Drug Abuse Detection in Twitter-sphere: Graph-Based Approach

Khaled Mohammed Saifuddin
Department of Computer Science
Oklahoma State University
Stillwater, OK 74078, USA
khaled_mohammed.saifuddin@okstate.edu

Muhammad Ifte khairul Islam
Department of Computer Science
Oklahoma State University
Stillwater, OK 74078, USA
ifte.islam@okstate.edu

Esra Akbas
Department of Computer Science
Oklahoma State University
Stillwater, OK 74078, USA
eakbas@okstate.edu

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671532

***Abstract*—The rate of non-medical use of opioid drugs has increased markedly since the early 2000s. Due to this non-medical use, abusers suffer from different adverse effects that include physical and psychological problems. Many studies have been done to detect Drug Abuse (DA) events from social media data using machine learning and deep learning concepts. Moreover, Graph Neural Networks (GNNs) have recently become popular in text classification tasks due to their high accuracy and capability to handle complex structures. In this work, we collect drugs-related Twitter data (tweets) and build text graphs (corpus-level and document-level) to capture word-word, document-word, and document-document relations. Then we apply different GNN models on those text graphs and thus turn the text classification task into a node classification (for corpus-level graph) and graph classification (for document-level graph) task to detect DA events. Finally, we compare our graph-based DA detection models with different types of baselines models, including rule-based, traditional machine learning, and deep learning models. Our result shows graph-based models outperform the traditional machine learning and deep learning-based models.**

***Index Terms*—Opioid, Drug Abuse, Corpus-level Text graph, Document-level Text graph, Graph Neural Network**

I. Introduction

Non-medical use of prescription drugs and illicit drugs is increasing rapidly over the nation [1]. Drug Abuse Warning Network (DAWN) published a report claiming that in the year 2011, more than 5 million patients related to Drug Abuse (DA) and misuse visited the emergency department [2]. Despite this severe crisis, getting public data in real-time is a big problem. Also, there is a lack of surveillance systems to detect and monitor opioid DA except for the survey-based system like National Poisoning Data System (NPDS) [3], Centers for Disease Control and Prevention (CDC), and DAWN, etc. that are failed to report in real-time. On the other hand, social media have become a significant platform for information sharing. Drug-addicted people willingly share their own problems and personal things on social media, i.e., Twitter, Reedit, etc. Therefore social media provides real-time data that helps us to monitor and detect DA events in real-time [4].

However, it is not possible to manually curate information from large volumes of data. A number of recent studies have focused on automatically detecting drug use/abuse in social media [2], [4]–[6]. Large amounts of noise and the use of non-standard languages and terms create a challenge to work on this data. Some researchers detected DA events using a rule-based concept with a dictionary of words [5]. However, this may not be efficient due to the huge use of informal and slang terms on social media by users. Moreover, some researchers have considered this problem as a text classification problem and explored different machine learning models in this domain [6]–[9]. However, the selection of the best features (e.g., count vectors, TF-IDF vectors, bag-of-words, n-grams, etc.) is often a very challenging task due to the presence of huge variations of the social text.

As a solution to this problem, deep learning models, which learn features automatically, have been used widely for different text classification tasks, including DA [6], [10]–[13]. These deep learning models can capture semantic relation in consecutive word sequences but may fail in the case of global word co-occurrences when there are very long-distance dependencies. To solve these issues, different Graph Neural Network (GNN) based models have been proposed recently for text classification [14]–[16]. However, those GNN models have not been used for DA related text classification problems.

In this work, to detect DA events from Twitter text data, we build text graphs (corpus-level and document-level) from tweets to capture word-word, document-word, and document-document relations. Then we apply text-based different GNN models on those text graphs and thus turn the text classification task into a node classification (for corpus-level graph) and graph classification (for document-level graph) task. As per our knowledge, this is the first work to detect DA events from Twitter data using GNN. We work on detecting illicit and prescription DA-related tweets. We focus on opioid drugs that include both prescription drugs and illicit drugs as well [8] and collect related tweets. A list of commonly abused illegal schedule 1 drugs like heroin and schedule 2 prescription drugs like fentanyl, oxycontin, etc., are used to collect tweets on Twitter. [8].

We present different GNN-based models for DA-related text classification on Twitter data. We construct different types of text graphs; (1) corpus-level text graphs and (2) document-level text graphs from tweets. As the corpus-level text graphs,

we create a single graph from all tweets of the corpus. First, we use document-word co-occurrence relations to create a bipartite text graph that includes document and word nodes. Then we convert the bipartite text graph into Homogeneous Corpus-level Text (HoCT) graph and Heterogeneous Corpus-level Text (HeCT) graph. We create another corpus-level text graph based on the similarity between documents (i.e., Similarity-based Corpus-level Text (SCT) graph). After representing each tweet as a node of the graph, we add edges between them based on their similarity calculated with Word2Vec representations of tweets. As the document level, for each tweet, we create a text graph that contains nodes as words and word-word edges based on the co-occurrence of words in the tweet. First, we obtain edges from the local adjacency of words which is the adjacency within the same document and create Local Document-level Text (LDT) graph. We also create a global word adjacency matrix from the whole corpus and use it to obtain word-word edges in each document graph that produce Global Document-level Text (GDT) graph.

After creating these different types of text graphs, we extract the features of documents from them. Then, we use extracted features to detect DA-related tweets, thus turning the DA-related text classification task into a node classification and graph classification problem. For this goal, we apply different GNN models on those constructed graphs individually. GNN models are trained as a semi-supervised classification using cross-entropy loss.

To demonstrate the advantages of graph-based models for the DA problem, we evaluate its performance with our Twitter dataset collected with a list of commonly abused illegal and prescription drugs such as heroin, fentanyl. We compare our results with different types of baselines models, including rule-based models, traditional machine learning models, and deep learning models. On the DA detection task, graph-based models, especially SCT graph with Graph Convolution Network (GCN), outperforms all other baseline approaches and achieve state-of-the-art performance. A summary of our contributions is provided below:

- We propose a graph-based model for DA detection problem with converting DA-related text classification task into a node classification and graph classification problem.
- We create different types of text graphs as document-level and corpus-level text graphs.
- We apply different GNN models on the created text graphs and use them for node classification and graph classification
- We present extensive performance comparisons of several baseline model learning, including rule-based, traditional machine learning, and deep learning models using a collected Twitter DA dataset.

The rest of the paper is organized as follows. In section II, related works are presented. In section III, we discuss our graph-based DA detection system. Experiments and results are presented in section IV. In section V, we present the conclusion.

II. LITERATURE REVIEW

A. DA detection from online text data

Jenhani et al. [5] propose a lexicon-based approach to extract DA entities from Twitter data. They build a dictionary of words related to DA and then extend the Standford CoreNLP pipeline to annotate the tweet text. Though drug-related vocabulary is always changing over the internet, they fail to provide any model to update the dictionary automatically.

To automate and to get better accuracy supervised machine learning models are used widely in text classification tasks, particularly in online medical-related text data [4], [7], [9], [17]. Sarker et al. [9] present an approach to perform localized surveillance of opioid abuse using machine learning models. They extract word n-grams and abuse indicating terms as features from tweets and explore different machine learning-based models to detect opioid abuse. Kim et al. [7] analyze the non-medical use and side effects of methylphenidate on Twitter. They exploit non-medical use terms, medical use terms, and side effects of methylphenidate as features. Moreover, they use personal nouns as another feature in the machine learning classifier, which indicates whether tweets are coming from first-hand experience or not.

As annotating social media data is very challenging due to the presence of informal and vague languages, Hu et al. [6] come up with an approach named deep self-taught learning to automatically annotate a big corpus. As Twitter users post on varieties of topics, it becomes difficult to detect DA-related tweets properly. Moreover, it creates data imbalance problems, especially when tweets include abuse-related slang terms. To solve this data imbalance problem, Hu et al. [10] propose an ensemble deep learning model that leverages both word-level CNN and character-level CNN to classify abuse-related tweets.

B. Graph Neural Networks-based text classification

Recently GNNs have been used widely in Natural Language Processing (NLP) tasks like sentiment classification [18], [19], text classification [15], [16], question-answering [20], [21]. Kipf et al. [22] propose GCN for the node classification task; later Yao et al. [14] turn the node classification task into a text classification task using GCN. Moreover, different modified GCN like tensor graph convolution networks [23], and heterogeneous graph convolutional neural networks [24] are also proposed for text classification. Huang et al. [15] use a text-level GNN model for text classification. They represent each text as a graph where each node is represented by aggregating the neighbor's information using a message passing mechanism. Finally, they use a readout function to do the graph classification (i.e., text classification). TextING [16] also represents each text as a graph and exploit gated graph neural network architecture to get the word level embedding and use max pooling function to pool the word level embedding to graph level for text classification. However, these GNN models has not been applied to DA related text classification problem.

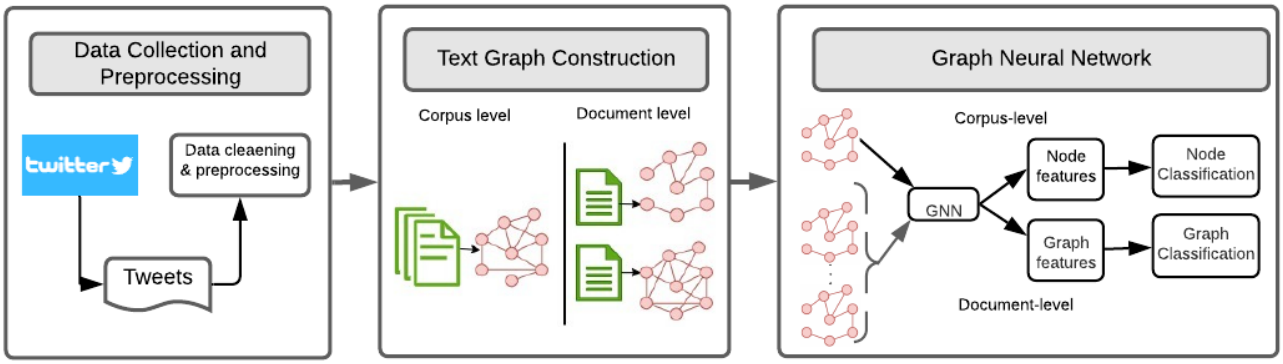

Fig. 1: The outline of the proposed graph-based DA detection system.

III. Graph-Based DA Detection System

In this section, we present our graph-based DA detection system on tweets. First, we collect our data from Twitter using drug-related keywords, annotate and apply preprocessing steps. Then, we create different types of graphs from tweets and apply different GNN-based models to decide whether a tweet is a DA-related tweet. As illustrated in Fig. 1, our proposed system includes the following steps; (1) data collection, annotation, and preprocessing, (2) text graph construction, (3) feature extraction with GNN, and (4) DA classification as a means of node or graph classification task.

A. Data collection, annotation and preprocessing

To create our dataset, we collect raw tweets through Twitter APIs. A list of commonly abused illegal schedule 1 drugs like heroin and schedule 2 prescription drugs like fentanyl, oxycontin are used to collect tweets on Twitter. Moreover, people very often use slang and street terms on social media, which is not exceptional for drug-related posts. That is why, while collecting and analyzing data, we also consider the street terms of drugs. In Table I, we show the list of drug names with their street terms that are used to collect the tweets. These street terms are collected from some online platforms such as [1]Addiction Center, [2]DEA, [3]DRUGBANK.

On the other hand, just mentioning a drug name or drug street name, even an illicit drug on a Twitter post, does not always mean that the tweet is related to DA. To overcome this problem, while collecting Twitter data, a list of DA terms is used with the drug names. Some of them are given in Table II. A tweet that comes with any of the listed drug names and the DA term might imply that the tweet is related to DA. We also obtain DA-related Twitter data set from the authors of [4]. From the collected and obtained tweets, 1500 tweets are randomly chosen and annotated as DA and NDA (non-drug abuse) tweets by a group of health informatics expertise. Out of 1500 tweets, 380 are abuse-related (DA), and 1120 are non-abuse-related (NDA) tweets. Some instances of DA and NDA tweets are shown in Table III.

Twitter data is a short text and carries random characters, emoticons, links, retweets, and colloquial words [5]. It is essential to clean it before feeding it to the NLP and text mining tools. So, we preprocess the corpus properly. At first,

[1]https://www.addictioncenter.com
[2]https://www.dea.gov/
[3]https://go.drugbank.com/

TABLE I: Drug name with street terms

Drug name	Street terms
Heroin	Brown sugar, China White, Chiva Dope, Horse, Junk, Skag, Skunk, Smack, White Horse, big H
Marijuana	Weed, Pot, Reefer, Grass, Dope, Ganja, Mary Jane
Cocaine	Blanca, Crack, Flake, Big flake, Rail, Candy C, Stash
Fentanyl	Apace, Tango, Laced, Actiq, Duragesic, Sublimaze
Oxycontin	Kickers, Blues, Oxycotton, Hillbilly heroin, Oxy
LSD	Acid, Blotter Acid, Dots, Mellow Yellow, Window Pane
Codeine	Captain Cody, Cody, Little C, Schoolboy

TABLE II: List of DA indicating terms

Overdose, High, Addiction, Abuse, Feel, Need, Sniff, Swollen, etc.

TABLE III: Some examples of manually annotated tweets

Tweets	
Abuse	”Update I have overdosed on ritalin and monster and my skin is leaving my bones” ”im sure ive been drinking alcohol while on codeine tablets as well lol” ”Who has some cocaine they can let me borrow” ”I see it in ya face I got the good stuff cocaine pills whatever ya want”
Non-Abuse	”How Doctors And Big Pharma Helped Create N. Americas Fentanyl Crisis” ”Doctor prescribed benzodiazepines lead to physical dependence. Man forced to cold turkey dies” ”Dentists Are patients seeking prescriptions for benzodiazepines or opioids to sell them” ”historyepics Coca cola started out as wine with cocaine in it.”

we remove tweets with URLs and remove retweets, and then we utilize the NLTK [25] sentence and word tokenizers and lemmatize the output word tokens to convert different forms of the same word to its main format. Then we remove extra spaces, emojis, hashtags, mentions, punctuation, numbers, and stop words.

B. Text Graph Construction

An important step in a graph-based system is how one defines the nodes/relations and builds the graph from the data instances. There are different ways to create graphs from text. We construct two different types of text graphs from our corpus: corpus-level text (CT) graph and document-level text (DT) graph.

a) Corpus-level Text Graph

As the corpus level, we create one single graph from all texts in our corpus. We consider each document/tweet as nodes of the graph and define relations between them. Based on the definition of relations, we consider two types of CT graphs: co-occurrence which is based on common words of documents, and similarity, which is based on the similarity of documents' features.

Co-occurrence graph: Relations between documents shows their similarity. The similarity between a pair of documents can be assumed to arise from the words that co-occur in them, and we model this document-word co-occurrence relationship as a bipartite graph. A bipartite graph $G = (U, V, E)$, includes two sets of nodes U and V with one set of edges $E \subseteq U \times V$ between these two node set. At first, we construct a bipartite graph B from the whole corpus where its two types of nodes are unique words, and documents (tweets). We calculate the term frequency-inverse document frequency (TF-IDF) score between all documents and unique words, and if a document contains a word, we build an edge among that document and word and TF-IDF value, w is given to the edge as the weight of it. This bipartite graph B has only document-word edges. From this bipartite graph, we construct 2 different corpus-level text graphs, homogeneous and heterogeneous.

a.1. ***Homogeneous Corpus-level Text (HoCT) graph*:** HoCT graph, G_{HoCT}, is a weighted projection graph from the bipartite graph B. G_{HoCT} is a homogeneous graph with only document nodes. We add edges between document nodes if they have a common word node neighbor in bipartite graph B. We also assign the number of shared neighbors as a weight $weight_{i,j}$ to the edge between document i and j. Adjacency matrix of G_{HoCT} is defined as

$$A_{HoCT_{i,j}} = \begin{cases} weight_{i,j} & \text{number of shared neighbors} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

a.2. Heterogeneous Corpus-level Text (HeCT) graph: HeCT graph, G_{HeCT}, on the other hand, keep both word nodes and document nodes from bipartite graph. In addition to the document-word edges in the bipartite graph, we add word-word edges representing global word co-occurrence in the whole corpus. We also give weights to edges between two-word nodes. For word-word edges, we employ point-wise mutual information (PMI), which is a popular measure for word associations that achieves better results than word co-occurrence count. Adjacency matrix of G_{HeCT} is defined as

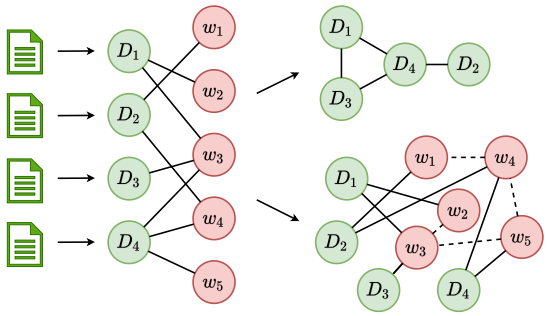

Fig. 2: Document-word bipartite graph (left) and its projection to a homogeneous corpus level text graph (right-top) and a heterogeneous corpus level text graph (right-bottom).

$$A_{HeCT_{i,j}} = \begin{cases} TF-IDF_{i,j} & \text{where i is document and j is word} \\ PMI_{i,j} & \text{where i and j both are words and } PMI_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $PMI_{i,j} = log(\frac{p(i,j)}{p(i)p(j)})$ with $p(i,j)$ as the probability of co-occurrence of word i and j in a fixed size window and $p(i)$, $p(j)$ as the individual probability of occurrence of i and j in a fixed-size sliding window.

We present examples for bipartite graph as CT graph and projection of it into homogeneous corpus-level text graph G_{HoCT} (top) and heterogeneous corpus-level text graph G_{HeCT} (bottom) in Fig. 2. This G_{HoCT} has only document nodes. Document nodes share edges between them if they have common words. And, G_{HeCT} has word and document nodes with word-word and document-word edges.

a.3. Similarity-based Corpus-level Text (SCT) Graph: The idea behind similarity graph G_{SCT} is building edges among the similar nodes based on features of the nodes. Since other than just co-occurrence relations between nodes that we get using the sliding window method, there are also semantic meanings of the words and semantic relations between them. We can get this semantic information about words and documents using word and document embedding models, e.g., word2vec, GloVe, doc2vec. Also, we can use this information to measure the similarity between documents and create a relation between them. We represent each tweet as a node in G_{SCT} graph. At first, We apply the GloVe model on tweets to get the representation of words and take the average of the words in a tweet to obtain the feature vector of each tweet node. We then apply the K-nearest neighbors model, which calculates the similarity between each pair of tweets

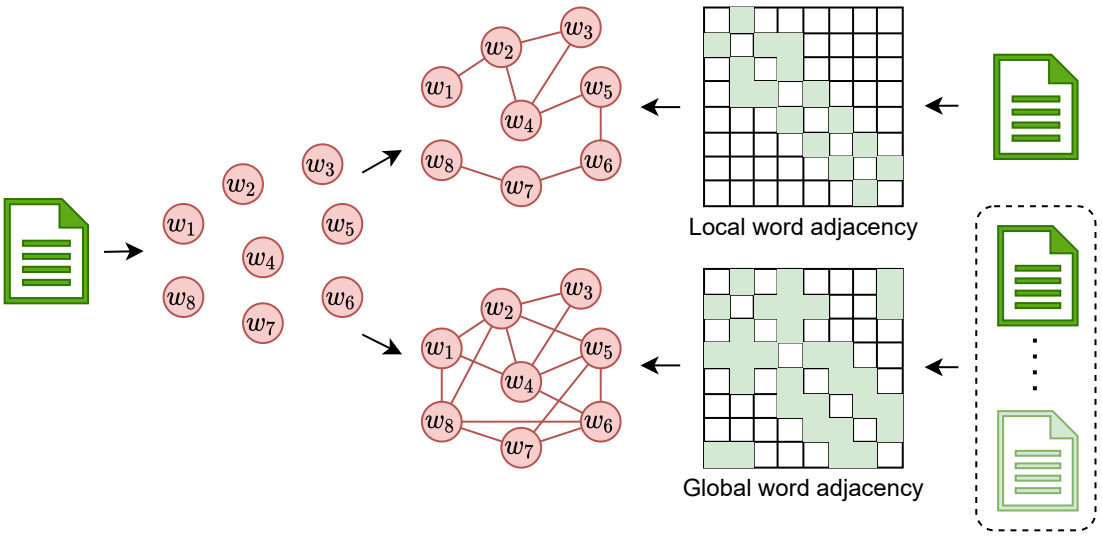

Fig. 3: Local (top) and global (bottom) document-level text graphs.

using their feature vector and then returns k most similar documents of each document. We consider this k most similar documents as the neighbors of it, and we build edges among the neighbor nodes that result in a similarity text graph G_{SCT}. This G_{SCT} is a weighted graph where the weight of an edge between two nodes is calculated using Gaussian kernel (GK) weighting [26], which can effectively capture the sample similarity. The adjacency matrix of G_{SCT} is defined as

$$A_{SCT_{i,j}} = \begin{cases} GK\_weight_{i,j} & \text{GK weighting between} \\ & \text{two connected nodes i and j} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$GK\_weight_{i,j} = \exp(-\frac{d(i,j)}{2 * \sigma^2})$$

where $d(i,j)$ is a distance function that could be as simple as cosine distance, and σ is the kernel bandwidth parameter.

b) Document-level Text Graph

As the document-level, we construct individual text graphs from each tweet by following two different approaches: local DT graph and global DT graph.

b.1. Local Document-level Text (LDT) Graph. To construct an LDT graph G_{LDT} for each tweet, we consider each unique word of the tweet as a node and build a word-word edge based on the co-occurrence of words in the tweet. We consider a fixed-size sliding window (length three at default) to capture the word co-occurrence. We add an edge between two-word nodes if they appear in the same window. This captures the local adjacency of the words. In Fig. 3 (top), we present an example for a local document-level text graph, where an adjacency matrix is constructed from each document.

b.2. Global Document-level Text (GDT) Graph. Since tweets are short text, when we use local adjacency of words in a single text to create text graph, we may get less number of edges in the graph. Therefore, to solve this problem, we construct a GDT graph G_{GDT}, where we use global word adjacency to create the text graph. We create a $U \times U$ word-word adjacency matrix where U is the set of unique words of the entire corpus by maintaining a fixed-size sliding window on text. Also, we represent every unique word of the entire corpus by a vector of d dimension initialized by word embedding; thus, we create a matrix of shape $U \times d$. These two matrices are globally shared by all texts in the corpus. To create graph G_{GDT} for each individual tweet, while we obtain nodes from words of individual text, edge information and the word representations are taken from the global shared matrices. So, we add edges between 2 nodes if they appear in the same window in any document, even if they do not appear in the same window for that document. In this way, we capture the global relations of words. In Fig. 3 (bottom), we present an example for a global document-level text graph, where an adjacency matrix is constructed from the full corpus that is shared globally while constructing a graph from each document.

C. Graph Neural Network (GNN)

We apply different GNN models on our constructed graph to learn the features of tweets and train the model for DA classification. GNN is a multilayer neural network that is designed for graphs and can generate embeddings of nodes using global and local structural patterns from graphs [27]. It has a propagation module that is used to propagate information between nodes so that the aggregated information could capture both feature and topological information in the graph. A layer of a GNN can be expressed as

$$F^{(1)} = \sigma(\hat{A}F^{(0)}W^{(0)}) \tag{4}$$

where $\hat{A}$ stands for adjacency matrix with self-loop of a graph G, $F^{(0)} \in R^{n \times m}$ be a matrix containing all n nodes with their

features, $W_0 \in R^{m \times k}$ is a weight matrix, and σ is an activation function. We use the GloVe embedding model [28] to get the initial features of word nodes of the DT graphs. To get the initial features of a document node of the CT graphs, we take the average of the words features generated by GloVe of that document.

For corpus-level text graphs, we need to get node embedding which will be the embedding of the tweets. We apply the GCN model to get the node embedding on corpus level graphs, which are G_{HoCT}, G_{HeCT}, and G_{SCT}.

For document-level text graphs, we need to get graph embedding which will be the embedding of the tweets. For graph embedding, we apply message passing mechanism-based GNN, and gated graph neural networks on G_{GDT}, and G_{LDT} respectively.

1) ***GCN****:* We apply GCN to CT graphs, which are G_{HoCT}, G_{HeCT}, and G_{SCT} graphs explained in Section III-B(a). In GCN, each node is represented by aggregating its neighbors' features along with its own features. Kipf et al. [22] designed propagation rules that sum up features from the previous layer to form the next layer's features with a spectral approach. A two-layer GCN can be expressed as

$$F^{(1)} = Relu(ZF^{(0)}W^{(0)}), F^{(2)} = softmax(ZF^{(1)}W^{(1)}) \tag{5}$$

where $Z = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ is the normalized symmetric adjacency matrix, and $\hat{A}$ stands for adjacency matrix with 1s on its diagonal for the self-loops. $\hat{D}$ is the degree matrix of adjacency matrix $\hat{A}$, where $\hat{D}_{\text{ii}} = \sum_j \hat{A}_{\text{ij}}$. $F^0 = X$, where X is the initial features of nodes, W^0 and W^1 are the trainable weight.

Loss function to train GCN: To train and thus minimize the loss between ground truth label and predicted label, we set a cross-entropy loss function:

$$Loss = \sum_{t \in T} \sum_{c=1}^{C} Y_{tc} \ln F_{tc}^{(2)} \tag{6}$$

Where T is the set of tweets/documents that have labels, and C is the dimension of the output features, which is basically equal to the number of classes, and Y is the label matrix. Here, as an optimizer, we use adam optimizer. In equation 7, $F^{(2)}$ is the embedding of tweets.

2) ***Message Passing Mechanism based GNN (MPM-GNN)****:* Similar to [15], we apply a message passing mechanism (MPM) as non-spectral convolution on G_{GDT} graph explained in III-B(b.2). After the text is processed, the embedding of the vertices is initialized with word features, denoted as $x \in R^{|V| \times d}$ where d is the embedding dimension. For each graph, after collecting information from adjacent nodes, MPM applies a reduction function on them, which combines the maximum values on each dimension to form a new vector. Then it updates nodes' representations based on their original representations and collected information, which is defined as:

$$h_i = \max_{a \in N_i^p} e_{ji} x_j, \ \bar{x_i} = (1 - \eta_i)h_i + \eta_i x_i \tag{7}$$

Where h_i is the message that node i receives from its neighbors, N_i^p is the closest p words of node i, e_{ij} is the edge weight between node i and j. x_i and x_j are the previous representation of nodes i and j, respectively; max is a reduction function, η_i is a trainable variable of node i that decides the portion of the information that should be kept. $\bar{x_i}$ represents the updated representations of node i. Finally, the representations of all nodes of a text graph are summed up and passed through a softmax function to predict the label of the text:

$$y_i = softmax(Relu(W * sum + b)), \ sum = \sum_{n \in N_i} \bar{x}_i \tag{8}$$

W and b are trainable weights and biases.

3) ***Gated Graph Neural Network (GGNN)****:* We apply gated graph neural network [29] to our local document-level text graph G_{LDT} to learn the representations of graphs similar to TextIng [16]. Similar to the message passing, a node receives information from its neighbors and then combine it with its own feature to get an updated feature which is defined as:

$$m^t = Ah^{t-1}W_m \tag{9}$$

$$u^t = \sigma(W_{z1}m^t + W_{z2}h^{t-1} + b_z) \tag{10}$$

$$r^t = \sigma(W_{r1}m^t + W_{r2}h^{t-1} + b_r) \tag{11}$$

$$\bar{h}^t = tanh(W_{h1}m^t + W_{h2}(r^t \odot h^{t-1}) + b_h) \tag{12}$$

$$h^t = \bar{h}^t \odot u^t + h^{t-1} \odot (1 - u^t) \tag{13}$$

Where A is the adjacency matrix, m is the messages that a node receives from its neighbors. h^{t-1} and h^t are the representation of nodes at layer $t-1$ and t, respectively. W and b are trainable weights and biases. u and r are used as update gate and reset gate to determine the degree of neighbors' information that contributes to the representation of nodes. Finally, a readout function is used to get the text level representation from nodes which is defined as:

$$h_v = \sigma(f_1(h_v^t)) \odot tanh(f_2(h_v^t)) \tag{14}$$

$$h_g = 1/V \sum_{v \in V} h_v + \max(h_1 ..... h_v) \tag{15}$$

f_1 and f_2 represent multilayer perceptrons. In equation 15, along with averaging the node representations, a max-pooling functions has been used to get the text-graph representations h_g. Then h_g is passed through a softmax function to get the predicted label. For both MPM-GNN and GGNN, we use the same type of cross-entropy loss function that is described in equation 6.

IV. Experiments

In this section, we test and evaluate the proposed models on our dataset. Here, we first explain the baseline models that we compare. We then discuss the parameter settings and evaluation measures. We finally present the experimental results.

A. Baseline models

In this research, we consider three types of models as the baseline models for DA text classification. These models have been used widely in the DA detection task from the text data. They are: (1) The rule-based model, (2) traditional machine learning models, (3) deep learning models.

1) Rule-based model: To detect DA-related tweets using the rule-based model, we go through each tweet and check each word in the tweet against a drug-related word dictionary given in Table I and II. Since tweets are non-formal and include typos, we use similarity measures between words for matching. We measure the similarity of words of a tweet with the dictionary words using the Jaro-Winkler similarity measure [5]. If the similarity score between the word in a tweet and the word from the dictionary is greater than a given similarity threshold value, we consider that the word is present in our dictionary. If a tweet contains at least one word from each table (i.e., Table I and II), we label that tweet as DA, otherwise NDA.

2) Traditional machine learning models: We refer these models as traditional since they consist of non-deep neural network classifiers. The most common text classification method via machine learning is applying a machine learning classifier to the embedding of the test. Here, we follow the same method. For the text embedding, we use pre-trained GloVe to get the vector representation of text. First, we take the average representations of all words in a tweet as the representation for each tweet. We then apply three machine learning classifiers: Naive Bayes (NB), Decision Tree (DT), and Logistic Regression (LR). These are the most common classifiers for text classification and DA detection [4].

3) Deep learning-based models: While deep learning models usually show good performance for many different problems; they are also commonly used for different text classification tasks. Here, we explore five different deep learning models that have been used for DA detection.

CNN: Convolutional Neural Network (CNN) is one of the most common deep learning model. It is frequently used for DA text classification [6]. It has a convolution layer, activation layer, pooling layer, and fully connected layer that are stacked one by one.

LSTM: Long Short Term Memory (LSTM), as a very popular Recurrent Neural Network (RNN) technique [30], has been widely used in text classification [31] particularly DA detection task [6]. RNN is good when dealing with short text but not efficient for long text because of long-term dependency, which has been solved by LSTM [32].

GRU: Gated Recurrent Unit is another type of RNN. Instead of having a memory cell, it has two types of gates called update gate and reset gate that controls the flow of information [33]. Update gate controls the flow of information that is supposed to be passed to the next state, and reset gate decides how much past information is needed to neglect.

RCNN/CRNN: Recurrent Convolutional Neural Network and Convolutional Recurrent Neural Network are hybrid models where the advantages of both CNN and RNN are combined [34].

BiLSTM: One of the most cutting-edge RNN architecture for text classification use the BiLSTM architecture. Bidirectional LSTM utilizes both left and right context to get the current prediction so it can incorporate both past and feature information [35]. It is also more efficient than the traditional LSTM model.

B. Parameter Setting and Evaluation Measures

In the GNN and the baseline models, except the rule-based model, we use the pre-trained GloVe word embedding with dimension d=300 as the initial features. We empirically set the learning rate 0.02 with a dropout 0.5 to avoid overfitting. We use Adam optimizer to minimize cross-entropy loss in the GNN and deep learning-based models. We choose ReLU as an activation function in the hidden layers and Sigmoid in the output layer of deep learning-based models.

As evaluation measures, accuracy, precision, recall, and $F1$-value are used to compare the effectiveness of the baseline approaches with proposed graph-based models. We split our data as 67% for train and 33% for test. The experiment is run 10 times for each model, and the average of the 10 trials is taken to obtain more reliable results. Here we report the weighted average of precision, recall, and $F1$-value since it takes label imbalance into account.

C. Result analysis

We analyze each type of baseline model separately to select the best model and parameters within each type. Then we compare the best ones with our proposed models.

Table IV represents the rule-based model accuracy for different threshold values from similarity scores ranging from 0.65 to 1. When the threshold value is 0.65, the accuracy is too low that is around 33%. With the increment of threshold, the accuracy also increases. For the threshold 0.75 and 0.85, the accuracy is around 54% and 72%, respectively. However, when the threshold is high, like 1, the accuracy again decreases. Using 0.85 value as the threshold gives the best results for all evaluations measures. The reason behind these accuracy fluctuations is that in the rule-based model, we calculate the similarity measure between words from each tweet and dictionary. Hence, when the similarity threshold is too low, a lot of words from the tweets are counted as similar words of the dictionary, though they are not. Similarly, as tweets contain typos and informal language, when the threshold is too high, the model tries to match the words from tweets to the dictionary exactly, thus losing many interesting tweets.

Next, we present the results for traditional machine learning-based models in Table V. After converting the text into a

TABLE IV: Experimental results of the rule-based model with different threshold values.

Threshold	Accuracy	Precission	Recall	F1 score
0.65	0.330	0.592	0.330	0.292
0.75	0.541	0.617	0.541	0.567
0.85	**0.725**	**0.681**	**0.725**	**0.686**
1.00	0.724	0.648	0.724	0.665

TABLE V: Experimental results of the traditional machine learning models.

Model	Accuracy	Precision	Recall	F1 score
NB	0.558	0.671	0.558	0.584
DT	0.687	0.693	0.687	0.689
LR	**0.759**	**0.746**	**0.759**	**0.691**

TABLE VI: Experimental results of the deep learning models.

Model	Accuracy	Precision	Recall	F1 score
CNN	0.744	0.742	0.743	0.740
RCNN	0.750	0.740	0.753	0.733
GRU	0.758	0.745	0.758	0.750
LSTM	0.758	**0.752**	0.762	**0.751**
BiLSTM	**0.766**	0.746	**0.766**	0.737

fixed dimension vector by exploiting pre-trained GloVe word embedding, the vector representations are used as features in the machine learning classifier, which are NB, DT, and LR. As we see in Table V, LR is the best performer among the three for all measures. While LR's accuracy is around 76%, NB's accuracy is much lower than others as it is around 55%.

Furthermore, Table VI shows the results for deep learning models. While results for all deep learning models are close to each other, out of five models, we get the best accuracy and recall (76.6%, 76.6%) from BiLSTM and best precision and F1 (75.2%, 75.1%) from LSTM. One of the advantages of using BiLSTM is that it can preserve information from both past and feature blocks that help to learn better, whereas unidirectional LSTM can only preserve information from past blocks. Moreover, the performance of RNN models (i.e., LSTM, GRU, BiLSTM) with respected all measures is higher than the CNN model. The reason actually lies in their structures. RNNs consist of a sequence of neural network blocks that are a good fit for sequential data like text data where the output of the current step depends on the previous step. RNN can capture long-range semantic dependency, whereas CNN is good for grid structure data like images and can capture local features. That is why in our experiment, all RNN-based models outperform CNN. Moreover, the hybrid model RCNN outperforms CNN in terms of accuracy and recall as it utilizes the advantages of both CNN and RNN.

Last, in Table VII, we present our models' performance. From this table, we can see that while G_{HeCT} with GCN is the least performer, G_{SCT} with GCN is the best performer among our graph-based models. G_{HeCT} is a heterogeneous corpus-level co-occurrence text graph that has word-word and word-document edges but not the document-document edges. However, the performance of the other co-occurrence

TABLE VII: Experimental results of the proposed graph-based models.

Model	Accuracy	Precission	Recall	F1 score
HeCT + GCN	0.734	0.738	0.734	0.738
HoCT + GCN	0.777	0.807	0.777	0.707
SCT + GCN	**0.964**	**0.965**	**0.964**	**0.963**
LDT + GGNN	0.786	0.781	0.786	0.774
GDT + MPM-GNN	0.779	0.766	0.779	0.766

graph G_{HoCT} with GCN is comparatively better, which has document-document edges and can pass messages easily in GCN hidden layers. Performance of LDT is slightly better than the performance of GDT. Though these models are document-level text graphs, in GDT, we use a globally shared word-word adjacency matrix; thus, the edges of the graph are globally fixed. However, in LDT, there are no globally fixed edges; rather, we focus on capturing the local structure of each text graph by setting a fixed sliding window. Global information may provide noisy data into model. G_{SCT} with GCN has the best performance among all other models for all measures. A possible reason lies in its construction. The G_{SCT} graph is based on the k-nearest neighbors (KNN) model, where each node is connected to its k most similar node. The similarity is calculated based on document embedding that also includes the semantic information of words and documents. It provides valuable information for drug abuse detection.

Moreover, in Figure 4, a performance comparison (i.e., accuracy, precision, recall, F1-score) between our graph-based models and baseline models (best performer of each type of baseline model) is shown. The best results for the rule-based model are obtained with threshold k = 0.85, for traditional machine learning models with LR and for deep learning models with BiLSTM, where the accuracy of them is 72.5%, 75.9% and 76.6%, respectively. Therefore, we compare their results with our model's results. We can see that all of our graph-based models outperform baseline models except G_{HeCT} with GCN.

The performance of the rule-based model is the worst among the others. The reason is that this model depends on a dictionary of keywords that are built manually and needs domain expertise. Moreover, social media chatter changes over time, and this model does not have any learning parameter to update the dictionary automatically. Thus, it may fail to work in the future. Similarly, though the performance of LR is good enough and close to deep learning models, extracting and selecting the best features is a costly process. However, the good performance of our graph-based models, especially G_{HeCT} with GCN, implies that the graph-based models are very promising solutions for text classification and drug abuse detection tasks.

Also, defining relations between documents with local information, especially based on their semantic similarities, gives better performance for the drug abuse detection problem. Instead of treating DA detection as a regular text classification task, we treat it as a node classification task, thanks to the corpus-based text graph models, and as a graph classification

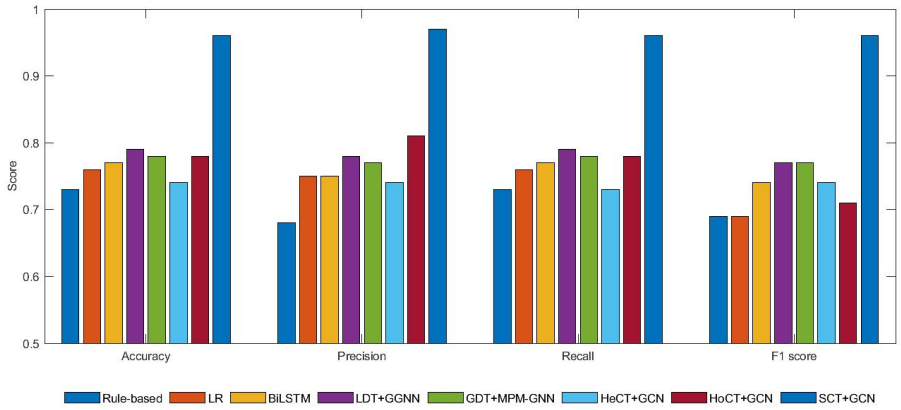

Fig. 4: Comparison between the performance of the proposed graph-based DA detection models and the baseline models. Here we present the best baseline models from each type.

task, thanks to the document-level text graph models. The main assumption of node classification is that similar nodes are close to each other in the network and passage messages between them. For graph classification, similar texts have a similar graph structure. That is why employing relational information within text via graph models increases the effectiveness compared with the regular text classification models, as we see in the results.

D. Insight analysis on abuse related tweets

We further apply our model to see the difference between drug abuse-related and non-abuse-related tweets. We present the comparison of frequent words in DA tweets and NDA tweets in Figures 5 and 6. Since similarity-based text graph, G_{SCT} with GCN model has the best performance among all, we use this model to label 2,000 tweets data. Then, we extract the 10 most frequent words in DA tweets and find their frequency in NDA tweets. We present the normalized frequency of these words in DA tweets and NDA tweets in Figure 5. Similarly, we extract the 10 most frequent words in NDA tweets and then find their frequency in DA tweets. We present the normalized frequency of those words in DA tweets and NDA tweets in Figure 6.

As we see from these two figures, frequent words of DA tweets and NDA tweets are different. While we see abuse-related words frequently in DA tweets, we see regular words frequently in NDA tweets. These results indicate that our graph-based models are effective in drug abuse detection.

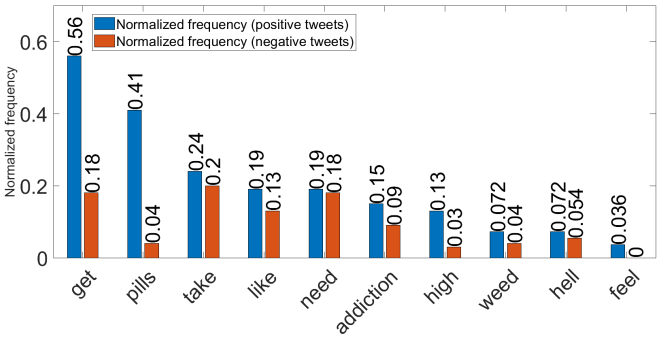

Fig. 5: Comparison between the normalized frequency score of top 10 frequent words in the positive tweets and the negative tweets.

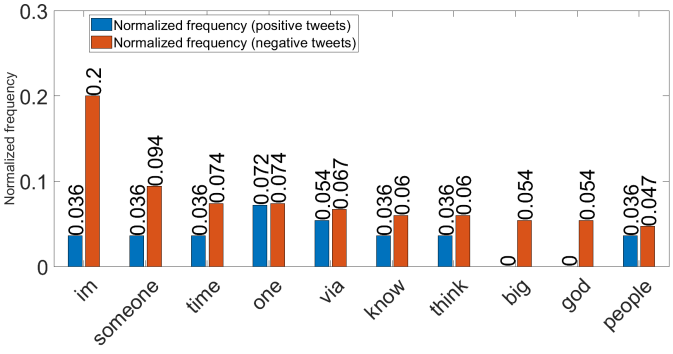

Fig. 6: Comparison between the normalized frequency score of top 10 frequent words in the negative tweets and the positive tweets.

V. Conclusion

This paper presents different GNN-based approaches for drug abuse detection as a node (corpus-level) and a graph (document-level) classification task from social media data. To the best of our knowledge, the present work is the first study that employs different GNNs for DA detection in Twittersphere. Here, we show different approaches to represent text data as a graph and the impact of GNN models on graph structure in text classification problems. Our experiments suggest that GNN-based models are very promising in DA text classification, where we get the highest accuracy for G_{SCT} with GCN compared to the traditional machine learning and deep learning models. Defining relations between documents with local information, especially based on their semantic

similarities, gives promising performance for the DA detection problem.

The noisy and sparse characteristics of Twitter data and the limited availability of annotated data are the major challenges for this study, as they are in social network analysis. We will increase our corpus size and incorporate attention mechanisms in GNN to improve performance in future work. Furthermore, the extra information that we can extract from social media, such as user information, connections among users, and tweets, may be valuable for the DA detection problem.

REFERENCES

[1] Janani Kalyanam, Takeo Katsuki, Gert R.G. Lanckriet, and Tim K. Mackey. Exploring trends of nonmedical use of prescription drugs and polydrug abuse in the Twittersphere using unsupervised machine learning. *Addictive Behaviors*, 2017.

[2] Abeed Sarker, Karen O'Connor, Rachel Ginn, Matthew Scotch, Karen Smith, Dan Malone, and Graciela Gonzalez. Social media mining for toxicovigilance: Automatic monitoring of prescription medication abuse from twitter. *Drug Safety*, 2016.

[3] Aapcc.org. "National Poisoning Data system." [Online] Available: https://www.aapcc.org/national-poison-data-system.

[4] Nhathai Phan, Manasi Bhole, Soon Ae Chun, and James Geller. Enabling real-Time drug abuse detection in tweets. In *Proceedings - International Conference on Data Engineering*, 2017.

[5] Ferdaous Jenhani, Mohamed Salah Gouider, and Lamjed Ben Said. Lexicon-based system for drug abuse entity extraction from Twitter. *Communications in Computer and Information Science*, 613:692–703, 2016.

[6] Han Hu, Nhat Hai Phan, Soon A. Chun, James Geller, Huy Vo, Xinyue Ye, Ruoming Jin, Kele Ding, Deric Kenne, and Dejing Dou. An insight analysis and detection of drug-abuse risk behavior on Twitter with self-taught deep learning. *Computational Social Networks*, 6(1), 2019.

[7] Kim M.G., Kim J., Kim S.C., and Jeong J. Twitter Analysis of the Nonmedical Use and Side Effects of Methylphenidate: Machine Learning Study. *Journal of medical Internet research*, 2020.

[8] Stevie Chancellor, George Nitzburg, Andrea Hu, Francisco Zampieri, and Munmun De Choudhury. Discovering alternative treatments for opioid use recovery using social media. In *Conference on Human Factors in Computing Systems - Proceedings*, 2019.

[9] Abeed Sarker, Graciela Gonzalez-Hernandez, and Jeanmarie Perrone. Towards automating location-specific opioid toxicosurveillance from twitter via data science methods. In *Studies in Health Technology and Informatics*, 2019.

[10] Han Hu, Nhat Hai Phan, James Geller, Stephen Iezzi, Huy Vo, Dejing Dou, and Soon Ae Chun. An ensemble deep learning model for drug abuse detection in sparse twitter-sphere. In *Studies in Health Technology and Informatics*, volume 264, 2019.

[11] Mohammed Ali Al-Garadi, Yuan Chi Yang, Haitao Cai, Yucheng Ruan, Karen O'Connor, Gonzalez Hernandez Graciela, Jeanmarie Perrone, and Abeed Sarker. Text classification models for the automatic detection of nonmedical prescription medication use from social media. *BMC Medical Informatics and Decision Making*, 21(1), 2021.

[12] Han Hu, Pranavi Moturu, Kannan Neten Dharan, James Geller, Sophie Di Iorio, and Hai Phan. Deep learning model for classifying drug abuse risk behavior in tweets. In *2018 IEEE International Conference on Healthcare Informatics, ICHI 2018*, 2018.

[13] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to Fine-Tune BERT for Text Classification? In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11856 LNAI, 2019.

[14] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, 2019.

[15] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Text level graph neural network for text classification. In *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 2020.

[16] Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. Every Document Owns Its Structure: Inductive Text Classification via Graph Neural Networks. 2020.

[17] Azadeh Nikfarjam, Abeed Sarker, Karen O'Connor, Rachel Ginn, and Graciela Gonzalez. Pharmacovigilance from social media: Mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, 22(3), 2015.

[18] Binxuan Huang and Kathleen M. Carley. Syntax-aware aspect level sentiment classification with graph attention networks. In *EMNLP-IJCNLP 2019*, 2020.

[19] Chen Zhang, Qiuchi Li, and Dawei Song. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 2020.

[20] Nilesh Chakraborty, Denis Lukovnikov, Gaurav Maheshwari, Priyansh Trivedi, Jens Lehmann, and Asja Fischer. Introduction to neural network-based question answering over knowledge graphs, 2021.

[21] Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Soren Auer. Neural network-based question answering over knowledge graphs on word and character level. In *26th International World Wide Web Conference, WWW 2017*, 2017.

[22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[23] Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. Tensor graph convolutional networks for text classification. In *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, 2020.

[24] Rahul Ragesh, Sundararajan Sellamanickam, Arun Iyer, Ramakrishna Bairi, and Vijay Lingam. HeteGCN: Heterogeneous Graph Convolutional Networks for Text Classification. In *WSDM 2021 - Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021.

[25] Edward Loper Bird, Steven and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[26] Tony Jebara, Jun Wang, and Shih Fu Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th International Conference On Machine Learning, ICML 2009*, 2009.

[27] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep Learning on Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[28] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014.

[29] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[30] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[31] Ran Jing. A Self-attention Based LSTM Network for Text Classification. In *Journal of Physics: Conference Series*, volume 1207, 2019.

[32] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *13th Annual Conference of the International Speech Communication Association 2012, INTERSPEECH 2012*, volume 1, 2012.

[33] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014.

[34] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the National Conference on Artificial Intelligence*, volume 3, 2015.

[35] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, Sang Ug Kang, and Jong Wook Kim. Bi-LSTM model to increase accuracy in text classification: Combining word2vec CNN and attention mechanism. *Applied Sciences (Switzerland)*, 10(17), 2020.