

Spring 2023

Twitter Bot Detection using NLP and Graph Classification

Warada Jayant Kulkarni
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects



Part of the [Artificial Intelligence and Robotics Commons](#), and the [Other Computer Sciences Commons](#)

Recommended Citation

Kulkarni, Warada Jayant, "Twitter Bot Detection using NLP and Graph Classification" (2023). *Master's Projects*. 1263.

DOI: <https://doi.org/10.31979/etd.ms4h-hj9x>

https://scholarworks.sjsu.edu/etd_projects/1263

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Twitter Bot Detection using NLP and Graph Classification

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Warada Jayant Kulkarni

May 2023

© 2023

Warada Jayant Kulkarni

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

Twitter Bot Detection using NLP and Graph Classification

by

Warada Jayant Kulkarni

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2023

Dr. Katerina Potika Department of Computer Science

Dr. William Andreopoulos Department of Computer Science

Dr. Genya Ishigaki Department of Computer Science

ABSTRACT

Twitter Bot Detection using NLP and Graph Classification

by Warada Jayant Kulkarni

Social media platforms are one of the primary resources for information as it is easily accessible, low in cost, and provides a high rate of information spread. Online social media (OSM) have become the main source of news information around the world, but because of the distributed nature of the web, it has increased the risk of fake news spread. Fake news is misleading information that is published as real news. Therefore, identifying fake news and flagging them as such, as well as detecting sources that generate them is an ongoing task for researchers and OSM companies. Bots are artificial users and are useful for various tasks. Unfortunately, bots do often disseminate inaccurate news and low-quality information [1]. Discriminating bots from real users is a difficult task to perform just based on content.

The aim of this project is to explore and expand techniques for bot detection. A novel approach is proposed by combining text features of the news and the graph structure of the diffusion of the specific news on the OSM. In our methodology, we first perform feature extraction using Natural Language Processing techniques, like the pre-trained models, BERT [2] and spaCy [3] on the news. Next, we create diffusion graphs of the news posted by bots, and news posted by real users based on how it is propagated through the OSM. We model our problem as a graph classification one and apply graph convolutional network approaches to solve it. We use the Cresci-2017 [4] Twitter dataset for our experiments, which contains real and bot users and their tweets. We expand this dataset by performing dataset augmentation using web-scraping to fetch additional tweets and user data to create the graphs. We analyze the graphs that we constructed based on news from bots and from real users' graphs. An experimental

comparison between the existing techniques and the proposed methodology will be performed to better understand the scope for optimization and improvement.

Keywords - Natural language processing, graph classification, embedding techniques, Twitter

ACKNOWLEDGMENTS

I would like to thank Dr. Katerina Potika, for her guidance and support throughout the research work. I was able to progress through this work due to her constant advice and input.

I also appreciate the inputs provided by my committee members, Dr. William Andreopoulos and Dr. Genya Ishigaki. I want to thank the Department of Computer Science for giving me this opportunity to conduct my research work on this topic. Finally, I want to express my gratitude to my family members and friends for their unwavering support.

TABLE OF CONTENTS

CHAPTER

1	Introduction	1
1.1	Problem Definition	1
1.2	Terminologies	2
1.2.1	Graph Classification	2
1.2.2	Graph Isomorphism	3
1.2.3	NLP Pre-processing Techniques	3
1.2.4	NLP Feature Extraction	6
1.2.5	Graph Convolutional Network	10
2	Related Work	12
3	Methodology	16
3.1	Twitter Data	17
3.2	Phases in Implementation Plan	17
3.3	Dataset Augmentation	19
3.4	Creation of Graphs	19
3.5	Classical Machine Learning Approach Architecture	22
3.5.1	K-Nearest Neighbours	23
3.5.2	Support Vector Machine	25
3.5.3	Random Forest Classifier	25
3.6	GCN Model Architecture	26
4	Experimental Evaluation	28

4.1	Dataset Description	28
4.2	Innovative and challenging Aspects	29
4.3	Graph Properties	30
4.4	Classical Machine Learning Approach	30
4.5	GCN Approach Results	32
5	Conclusion and Future Work	36
	LIST OF REFERENCES	38
	APPENDIX	

LIST OF TABLES

1	Performance comparison for benchmark testing [5]	15
2	Statistics for graph densities observed	30
3	Dataset description for classical machine learning approach	31
4	spaCy results for classical machine learning approach	31
5	BERT results for classical machine learning approach	31
6	Hyper-parameters for GCN	33
7	Results using GCN	33
8	Hyper-parameter tuning of activation function	34

LIST OF FIGURES

1	Graph Classification [6]	2
2	Segmentation	4
3	Tokenization	5
4	Stop Words Removal	5
5	Stemming	6
6	Lemmatization	7
7	BERT Embeddings [7]	8
8	spaCy Embeddings [8]	9
9	Aggregation in GCN [9]	10
10	Homogeneous graph structure [10]	11
11	Heterogeneous graph structure [10]	11
12	Graph Classification for Twitter Bot or User Classification	16
13	Phases in Implementation	18
14	Dataset folder structure	19
15	Graph construction - Level 1	21
16	Graph Construction - Level 2	21
17	Architecture Diagram for Classical Machine Learning Approach	22
18	K-Nearest Neighbours [11]	24
19	Support Vector Machine [12]	25
20	Random Forest Classifier [13]	26
21	Architecture Diagram for GCN methodology	27

22	Accuracies for classical machine learning approach	32
23	F1 scores for classical machine learning approach	32
24	Loss observed per epoch	34
25	Loss observed per epoch	35

CHAPTER 1

Introduction

1.1 Problem Definition

Fake news can be defined as a piece of misleading information circulated as news. It can cause harm and issues such as anxiety, and stress to individuals. It can also lead to the spread of conspiracy theories and the creation of culture wars. One of the 5G conspiracy theories suggested that 5G is the cause of Covid 19 and people in the UK started torching 5G towers. During the Elections of 2016, the term fake news was declared as Word of the Year in the Collins Dictionary in 2017 [14]. In March 2020, during the Covid-19 pandemic, as a result of fear and panic, people made bad choices that led to poisoning and a significant increase in deaths as compared to March 2019 [15].

People share news without verifying if it is true or false. It is mostly done unconsciously. As per the study performed on fake news [16], given a clear classification of real and fake news, it is observed that the same people share real news and not false ones. Software companies like Twitter and Facebook have a mechanism inbuilt on their social media platforms that flag any post or news if it is unverifiable [17].

This report is structured as - Section 1.2 describes the terminologies, Section 2 elaborates on the literature survey and related works, followed by Section 3, where the current methodology is described, Section 4 shows the experimentation and results, and finally, Section 5 presents the conclusion and future work.

1.2 Terminologies

1.2.1 Graph Classification

The problem of graph classification states that given a family of graphs and different categories, the goal is to classify the graphs in their respective categories [18]. For instance, we can identify whether a chemical compound is toxic or not toxic by its structure created using graphs where nodes are atoms and edges is the bond shared between atoms. Known samples for toxic and non-toxic compounds can be utilized for training.

Graph classification is used in several domains where graphs between different classes are distinguished using graph statistics or graph features. One of the possible ways is to use graph kernels. A kernel is a function that provides the inner product. Kernel methods [18] perform mapping of graphs in feature space, and the inner products of feature vectors are considered during machine learning. This approach has proven to be efficient in high dimensional spaces as well.

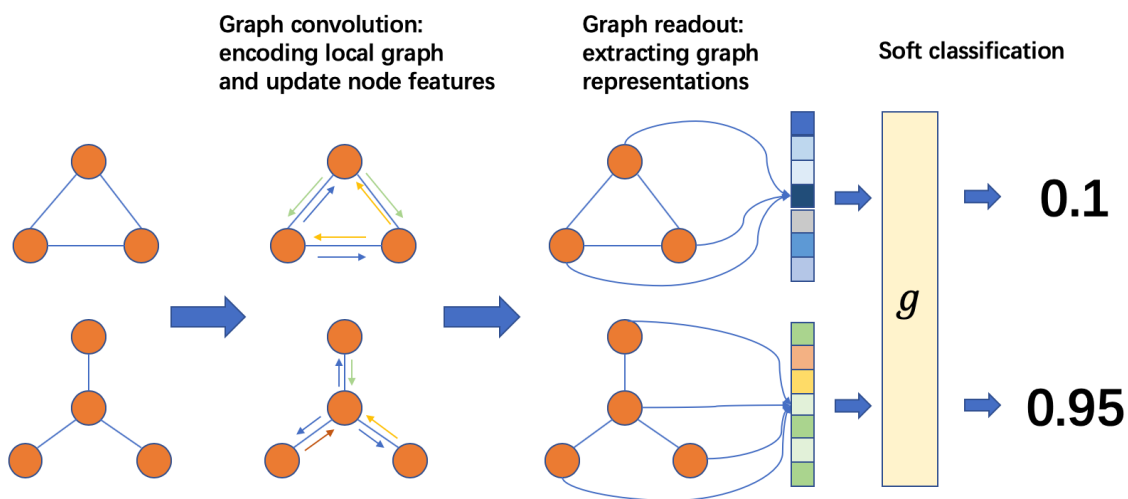


Figure 1: Graph Classification [6]

1.2.2 Graph Isomorphism

The concept of graph comparison can be applied in multiple domains such as structural comparison between chemical compounds, social networks, semantic structures in NLP, etc.

Computing a mapping between the vertices of G to G' such that G and G' are similar, that mapping is an isomorphism, and G and G' are isomorphic. For example, consider (u,v) to be an edge of G if $(f(u),f(v))$ is an edge in G' . This problem is referred to as Graph Isomorphism. There is no polynomial-time algorithm for graph isomorphism is known and also, it is not NP-complete. Within isomorphism, a concept called subgraph isomorphism is also used where a subset of nodes and edges are compared. This problem of subgraph isomorphism is known to be NP-complete.

Weisfeiler-Lehman (WL) Kernel is used for the purpose of graph isomorphism. It checks the similarity between two input graphs. A distinguishing property of the WL kernel is that it considers the attributes specific to nodes, referred to as node tags. Rosenfeld et. al [19] utilized the patterns in information propagation to identify misinformation on the social media platform Twitter. They utilized the WL graph kernel for their study on a dataset prepared by Vosoughi et al. [20] that has 126,000 cascades containing 4.5 million tweets. Through their experiments, they proved that the collective coexisting pattern of the population can reveal the truthfulness of the information.

1.2.3 NLP Pre-processing Techniques

The text from datasets is usually unstructured way and contains noise. Cleaning the text data and having it in a consistent format is very crucial to obtain efficient results. NLP defines some standard methods and techniques for data pre-processing

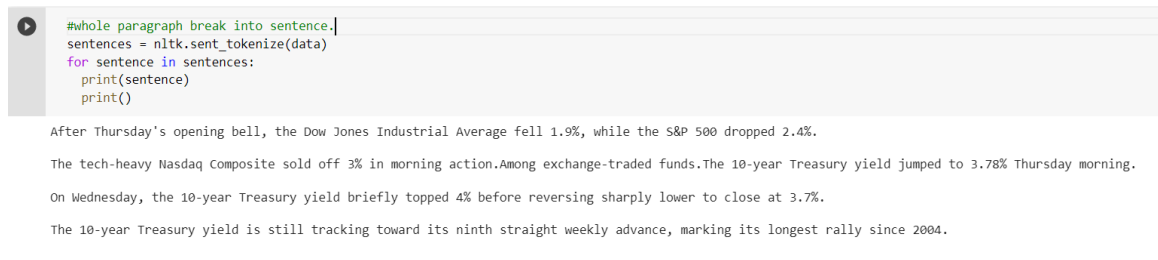
that can be used in a pipeline to streamline the entire process.

1. Segmentation

The task of dividing a document of text into continuous semantically meaningful segments is called segmentation. It performs dividing a text into corresponding sentences. It is used in text summarization, chatbots, in understanding the context, etc. This is also referred to as sentence tokenization. Example in Figure 2 demonstrates this step on a dummy text data.

Input text data considered are -

"After Thursday's opening bell, the Dow Jones Industrial Average fell 1.9%, while the S&P 500 dropped 2.4%. The tech-heavy Nasdaq Composite sold off 3% in morning action. Among exchange-traded funds. The 10-year Treasury yield jumped to 3.78% Thursday morning. On Wednesday, the 10-year Treasury yield briefly topped 4% before reversing sharply lower to close at 3.7%. The 10-year Treasury yield is still tracking toward its ninth straight weekly advance, marking its longest rally since 2004."



```
#whole paragraph break into sentence.
sentences = nltk.sent_tokenize(data)
for sentence in sentences:
    print(sentence)
    print()
```

After Thursday's opening bell, the Dow Jones Industrial Average fell 1.9%, while the S&P 500 dropped 2.4%.
The tech-heavy Nasdaq composite sold off 3% in morning action. Among exchange-traded funds. The 10-year Treasury yield jumped to 3.78% Thursday morning.
On Wednesday, the 10-year Treasury yield briefly topped 4% before reversing sharply lower to close at 3.7%.
The 10-year Treasury yield is still tracking toward its ninth straight weekly advance, marking its longest rally since 2004.

Figure 2: Segmentation

2. Tokenization

Next to segmentation, tokenization performs dividing a sentence into words, known as tokens in NLP. Using these tokens as basic blocks, analysis, and

statistical measures can be performed.

While creating tokens, different boundaries can be specified. For example, space between two words or any other punctuation can be any used as a boundary. Using the same dummy text data mentioned above, the result after tokenization is shown in Figure 3.

```
#tokenization
sentences = nltk.sent_tokenize(data)
for sentence in sentences:
    tokens = []
    tokens = nltk.word_tokenize(sentence)
    print(tokens)
    print()

['After', 'Thursday', "'s", 'opening', 'bell', ',', 'the', 'Dow', 'Jones', 'Industrial', 'Average', 'fell', '1.9', '%', ',', 'while', 'the', 'S', '&', 'P', '500', 'dropped', '2.4', '%',
['The', 'tech-heavy', 'Nasdaq', 'Composite', 'sold', 'off', '3', '%', 'in', 'morning', 'action.Among', 'exchange-traded', 'funds.The', '10-year', 'Treasury', 'yield', 'jumped', 'to',
['On', 'Wednesday', ',', 'the', '10-year', 'Treasury', 'yield', 'briefly', 'topped', '4', '%', 'before', 'reversing', 'sharply', 'lower', 'to', 'close', 'at', '3.7', '%', '.']
['The', '10-year', 'Treasury', 'yield', 'is', 'still', 'tracking', 'toward', 'its', 'ninth', 'straight', 'weekly', 'advance', ',', 'marking', 'its', 'longest', 'rally', 'since', '2004'
```

Figure 3: Tokenization

3. Stop-words removal

In NLP, along with the actual context of the data, certain words that do not add any value to the features are called stop words. For example, *a, the, in, of*. After the removal of stop words, the dummy text data looks as shown below in Figure 4.

```
#remove stop words
stop_words = set(stopwords.words('english'))

sentences = nltk.sent_tokenize(data)
for sentence in sentences:
    tokens = []
    #tokens = nltk.word_tokenize(sentence)
    #filtered_sentence = [w for w in nltk.word_tokenize(sentence) if not w.lower() in stop_words]
    filtered_sentence = []
    for w in nltk.word_tokenize(sentence):
        if w not in stop_words:
            filtered_sentence.append(w)
    print(filtered_sentence)
    print()

['After', 'Thursday', "'s", 'opening', 'bell', ',', 'Dow', 'Jones', 'Industrial', 'Average', 'fell', '1.9', '%', ',', 'S', '&', 'P', '500', 'dropped', '2.4', '%', '.']
['The', 'tech-heavy', 'Nasdaq', 'Composite', 'sold', '3', '%', 'morning', 'action.Among', 'exchange-traded', 'funds.The', '10-year', 'Treasury', 'yield', 'jumped', '3.78', '%', 'Th
['On', 'Wednesday', ',', '10-year', 'Treasury', 'yield', 'briefly', 'topped', '4', '%', 'reversing', 'sharply', 'lower', 'close', '3.7', '%', '.']
['The', '10-year', 'Treasury', 'yield', 'still', 'tracking', 'toward', 'ninth', 'straight', 'weekly', 'advance', ',', 'marking', 'longest', 'rally', 'since', '2004', '.']
```

Figure 4: Stop Words Removal

4. Stemming and Lemmatization

Stem is referred to as the root of a word in NLP. The process of stemming is converting the tokens to their root form. A hash table is maintained for words and their corresponding stem. With recent advancements in deep learning, stemming is also performed intuitively by neural networks. This concept of stemming is used by search engines like Google to fetch documents matching the user queries.

Going one more step ahead of stemming, lemmatization converts a token to its base form. A lemmatizer uses more rules of language and the information obtained from the context. For example, the word huge can be converted to great or the different usage of the token right can be understood by a lemmatizer. Similar to stemming, a hash table can be maintained for words and their base form. Results after stemming and lemmatization after the removal of stop words are as illustrated in Figure 5 and 6 respectively.

```
#stemming and lemmatization
stop_words = set(stopwords.words('english'))
ps = PorterStemmer()
sentences = nltk.sent_tokenize(data)
for sentence in sentences:
    stemming_result = []
    for w in nltk.word_tokenize(sentence):
        if w not in stop_words:
            stemming_result.append(ps.stem(w))
    print(stemming_result)
print()
```

```
['after', 'thursday', "'s", 'open', 'bell', ',', 'dow', 'jone', 'industri', 'averag', 'fell', '1.9', '%', ',', 's', '&', 'p', '500', 'drop', '2.4', '%', '.']
['the', 'tech-heavi', 'nasdaq', 'composit', 'sold', '3', '%', 'morn', 'action.among', 'exchange-trad', 'funds.th', '10-year', 'treasuri', 'yield', 'jump', '3.78', '%', 'thursday', 'mor
['on', 'wednesday', ',', '10-year', 'treasuri', 'yield', 'briefli', 'top', '4', '%', 'revers', 'sharpli', 'lower', 'close', '3.7', '%', '.']
['the', '10-year', 'treasuri', 'yield', 'still', 'track', 'toward', 'ninth', 'straight', 'weekli', 'advanc', ',', 'mark', 'longest', 'ralli', 'sinc', '2004', '.']
```

Figure 5: Stemming

1.2.4 NLP Feature Extraction

NLP deals with text as input data. The data, after pre-processing and cleaning is used for modeling purposes. Since computers understand numbers, this data is

```

#stemming and lemmatization
stop_words = set(stopwords.words('english'))
wordnet_lemmatizer = WordNetLemmatizer()
sentences = nltk.sent_tokenize(data)
for sentence in sentences:
    lemma_result = []
    for w in nltk.word_tokenize(sentence):
        if w not in stop_words:
            lemma_result.append(wordnet_lemmatizer.lemmatize(w))
    print(lemma_result)
print()

['After', 'Thursday', '', 'opening', 'bell', ',', 'Dow', 'Jones', 'Industrial', 'Average', 'fell', '1.9', '%', ',', 'S', '&', 'P', '500', 'dropped', '2.4', '%', '.']

['The', 'tech-heavy', 'Nasdaq', 'composite', 'sold', '3', '%', 'morning', 'action.Among', 'exchange-traded', 'funds.The', '10-year', 'Treasury', 'yield', 'jumped', '3.78', '%', 'Thurs

['On', 'Wednesday', ',', '10-year', 'Treasury', 'yield', 'briefly', 'topped', '4', '%', 'reversing', 'sharply', 'lower', 'close', '3.7', '%', '.']

['The', '10-year', 'Treasury', 'yield', 'still', 'tracking', 'toward', 'ninth', 'straight', 'weekly', 'advance', ',', 'marking', 'longest', 'rally', 'since', '2004', '.']

```

Figure 6: Lemmatization

mapped to numerical representation using different feature extraction methods.

1. Word Embedding

In order to have a numerical representation of text, every word in a document can be treated as a class and can be assigned a value of 1 wherever the word is present in the document and all the other words as 0. This representation is called as One-Hot Encoding. The next matrix is created to store their numerical values. This technique might work for small data. With a large vocabulary, the complexity of this technique will increase to a great extent.

Word embedding is a method that performs efficiently on large data. It helps in finding other features of the data such as the similarity, distance, and analogies between two words in a document. This provides the numerical representation more meaning and can provide better results.

2. BERT [2]

BERT (Bidirectional Encoder Representations from Transformers) [2] is a pre-trained transformer-based NLP architecture. It helps in understanding the context of the sentence. It is trained on the Masked Language Model and next-sentence prediction tasks. The masked language model works to find the

randomly hidden words. The reason for the BERT model being a bidirectional model is that it considers both preceding and succeeding words for the prediction tasks. This provides it with a better understanding of the context of the textual data. It takes the embedding tokens of the pre-processed data as input and provides the embedding per token, also called a hidden state.

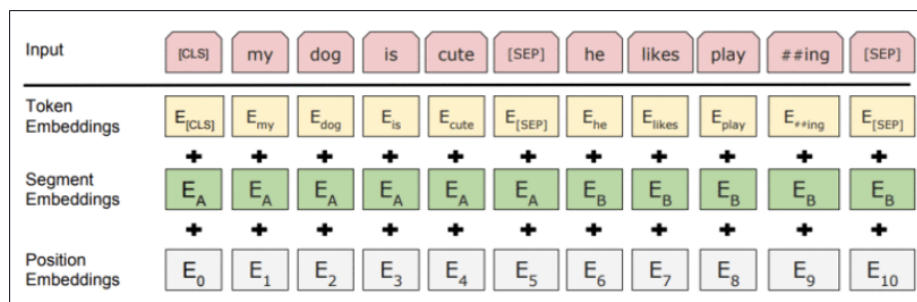


Figure 7: BERT Embeddings [7]

Every word in the given input text gets its own token embedding. Before it is passed on to BERT, all words are transformed into numerical vectors. The output of the model represents the context of every word in a particular sentence. Sentence embeddings are vectors created for every sentence. Each sentence gets its own embedding, thus enabling the model to evaluate the relationships between sentences and also identify which words belong to which sentences.

Using position embeddings, the positions for every token in the input sequence are encoded. In order to achieve a unique vector for each token that accounts for the meaning and the location in the input, these position embeddings are combined with the token embeddings.

Using these three kinds of embeddings, BERT proves to be very effective for a variety of NLP problems as it describes the positioning and context of tokens. Examples of applications of BERT are sentiment analysis, text summarization,

text classification, language translation, etc.

3. spaCY [3]

spaCY [3] provides faster and more efficient NLP algorithms as compared to BERT. The pre-trained model - `en_core_web_sm` - is a statistical model. It is trained on a large English corpus and is the lightweight version. It focuses more on the NLP concepts such as POS tagging, NER, and dependency parsing and hence is a statistical model. Similar to BERT, it takes text as input and provides a high-dimensional feature embedding for further processing.

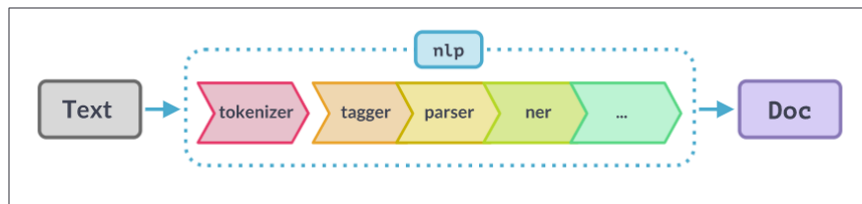


Figure 8: spaCy Embeddings [8]

To extract embeddings, spaCy uses convolutional neural networks (CNN) on text data and returns the features as vectors. Using these word vectors obtained from CNN, words from input text are represented numerically to utilize them further in ML models. spaCy has the ability to handle words that are out of vocabulary. It uses character-level embedding to obtain a word vector for such vocabulary.

SpaCy embeddings have demonstrated outstanding performance in a range of NLP applications, including sentiment analysis, named entity recognition, and text categorization. They are a part of the free and open-source spaCy package and are frequently used by NLP researchers and programmers.

1.2.5 Graph Convolutional Network

Graph Convolutional Networks (GCN) are based on the idea of a Convolutional neural network (CNN). Like CNN aggregates the information from the neighboring pixels, GCN aggregates the features from the neighboring graph elements. The basic concept is to perform convolutional operations on the graph in which every node is treated as a vector.

The working of GCN aggregation is illustrated in Figure 9.

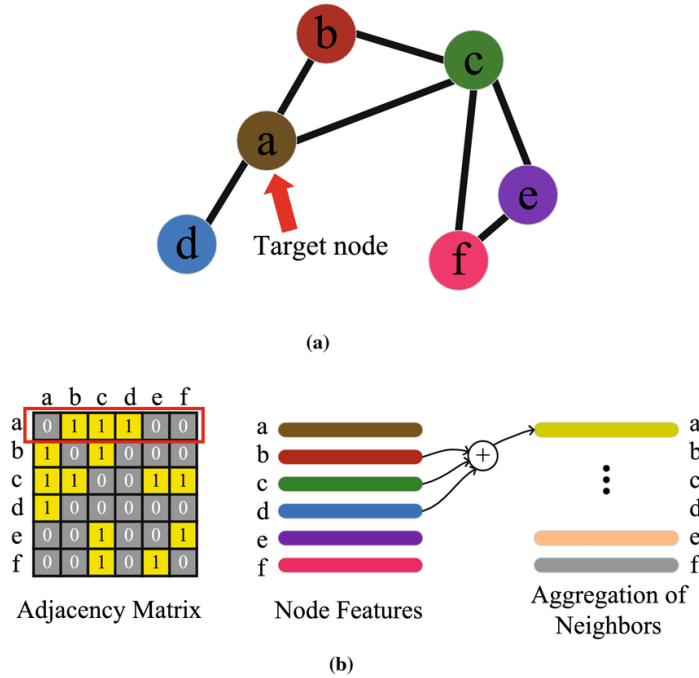


Figure 9: Aggregation in GCN [9]

Depending on the type of graph, GCNs can be modified specific to the problem.

1.2.5.1 Homogeneous GCN

When a graph has the same types of nodes and edges, the structure of the graph is homogeneous. The feature set dimension for nodes and edges is also the same. For example, in Figure 10, the nodes are of one type and the edges

are also undirected and of the same type. This makes the graph homogeneous. For homogeneous graphs, while passing the graphs as an input, a simple Data Object can be passed that contains the edge matrix, feature vectors for nodes and edges, and the label.



Figure 10: Homogeneous graph structure [10]

1.2.5.2 Heterogeneous GCN

The structure of a heterogeneous graph can contain nodes of multiple types, having different feature sets, different types of edges, and edge weights. The input data object created for GCN to process this type of graph is called as HeteroData. An example of a heterogeneous graph is shown in Figure 11. Nodes are of type building or user. The relationship between every node is different.

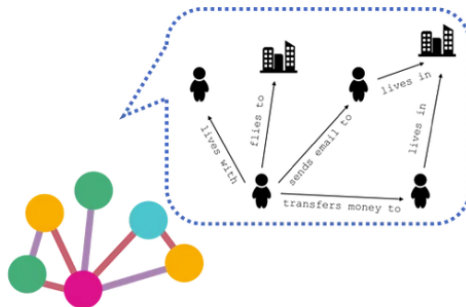


Figure 11: Heterogeneous graph structure [10]

CHAPTER 2

Related Work

Researchers [21] have worked on identifying the truthfulness of an article on social media platforms such as Twitter, using fact-checking websites. This technique is lengthy and tiresome as every tweet will be cross-verified, which might provide erroneous results. Using network analysis on the spread of the news, fake news can be detected. Garcia-Gasulla et. al [22] performed sentiment distribution among people on Twitter to analyze the impact of Covid-19 on society's health, social, economic behavior, and mobility. They used the BERT deep learning model [2] and STANZA tool [23]. As a result of their study, they found that there was an increase in people's fear, sadness, anger, and anticipation.

The technique proposed by Phayung M. [24] analyzes the data using natural language processing techniques to extract the features which the machine learning models then use to classify a piece of information as real, fake, or suspicious. Classical machine learning models were used. Their experiments concluded that LSTM performed the best. However, this methodology utilizes only text, and our Twitter dataset contains the connectivity between tweets, followers, and retweets additionally, and thus should be leveraged.

Rosenfeld et. al [19] utilized the patterns in information propagation to identify misinformation on the social media platform Twitter. Their research proves that the collective coexisting pattern of the population can reveal the truthfulness of the information. They utilized graph kernels for their study. Mapping of graphs into feature vectors is involved in kernel methods [25, 26] and the inner products

of these vectors are considered during the machine learning. A kernel is a function that results from the inner products of the feature vectors. Weisfeiler-Lehman (WL) Kernel performs the test of graph isomorphism between two given graphs. Rosenfeld et al. [19] used the WL kernel for their study on a dataset prepared by Vosoughi et al. [20] that has 126,000 cascades containing 4.5 million tweets.

In [16], observations from [17] are used, but on the Covid-19 specific data. Multiple fake/real information graphs are developed using the Twitter dataset. This dataset is created by the authors which contain tweets particular to Covid-19. Inspired by the model design proposed by Vosoughi et al. [20], nodes of a graph are Twitter IDs and edges are the retweeted data. For classification, the WL kernel algorithm is trained on the constructed real and fake information graphs. During testing, graphs of fake news are created and their similarity with real and fake news graphs is computed. As suggested for the WL kernel, the similarity between the two graphs is computed using cosine vectors of their respective graph embeddings. Prediction accuracy is thus computed for news to be retweeted.

Machine learning is one of the earliest approaches to detecting fake news and modeling credibility over news datasets. Support vector machines (SVM), decision trees, Naive Bayes, random forests, and ensemble methods are supervised learning methods extensively studied in fake news detection [27, 28, 29, 30, 31, 32]. In addition to the variability in models, the approach requires a manual selection of features. Afroz et al. applied k-nearest neighbor (KNN), Naive Bayes, Decision Tree, Logistic Regression, and SVM on three textual corpora including blog posts [31]. With a focus on lexical, syntactic, and content-specific features, the study found the SVM classifier with sequential minimal optimization with the best prediction performance. Classifying with similar models, Castillo et al. performed the task on a set of messages

collected on Twitter with additional propagation-related features and top elements like most frequent URL, authors, and hashtags [32]. BotOrNot is another random forest-based system that utilizes not only users, friends, and network features but also temporal features to capture the credibility of tweets [30]. Though potentially resulting in high accuracy, supervised learning relies on the selection of features that can best predict credibility.

Deep learning approaches, nevertheless, manage to classify with model-generated representations on datasets. Ma et al. trained models with recurrent neural networks (RNN), long short-term memory (LSTM), and gated recurrent units (GRU) architecture models [33]. A recurrent neural network (RNN) is a common architecture for the deep learning approach by processing the input words and extracting features through hidden states. The authors also employed LSTM and GRU to capture the patterns and hidden states throughout the life cycle of associated events in the inputs. Combining RNN and LSTM, Ruchansky et al. proposed a hybrid model that captures fake news representations as well as user behaviors for the classification of fake news on Twitter and Weibo [34]. The approach is also robust against limited labeled data.

The selected dataset [4] has been used by multiple papers for performance analysis. [35] have proposed a clustering technique to distinguish genuine users from the bots. The authors employ various features extracted from Twitter user profiles, such as the number of followers, friends, tweets, and user description, as input to the clustering algorithm. [36] propose a technique that captures and represents users' online behaviors by considering sequences of activities and interactions, similar to how DNA represents genetic information. They used this representation, also called "Behavioral DNA", and used it to create profiles that reflect the behavior patterns of both legitimate users and spam bots. The paper [30] describes the methodology used in the development of

	Cresci et al. [36]	Miller et al. [35]	Botometer [30]	Yang et al. [37]
Accuracy	0.402	0.520	0.959	0.984
F1 Score	0.292	0.473	0.973	0.989

Table 1: Performance comparison for benchmark testing [5]

Botornot, which involves a combination of machine learning techniques and analysis of various account features and behaviors. The system utilizes a large dataset of Twitter accounts, with each account assigned a score indicating the likelihood of being a social bot. The paper [37] presents a data selection approach for scalable and generalizable social bot detection. By leveraging active learning techniques and prioritizing informative instances, the method reduces the reliance on labeled data while maintaining or improving detection performance. The research contributes to the field of social bot detection and offers insights into addressing the challenges of scalability and generalizability in bot detection methods. These methodologies were used by [5] for testing on the proposed dataset "TwiBot20" along with other benchmark datasets - cresci-2017 and PAN-19. Below are the observed results from [5].

CHAPTER 3

Methodology

A lot of real-world problems in social networks, biomedical, the world wide web, and networking can be defined in terms of graphs. Recent research work has been on restructuring neural networks to be used for graph datasets. Social networks involve users, their posts, media shared on the internet, and their connections. This research focuses on the user posts and the connections shared. Classical Machine Learning approach experimentation uses the tweets posted by one user and their word embeddings, and provides a prediction of whether the user is a bot or a real user. This project proposes a novel hybrid method that addresses the respective limitations of NLP and neural networks as individual tools for text classification. This hybrid method will involve word embeddings, which will then be passed on to the graph classifiers for classification purposes. The implementation is described below.

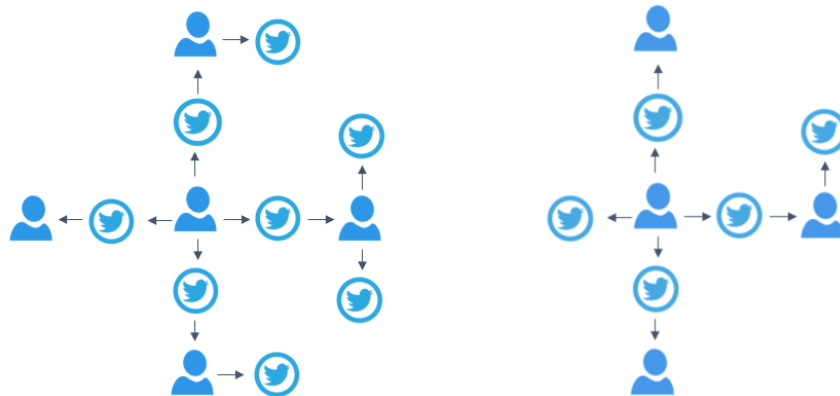


Figure 12: Graph Classification for Twitter Bot or User Classification

3.1 Twitter Data

Twitter is an online social media platform for users to post media and share among their connections. It can be categorized into two kinds of entities - tweets, and users. A Twitter profile for a user has its own metadata such as the number of followers, friend list, number of likes, number of tweets, profile summary, number of comments, and so on. Similarly, a tweet has metadata that contains attributes like user ID, in reply to user ID, tweet text, media shared, retweet count, comments count, etc. There are users connected to multiple other users and the connection grows with every user. These connections and attributes can be realized as a giant graph and can be utilized to study the social network.

3.2 Phases in Implementation Plan

The implementation plan is described in Figure 13.

The explanation of each phase is as below -

- Phase 1 - The original dataset Cresci-2017 [4] contains two folders - genuine users and bot users. Create a dataset combining the separate datasets and label them as 0 for the bot and 1 for the real user.
- Phase 2 - The tweets posted by the user and the user description can contain a lot of information that is not useful for the model. For example, removing unnecessary URLs, and mentions in the tweets. Using the standard Python NLTK functionalities, perform pre-processing on the original text data. After pre-processing, use NLP pre-trained models BERT [2] and spaCy [3] for extracting word embeddings for the text.
- Phase 3 - Using the embeddings obtained in Phase 2, train and test the Machine Learning models - KNN, SVM, and Random forest for classical machine learning

Phase 1

Perform data labelling and group original dataset of users into bot and real users.

Phase 2

Perform pre-processing of tweets and extract word embeddings using NLP pretrained models.

Phase 3

Obtain baseline results using classical ML algorithms

Phase 4

Perform dataset augmentation and generate graphs using the graph construction algorithm

Phase 5

Perform comparison and analysis between different methods and define future scope.

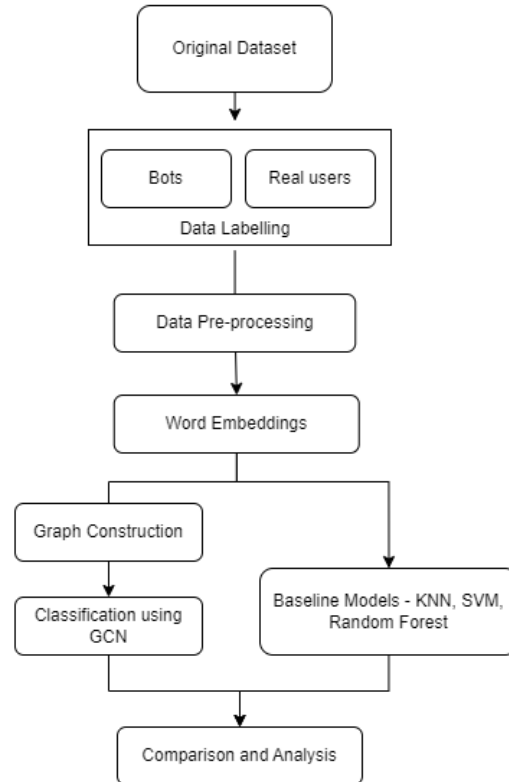


Figure 13: Phases in Implementation

approach results.

- Phase 4 - Using the base user and the corresponding tweets, augment the dataset with level 2 user nodes and tweet nodes using real-time Twitter web-scraping.
- Phase 5 - Design and utilize a homogeneous GCN model for graph classification purposes.
- Phase 6 - Using the classical machine learning approach results and GCN results, perform analysis and performance of the GCN method and define the future scope.

3.3 Dataset Augmentation

The original dataset considered for experimentation is Cresci-2017. This dataset consists of folder structure as shown in Figure 14. `users.csv` file contains the user details - user ID, user description, number of followers, number status, and number of friends. The `tweet.csv` file contains - the tweet ID, user ID of the user who posted the tweet, and user name of the user for which the tweet was posted (*in_reply_to_userID*), and the actual tweet text. Using the *in_reply_to_userID* field as input to Python library `snsrape`, a list of tweets and their attributes including user details are obtained. Using `snsrape`, level 2 of user nodes and tweet nodes are added to the original graphs.

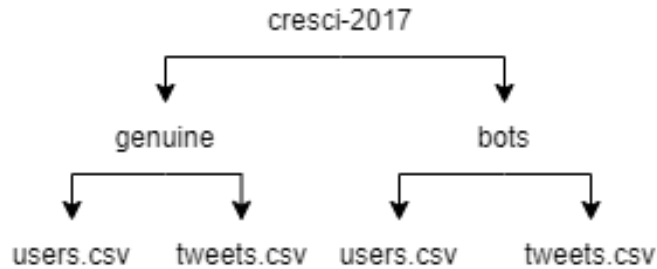


Figure 14: Dataset folder structure

3.4 Creation of Graphs

Graph generation and dataset augmentation are done hand in hand. Firstly, nodes for the user, tweets and *in_reply_to_userID* are created. This gives us the graph with level 1 of tweets. Using *in_reply_to_userID*, further tweets are augmented, and a node for every tweet is created and added to the graph. This provides the level 2 tweets. The graph construction is performed in a Breadth First Search manner as described in the algorithm below.

Using the Cresci-2017 [4] dataset, a level 1 graph as shown in Figure 15 can be

Algorithm 1 Construction of Graphs using dataset augmentation

Require: user.csv, tweet.csv

Ensure: List of graphs G

```
1:  $user\_df \leftarrow$  user dataframe
2:  $tweet\_df \leftarrow$  tweet dataframe
3:  $graphs \leftarrow$  empty list
4: for every user in  $user\_df$  do
5:    $G \leftarrow$  new graph()
6:    $G.add\_node(user)$ 
7:    $tweet\_data \leftarrow$  Fetch the list of tweets and corresponding in_reply_to_user
   from tweet dataframe
8:   for every tweet in  $tweet\_data$  do
9:      $G.add\_node(tweet)$ 
10:     $G.add\_node(in\_reply\_to\_user)$ 
11:     $G.add\_edge(user, tweet)$ 
12:     $G.add\_edge(tweet, in\_reply\_to\_user)$ 
13:   end for
14:    $graphs.add(G)$ 
15: end for
16: return  $graphs$ 
```

obtained. This graph contains the primary user node in the center connected to the tweets it has posted shown as yellow nodes. These yellow nodes (tweets) are connected to the user for whom this tweet was posted. The third level in this tree is the users to which the primary user has replied to.

Using the outer layer of nodes, level 2 of tweets is extracted using the web-scraping library - sncrape - provided by Python. The final generated graph with level 2 tweets looks as shown in Figure 16. The height of the final tree structure generated is 4. Method for graph construction using dataset augmentation is described in Algorithm 1.

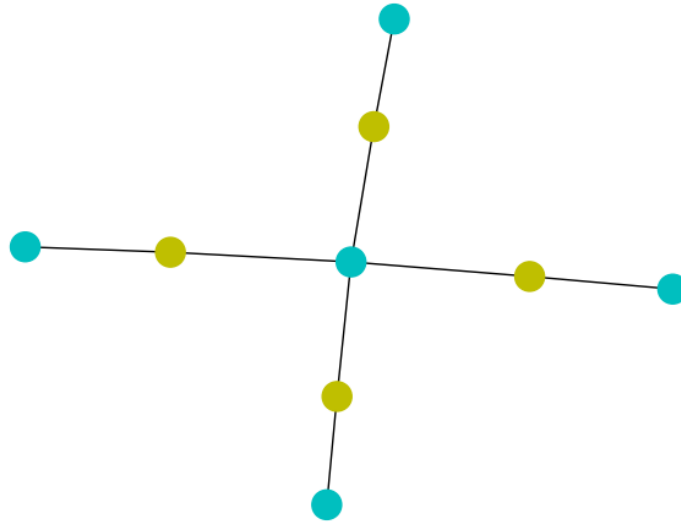


Figure 15: Graph construction - Level 1

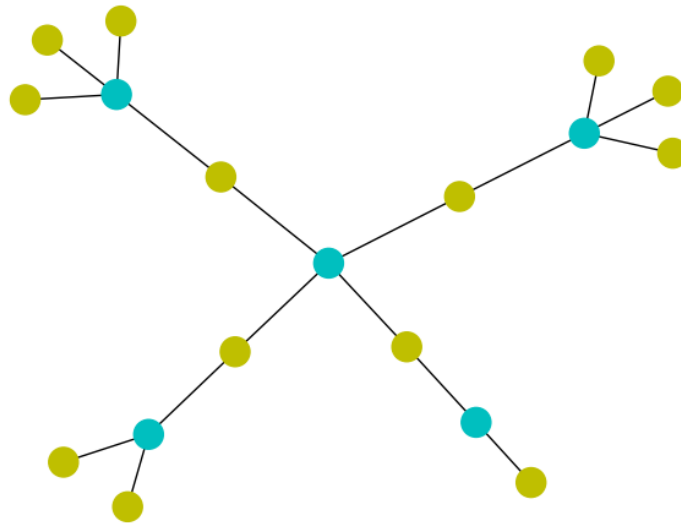


Figure 16: Graph Construction - Level 2

3.5 Classical Machine Learning Approach Architecture

The feature embeddings obtained using BERT and spaCy pre-trained models are grouped for every user. For example, BERT feature embedding for all tweets posted by user 1 will be combined as one feature vector. Similarly, one vector will be obtained for all tweets posted by user 1 for spaCy embeddings. These grouped features for every user will be used to train and test classical machine learning algorithms such as K-Nearest Neighbours, Support Vector Machines, and Random Forest. Figure 17 shows the architecture for classical machine learning models and Algorithm 2 describes the algorithm.

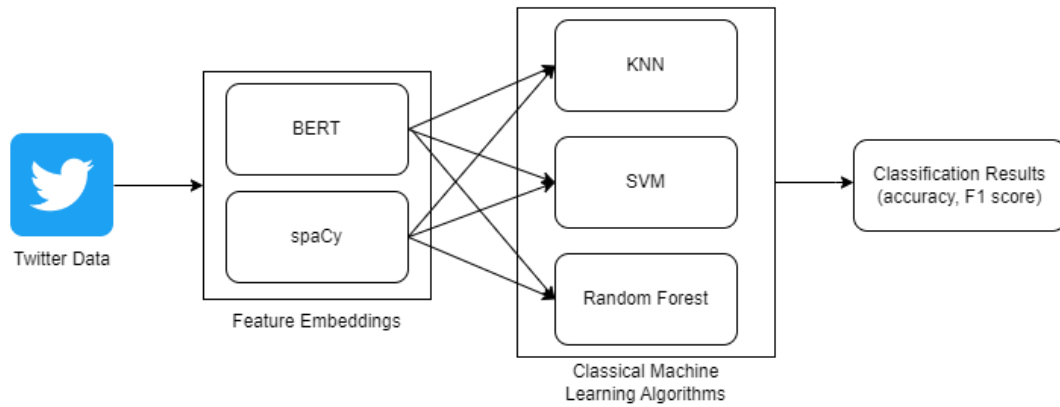


Figure 17: Achitecture Diagram for Classical Machine Learning Approach

Algorithm 2 Classification using Classical Machine Approach

Require: user.csv, tweet.csv

Ensure: Accuracy and F1 score for 3 ML algorithms

```
1: for every review in tweets.csv do
2:   Extract BERT feature
3:   Extract spaCy feature
4: end for
5:
6: for every user in tweets.csv do
7:   Group all tweets
8:   Aggregate all BERT features
9:   Aggregate all spaCy features
10:  Add user, bert_feature, spacy_feature, label in features.csv file
11: end for
12:
13: While features.csv is open:
14:
15: Split the dataset into train and test records in 80:20 ratio
16:
17: Define KNN model with k=5:
18:   Train with train dataset
19:   Test with test dataset and capture accuracy and F1 score
20:
21: Define Random Forest model with max_depth = 5:
22:   Train with train dataset
23:   Test with test dataset and capture accuracy and F1 score
24:
25: Define SVM model with linear kernel:
26:   Train with train dataset
27:   Test with test dataset and capture accuracy and F1 score
28:
29: Return: accuracies, F1 scores
```

3.5.1 K-Nearest Neighbours

This algorithm classifies the given object on the basis of its frequency in the given k samples during training. For example, in the below Figure 18, since red is in majority in 3 closest objects for the green object, it will be classified red when k is 3.

If k is 5, then it will be classified as blue as blue is in majority.

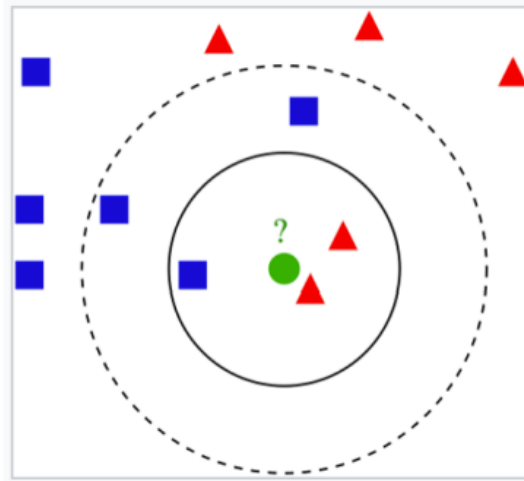


Figure 18: K-Nearest Neighbours [11]

3.5.2 Support Vector Machine

Support Vector Machine (SVM) is a supervised model that works very well on linear as well as non-linear classification problems. It uses kernels specific to the problem type and divides the input data into classes using a hyperplane. Figure 19 shows an illustration of SVM.

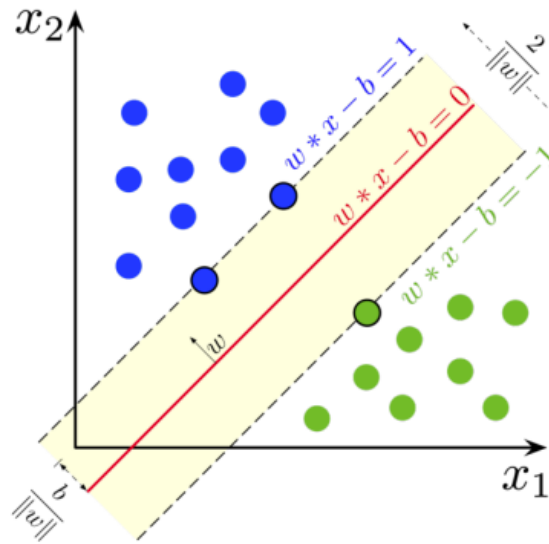


Figure 19: Support Vector Machine [12]

3.5.3 Random Forest Classifier

A random forest is a collection of several decision trees. Classification is done using the majority of the votes by the decision trees. The performance of the model enhances by bagging many decision trees together. Figure 20 shows an illustration of the Random Forest Classifier.

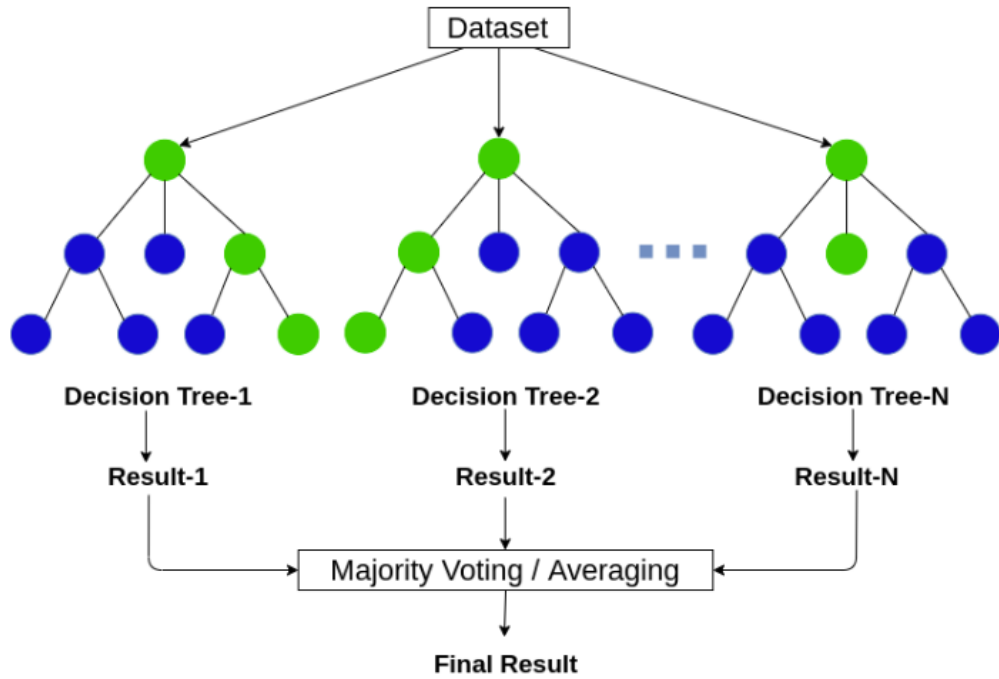


Figure 20: Random Forest Classifier [13]

3.6 GCN Model Architecture

In order to understand whether a Twitter user is real or a bot, we need to understand the context of the tweet and the user connections. The GCN approach overcomes the limitations of NLP and graph classification as individual tools. The preprocessed tweets and user descriptions obtained from the dataset and by performing web scrapping, are used to get the word embeddings from BERT and spaCy. Using these word embeddings and graph structure as input, the GCN is trained and tested in order to classify whether a graph belonging to a user is a real user graph or a bot graph.

The graph has two types of nodes - user node and tweet node. Since the classification is done using only text data, for user nodes, the user description is considered. The standard spaCy vector size is [96x1]. Hence, the feature vector size

for the user node as well as the tweet node is $[96 \times 1]$. Since the vector sizes are the same, homogeneous GCN can be used for classification purposes. Figure 21 illustrates the working of the GCN approach.

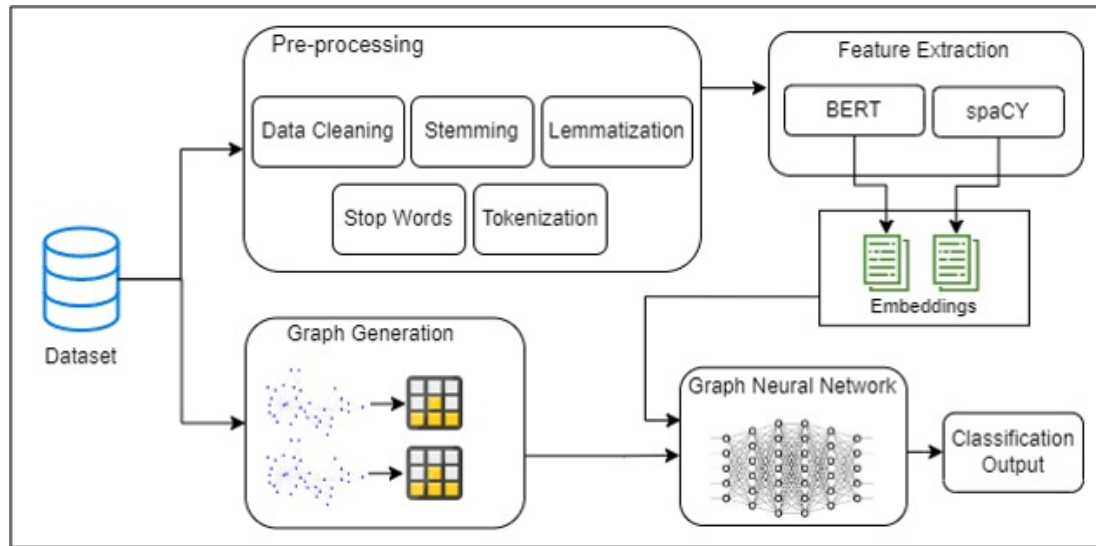


Figure 21: Achitecture Diagram for GCN methodology

CHAPTER 4

Experimental Evaluation

The evaluation of all the methods and models is done on the basis of their accuracies and F1 scores. Definitions for accuracy and F1 are -

- Accuracy - Accuracy is used to measure the performance of a model. It depicts the percentage of the correct predictions made by the model. It is the number of correct predictions over total predictions.

$$Accuracy = \frac{\text{number of correct predictions}}{\text{total number of predictions}} \quad (1)$$

- F1 Score - F1 score is another measure indicating the performance of a model. It presents the harmonic mean given precision and recall. Precision is the number of true positives among all positive predictions. The recall is the true positive predictions made by the model among all actual positive dataset instances.

$$F1score = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (2)$$

4.1 Dataset Description

The Cresci-2017 dataset [4], which is openly accessible, includes a sizable number of Twitter accounts, as well as the tweets and information that go with them.

The metadata for users contain - *id, name screen_name, statuses_count, followers_count, friends_count, favourites_count, time_zone, location, description, created_at, timestamp.*

The metadata for tweets contain - *id, text, source, user_id, truncated, in_reply_to_status_id, in_reply_to_user_id, in_reply_to_screen_name, retweeted_status_id, geo, place, contributors, retweet_count, reply_count, favorite_count, favorited, retweeted, possibly_sensitive, num_hashtags, num_urls, num_mentions, created_at, timestamp, crawled_at, updated*

The dataset is intended for use in studies on bot detection, social media manipulation, and fake news. The phony accounts are grouped into four groups: social spambots, social bots, social malware, and fake followers. It contains both real and fake accounts. The collection contains tweets over the course of more than four years, from 2013 to 2017.

Examples of tweets posted by genuine users -

- @bookmeister @studiomondeo nope. Thanks for giving it a shot
- The last video game I was good at: <http://t.co/tla2MOGccF>
- Picked up another spectacular middle-aged accomplishment: slept through the night last night! #newsuperpower#fb

Examples of tweets posted by spam bot users -

- I posted a new photo to Facebook <http://t.co/q4I4ssFXMr>
- <http://t.co/1RabKjTA68>
- <http://t.co/FhFgGSDuH7>

4.2 Innovative and challenging Aspects

The GCN approach is a novel hybrid method for classifying whether a Twitter user is a bot or a real user based on the profile summary and the tweet text posted by that user. The *cresci-2017* [4] dataset contained the folder structure as shown in

Figure 14. The CSV files were pre-processed and merged with the appropriate labels (bot and real user) for further processing. To increase the depth of the graph generated for one user, dataset augmentation is performed to fetch the user and tweet details in real-time for graph construction. Feature vectors for the graph nodes are generated using BERT [2] and spaCy [3] embeddings. Lastly, a GCN is created and trained on the graphs for classification purposes. This way, the GCN approach combines both, NLP techniques and graph classification.

4.3 Graph Properties

To study the graph properties for the graphs generated for bot users and real users, their densities were studied. Density measures the connections that exist between the nodes in comparison to the number of connections that could be possible between nodes. We measure the average, maximum, and minimum densities for bot users and real users. Table 2 describes the results obtained for densities of the graphs for bot and real users.

	Bot user	Genuine User
Average graph density	0.0313	0.0326
Maximum density	0.0124	0.01415
Minimum density	0.1090	0.1090

Table 2: Statistics for graph densities observed

4.4 Classical Machine Learning Approach

There are multiple tweets per user. In order to create a single feature vector for every user, all the tweets are combined and grouped by the user ID. This condensed the dataset of 57054 rows into 804 rows where we have a user ID, feature vector, and label as columns.

The dataset considered for this experimentation is described in Table 3.

Number of Users	804
Number of Tweets	57054
Bot Users	437
Real Users	367

Table 3: Dataset description for classical machine learning approach

Since the feature vector obtained using BERT and spaCy are highly dimensional, the classical machine learning algorithms cannot process these vectors. To reduce the dimensionality of the features, Principal Component Analysis (PCA) was performed with n equals 10. Now the features are ready to be used by Machine Learning algorithms. Both, BERT and spaCy feature vectors were used separately for classification purposes. Results obtained using spaCy features and BERT features are mentioned separately in Table 4 and Table 5 separately. Figure 22 depicts the accuracies and Figure 23 depicts the F1 scores for the same results.

	Accuracy	F1 Score
KNN	0.888	0.887
Random Forest	0.882	0.881
SVM	0.900	0.900

Table 4: spaCy results for classical machine learning approach

	Accuracy	F1 Score
KNN	0.894	0.892
Random Forest	0.863	0.861
SVM	0.900	0.899

Table 5: BERT results for classical machine learning approach

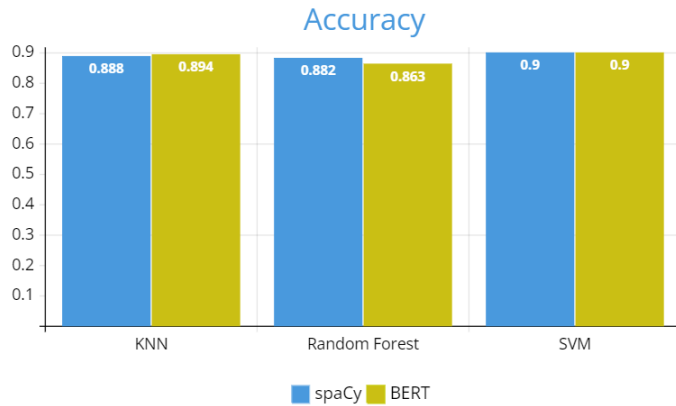


Figure 22: Accuracies for classical machine learning approach

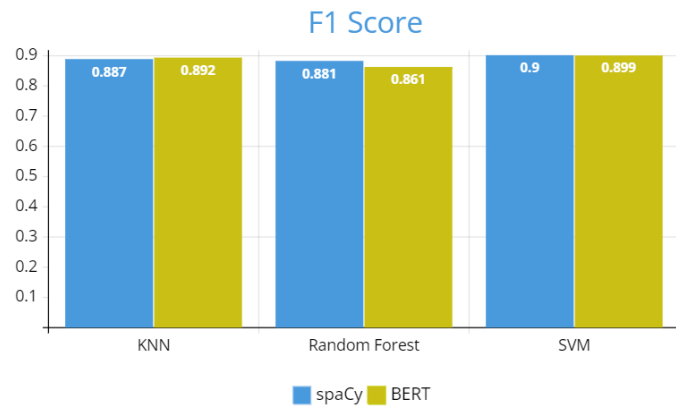


Figure 23: F1 scores for classical machine learning approach

4.5 GCN Approach Results

For this methodology, the original dataset was augmented with the tweet nodes obtained using web scraping. The data augmentation was done in real time and at the same time, feature vectors were generated. Since storing and processing the data was heavy, the run-time solution was used.

The hyperparameters used to create a homogeneous GCN model are described in 6.

For every user, one graph was created. The entire dataset is shuffled and

Number of layers	3
Aggregation for GCN	tanh
Activation Function	log_softmax
Number of epochs	10
Learning Rate	0.01
Optimizer	Adam

Table 6: Hyper-parameters for GCN

partitioned into train and test datasets in an 80:20 ratio. The GCN model was trained for every graph and loss per epoch was captured. Loss is used as a criterion for backward propagation. For every graph, the loss per epoch was observed to reduce. Sample graphs for loss are shown in Figure 24. The training and testing classification results for the GCN approach are described in Table 7

	Training	Testing
Accuracy	0.898	0.721
F1 Score	0.915	0.758

Table 7: Results using GCN

As a part of hyper-parameter tuning, the activation functions were tested. They were - relu and tanh. Relu maps the negative values to 0 and positive values to positive values, and tanh maps the values between -1 to 1 thus allowing negative values. Since the embeddings obtained from spaCy and BERT had negative values, it is important that those negative values are preserved. The embeddings for every word represent how that word is situated in a particular sentence. If the embedding is negative, it can mean that the word is not representing the same meaning as the sentence. Having negative values will provide better values and the same was observed by changing the activation function from relu to tanh. Accuracy results are shown in Table 8.

	relu	tanh
Accuracy	0.645	0.762
F1 Score	0.656	0.798

Table 8: Hyper-parameter tuning of activation function

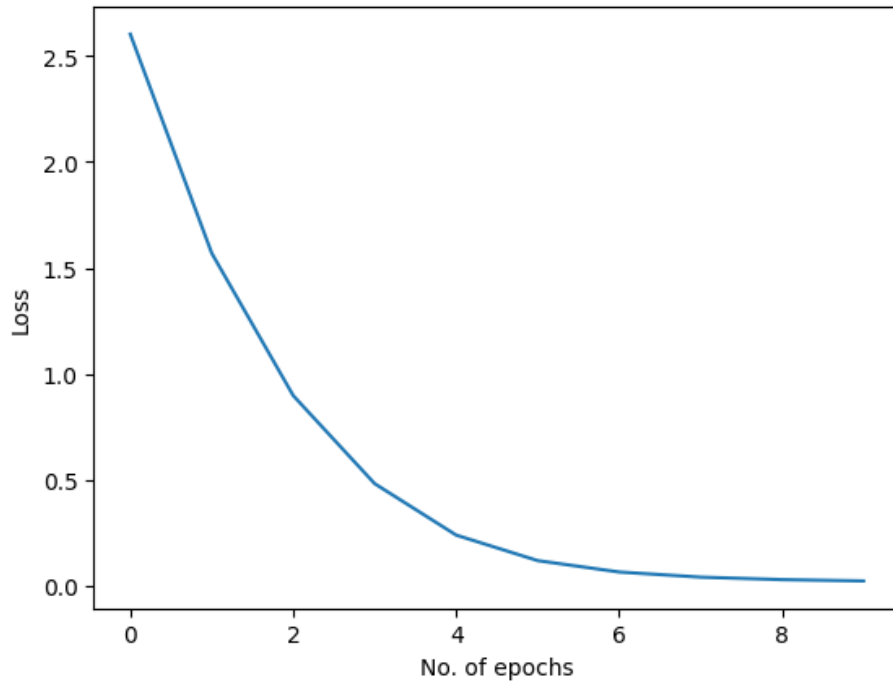


Figure 24: Loss observed per epoch

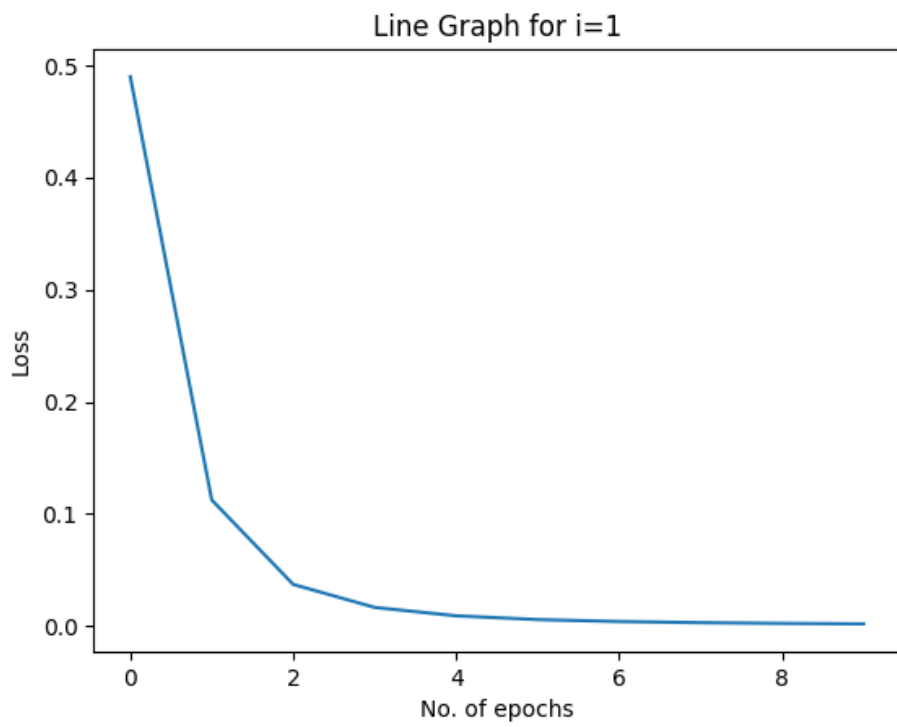


Figure 25: Loss observed per epoch

CHAPTER 5

Conclusion and Future Work

This research work has successfully proposed and proved the working of a hybrid model of NLP and graph classification to classify a Twitter user as a bot or a real user. This methodology uses tweets and the first level of connections for the user. The GCN methodology comes with some assumptions. The level 1 tweet should be posted by the user as a response to a particular user. This provides the ability to create a response network between users connected through the tweets. Since this methodology utilizes only text data as input - tweets and user description - it is recommended that the user should have a description or a profile summary as the feature embedding for an empty description will be an empty tensor array, which might not be very helpful for the model in training.

The research in graph-based real-world problems and the databases corresponding to that has started to evolve recently, making machine learning and deep learning more generalized to accommodate graph-based experimentation as well. The classical machine learning approach results are better than the results obtained using the GCN hybrid approach. The primary reason for this is having less data for training and testing the model. Recently, [38] developed a huge graph-based Twitter dataset. As the next steps, using the TwiBot dataset, and extra processing power, the proposed hybrid model can be trained and tested.

The GCN approach utilizes the features obtained only from the text data. There are other properties of the user node such as - number of friends, number of followers, and number of tweets - that can be leveraged. For tweets, the number of retweets and

number of replies can also be leveraged. However, having these features in addition to the embedding vector will lead to a change in the feature set dimension. [39] recently proposed an approach for graphs with multiple node types and edge types having different feature sets and edge weights. The graphs generated are heterogeneous in nature and to classify such graphs, heterogeneous GCN can be used. The GCN methodology can also be used to perform the analysis of fake news spread. Propagation graphs can be created for every tweet. These graphs will depict how the tweet is being spread across Twitter users. Similar features for user and tweet nodes can be considered for classification purposes.

The GCN approach can be extended to other social network sites such as TikTok, Facebook, and LinkedIn. This will provide a better understanding of how fake news is propagated and the source of it. It will also help to understand which social media platform is used by a society or an individual the most for influence maximization.

LIST OF REFERENCES

- [1] M. Heidari, J. H. J. Jones, and O. Uzuner, “Online user profiling to detect social bots on twitter.”
- [2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” no. arXiv:1810.04805, May 2019, arXiv:1810.04805 [cs]. [Online]. Available: <http://arxiv.org/abs/1810.04805>
- [3] M. Honnibal and I. Montani, “Efficient natural language processing with spacy,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017, pp. 69–74.
- [4] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race,” in *Proceedings of the 26th International Conference on World Wide Web Companion*. ACM, 2017, pp. 963–972.
- [5] S. Feng, H. Wan, N. Wang, J. Li, and M. Luo, “Twibot-20: A comprehensive twitter bot detection benchmark,” *CoRR*, vol. abs/2106.13088, 2021. [Online]. Available: <https://arxiv.org/abs/2106.13088>
- [6] D. Team, “5.4 graph classification,” <https://docs.dgl.ai/en/0.8.x/guide/training-graph.html>, 2018.
- [7] M. Blog, “How the embedding layers in bert were implemented,” https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a, 2019.
- [8] spaCy documentation, “Embeddings, transformers and transfer learning,” <https://spacy.io/usage/embeddings-transformers>.
- [9] W. Liang, J. Jin, I. Daly, H. Sun, X. Wang, and A. Cichocki, “Novel channel selection model based on graph convolutional network for motor imagery,” *Cognitive Neurodynamics*, pp. 1–14, 10 2022.
- [10] M. Blog, “Knowing your neighbours: Machine learning on graphs,” <https://medium.com/stellargraph/knowing-your-neighbours-machine-learning-on-graphs-9b7c3d0d5896>, 2019.
- [11] “k-nearest neighbors algorithm,” https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm, 2023.

- [12] “Support vector machine,” https://en.wikipedia.org/w/index.php?title=Support-vector_machine&, 2021.
- [13] “Random forest algorithm explained,” <https://anasbrital98.github.io/blog/2021/Random-Forest/>.
- [14] L. Ha, L. Andreu Perez, and R. Ray, “Mapping recent development in scholarship on fake news and misinformation, 2008 to 2017: Disciplinary contribution, topics, and impact,” *American Behavioral Scientist*, vol. 65, no. 2, p. 290–315, Feb 2021.
- [15] “Accidental poisonings rise after trump disinfectant comments | time.” [Online]. Available: <https://time.com/5835244/accidental-poisonings-trump/>
- [16] G. Pennycook, J. McPhetres, Y. Zhang, J. G. Lu, and D. G. Rand, “Fighting covid-19 misinformation on social media: Experimental evidence for a scalable accuracy-nudge intervention,” *Psychological Science*, vol. 31, no. 7, p. 770–780, Jul 2020.
- [17] “Facebook to label misleading politicians’ posts, including trump’s - los angeles times.” [Online]. Available: <https://www.latimes.com/business/technology/story/2020-06-26/facebook-following-twitter-will-label-posts-that-violate-its-rules-including-trumps>
- [18] M. G. Seenappa, K. Potika, and P. Potikas, “Short paper: Graph classification with kernels, embeddings and convolutional neural networks,” in *2019 First International Conference on Graph Computing (GC)*, Sep 2019, p. 88–93.
- [19] N. Rosenfeld, A. Szanto, and D. C. Parkes, “A kernel of truth: Determining rumor veracity on twitter by diffusion pattern alone,” in *Proceedings of The Web Conference 2020*, ser. WWW ’20. New York, NY, USA: Association for Computing Machinery, Apr 2020, p. 1018–1028. [Online]. Available: <https://doi.org/10.1145/3366423.3380180>
- [20] S. Vosoughi, D. Roy, and S. Aral, “The spread of true and false news online,” *Science*, vol. 359, no. 6380, p. 1146–1151, Mar 2018.
- [21] B. Anderson, “Graph-based malware detection using dynamic analysis,” *Journal in Computer Virology*, vol. 7, no. 4, pp. 247–258, 2011.
- [22] T. Suzumura, D. Garcia-Gasulla, S. A. Napagao, I. Li, H. Maruyama, H. Kanezashi, R. P’erez-Arnal, K. Miyoshi, E. Ishii, K. Suzuki, S. Shiba, M. Kurokawa, Y. Kanzawa, N. Nakagawa, M. Hanai, Y. Li, and T. Li, “Global data science project for covid-19,” no. arXiv:2006.05573, Aug 2021, arXiv:2006.05573 [physics]. [Online]. Available: <http://arxiv.org/abs/2006.05573>

- [23] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, “Stanza: A python natural language processing toolkit for many human languages,” no. arXiv:2003.07082, Apr 2020, arXiv:2003.07082 [cs]. [Online]. Available: <http://arxiv.org/abs/2003.07082>
- [24] P. Meesad, “Thai fake news detection based on information retrieval, natural language processing and machine learning,” *SN Computer Science*, vol. 2, no. 6, p. 425, Aug 2021.
- [25] M. Borhani and H. Ghassemian, “Hyperspectral image classification based on spatial graph kernel,” in *2014 22nd Iranian Conference on Electrical Engineering (ICEE)*, May 2014, p. 1811–1816.
- [26] A. Mahboubi, L. Brun, and F.-X. Dupé, “Object classification based on graph kernels,” in *2010 International Conference on High Performance Computing & Simulation*, Jun 2010, p. 385–389.
- [27] S. Helmstetter and H. Paulheim, “Weakly supervised learning for fake news detection on twitter,” in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018, pp. 1029–1036.
- [28] I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, “Fake news detection using machine learning ensemble methods,” *Complexity*, vol. 2020, pp. 1–11, 2020.
- [29] Y. Chen, N. J. Conroy, and V. L. Rubin, “Misleading online content: Recognizing clickbait as ‘false news’,” in *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*. ACM, 2015, pp. 15–19.
- [30] 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*. ACM, 2016, pp. 273–274.
- [31] S. Afroz, M. Brennan, and R. Greenstadt, “Detecting hoaxes, frauds, and deception in writing style online,” in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 461–475.
- [32] C. Castillo, M. Mendoza, and B. Poblete, “Information credibility on twitter,” in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 675–684.
- [33] J. Ma, W. Gao, Y. Wei, X. Lu, and K.-F. Wong, “Detecting rumors from microblogs with recurrent neural networks,” in *IJCAI*, vol. 16, 2016, pp. 3818–3824.
- [34] N. Ruchansky, S. Seo, and Y. Liu, “Csi: A hybrid deep model for fake news detection,” in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 797–806.

- [35] 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.
- [36] S. Cresci, R. DiPietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “Dna inspired online behavioral modeling and its application to spam bot detection,” *IEEE Intelligent Systems*, vol. 31, no. 5, pp. 58–64, 2016.
- [37] K.-C. Yang, O. Varol, P.-M. Hui, and F. Menczer, “Scalable and generalizable social bot detection through data selection,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 1096–1103.
- [38] 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,” 2023.
- [39] R. Ragesh, S. Sellamanickam, A. Iyer, R. Bairi, and V. Lingam, “Hetegen: Heterogeneous graph convolutional networks for text classification,” 2020.