data, enhancing the precision and reliability of bot detection capabilities on social media platforms.

D. Data Preprocessing

In this paper, the Twibot-22 and Cresci-15 datasets are employed, both of which offer a rich array of data conducive to identifying automated behaviors. These datasets provide detailed user profiles, including numerical data such as the number of followers, the number of people followed, and tweet counts, metadata of users and all tweets by each user. Given the similarities in the data offered by both datasets, uniform preprocessing methodology is applied to both. This approach involves cleaning and normalizing the data, extracting relevant numerical features from user profiles, standardizing data formats to facilitate effective feature utilization and creating edge index files. By employing such consistent preprocessing steps, it was ensured that the datasets are optimally prepared for the subsequent stages of analysis, which involve leveraging both textual and graph-based machine learning techniques to detect and analyze bot activity. This uniform preprocessing not only streamlines the workflow but also ensures comparability and consistency in the treatment and analysis of the data across different dataset structures.

1) User Data Preprocessing: The initial step in preprocessing involves cleaning the user data from the datasets. This step is crucial for ensuring the quality and reliability of the features used in later stages. Cleaning user data for this paper includes:

- Removing duplicates: Identifying and removing duplicate entries to prevent skewed analysis.
- Handling missing values: Omitting rows where values in critical fields such as user ID, tweet count, followers count, following count, favorites count and creation date are missing.

The next step is the creation of the edge indices required for graph construction. This data essentially contains the adjacency matrix in a coordinate list format, where each line represents an edge linking two nodes (users). The steps include:

- Extracting relationships: Identify and extract user relationships from the user data. This includes followers, friends and interactions such as retweets and mentions. The goal is to construct a homogeneous graph, so the type of relation does not matter in this case. Every relation is treated the same. If a relation exists, it considered to be an edge between the nodes.
- Mapping to indices: Users are mapped to unique indices that serve as nodes in the graph. The user ID is replaced with a sequential index to facilitate easier processing.
- Building edge list: An edge list is created where each row represents a connection between two users. This list includes source and destination indices derived from user relationships.
- Conversion to tensor format: Finally, the edge list is converted into a tensor format suitable for use with GraphSage.

2) Tweet Data Preprocessing: Since the quality of text data directly impacts the performance of BERT and DistilBERT, cleaning tweet data is a fundamental step. This involves:

- Text normalization: Lowercasing the text and removing numbers to reduce the variability in the text.
- Removing noise: Stripping out URLs, hashtags, mentions, and special characters that do not contribute to the understanding of the tweet's context.
- Handling emojis and symbols: Removing emojis and symbols that are not standard ASCII characters.
- Stop words removal: Eliminating common words (e.g., "and", "the") that are not be useful for analysis to focus on meaningful words.

Specifically for the Twibot-22 dataset, only the most recent 15 tweets are considered for each user as highlighted in the Background section. For Cresci, all tweets of each user are considered.

The preprocessing steps detailed above are critical for ensuring the data is clean, structured, and ready for subsequent stages of modeling, particularly when employing sophisticated techniques like BERT for text embeddings and GraphSage for analyzing user interactions on social platforms.

E. Generating BERT embeddings from Tweet Data

After cleaning the tweets, the next step involves generating embeddings using DistilBERT. This process translates the textual data into a numerical form that captures semantic meanings, which is used for further analysis.

The followings steps are done to generate the text embeddings:

- Tokenization: The process begins with tokenization, where the cleaned tweets are converted into a format that BERT can process. This involves breaking down the text into words or subwords, known as tokens, and mapping these tokens to their respective indices in DistilBERT's vocabulary.
- Generating Embeddings: After tokenization, the tokenized tweets are fed into the DistilBERT model. It processes these tokens and generates embeddings for each tweet. These embeddings are high-dimensional vectors

that capture the contextual relationships between words in a tweet.
- Aggregating Embeddings: This step involves aggregating the embeddings of all user tweets. This is achieved by taking the mean of all tweets per user and getting a single embedding value for each user of size 768.

After completing the above steps, the generated embeddings are stored as a tensor file which will be used for concatenation with GraphSage embeddings.

F. Feature Extraction from user data

The following features are extracted from the user data to be used as node features for GraphSage:

- Tweet Count: Reflects the total activity level of the user, indicating how often they interact with the platform.
- Followers Count: Provides insight into the user's social reach and popularity.
- Following Count: Helps assess the user's networking behavior, which in excessive amounts can be indicative of spam or bot-like operations.
- Favorites Count: Shows the extent of the user's engagement with content other than their own, giving a sense of how they interact with the platform.
- Account Age: Unlike the previous features which are readily available in the chosen datasets, this is derived from the creation date. The current date is subtracted from the user's account creation date. The result is expressed days.

After extracting and calculating the necessary features, they are compiled into a structured tensor. This tensor aggregates the key numerical features for each user into a single, coherent format, which is then used as node features for GraphSage.

G. Graph Construction

Graph construction is a critical component in analyzing social networks, especially in tasks like bot detection where the relationships between entities (users) can provide significant insights. This process involves representing the social interactions between users as a graph, where nodes represent the users and edges represent the relationships or interactions between them.

The structure of the graph is defined by the edge index tensor file generated previously, This file contains information about the edges between nodes in a format that is optimized for processing with graph-based machine learning models.

The file is structured as a two-row tensor, where each column represents an edge. The first row contains the source nodes, and the second row contains the destination nodes. For example, if the first row is [0, 2] and the second row is [1,0], this means that an edge needs to be drawn between 0 and 1 and 2 and 0. This is highlighted in the figure below.

The graph is constructed where Each user in the dataset is represented as a node in the graph. The numerical features extracted from user data previously are used as node features.

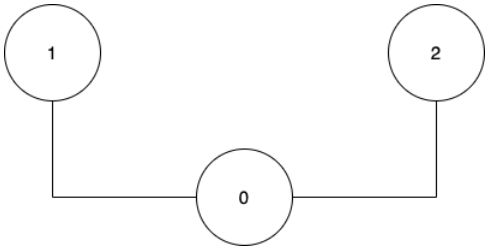

Fig. 2. Graph Construction Example

While edges are added between nodes according to the relationships outlined in the Edge Index file. Once constructed, this graph serves as the input to GraphSage.

H. GraphSage Embeddings

In this paper, the GraphSage model is used to generate node embeddings directly from the input features and the graph structure, without the need for training epochs or optimization tasks typically associated with learning models. This method provides a straightforward mechanism to capture the structural and feature-based information of the nodes in a single pass.

Initializing the GraphSage Model involves the following:

- The GraphSAGE model is initialized with a single SAGEConv layer, with the in_channels set to 5 (input node features) and out_channels set to 128 (embedding vector size).
- A graph data object is created using the Data class from PyTorch Geometric, which encapsulates the node features and the graph connectivity using the egde index file.
- With the model in evaluation mode, embeddings are generated in a single forward pass. This operation does not require gradient calculations, as this process is purely for embedding generation rather than training.
- The SAGEConv layer processes the node features along with the graph structure to produce the final node embeddings. The embeddings are generated by aggregating features from each node's immediate neighbors (and potentially their own features), transforming this aggregated information through the convolution layer to produce a new set of features (embeddings) for each node.
- The resulting GraphSAGE embeddings contain the transformed features for each user, encapsulating both their inherent attributes and the contextual information provided by their connections in the graph. These embeddings are concatenated with text embeddings to be used downstream for processing through an SVM.

This concise approach to generating GraphSage embeddings provides a snapshot of the node's position and role within the broader network, effectively capturing both individual and collective characteristics that are crucial for analyzing complex social interactions, particularly in the context of bot detection.

I. Concatenation of Embeddings

In this paper, two distinct types of embeddings are generated for each user: BERT embeddings from textual data

(tweets) and GraphSage embeddings from user relationship graphs. Concatenating these embeddings forms a comprehensive feature vector that captures both textual content and social network structure, crucial for identifying nuanced patterns of bot behavior.

Each user's BERT embeddings (generated from their tweets and aggregated by mean) and their GraphSage embeddings (derived from the graph structure) are concatenated into a single vector. GraphSage embeddings have a dimension of 128 and BERT embeddings have a dimension of 768, the concatenated vector will have a dimension of 896. The two embeddings are basically stitched together side by side. This extended feature vector offers a comprehensive view of each user, integrating their content generation and interaction patterns.

Purpose of Concatenation:

- Dual Representation: BERT embeddings provide a deep contextual understanding of the language while GraphSage embeddings encapsulate the users' positions and roles within the network.
- Enhanced Feature Set: By concatenating these embeddings, the resulting vector for each user contains a richer set of features.

The concatenated embeddings serve as the input to the final machine learning models tasked with distinguishing between bots and genuine users. The enriched data representation provided by the concatenated embeddings allows these models to detect and learn from complex patterns that might be indicative of automated behaviors.

J. Classification using SVM

For the classification task, the input data consisted of concatenated embeddings derived from the user's text and network-based features. Specifically, these embeddings combined the 768-dimensional BERT embeddings, which capture textual nuances, with the 128-dimensional GraphSage embeddings that encapsulate the structural information of the social network. This approach created a comprehensive 896-dimensional feature vector for each user, providing a rich representation of both their textual content and their position within the network.

The choice of Support Vector Machine (SVM) with a linear kernel for this task was motivated by several factors. Firstly, SVMs are renowned for their effectiveness in high-dimensional spaces [11], which is pertinent given the dimensionality of the input embeddings. Additionally, the linear kernel was selected to maintain computational efficiency and to benefit from SVM's capacity for handling linearly separable data. It offers a good balance between complexity and performance, especially when dealing with potentially linearly separable transformations of high-dimensional data.

The parameters of the SVM were carefully chosen to optimize the model's performance. The linear kernel was employed without the addition of a kernel trick, relying instead on the inherent separability of the data in its transformed feature space. Other parameters, such as the regularization term, were adjusted to prevent overfitting while maintaining the model's ability to generalize well to unseen data.

The training process utilized a 5-fold cross-validation approach. This method involved dividing the complete dataset into five distinct subsets, using each in turn as a test set while training on the remaining four. This cross-validation technique not only helped in assessing the model's performance more reliably but also ensured that the training was robust, minimizing the influence of any particular subset of the data on the model's final parameters.

K. Evaluation

Evaluation of the model was carried out by calculating the accuracy and F1 scores, which are critical metrics for assessing classification models. Accuracy provided a straightforward proportion of correctly predicted instances against the total, offering a quick measure of the model's effectiveness. The F1 score, being the harmonic mean of precision and recall, gave a more nuanced view of the model's performance, especially important in scenarios where class imbalances might exist. These metrics together offered a comprehensive overview of the model's performance. The results are highlighted in the next section.

L. Execution Details

This section highlights the some of the key execution details for training each of models.

Method	**Dataset**	**Training Time (hours:mins)**
BERT+GraphSage	Cresci-15	3:47
DistilBERT+GraphSage	Cresci-15	2:32
GraphSage+SVM	Twibot-22	10:55
DistilBERT+GraphSage	Twibot-22	14:29
BERT+GraphSage*	Twibot-22	300>

TABLE III
TRAINING TIME COMPARISON OF DIFFERENT METHODS

From the above table we can observe that BERT+GraphSage* (Includes preprocessing and training time) requires more than 300 hours for the Twibot dataset if all tweets of all accounts are used. Since execution for such a long duration is infeasible, constraints to limit number of tweets per user were added along with faster BERT models such as DistilBERT.

All the experiments were conducted on a MacBook Pro 2019 with an 2.4 GHz Quad-Core Intel Core i5 Processor and 8 GB 2133 MHz LPDDR3 of RAM.

IV. RESULTS

This section presents the results of the bot detection model, which was trained and tested using the Cresci-2015 dataset and the Twibot-22 dataset. The approach utilized a combination of BERT and GraphSage embeddings to capture both textual and structural information of user profiles and interactions. The performance of the model was evaluated based on standard metrics such as accuracy and F1-score. Here, the effectiveness of the model in distinguishing between genuine accounts and

bots is detailed, highlighting the impact of various features and the robustness of the method across different testing scenarios. These results underscore the potential of the model to contribute significantly to ongoing efforts in enhancing the detection of sophisticated social bots.

A. Cresci-15 Results

Two experiements were conducted on the Cresci-15 dataset to offer a more holistic approach, once using BERT and then using DistilBERT as the text encoders while the GraphSage and SVM models remain the same between the two. This section highlights the results of both methodologies compared to standard as well as current state-of-art models. GraphSage+BERT and GraphSage+DistilBERT are the models implemented in this paper. The comparison to other state-of-the-art methods on the Cresci-15 dataset as summarized in the below Table. The results are based on models trained using a 5-fold cross-validation scheme to ensure consistency and reliability of the performance metrics.

Method	Type (F/G)	Accuracy (%)	F1 score (%)
SATAR [12]	F	93.42	95.05
GCN [13]	G	77.08	77.91
BotRGCN [6]	G	96.52	97.30
RGT [12]	G	97.15	97.78
BIC [12]	G	98.35	98.71
GraphSage+BERT	G	98.68	98.92
GraphSage+DistilBERT	G	98.56	98.88

TABLE IV
PERFORMANCE COMPARISON OF DIFFERENT METHODS ON THE CRESCI DATASET

The approach using GraphSage combined with BERT and DistilBERT embeddings yields high performance, as evidenced by accuracy and F1 scores:

- GraphSage+BERT achieves an accuracy of 98.68% and an F1 score of 98.92%.
- GraphSage+DistilBERT records slightly lower metrics, with an accuracy of 98.56% and an F1 score of 98.88%.

Both configurations surpass the performance of traditional graph-based models such as GCN, which has an accuracy and F1 score of 77.08% and 77.91%, respectively. Notably, the models also exhibit competitive or superior performance compared to other recent methods:

- BotRGCN reports an accuracy of 96.52% and an F1 score of 97.30
- RGT shows an accuracy of 97.15% and an F1 score of 97.78%, and
- BIC presents the highest scores among other compared models with an accuracy of 98.35% and an F1 score of 98.71%.

The improvement in accuracy and F1 score in the models can be attributed to the effective combination of GraphSage's capability to capture neighborhood information and BERT's sophisticated handling of textual features. While GraphSage+DistilBERT offers a small reduction in performance relative to GraphSage+BERT, it provides a beneficial trade-off between efficiency and effectiveness, leveraging the reduced complexity of DistilBERT.

B. Twibot-22 Results

Two experiments were conducted for Twibot 22, once without any BERT embeddings using only the GraphSage embeddings and once with DistilBERT embeddings. The models GraphSage+SVM and GraphSage+DistilBERT are compared against other contemporary approaches as presented in the below Table. Each model was evaluated using a 5-fold cross-validation method to ensure the integrity and comparability of the results.

Method	Type (F/G)	Accuracy (%)	F1 score (%)
SVM [14]	F	49.30	-
GCN [15]	G	47.72	38.10
HGT [16]	G	74.91	39.60
BotRGCN [6]	G	79.66	57.50
RGT [9]	G	76.47	49.24
BGSRD [3]	G	71.88	21.14
GraphSage+SVM	G	67.21	48.32
GraphSage+DistilBERT	G	74.62	51.69

TABLE V
PERFORMANCE COMPARISON OF DIFFERENT METHODS ON THE TWIBOT DATASET

The performance of the models in question, based on accuracy and F1 scores, are as follows:

- GraphSage+SVM showed an accuracy of 67.21% and an F1 score of 48.32%.
- GraphSage+DistilBERT delivered an accuracy of 74.62% and an F1 score of 51.69%.

Comparing and contrasting these results against other notable methods tested on the same dataset:

- SVM alone had a notably lower accuracy of 49.30%. F1 score was not reported.
- GCN and HGT delivered accuracy scores of 47.72% and 74.91% respectively, with F1 scores at 38.10% and 39.60%.
- BotRGCN and RGT performed better, with BotRGCN achieving an accuracy of 79.66% and an F1 score of 57.50%, while RGT reported an accuracy of 76.47% and an F1 score of 49.24%.
- BGSRD presented an accuracy of 71.88% but a much lower F1 score of 21.14%.

The incorporation of GraphSage with DistilBERT significantly improved both accuracy and F1 scores over using GraphSage with a traditional SVM classifier. This suggests that the textual context captured by DistilBERT embeddings provides meaningful enhancement over the graph structural information alone, as provided by GraphSage.

C. Summary of Main Results

The above illustrated the results of enhancing the effectiveness of bot detection focusing on the Cresci-15 and Twibot-22 datasets. On the Cresci-15 dataset, the GraphSage+BERT model demonstrated exceptional performance, besting all of

the state-of-the-art models compared. This underscores the effectiveness of combining rich textual features extracted by BERT with the structural insights provided by GraphSage. Although the GraphSage+DistilBERT model slightly underperformed compared to its BERT counterpart, it offered a significant computational advantage. This model's balance between efficiency and performance makes it particularly suitable for scenarios where computational resources are constrained. The trends observed on the Cresci-15 dataset were mirrored on the Twibot-22 dataset, albeit with overall lower performance metrics across all models. This was expected due to the inherent challenges and complexities of the Twibot-22 dataset. Even here, the GraphSage+DistilBERT model stood out, providing a good balance between accuracy and computational demand. By contrasting with the GraphSage+SVM only model, it also demonstrated that the addition of the DistilBERT embeddings greatly increased the accuracy.

V. Conclusion and Future Work

This paper aimed to explore and improve the efficacy of bot-detection methodologies on social media platforms, centering on the Cresci-15 and Twibot-22 datasets. The system has successfully demonstrated the integration of GraphSage with BERT embeddings which offers a robust approach to bot detection on social media. This approach has effectively harnessed the synergy between advanced natural language processing techniques and graph-based learning algorithms while addressing the intricate challenges of identifying inauthentic behavior across diverse social media platforms. By integrating GraphSage with both BERT and its distilled version, DistilBERT, the developed models have shown to outperform the main traditional models while remained competitive comparing with most of the existing, state-of-the-art methods. The exploration of both full and distilled versions of BERT within this framework has offered valuable insights into balancing computing efficiency with performance, making this methodology adaptable for various operational environments and resource constraints.

To further enhance the efficacy and applicability of the botfdetection models developed in this paper, several areas of future research are recommended: (1) The current limitation to the 15 most recent tweets per user due to computational constraints may restrict the model's learning potential. If computational limitation is not a concern, more tweets of each user should be considered, which would potentially lead to significantly better results. (2) While DistilBERT offers a balance between performance and efficiency, employing the full BERT model may improve the detection capabilities due to its richer linguistic understanding. (3) The current implementation uses only 5 numerical features; by modifying the GraphSage model and including more node features and by using heterogeneous graphs to add more information for the feature set, more numerical and non-numerical features may be tested to evaluate the extended feature set. (4) Finally, the idea of concatenating embeddings may be applied to other areas such as detecting fake news and combating misinformation [16]; moreover, it is worthy to integrate text-based features using BERT, aiming to significantly improve the present findings.

References

[1] S. Qian, J. Hu, Q. Fang, and C. Xu, "Knowledge-aware multi-modal adaptive graph convolutional networks for fake news detection," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 17, no. 3, jul 2021. [Online]. Available: https://doi.org/10.1145/3451215

[2] I. Varlamis, D. Michail, F. Glykou, and P. Tsantilas, "A survey on the use of graph convolutional networks for combating fake news," *Future Internet*, vol. 14, no. 3, 2022. [Online]. Available: https://www.mdpi.com/1999-5903/14/3/70

[3] 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, 2022. [Online]. Available: https://www.mdpi.com/2073-8994/14/1/30

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[6] 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*, ser. ASONAM '21. New York, NY, USA: Association for Computing Machinery, 2022, p. 236–239. [Online]. Available: https://doi.org/10.1145/3487351.3488336

[7] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[8] S. Feng, H. Wan, N. Wang, J. Li, and M. Luo, "Twibot-20: A comprehensive twitter bot detection benchmark," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ser. CIKM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 4485–4494. [Online]. Available: https://doi.org/10.1145/3459637.3482019

[9] S. Feng, Z. Tan, H. Wan, N. Wang, Z. Chen, B. Zhang, Q. Zheng, W. Zhang, Z. Lei, S. Yang, X. Feng, Q. Zhang, H. Wang, Y. Liu, Y. Bai, H. Wang, Z. Cai, Y. Wang, L. Zheng, Z. Ma, J. Li, and M. Luo, "Twibot-22: towards graph-based twitter bot detection," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2024.

[10] M. Mazza, S. Cresci, M. Avvenuti, W. Quattrociocchi, and M. Tesconi, "Rtbust: Exploiting temporal patterns for botnet detection on twitter," 2019.

[11] Y. Yang, J. Li, and Y. Yang, "The research of the fast svm classifier method," in *2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2015, pp. 121–124.

[12] Z. Lei, H. Wan, W. Zhang, S. Feng, Z. Chen, J. Li, Q. Zheng, and M. Luo, "Bic: Twitter bot detection with text-graph interaction and semantic consistency," 2023.

[13] Y. Wu, D. Lian, Y. Xu, L. Wu, and E. Chen, "Graph convolutional networks with markov random field reasoning for social spammer detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 1054–1061, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5455

[14] J. Rodríguez-Ruiz, J. I. Mata-Sánchez, R. Monroy, O. Loyola-González, and A. López-Cuevas, "A one-class classification approach for bot detection on twitter," *Computers & Security*, vol. 91, p. 101715, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820300031

[15] S. Ali Alhosseini, R. Bin Tareaf, P. Najafi, and C. Meinel, "Detect me if you can: Spam bot detection using inductive representation learning," in *Companion Proceedings of The 2019 World Wide Web Conference*, ser. WWW '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 148–153. [Online]. Available: https://doi.org/10.1145/3308560.3316504

[16] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," 2020.