Learning representations for information mining from text corpora with applications to cyber threat intelligence

by

Avishek Bose

B.S., University of Dhaka, 2012

M.S., University of Dhaka, 2014

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2023

Abstract

This research develops learning representations and architectures for natural language understanding, within an information mining framework for analysis of open-source cyber threat intelligence (CTI). Both contextual (sequential) and topological (graph-based) encodings of short text documents are modeled. To accomplish this goal, a series of machine learning tasks are defined, and learning representations are developed to detect crucial information in these documents: cyber threat entities, types, and events. Using hybrid transformer-based implementations of these learning models, CTI-relevant key phrases are identified, and specific cyber threats are classified using classification models based upon graph neural networks (GNNs). The central scientific goal here is to learn features from corpora consisting of short texts for multiple document categorization and information extraction sub-tasks to improve the accuracy, precision, recall, and F1 score of a multimodal framework.

To address a performance gap (e.g., classification accuracy) for text classification, a novel multi-dimensional Feature Attended Parametric Kernel Graph Neural Network (APKGNN) layer is introduced to construct a GNN model in this dissertation where the text classification task is transformed into a graph node classification task. To extract key phrases, contextual semantic tagging with text sequences as input to transformers is used which improves a transformer's learning representation. By deriving a set of characteristics ranging from low-level (lexical) natural language features to summative extracts, this research focuses on reducing human effort by adopting a combination of semi-supervised approaches for learning syntactic, semantic, and topological feature representation. The following central research questions are addressed: can CTI-relevant key phrases be identified effectively with reduced human effort; whether threats be classified into different types; and can threat events be detected and ranked from social media like Twitter data and other benchmark data sets.

Developing an integrated system to answer these research questions showed that user-specific information in shared social media content, and connections (followers and followees) are effective and crucial for algorithmically tracing active CTI user accounts from open-source social network data. All these components, used in combination, facilitate the understanding of key analytical tasks and objectives of open-source cyber-threat intelligence.

Learning representations for information mining from text corpora with applications to cyber threat intelligence

by

Avishek Bose

B.S., University of Dhaka, 2012

M.S., University of Dhaka, 2014

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2023

Approved by:

Major Professor
William H. Hsu

Copyright

Abstract

This research develops learning representations and architectures for natural language understanding, within an information mining framework for analysis of open-source cyber threat intelligence (CTI). Both contextual (sequential) and topological (graph-based) encodings of short text documents are modeled. To accomplish this goal, a series of machine learning tasks are defined, and learning representations are developed to detect crucial information in these documents: cyber threat entities, types, and events. Using hybrid transformer-based implementations of these learning models, CTI-relevant key phrases are identified, and specific cyber threats are classified using classification models based upon graph neural networks (GNNs). The central scientific goal here is to learn features from corpora consisting of short texts for multiple document categorization and information extraction sub-tasks to improve the accuracy, precision, recall, and F1 score of a multimodal framework.

To address a performance gap (e.g., classification accuracy) for text classification, a novel multi-dimensional Feature Attended Parametric Kernel Graph Neural Network (APKGNN) layer is introduced to construct a GNN model in this dissertation where the text classification task is transformed into a graph node classification task. To extract key phrases, contextual semantic tagging with text sequences as input to transformers is used which improves a transformer's learning representation. By deriving a set of characteristics ranging from low-level (lexical) natural language features to summative extracts, this research focuses on reducing human effort by adopting a combination of semi-supervised approaches for learning syntactic, semantic, and topological feature representation. The following central research questions are addressed: can CTI-relevant key phrases be identified effectively with reduced human effort; whether threats be classified into different types; and can threat events be detected and ranked from social media like Twitter data and other benchmark data sets.

Developing an integrated system to answer these research questions showed that user-specific information in shared social media content, and connections (followers and followees) are effective and crucial for algorithmically tracing active CTI user accounts from open-source social network data. All these components, used in combination, facilitate the understanding of key analytical tasks and objectives of open-source cyber-threat intelligence.

Table of Contents

List of Figures xii

List of Tables xiv

Acknowledgements xvi

Dedication xvii

1 Introduction 1

1.1 Motivation 2

1.2 Problem Statement 3

1.3 Contributions 4

1.4 Dissertation Outline 5

2 Extracting Key Phrases from Short Texts for Cyber-Threat Intelligence Tasks . . 7

2.1 Introduction 7

2.2 Related Work 9

2.2.1 Statistical Learning Approaches 9

2.2.2 Feature-Based Heuristic Approaches 10

2.2.3 ML/DL Based Approaches 10

2.2.4 NE Extraction for CTI 10

2.3 Cybersecurity Keyphrase Identification Framework 13

2.3.1 Incorporating Heuristic Rules 14

2.3.2 Adopted Statistical and Neural Network Models 18

2.4 Experiment and Evaluation 23

2.4.1 Data sets . . . 23
2.4.2 Environment Setup . . . 24
2.4.3 Compatibility of Tagging Rules . . . 24
2.4.4 Our Developed Tagging Rule Validation . . . 26
2.4.5 Analysis of Results . . . 26
2.5 Conclusion and Future Work . . . 28

3 Attention Aware Parametric Kernel Graph Neural Network for Classifying Cyber-Threat Types . . . 30
3.1 Introduction . . . 30
3.2 Related Work . . . 33
3.3 Deep Learning on Graphs . . . 34
3.3.1 Spectral and Spatial based approaches . . . 34
3.3.2 Parametric Kernel for Extracting Patches . . . 36
3.3.3 Self Attention in GNN . . . 37
3.4 Implemented Technique . . . 39
3.5 Experiments . . . 43
3.5.1 Data sets . . . 43
3.5.2 Experimental Setup . . . 44
3.6 Results and Performance Evaluation . . . 45
3.6.1 Discussion . . . 53
3.7 Conclusion and Future Work . . . 54

4 Event Detection and Ranking of Cyber-Threat Events . . . 55
4.1 Introduction . . . 55
4.2 Related Approaches . . . 57
4.3 Background . . . 59
4.3.1 Named Entity Recognition (NER) . . . 59

4.3.2 TextRank . . . 59
4.3.3 TFIDF . . . 59
4.3.4 DBSCAN . . . 60
4.4 Methods . . . 60
4.4.1 Tweet Collection and Early Annotation . . . 60
4.4.2 Tweet Pre-processing and Cleaning . . . 61
4.4.3 Influential Twitter User Impact . . . 61
4.4.4 Determining Algorithm Design Architecture . . . 62
4.4.5 Event Detection Heuristics and Scoring . . . 62
4.4.6 Annotation Approach . . . 67
4.5 Experimental Results . . . 68
4.5.1 Simulation . . . 68
4.5.2 Annotation-Based Validation . . . 71
4.6 Conclusion . . . 74

5 Tracing Relevant Twitter Accounts in CTI Domain . . . 75
5.1 Introduction . . . 75
5.2 Related Work . . . 77
5.3 Methods . . . 78
5.3.1 Graph Construction . . . 79
5.3.2 Text Pre-processing . . . 79
5.3.3 Community Detection . . . 80
5.3.4 Calculating Community Weight and User Weight . . . 80
5.3.5 Calculating Text Similarity and Community Ranking . . . 82
5.3.6 Scoring and Ranking User Nodes . . . 83
5.3.7 Validating Model Output . . . 84
5.4 Collecting and Preparing Data . . . 86
5.4.1 Text Rating . . . 86

5.4.2 Extracting Sample Network 87
5.4.3 Annotation of Sample Network 87
5.5 Experimental Setup 88
5.6 Results and Discussion 89
5.7 Conclusion and Future Work 92

6 Conclusions and Future Research 93
6.1 Summary of Contributions 93
6.2 Future Research 95

Bibliography 98

List of Figures

1.1 Proposed Cyber-Threat Awareness System 4

2.1 Workflow diagram of new context-augmented key phrase extractor 11

2.2 Illustration of our new tagging rules with their respective steps for key phrase tagging 19

2.3 BERT-BiLSTM-CRF model[1] 23

2.4 Precision, Recall, and F1-Value of Rouge 1 score of developed rule-based tagger 27

2.5 Precision, Recall, and F1-Value of Rouge 1 score for baseline rule-based tagger 27

3.1 Euclidean and Non-euclidean data structure[2] 31

3.2 A two leyer GNN network[3] 36

3.3 Operational steps of node classification using novel implemented APKGNN layer) 42

3.4 Construction of APKGNN layer) 42

3.5 Updating node representation using novel APKGNN layer) 43

3.6 Test accuracy curve up to fifty epochs for the CORA data set for all GNN layers 46

3.7 Training accuracy curve up to fifty epochs for the CORA data set for all GNN layers 47

3.8 Validation accuracy curve up to fifty epochs for the CORA data set for all GNN layers 47

3.9 Testing accuracy curve up to fifty epochs for the CITESEER data set for all GNN layers 48

3.10 Training accuracy curve up to fifty epochs for the CITESEER data set for all GNN layers . . . 48
3.11 Validation accuracy curve up to fifty epochs for the CITESEER data set for all GNN layers . . . 49
3.12 Testing accuracy curve up to fifty epochs for the PUBMED data set for all GNN layers . . . 50
3.13 Training accuracy curve up to fifty epochs for the PUBMED data set for all GNN layers . . . 50
3.14 Validation accuracy curve up to fifty epochs for the PUBMED data set for all GNN layers . . . 51
3.15 Testing accuracy curve up to fifty epochs for the CTI data set for all GNN layers . . . 52
3.16 Training accuracy curve up to fifty epochs for the CTI data set for all GNN layers . . . 52
3.17 Validation accuracy curve up to fifty epochs for the CTI data set for all GNN layers . . . 53

4.1 Graphical representation of *commonSet*, *keywordSet* and *namedEntitySet* . . . 63
4.2 Flowchart of the proposed approach . . . 67
4.3 Event plot of the second time interval proposed approach . . . 69

5.1 Flow diagram of our proposed approach . . . 79
5.2 High level structure of a user community in our analysis . . . 82
5.3 User weight calculation using the predicted values of account tweets and descriptions . . . 83
5.4 Precision, Recall, and $F_1\beta$ comparison of our approach against two relevant user recommendation approaches . . . 91

List of Tables

2.1 Comparing our developed hybrid system and current tools for identifying key phrases from a short text . . . 12

2.2 Comparing our developed hybrid system and current tools for identifying key phrases from a short text . . . 13

2.3 Results for baseline tagging rules and our rules tandem with different statistical and transformer-based models[1] applied to CVE & NVD-specific and general cybersecurity corpora . . . 20

2.4 Results for BERT-base-uncased model on Twitter general cybersecurity corpus 25

2.5 ROUGE-1 and Rouge-L score comparing generated key phrases against the annotated data set for each set of rules . . . 26

3.1 Data set statistics for benchmark data sets and CTI data . . . 43

3.2 Training statistics for benchmark data sets and CTI data . . . 44

3.3 Performance of APKGNN compared to SOTA methods on validation and test accuracy for benchmark and CTI data sets based on implemented experimental set up . . . 45

3.4 Performance of APKGNN compared to SPLINECNN on validation and test accuracy for benchmark data sets based on SPLINECNN experimental set up 46

4.1 Summary results of five time intervals; NT:Number of Tweets; JT: Just Trendy; TN: Trendy and Novel; FS: First Story; TE: Total Number of Events 69

4.2 Summary results of time interval 1 . . . 70

4.3 Sample results of doc2Vec . . . 72

4.4 Confusion matrix of the algorithm's generated result . . . 73

4.5 Summary results of time interval 1; EN: Event Number;EL: Event Link; TC:Tweets Count
NESR:Normalized Event Score Rank;ET: Event Type; AER: Annotator Event Ranking; DBR: Difference between Rankings; FS: First Story ("Just Novel"); FST: First Story and Trendy (developing) 73

5.1 Performance comparisons with different methods on the sampled annotated data set 90

Acknowledgments

I thank all of the people and well-wishers who have helped, supported, and encouraged me throughout my Ph.D. journey at Kansas State University.

First, I thank my advisor, Dr. William Hsu. Without his support, completing my degree would have been unimaginable. He guided me through a way full of obstacles. He is one of the kindest people I have ever met in my life. He is a knowledgeable, versatile, and enthusiastic mentor. He not only provided proper guidance and insightful ideas but also provided the freedom to become a self-reliant, independent researcher in my areas of interest. In addition to academic and research experience, I have developed soft skills such as leadership, communication, and cooperation working under him and with my peers in the Knowledge Discovery in Databases (KDD) laboratory. It is an honor to have been a part of this wonderful group.

Dr. Doina Caragea, Dr. Pascal Hitzler, and Caterina Scoglio who served as my committee members and provided valuable guidance and insight on my dissertation. I thank them all.

Dr. Daniel Anderson helped me with my High-Performance Computing (HPC) research. Thank you! I also thank Dr. Mustaque Hossain who supported me in an inter-departmental project.

I also want to thank Nora Ransom who helped me revise my dissertation manuscripts.

Finally, I could not have completed my doctoral program without unconditional love, support, and encouragement from my parents, especially my mom, my younger brother, and my wife.

Dedication

To my lovely family, especially my son Aradhyo Bose (Ved), for their support and love.

Chapter 1

Introduction

Despite being a highly dynamic domain, cyber-threat intelligence (CTI) from open-source platforms often lacks the information needed to characterize CTI using specific corresponding components such as threat types, key entities, and actors. Although earlier studies[4–8,9–13,14–18,19–22] of these components have shown performance gain individually, a complete framework for understanding cyber-threat intelligence was needed. To address this research gap, I present a combined framework that focuses on understanding text information using novel syntheses of machine learning methods including a hybrid set up of tagging rules and transformers, graph neural networks (GNNs)[3;23;24], and other semi-supervised models for analyzing syntactic and semantic patterns. The central objective is to learn improved feature representations in multiple, domain-independent sub-tasks, such as structured information extraction, detection, and classification, that use social media data and other benchmark data sets[25–27] to (1) predict cyber-threat types[25]; (2) automatically label key phrases using semantic rules[14–17]; (3) rank emergent threat events based on novelty and trendiness[4–8]; and (4) perform relevant user account detection in social networks[19–22].

To improve the performance of cyber-threat type detection or, more generally, to improve text classification performance that was insufficient due to limitations in learning edge feature dimensionality, I introduce a novel node feature attended parametric kernel Graph Neural Network (APKGNN) in this dissertation. For key phrase extraction, I show that aug-

menting documents with contextual semantic tags can improve the accuracy of transformers used to encode text sequences. Both of these approaches support semi-supervised learning principles that are one of our implicit objectives because of the scarcity of labeled data among abundant, unlabeled, open-source data. In addition to the scientific goal of developing improved feature representations, the secondary goal was to present a robust cyber-threat awareness system comprising cyber-threat event detection and cyber threat-relevant user tracing methods. Therefore, I empirically demonstrate that incorporating user community structural information in social networks with their shared text content can trace influential user accounts effectively. This research thus incorporates multiple sub-modules such as a new GNN architecture, a hybrid system of context-aware transformers, and a system for tracing active user accounts using both network structures and text contents, and CTI events and ranking to reach both the scientific and application goals. In each of the sub-modules, experimental results on cyber-threat and other benchmark corpora demonstrate the improvements over existing state-of-the-art models, providing support for the hypothesis that the use of a unified framework consisting of multiple hybrid components and an implemented GNN layer (APKGNN) facilitate domain adaptation for characterizing cyber-threat intelligence.

1.1 Motivation

- Learning high-quality feature representation via deep learning models, such as transformers and GNNs, from short and raw text corpora, remains an open research problem. Transformer models often fail to learn improved representation for sequence labeling tasks such as key phrase identification because plain tokenized text data is fed into models. However, apart from improving the transformer models themselves, generating contextualized tagging data to feed into transformer models would be a new research direction to boost transformer performance because context information has provided performance gains in predictive NLP tasks. On the other hand, GNNs have begun outperforming CNNs in text classification tasks, and GNNs are very effective in semi-supervised learning settings. Using a parametric kernel to attend over the node

features in multi-dimensional vector space settings has not yet been explored for learning text or node feature representation. Therefore, introducing a novel efficient GNN layer would not only facilitate generic text classification, but also it allow classifying cyber-threat types as an application domain.

- Moreover, in dealing with emerging cyber-threats to online systems, updating security measures must be proactive rather than incremental. Open-source cyber-threat intelligence (CTI) provides predictive indicators to support such a strategy. Research gaps in earlier relevant research has exposed ambiguities in characterizing CTI through distinct components, which include CTI key phrases, cyber-threat types, tracing malicious users, and finding and ranking events.

- The Keyphrase identification process is a principal component of understanding CTI, but it supplies only limited information to characterize OSINT. However, predicting cyber-threat types can address this limitation. In addition, the cyber-threat event detection method will activate the expected functionality of the system, and tracing user accounts to monitor CTI instances will allow the construction of a robust CTI awareness system. So, analysis that points out these key components before incorporating them and improving the performance of each component over the state-of-the-art models is highly important.

1.2 Problem Statement

In this dissertation, I implemented novel techniques to learn sequential representations for the extraction task and topological representations for the text classification task using the short text and applying them to the CTI domain. From an application perspective, this research characterizes and then addresses the limitations of individual components of CTI, such as cyber-threat entity extraction, cyber-threat classification, CTI-relevant user account tracing, and event identification. The problem statement addressing these techniques follows:

- Improving classification performance on text data by implementing a novel graph neu-

ral network (GNN) convolution technique in which the text classification task is considered as a node classification task and raw text corpus is transformed into graph data;

- Boosting transformer-based language model performance by feeding contextualized semantic information into hybrid transformer models for the key phrase identification task;
- Identifying the key components of CTI and their correlations to facilitate constructing an end-to-end cyber threat awareness system;
- Reducing the human effort in data annotation and labeling by providing a set of semi-supervised and unsupervised machine learning approaches.

Figure 1.1 illustrates the block diagram of the proposed cyber threat awareness system.

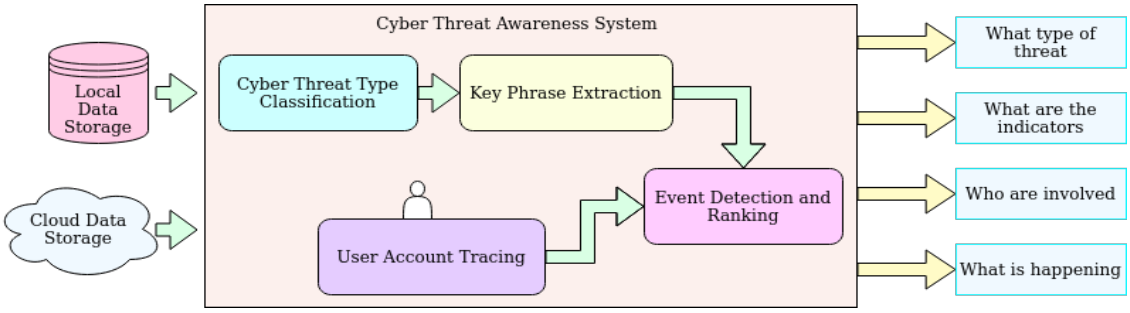

Figure 1.1: *Proposed Cyber-Threat Awareness System*

1.3 Contributions

Much of the research behind this dissertation has been published in peer-reviewed conference proceedings, while the rest of the research has been submitted and is under review. The research contributions can be summarised as follows:

- I introduce a new attention-based GNN that uses a Gaussian mixture model for short-text classification tasks.

- In the dissertation, I address the contextual limitations of current deep learning-based and heuristic key phrase extraction tools as applied to the domain of cybersecurity. To address these limitations, I develop a hybrid system that augments the performance of state-of-the-art (SOTA) transformers for the task of key phrase sequence labeling, using a novel set of semantic role-aware tagging rules to generate effective tag sequences from short text corpora. Next, I fine-tune multiple SOTA deep learning (DL) language model (LM) architectures to these transformed sequences.

- I present a new machine learning and text information extraction approach for detecting cyber-threat events in Twitter that are *novel* (previously non-extant) and *developing* (marked by significance concerning similarity with a previously detected event).

- In the dissertation, I propose a novel approach for cyber threat-associated user account tracing in the Twitter data stream based on the ranking of users according to their contextual relevance and topological information extracted from finding user communities in the Twitter network.

1.4 Dissertation Outline

- Chapter 2 - Extracting Key Phrases from Short Texts for Cyber Threat Intelligence Tasks

  This chapter presents how contextualized tagging can help transformer language models (LM) learn improved feature representation for the CTI keyphrase extraction task. This technique introduces a hybrid framework with a generalization objective that validates the importance of contextual tagging in fine-tuning transformer models by showing experimental results using standard performance metrics.

- Chapter 3 - Attention Aware Parametric Kernel GNN for Classifying Cyber-Threat Types

  This chapter introduces a novel GNN layer architecture to construct a GNN model as a

semi-supervised learning technique for a node classification task that is more accurate than all reported SOTA approaches on standard benchmark data. The implemented GNN model is adapted to classify cyber-threat type from raw tweets by transforming the text data into a text graph.

- Chapter 4 - Event Detection and Ranking of Cyber-Threat Events

  My application goal is to contribute to building an effective cyber-threat awareness system. Thus, in this chapter, I present an approach that identifies and ranks novel and trending cyber-threat events. To the best of my knowledge, this was the first time both novel and trending events were detected and ranked simultaneously on Twitter. The findings of this research are validated by investigating true incidents.

- Chapter 5 Tracing Relevant Twitter Accounts in CTI Domain

  This chapter introduces another essential component of cyber-threat awareness: tracing CTI-relevant users from the Twitter network by analyzing the content of user accounts and their relational structure.

- Chapter 6 - Conclusions and Future Research

  This chapter summarizes contributions and future direction of the continuing research.

The research in this dissertation contributes to efficient representation learning using short texts for downstream deep learning tasks. Altogether, the combined framework provides a cyber-threat awareness system that will facilitate the cybersecurity action process by (i) boosting the accuracy of threat classification using an APKGNN-based text classifier that outperforms SOTA baselines, (ii) identifying CTI-relevant key phrases through improved sequence labels from transformers, (iii) improving the account tracing of active CTI users over SOTA baselines, and (iv) implementing a novel event detection and ranking technique. Individual performance of each of the sub-tasks mentioned above is evaluated based on standard quantifiable metrics such as accuracy, precision, recall, and F1 score.

Chapter 2

Extracting Key Phrases from Short Texts for Cyber-Threat Intelligence Tasks

2.1 Introduction

Extracting cybersecurity-relevant informative tokens from text documents on open-source domains such as CVE and NVD reports and Twitter and then mapping them to respective labels as a sequence labeling task has particular significance for cybersecurity researchers in detecting emerging cyber-attacks. Key phrase extraction from text corpora has been unquestionably prolific but remains highly challenging because open-source short texts have a peculiar lack of structure. Because manual extraction of information from these massive data sets is often infeasible or prohibitively expensive, the only choice left is to use automatic extraction approaches.

Named Entity (NE) Extraction/identification as an information extraction process is used as a pipeline in many other applications[7;13;14] of CTI, mapping a sequence of text tokens to predefined classes. However, existing NE extraction tools can usually only find NEs that are nouns or conjunctions of nouns. Moreover, raw text that has vital information conveyed by

corresponding tokens does not qualify as NEs because those tokens may have been tagged as other parts of speech, are wrongly formed, or derived from different languages. Thus, existing NE extraction/identification methods often cannot identify crucial CTI information, triggering the need to develop more robust techniques than current NE extraction processes.

Deep learning neural networks such as sequence-to-sequence models and transformers have predominated among methods for key phrase extraction/identification from text[15] because of their domain-independent adaptability across fields. They do not require feature engineering based upon specialized knowledge. Heuristic-empowered extraction methods defined by a set of rules have some initial benefits, but incorporating extraction tagging rules with deep learning (DL) provides the most convenience in a model. On the other hand, DL methods have trouble extracting rare entities, acronyms, and abbreviations and are limited in learning from text documents relevant to cybersecurity if they were different lengths because information from short texts is condensed, whereas descriptive reports spread across pages.

Effective and insightful key phrase extraction from raw texts is not straightforward because contextual information is lacking; key information may remain unobserved by even some robust key phrase identification processes that are either DL-based or heuristic-based. DL-based LM architectures naturally facilitate context-learning of entities by focusing semantic structure of inputting text documents. On the other hand, the semantic role labeling (SRL) process can contextualize text documents based on the central verb. Therefore, a potential solution may be to unify these two context-aware modules to establish a hybrid system that adopts fruitful concepts from both domains. Thus, considering the contextual limitations of inputting text data and the huge effort required in annotating text corpora, we developed a set of SRL-powered generic extraction tagging rules. We fed text documents into this module to get a fine-grained, tagged sequence from the inputting documents. Next, we adopted a DL-based sequence labeling task of NE extraction[1] along with generalized extraction tagging rules to formulate a hybrid key phrase identification method that supports domain transferability.

The set of tagging rules we developed was empirically validated and analytically struc-

tured to generate compatible tagged sequence as input to the DL LM architectures. Our contributions are as follows:

1. We proposed a set of generalized extraction tagging rules for key phrase extraction;

2. We fed the tagged data set to some prominent DL-based transformers and statistical language model architectures to determine the suitable learning architectures for specific use cases;

3. We validated the applicability of extraction tagging rules by applying the ROUGE metric calculation between a sample tagged data set and its corresponding annotated data set;

4. We designed, implemented, and experimented with an end-to-end framework that combined the developed extraction tagging module and learning framework module which achieved better generalization performance.

2.2 Related Work

This section gives information on earlier approaches to key phrase extraction and NE extraction from text documents specifically in cybersecurity domain and that apply heuristic, statistical, ML/DL-based techniques.

2.2.1 Statistical Learning Approaches

As basic techniques key phrase extraction, statistical learning approaches compute the probabilities of a sequence of tokens in a text that enables labeling the tokens by suitable, particular classes. Examples of such approaches are Support Vector Machine (SVM) frameworks[16] or Conditional Random Field (CRF) frameworks[17] for extracting entity and context information about security vulnerabilities and attacks. Some models[17;18] took advantage of cyber security-specific rule dictionaries in conjunction with statistical ML methods to improve the performance of existing models. These statistical approaches depend on a large labeled text

corpus and require manual feature engineering steps that do not perform well nor are they cost-efficient.

2.2.2 Feature-Based Heuristic Approaches

The scarcity of labeled data sets from various open-source text corpora has paved the way for a generalized rule-based unsupervised approach to extract NEs[28]. However, the rules applied here require further domain-specific knowledge. Currently, the word embedding technique is replacing text-based features[29], but the technique cannot provide accurate information for finding key phrases. Knowledge based approaches[14;30;31] can increase accuracy in extracting key phrases, but the domain knowledge insufficiency, specifically in cybersecurity, has always been a problem.

2.2.3 ML/DL Based Approaches

Approaches that applied various ML architectures[32] such as vanilla BERT[33] for NE extraction are limited because they depend upon external knowledge bases. On the other hand, another approach applies bootstrap sampling for learning patterns generated from sample sets to label NEs from the test text sequence[34]. The label dependency tagging feature of CRF shows potential in the NE extraction tasks[35] and has been incorporated into different ML/DL architectures. More recently, NLP and cybersecurity researchers have used DL-based approaches applied to various DL architectures for key phrase extraction because DL is feature-independent. BiLSTM-CRF settings[36] and all its descendant approaches[37;38] for NE extraction have become state of the art (SOTA) techniques because they effectively apply word context as a primary step in token sequence tagging.

2.2.4 NE Extraction for CTI

The implications associated with extracting cybersecurity key phrases[39] can be well addressed using a combination of DL methods[40] that leverage the connection between the

character vector model and the word vector model with feature templates. The attention mechanism[41;42] and its combination with feature templates[43] facilitate extracting cyber security-relevant rare tokens from a text corpus. A joint module of a DL model and domain dictionary for generalized applicability and correctness for security entity extraction has been proposed[44], and in another study[14], a framework combined Stanford NER and Regular Expressions to detect cybersecurity NEs. Other research[45] uses vector space representations. DL architectures such as LSTM and BERT, have been frequently used in cybersecurity key phrase extraction tasks[46;47] that collect features locally and globally from the corpus.

This earlier research has shown incremental improvement in cybersecurity key phrases and entity extraction, but they did not provide more effective results through combining both context augmentation and DL techniques. We have results of different, open source, key phrase extraction tools that are very popular (see Tables 2.1 and 2.2). Therefore, the extraction performance must be further improved.

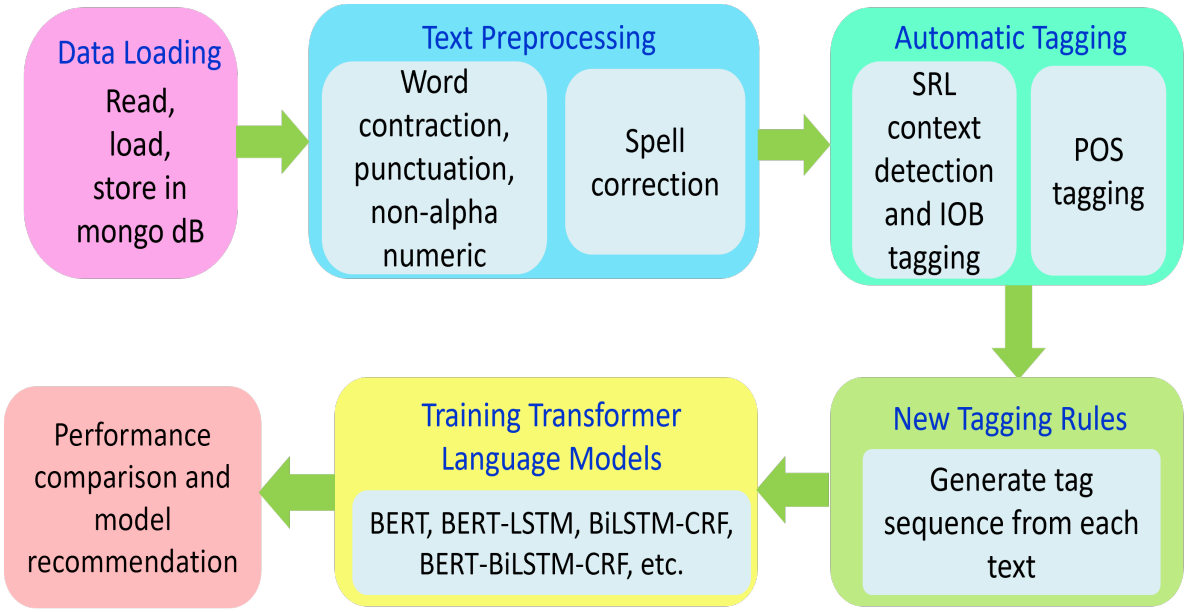

Figure 2.1: *Workflow diagram of new context-augmented key phrase extractor*

Table 2.1: *Comparing our developed hybrid system and current tools for identifying key phrases from a short text*

Clean Structured Text	A cyber attack in CSEDU has been committed by North Korean hackers	
Existing Tools	**Different test case of the given text**	**Remarks**
TextRazor	A cyber attack in csedu has been committed by north korean hackers	TextRazor could not detect all the key phrases after making all letters of certain NE tokens lower case, e.g., North Korean ⟶ north korean
	A cyber attack in CSEDU has been committed by north korea hackers	TextRazor detected only part of key phrase after nounifying the token *north korean* ⟶ north korea
Stanford NER	A cyber attack in csedu has been committed by north korean hackers	Stanford NER failed to detect any key phrase from the given lower case letter transformed text
	A cyber attack in Csedu has been committed by north korean hackers	Stanford NER detected only part of key phrase after making the token upper case *csedu* ⟶ Csedu
	A cyber attack in Csedu has been committed by North Korea Hackers	Stanford NER detected two key phrases only after nounifying and making the tokens upper case *csedu* ⟶ Csedu and *north korean* ⟶ North Korea
Spacy	A cyber attack in CSEDU has been committed by north korean hackers	Spacy detected only one after making the token upper case csedu ⟶ CSEDU
	A cyber attack in csedu has been committed by North Korean Hackers	Spacy failed to detect any key phrase from the given lower case letter transformed text
	A cyber attack in CSEDU has been committed by North Korea Hackers	Spacy failed to detect any part of a certain key phrase *North Korea Hackers* even though it was in upper case and nounified
	A cyber attack in CSEDU has been committed by North Korean Hackers	Spacy detected two key phrases, one after making the first NE token upper case; the second was only partly detected

Table 2.2: *Comparing our developed hybrid system and current tools for identifying key phrases from a short text*

Clean Structured Text	A cyber attack in CSEDU has been committed by North Korean hackers	
Existing Tools	**Different test case of the given text**	**Remarks**
IBM Watson	A cyber attack in csedu has been committed by North Korean Hackers	IBM Watson detected only one key phrase from the given text
Our developed tagging rules: tagging rules along with BERT-BiLSTM-CRF Model	A cyber attack in csedu has been committed by north korean hackers	All potential key phrases are tagged even after modifications
	A cyber attack in CSEDU has been committed by North Korean hackers	All potential key phrases are tagged even after modifications

2.3 Cybersecurity Keyphrase Identification Framework

Despite having many common ideas and processes, the difference between NE extraction and Key Phrase extraction lies in their purposes within a particular domain. Unlike NE extraction, which focuses only on extracting and identifying noun phrases, the key phrase extraction process tries to extract all key information that makes it appropriate for downstream CTI tasks. Nearly all generic NE extraction methods fail to detect text tokens mentioning emerging cyberthreats and foreign-language software entities because it is nearly impossible to assign a tag to each valid NE. On the other hand, key phrase extraction is important in the CTI domain because it accumulates subtle information from both structured long text and unstructured short text in which non-noun tokens also contain crucial information. Moreover, the purpose of CTI information extraction, even characterizing entity types, has significant value in CTI, although obtaining this vital information is a priority. Considering the requirement of the current needs in CTI, we designed a generalized set of extraction tagging rules to obtain key phrases by keeping their previously assigned types. Figure 2.1 presents our implemented framework for key phrase identification, which consisted of six steps.

2.3.1 Incorporating Heuristic Rules

Preparing Initial Data

To create primary test beds for applying the extractive tagging rules we developed, we first applied part of speech (POS) tagging and semantic role labeling (SRL) to every text document in the benchmark corpora. POS tagging defined micro-level feature information for each token present in a text, but semantic role labeling (SRL) defined verbs as predicates and associated other tokens with semantic roles represented as class-type arguments. In other words, SRL mapped the verb or predicate arguments as functions and other tokens to specific relations that corresponded to a specific function in a given sentence. NLTK's POS tagger tagged each token in an English text as follows: CC was the coordinating conjunction; CD was the cardinal digit; DT was the determiner; FW was a foreign word; IN was a preposition/subordinating conjunction; JJ+ was an adjective; MD was a modal; NN+ were nouns; RB+ were adverbs; VB+ were verbs. The symbol + showed all types of similar POS tags were included.

On the other hand, SRL-produced tagging included the only following three terms: (i) numbered arguments such as ARG0 and ARG1 plotted a middle course among possible different theoretical analyses and ensured consistent annotation; (ii) IOB tagging located the position relative to the verb; (iii) mnemonic ArgM modifier tags were LOC or location, CAU or cause, TMP or time, PNC or purpose, NEG or negation marker, DIR or direction. SRL usually generated two to four role arguments, but as many as six could appear based on the verbs and other tokens remaining adjuncts. A reference example follows: *i appreciate the love for hacking guys but let [ARG0: us] [ARGM-NEG: not] [V: dos] [ARG1: our own fun stream rip techno therapy] [ARG2: an above beyond].*

However, more than one verb in a sentence can cause ambiguity because of multiple SRL assignments. To solve this problem, we excluded all light verbs from being predicates and prioritized the verb written at the end of a sentence as a predicate. At the end of this step, we had POS tags and role labels for each token present in a given text.

Applying Extractive Tagging Rules

A crucial component of our new keyphrase extraction system was a set of tagging rules we developed that mapped POS and SRL-tagged sequences to the input of our transformer-based language model. Although the designed rules were not specific to cybersecurity, this simple but effective set of rules worked better for both structured and unstructured texts. These rules were formulated empirically to synthesize the intrinsic structure of sentences obtained from POS and SRL tags. These rules avoided using overconstrained, domain-specific pattern matching techniques such as gazetteers that are too brittle to adapt to new domains, such as different cybersecurity-related domains, and thus omitted key role-specific information in context. For example, for the text document *A cyber attack in CSEDU has been committed by North Korean hackers*, SRL tagged representation is A cyber attack in CSEDU[B-ARG1, I-ARG1, I-ARG1, I-ARG1, I-ARG1] has[O] been[O] committed[B-V] by North Korean hackers[B-ARG0, I-ARG1, I-ARG1, I-ARG1]. Here, the role verb *committed* created the relation context that the rule-based tagger could use with a token POS tag to extract the key phrases by tagging respective tokens.

SRL synthesizes a text document into multiple contexts (examples given in subsection 2.3.1.1 and subsection 2.3.1.2) according to the roles based on the central verb of the text document. Our rules mentioned (see subsection 2.3.1.2) considered the position of different arguments in the contexts and extracted noun phrases from them. If informative tokens in the context mapped to other POS other than nouns, the rules transformed them into Noun phrases. The resulting tagging sequence was mapped to the respective token where a tag determined a token's presence in the set of key phrases. Unlike existing tagging rules (too brittle for generic Benchmark 2[25] data set) that attempted to tag only specific CTI text tokens, our new rules considered the POS tag sequence of tokens inside the context (generated by SRL for tagging text tokens of a text document) for key phrase extraction tasks. Our new set of rules does not constrain itself to a particular domain data set (Benchmark 1[26]), so it supports cross-domain transfer. In the following itemized points, we describe the set of rules that we designed for tagging key phrases.

For any set of tokens T and SRL argument tags A , the set of all functions $f_{SRL}(T)$:T→A is denoted as A = $f_{SRL}(T)$ = *SRL(T)*

For any set of tokens T and POS tag P , the set of all functions $f_{POS}(T)$:T→P is denoted as P = $f_{POS}(T)$ = *POS(T)*

For any set of tokens T and Nounified token N, the set of all functions $f_{NOUN}(T)$:T→N is denoted as P = $f_{NOUN}(T)$ = *NOUN(T)*

1. Up to two consecutive noun phrases (NN+) where the first one is an initial argument (B-ARG) and the second is an intermediate argument (I-ARG) preceded by a determiner (DT/IN) are jointly considered a key phrase.

$$\begin{aligned} key_j = \{T_iT_{i+1} : i \in \{1, ..., |S_k|\}, \\ f_{SRL}(T_i) \in \{NN+\}, \\ f_{SRL}(T_{i+1}) \in \{NN+\}, \\ f_{POS}(T_i) \in \{B-ARG\}, \\ f_{POS}(T_{i+1}) \in \{I-ARG\}, \\ f_{POS}(T_{i-1}) \in \{DT/IN\}\}. \end{aligned} \tag{2.1}$$

2. A noun phrase (NN+) tagged by an initial argument (I-ARG) preceded by a determiner (DT/IN) is considered a key phrase.

$$\begin{aligned} key_j = \{T_i : i \in \{1, ..., |S_k|\}, \\ f_{SRL}(T_i) \in \{NN+\}, \\ f_{POS}(T_i) \in \{I-ARG\}, \\ f_{POS}(T_{i-1}) \in \{DT/IN\}\}. \end{aligned} \tag{2.2}$$

3. If any determiner (DT/IN) is present just before an adjective (JJ), the adjective is an intermediate argument (I-ARG), and a noun (NN+) is located just after the adjective (JJ+), the adjective and noun are jointly considered a key phrase.

$$\begin{aligned} key_j = \{T_iT_{i+1} : i \in \{1, ..., |S_k|\}, \\ f_{SRL}(T_i) \in \{JJ+\}, \\ f_{SRL}(T_{i+1}) \in \{NN+\}, \\ f_{POS}(T_i) \in \{I - ARG\}, \\ f_{POS}(T_{i+1}) \in \{I - ARG/B - ARG\}, \\ f_{POS}(T_{i-1}) \in \{DT/IN\}\}. \end{aligned} \tag{2.3}$$

4. If up to two adjectives (JJ) are followed by a noun phrase (NN), they are considered a key phrase.

$$\begin{aligned} key_j = \{T_{i-1}T_iT_{i+1} : i \in \{1, ..., |S_k|\}, \\ f_{SRL}(T_{i-1}) \in \{JJ+\}, \\ f_{SRL}(T_i) \in \{JJ+\}, \\ f_{SRL}(T_{i+1}) \in \{NN+\}\}. \end{aligned} \tag{2.4}$$

5. We nounify adjective (JJ+), adverb (RB), and verb (VB) and check to see if any of the noun forms are related to cybersecurity or not.

$$\begin{aligned} key_j = \{T_i : i \in \{1, ..., |S_k|\}, \\ f_{NOUN}(T_i) \in \{NN+\}\}. \end{aligned} \tag{2.5}$$

6. Any foreign language token or token length of more than 3 written with alphanumeric letters are considered key phrases.

$$key_j = \{T_i : i \in \{1, ..., |S_k|\}, \\ f_{POS}(T_i) \in \{FW\} \cup |T_i| \geq 3\}. \tag{2.6}$$

Here, i represents the token number of a token part of a key phrase that can be any number from 1 to $|S_k|$, where $|S_k|$ is the length of the key phrase.

Figure 2.2 shows a step diagram of our developed extraction tagging rules. DL transformer language models do not learn well if they receive highly constrained target labels against training data. Following this concept, too many type-specific CTI tags as labels would reduce the learning of LMs to predict key phrases from a text document. However, our contextualized set of rules produces generic tagging of tokens that does not limit the performance of LMs and results in a higher overall accuracy boost. So, new tagging rules are effective in tandem with transformers, generating better results, which means hybridizing two modules is transformer-friendly.

2.3.2 Adopted Statistical and Neural Network Models

BERT Pre-trained Language Model

Pre-trained language models of Bidirectional Encoder Representations from Transformers (BERT)[33] performed better than Word2Vec to generate contextualized embeddings for words present in a sentence by introducing two new ideas: (i) masked language model (MLM) and next sentence prediction (NSP). The MLM technique masks some words from a provided sentence to generate masked tokens according to the context. However, the second technique tries to predict the next sentence by feeding two randomly selected sentences as input. These two ideas leverage BERT as an embedding to include attention-focused, multi-layer, bidirectional, and nonlinear correlation constraint layers for sequence labeling tasks. BERT

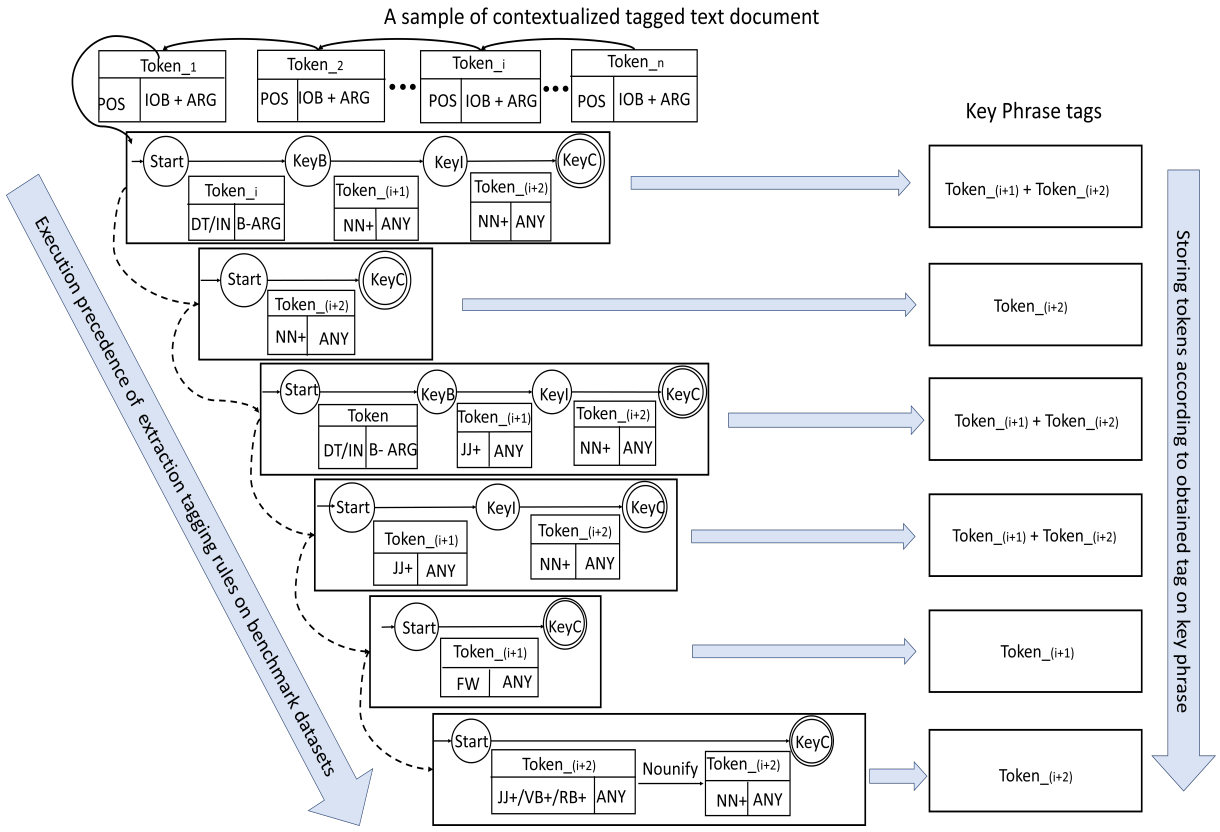

Figure 2.2: *Illustration of our new tagging rules with their respective steps for key phrase tagging*

Table 2.3: *Results for baseline tagging rules and our rules tandem with different statistical and transformer-based models*[1] *applied to CVE & NVD-specific and general cybersecurity corpora*

Corpora	Model	Prec	Rec	F1
Merged CVE &	BERT	96.40	97.13	96.77
NVD-specific	CRF	84.00	77.00	79.00
corpus tagged	BERT-CRF	97.36	97.29	97.33
using baseline	BiLSTM-CRF (Word2Vec)	94.78	89.70	92.17
rules	BERT-BiLSTM-CRF	**97.82**	**97.37**	**97.59**
Merged CVE &	BERT	90.34	91.54	90.94
NVD-specific	CRF	88.00	86.00	87.00
corpus tagged	BERT-CRF	91.39	92.16	91.77
using our new	BiLSTM-CRF (Word2Vec)	82.30	83.67	82.98
rules	BERT-BiLSTM-CRF	92.36	92.21	92.28
Twitter general	BERT	51.42	49.18	50.28
cybersecurity	CRF	70.00	54.00	60.00
corpus tagged	BERT-CRF	66.97	52.14	58.63
using baseline	BiLSTM-CRF (Word2Vec)	44.58	45.83	45.19
rules	BERT-BiLSTM-CRF	63.63	70.68	66.97
Twitter general	BERT	80.44	80.55	80.49
cybersecurity	CRF	82.00	81.00	82.00
corpus tagged	BERT-CRF	82.38	83.41	82.89
using our new	BiLSTM-CRF (Word2Vec)	70.02	69.60	69.81
rules	BERT-BiLSTM-CRF	**84.01**	**83.89**	**83.95**

must be trained on a large corpus, so adopting a pre-trained BERT model and fine-tuning it with cybersecurity data sets is reasonable. BERT has used contextual information learning and transferability, so it can be certainly used as an embedding layer for key phrase extraction as a sequence labeling task.

BiLSTM Layer

The applicability of Bidirectional LSTM or BiLSTM in sequence processing tasks leads to further improvement that simultaneously analyzes both forward (future) and backward (past) contextual information for a given text. Unlike RNN, BiLSTM inherits similar advantages that LSTM introduced, such as addressing vanishing and exploding gradient problems.

CRF Model

Unlike the Softmax layer on top of a neural network that independently predicts labels based on the highest score among label probability distributions, a CRF layer as an output layer formulated by Conditional Random Field performs better in constraint labeling settings where labels are interdependent. A CRF layer used as a classification layer along with an LSTM network decodes vector information generated by the LSTM network. CRF outperforms Softmax in automatic learning of constraints between labels, such as in IOB tagging in which the word label should begin with "O-type" or "B-type" but not with "I-type". The CRF layer predicts labels by combining the score Q of the output vector matrix generated from the BiLSTM layer and probabilities from label transition matrix T. For an input sequence X = $(x_1, x_2, ..., x_n)$ and label sequence Y = $(y_1, y_2, ..., y_3)$ Q_{iyi} represents the score of the ith word in the input sequence with y_i tag, and $T_{yi,yi+1}$ represents the transition probability from tag y_i to tag y_{i+1}. The following equation is used to calculate the score to predict sequence labeling:

$$score(X, Y) = [\sum_{i=1}^{n} Q_{i,yi} + \sum_{i=1}^{n} T_{yi,yi+1}] \tag{2.7}$$

BERT-CRF Model

Taking advantage of a pre-trained BERT model as a contextualized embedding layer along with a CRF layer correlation constraint feature, we used the BERT-CRF model for the sequence labeling task of cybersecurity key phrase extraction.

BiLSTM-CRF Model with Word2Vec

We adopted the classic BiLSTM-CRF model[36] comprising BiLSTM recurrent neural network and CRF statistical learning techniques for this sequence labeling task. In a sentence, BiLSTM effectively uses the word context while CRF layers use tag sequence information. We used a pre-trained Word2Vec model as an embedding layer to this model.

BERT-BiLSTM-CRF Model

This model[1] is an extension of the previous model but comes with the added advantage of BERT, which is used as a contextualized embedding layer. The BiLSTM model gets its embedding input from the BERT embedding layer, where BiLSTM's hidden states concatenate outputs from forward and backward LSTM networks together to generate a feature vector matrix for the CRF layer. Figure 2.3 shows a block diagram of this model.

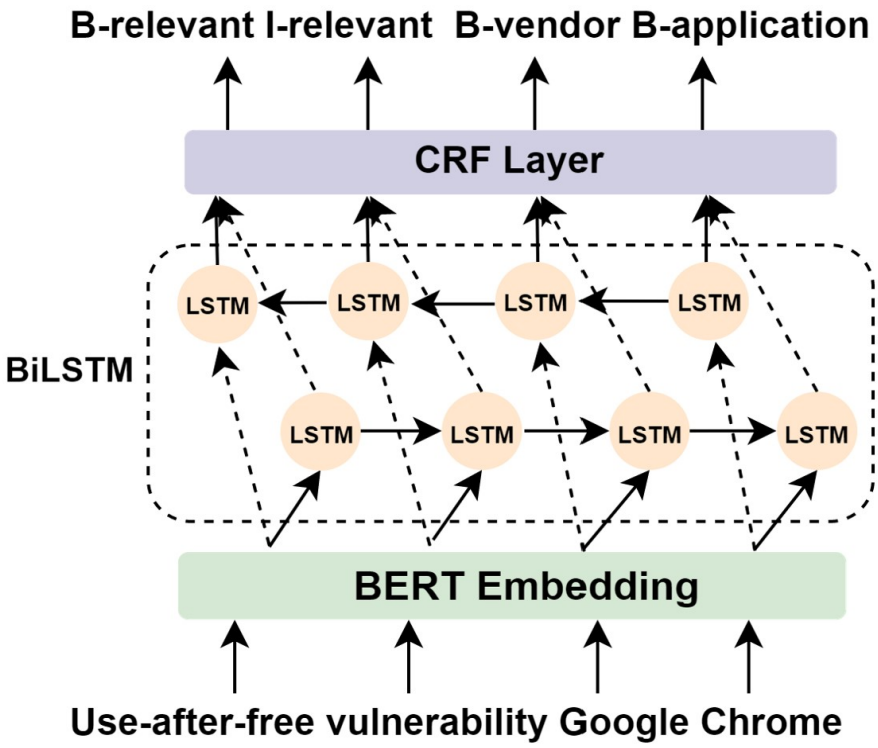

Figure 2.3: *BERT-BiLSTM-CRF model*[1]

2.4 Experiment and Evaluation

Block five and block six in Figure 2.1 present model training and experiment steps.

2.4.1 Data sets

Open-source cybersecurity data sets are significant to supporting new CTI applications. Here, we work with two benchmark data sets (i) benchmark 1[26] and (ii) benchmark 2[25] to train and evaluate different machine learning language models. The benchmark 1[26] data set comprises cybersecurity information from three different sources: NVD, Metasploit, and Microsoft Security Bulletins. The data set includes the following 15 entity types, among them, "software vendor", "software product", "software version", "software language", "vulnerability name", "software symbol", "OS", and "hardware", where the tagging rules are strictly constrained to tag entities from the data set. However, important cybersecurity information

beyond these 15 entity types, like rare entities, non-noun entities, and misspelled entities, cannot be detected using these tagging rules. This tagging rule thus works for a structured data set such as the benchmark 1 data set but is severely limited for an unstructured data set such as the benchmark 2 data set that was generated by a crawl from Twitter using security-related keywords. The data set was manually annotated based on the relevancy of each tweet to cybersecurity. The data set initially had 21368 clean tweet texts but, of those tweets, 11111 are related to cybersecurity. We used only cybersecurity-related tweets. We used the full text of a tweet if the tweet was not quoted or retweeted and the original tweet if the tweet was retweeted or quoted. We applied our extraction tagging rule to both data sets ([26] and[25]), tagging key phrases by KeyB, KeyI, KeyO, and KeyNone. To evaluate the validity of the purposed tagging rules, we annotated 100 tweets by extracting all possible combinations of key phrases from the tweet texts. Then we compared the tagged tweet texts from the 100-tweet sample data set against the annotated extracted key phrases from the same data set by calculating rouge scores.

2.4.2 Environment Setup

We used Python, PyTorch, and NLTK to implement our source code. We used two embedding layers, word2Vec and BERT in two different learning models, to compare performance. For the BiLSTM-CRF model, we set the maximum sequence length at 256, batch size at 8, learning rate at 0.00005, and round of training at 20 epochs. For the BERT-related models (BERT, BERT-CRF, BERT-BiLSTM-CRF), we used 512 for maximum sequence length, 32 for batch size, 0.00005 for learning rate, and 10 epochs for training rounds. The BERT pre-trained language models were tuned as the BERT embedding layer during the training process. All models were trained with a single Nvidia A40 GPU.

2.4.3 Compatibility of Tagging Rules

To evaluate the effectiveness of the tagging rules introduced above, we fine-tune our transformer models on a training data set **without** the tagging/extraction rules as a baseline, and

with them as an alternative treatment. Then we validate the resulting predicted sequences. Higher performance metric scores of a model obtained by fine-tuning with the tagged data set (e.g. Precision, Recall, F1) validate the importance of contextualized tagging of texts for transformers. Conversely, lower scores obtained from the model after fine-tuning with a small labeled data set represent a lack of expressiveness or generalization in the model. In other words, contextualized tagging can improve transformer model learning by guiding the transformer models to learn SRL information of texts. For this analysis, we use a pre-trained BERT-base-uncased model with two different model setups (I) without fine-tuning the rule-tagged data set from Twitter cybersecurity corpus, and (II) fine-tuned with this. We annotated a sample dataset of 200 tweets where we used the sample of the first 100 tweets for fine-tuning, and the remaining 100 tweets for testing. For the first model setup, we fine-tuned the whole tweet corpus and predicted the sequences for the 100 test tweet sample. For the second model setup, we use the first 100 labeled tweets for fine-tuning and the 100 test tweet sample for prediction. Table 2.4 shows the BERT-base-uncased model result for two experimental setups (batch size 4 and batch size 2) run on 15 epochs. The model performs worse when it is fine-tuned on 100 annotated tweets to predict sequence labels for 100 tweets. On the other hand, the model predicts a relatively correct sequence label for the 100 test tweet sample if this is fine-tuned on the data set generated by the newly introduced tagging rules.

Table 2.4: *Results for BERT-base-uncased model on Twitter general cybersecurity corpus*

Experiment Setup	Model Setup	Prec	Rec	F1
Batch size 4	Without tagging Rules	48.52	55.75	51.88
Batch size 4	With Tagging Rules	**65.45**	**63.78**	**64.60**
Batch size 2	Without tagging Rules	48.45	49.83	49.13
Batch size 2	With Tagging Rules	**62.40**	**66.90**	**64.57**

Table 2.5: *ROUGE-1 and Rouge-L score comparing generated key phrases against the annotated data set for each set of rules*

ROUGE Metric	Twitter data set	Prec	Rec	F1
ROUGE-1	Tagged with developed rule	52.19	35.05	40.09
	Tagged with baseline rule	20.45	28.52	21.49
ROUGE-L	Tagged with developed rule	31.96	27.61	31.96
	Tagged with baseline rule	19.40	26.93	20.45

2.4.4 Our Developed Tagging Rule Validation

The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) score[48] provided a set of performance evaluation metrics for various text sequence matching tasks such as summary generation[49], entity matching, and machine translation. We adopted the ROUGE score to evaluate how effective the developed extraction tagging rules were in tagging key phrases compared to the baseline tagging rules on the annotated sample of benchmark 2[25] data set. Table 2.5 provides the results of two ROUGE metrics ROUGE-N (where N=1) and Rouge-L with their corresponding performance indicators such as precision, recall, and F1 score for both developed and the baseline sets of rules. ROUGE-1 computed 1-gram matching performance and ROUGE-L computed the Longest Common Sub-sequence performance for the generated tagged data set and annotated data set. Figure 2.5 shows the density distribution of recall, precision, and F1 value of ROUGE-1 computed for the sample data set tagged by the baseline set of rules and the annotated tag labels. Figure 2.4 shows the density distribution of recall, precision, and F1 value of ROUGE-1 computed for the sample data set tagged by the set of developed rules and the annotated tag labels. Our developed set of rules outperformed the baseline tagging rules by a significant margin in tagging key phrases in a text. The tagged-data set was then fed as input to the machine learning models for sequence labeling.

2.4.5 Analysis of Results

Table 2.3 shows the performance evaluation of all models and the two different data sets ([42] and[25]). Clearly, any learning model using a generalized word2Vec embedding performs

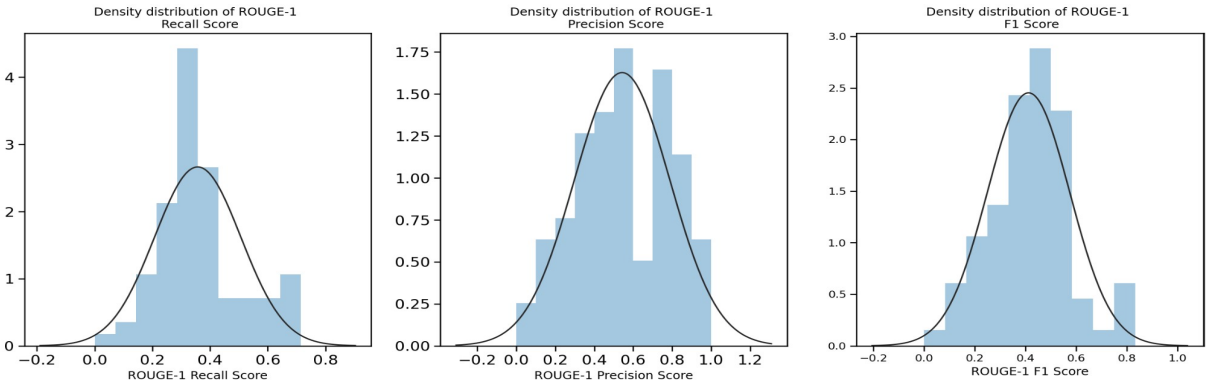

Figure 2.4: *Precision, Recall, and F1-Value of Rouge 1 score of developed rule-based tagger*

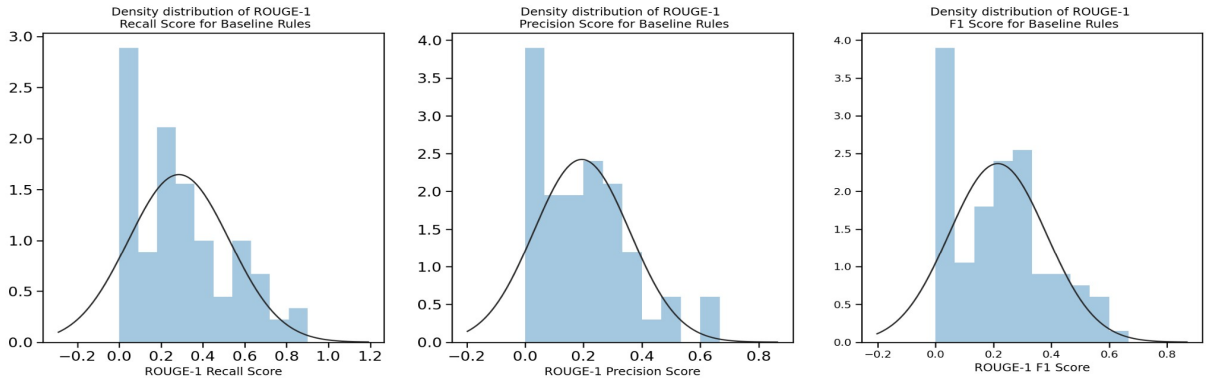

Figure 2.5: *Precision, Recall, and F1-Value of Rouge 1 score for baseline rule-based tagger*

the worst for both data sets. The BERT-BiLSTM-CRF model is the best-performing model overall, so we recommend this LM be used with our generic contextualized tagging rules for sequence labeling prediction tasks. All learning models trained on the benchmark 2 data set and tagged using our set of extraction tagging rules outperform learning models using baseline tagging rules. The BERT, BERT-CRF, BiLSTM-CRF, and BERT-BiLSTM-CRF models trained on the data set tagged by our new tagging rules outperform the same models trained on the data set tagged by the baseline rules. The BERT F1 scores are 60% higher, BERT-CRF results are 41.37% higher, BiLSTM-CRF 54.48% higher, and BERT-BiLSTM-CRF 25.35% higher. On the other hand, the models trained on the benchmark 1 data set tagged by the baseline rules outperform the same models trained on the benchmark 1 data set tagged by our newly developed set of rules. The F1 scores are 6.41%, 6.05%, 11.07%, and 5.7% higher (see Table 2.3). Baseline tagging rules slightly outperform our newly developed rules on the benchmark 1 data set because this data set is domain-dependent and has some repetitive and similar pattern security keywords (CVE-NVD-specific cyberthreat), easily identified by the highly-specific baseline rules. However, our generic set of rules worked much better for generic CTI data sets such as the benchmark 2 data set. The trade-off in using our new tagging rules includes slight performance loss in results on highly domain-dependent data sets. Attempting to improve the resulting performance on a domain-dependent data set would require the rules to be over-constrained, which in turn will reduce performance on more generic data. Because emerging cyberthreat incidents are ceaseless and have drawn the most scrutiny, limiting rules to a particular domain(s) would only make room for potential loss of security information.

2.5 Conclusion and Future Work

In this research, we presented a two-stage hybrid system combining a newly developed extraction tagging module with a recommended DL-based sequence label prediction module (BERT-BiLSTM-CRF) for key phrase extraction in the cyber-threat intelligence domain. We have also demonstrated in detailed experiments with different machine learning and sta-

tistical learning models that our claim is supported by the evaluation of their performances on two benchmark data sets tagged by both the newly developed and baseline rules. The ROUGE score validated the applicability of the newly developed set of tagging rules for key phrase extraction against a small sample annotated data set. Although our system works better for generic CTI (benchmark 2) data sets, this hybrid system can be further improved for particular data sets containing entities of specific patterns. We intend to apply the tagging rules to other knowledge bases to do so. Efficiency could also be increased by adding more layers to the current DL LMs to extract more discriminative features. We will continue developing our set of rules to obtain better performance indicator scores.

Chapter 3

Attention Aware Parametric Kernel Graph Neural Network for Classifying Cyber-Threat Types

3.1 Introduction

Despite being successfully applied to many real-life applications, such as image classification[50], object detection[51;52], machine translation[53;54], and speech recognition[55], CNN[56] and RNN[57] cannot be directly applied to generic graph structure data especially in non-Euclidean domains such as social networks, telecommunication networks, and epidemic networks because of the structural limitations of these deep neural networks (DNN). In general, grid-based (image) and sequential (text) data structures are independent and identically distributed (IID). On the other hand, images and text corpora can be considered subsets of graph-structured data; image pixels and documents can be considered as graph nodes in Euclidean space[58]. For the non-Euclidean domain, information from both properties of a graph, (i) the intrinsic features of nodes and (ii) the relationships between the nodes, is important. To learn information from both types of sources for a downstream predictive task such as classification, Graph Neural Networks (GNN) are a learning representation technique

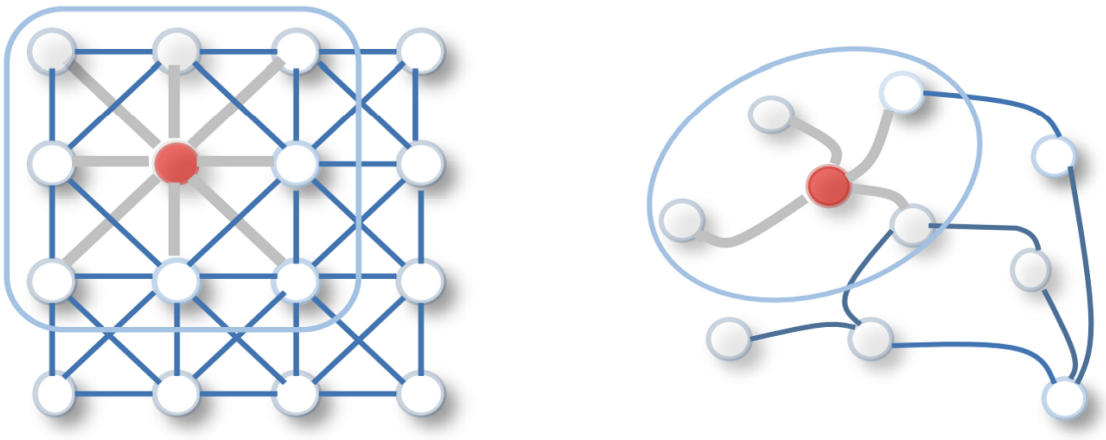

Figure 3.1: *Euclidean and Non-euclidean data structure*[2]

where the predictive result is calculated by neighborhood information aggregation. For many text classification tasks that are transformed into node classification tasks, semi-supervised learning approaches like GNNs outperform CNN models for which less labeled data limits learning models. Figure 3.1 shows the Euclidean and non-Euclidean data structure.

Existing GNN layers such as Monet[59] has a common problem in extracting representative local intrinsic patches because in it, a node feature attention is not considered by its neighbors. From a different direction, in an attention-based method such as GAT[23], edge attributes do not convolve with a parametric kernel, so the filters are not applied to edge attributes to project fixed-length transformed signals. Therefore, the results provide less representative patches for the downstream task of learning node representation. Consequently, node feature information and graph structural information without applying a parametric kernel (e.g., Gaussian mixture model) aggregated to form a scalar attention score for each edge will lead to poor learning representation. On the other hand, using a fixed kernel for neighborhood information aggregation as in GCN[3] also shows limitations in learning complex graph structural information to produce efficient learning representations for downstream node classification tasks.

For instance, a relationship between two humans depends on such things as their natures, favorite items, origins, interests, and food habits, and relationship elements like communication, proximity, and common social media platform. So the relationship between them

or the edge connecting two nodes having a single weight does not encode a graph structure efficiently in a latent space. Again, each edge feature is not equally important because they represent individual features with different objectives. So, a model aggregating two sub-module computations according to the number of specified kernels might be effective in learning a representation where the first one attends over the distribution of connected node features and the second one generates patches by convolving a parametric kernel over the edge attributes. The observation mentioned above becomes crucial when GNNs are applied to text graphs of short text corpora because the graph structure itself, by default, lacks contextual information, and any form of information loss information can affect the classification performance. Hence, we have low classification performance in many SOTA approaches.

In this method, the local intrinsic patches (receptive fields) are extracted after the parametric kernel is applied to edge attributes (pseudo-coordinates). The result is then linearly combined with node feature attention to form augmented local patches. The augmented local patches convolve with node features as a template matching process between augmented local patches and signals of nodes in a graph.

The cost of labeling open-source social media data is high enough that GNNs as semi-supervised learning techniques are applied to the CTI data set (Social media corpora containing raw tweets) for threat type classification. We applied the SOTA GNN models, including our implemented GNN model APKGNN on the CTI data for cyber-threat classification where the text categories of the CTI data set are mapped to different types of such cyber-threats as 0day, malware, and botnet.

In this study, I contributed (i) implementing a novel GNN architecture and (ii) its application to a cybersecurity raw text corpus obtained from Twitter. Our contributions are listed below:

- I implemented attention aware parametric kernel augmented GNN (APKGNN) layer architecture to implement a robust GNN model;
- I evaluated the implemented model on benchmark graph data sets, which gained performance over all of the SOTA models by a significant margin;

- I designed a GNN model building pipeline by applying all considered GNN layers where the pipeline results show the models outperform even their corresponding original implementations.

- I also applied the implemented GNN model on the CTI data set for cyber-threat type classification where the implemented model outperforms all SOTA models.

3.2 Related Work

The advent of graph neural network (GNN) was first inspired by recurrent neural networks applied to directed acyclic graphs Frasconi et al. (1998)[60] and Sperduti et al. (1997)[61]. Later GNN was introduced by Gori et al. (2005)[62] which was then extended by Scarselli et al. (2009)[61] and further improved by Li et al. (2016)[63] to handle general types of graphs. These methods recursively exchanged neighborhood information as a propagation method until a stable equilibrium was reached. Semi-supervised learning was first introduced in graph learning using a graph Laplacian regularization approach that includes methods such as label propagation by Zhu et al. (2003)[64], label spreading by Zhou et al. (2003)[65], manifold regularization by Belkin et al. (2006)[66], semi-supervised embedding by Weston et al. (2012)[67] applied to attribute graphs. Considering structural correlation among data samples, the non-spectral technique of graph convolution was introduced in Duvenaud et al. (2015)[68] and GraphSage Hamilton et al. (2017)[69] for graph-level classification, and in Atwood & Towsley (2016)[70] for node classification. These approaches require learning weight matrices either for each node degree or for both input channel and neighborhood degree (using the power of transition matrix) and do not scale to large graphs with a wide range of node degree distributions. Niepert et al. (2016)[71] requires normalizing fixed neighborhoods of a graph before converting them locally into ordered node sequences to feed into a convolutional neural network. Spectral graph convolutional networks, first introduced by Bruna et al, (2014)[72], are inspired by graph Fourier analysis, which assumes that a filter can be defined using a set of learnable parameters and considers graph signals with multiple channels. After that,

Henaff et al. (2015)[73] parameterized spectral filters to make them spatially localized using smoothing coefficients. In a follow-up, Defferrard et al. (2016)[24] extended the research by approximating the filters using Chebyshev expansion of the graph Laplacian through avoiding computation of Laplacian eigenvectors to yield spatially localized convolutions. After that, Kipf et al. (2017)[3] simplified the ChebNet approximation using filters restricted to operating up to the first order neighborhood of each node. An extension of this GCN was introduced by Klicpera et al. (2019)[74], improving GCN's over-smoothing by adopting the PageRank algorithm. Unlike GCN that assumes equal contributions of neighboring nodes by assigning non-parametric weights to the edges, GAT (Velickovic et al. (2018)[23]), inspired by the attention mechanism (Bahdanau et al. (2015))[75] that could handle variable size neighborhoods, attended highly important parts of the inputs to learn the relative weights between two connected nodes. A significant, parametric kernel-based approach (referred as GMM throughout this chapter) proposed by Monti et al. (2017)[59] assigned weights to a node's neighbors by mapping with relative positions of nodes within a neighborhood by introducing a feature called node pseudo-coordinates.

3.3 Deep Learning on Graphs

3.3.1 Spectral and Spatial based approaches

GNN architectures are theoretically categorized into two types of approaches: spectral approaches and spatial approaches. Spectral graph convolution[72;73] originated from network signal spectral analysis where a signal X $\in R^F$ or a node's set of input channel $X^{in} = (x_1^{in}, ..., x_p^{in})$ is multiplied with a parameterized filter G_θ in the Fourier domain:

$$g_\theta * X = VG_\theta V^T X = VG_\theta(\Lambda)V^T X \tag{3.1}$$

where V is the eigenvector matrix of the normalized graph Laplacian L, θ is the set of

learnable parameters, and R is the set of all real numbers. The normalized graph Laplacian matrix can be represented as follows:

$$L = I_N - D^{-1/2}AD^{-1/2} = V\Lambda V^T \tag{3.2}$$

where V^T X is the Fourier transform of graph signal X, Λ is a diagonal eigenvalue matrix, and G_θ is a function of eigenvalues $G_{l,j} = diag(g_{l,j,1}, ...g_{l,j,k})$ with learnable parameters of normalized graph Laplacian L.

Some major drawbacks of the basic spectral approaches are that the filter depends on a given graph and the learned filter cannot be applied to different graphs; computationally expensive Fourier transformation; and filters cannot guarantee spatial consistency. After simplifying the higher order Chebyshev polynomial by ChebyNet[24], the convolution of the input signal X with a filter is represented as

$$\begin{aligned} g_\theta * X &= V(\Sigma_{p=0}^{K}\theta_p T_p(\Lambda))V^T X \\ &\sim \Sigma_{p=0}^{K}\theta_p T_p(L')X \end{aligned} \tag{3.3}$$

where $T_p(x)$ is the recursively defined truncated expansion of the Chebyshev polynomials, K is the order of polynomials, and L' is the normalized Laplacian.

On the other hand, spatial GNN approaches[68–71] rely on the spatial information of a node: its location in a graph, its neighbors, its degrees, for learning graph representation where the convolution operation propagates node information along the edge connections. The spatial filter can work with variable-sized neighborhoods and can be generalized across other domains. Unlike earlier spatial-based approaches that had fixed filter operations[3;74] to extract local intrinsic patches on the graph, spatial approaches formulating patch operators as a function of pseudo-coordinates[59] perform better for the complex node classification task. The general form of a spatial GNN can be written as follows:

$$H^i = f(XW^i + \sigma_{i=1}^{i-1} AH^{i-1}\theta^i) \tag{3.4}$$

where H is the hidden layer feature matrix, and W represents the linear transformation weight matrix. Figure 3.2 shows a simple two-layer GNN network.

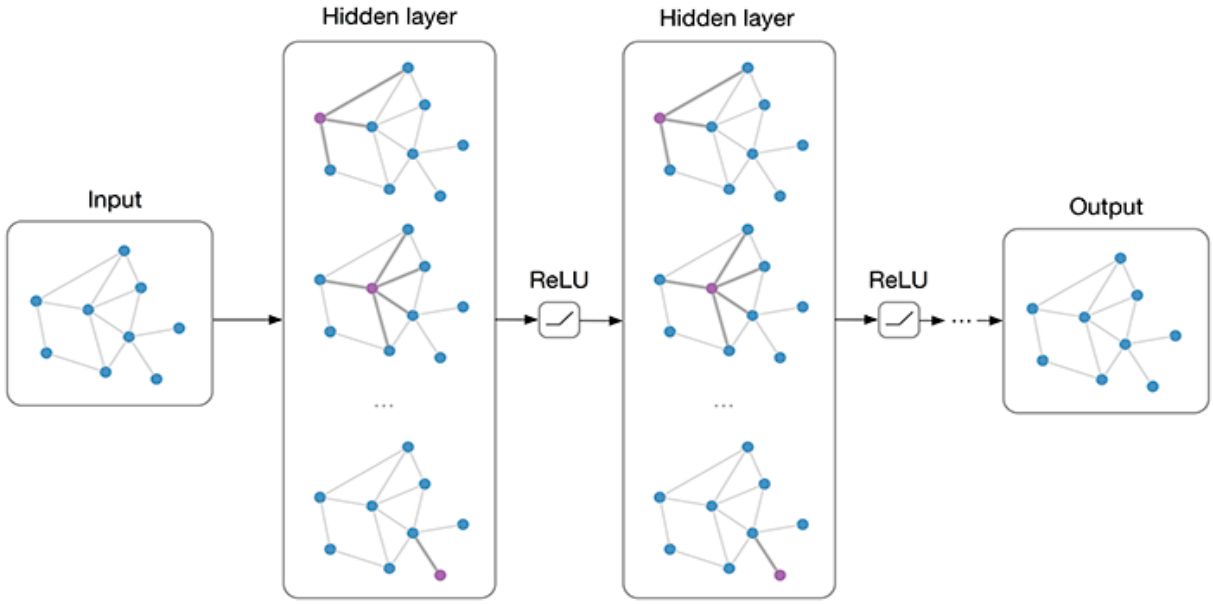

Figure 3.2: *A two leyer GNN network*[3]

3.3.2 Parametric Kernel for Extracting Patches

However, a parametric kernel[59] can significantly improve the performance of node classification of spatial GNNs if the patch operator can be represented by Gaussian mixture model (GMM) kernels. Parametric kernel function (weight function) $(w_\theta(u) = (w_1(u), ..., w_K(u)))$ can encode graph spatial information into low dimensional space where the dimension depends on the number of kernels applied to the d-dimensional vector of pseudo-coordinates u(x, y):

$$u(x,y) = (\frac{1}{\sqrt{deg(x)}}, \frac{1}{\sqrt{deg(y)}}) \tag{3.5}$$

Here x is a node in a graph, and y is the neighbor of x (e.g., $y \in N(x)$), θ is learnable parameters, and K is the dimensionality of the extracted patch. The patch operator can therefore be written in the following general form:

$$D_K(x)f = \Sigma_{y \in N(x)} w_K u(x,y) f(y) \tag{3.6}$$

On the other hand, the parametric kernel function (weight function) is defined as follows:

$$w_K(u) = exp(\frac{-1}{2}(u-\mu_k)^T \Sigma_k^{-1}(u-\mu_k)) \tag{3.7}$$

where μ represents the mean, and Σ represents the covariance matrix of the GMM.

3.3.3 Self Attention in GNN

Attention mechanism[75] applied to graph neural networks[23] allows the representation of a central node to attend to its most important nodes in the neighborhood. A graph attention network (GAT) as a spatial-based approach relies on the locality of nodes, so for a given node, this aggregates neighboring node features X ($\{\overrightarrow{x_1}, \overrightarrow{x_2}, ..., \overrightarrow{x_N}\}$, $\overrightarrow{x_i} \in R^F$) after attending them. Here N is the number of nodes, and F is the number of node features. Each GAT layer produces a new node feature vector X ($\{\overrightarrow{x'_1}, \overrightarrow{x'_2}, ..., \overrightarrow{x'_N}\}$, $\overrightarrow{x'_i} \in R^{F'}$) after a learnable linear transformation is computed by a weight matrix W $\in R^{F \times F'}$. To calculate the importance of neighboring node j features to node i ($j \in \mathcal{N}_i$), the concept of attention coefficients has been introduced:

$$e_{ij} = a(W\overrightarrow{x_i}, W\overrightarrow{x_j}) \tag{3.8}$$

Here, the attention function a is a single-layer, feed-forward neural network that applies LeakyRelu non-linearity parameterized by a weight vector $\overrightarrow{a} \in R^{2f'}$. Thus, Equation 3.8 can be rewritten:

$$a(W\overrightarrow{x_i}, W\overrightarrow{x_j}) = LeakyReLU(\overrightarrow{a}^T[W\overrightarrow{X}_i||W\overrightarrow{X}_j]) \tag{3.9}$$

where $.^T$ refers to matrix transposition, W refers to the linear transformation weight matrix, and || represents the concatenation of vectors. To scale the different coefficient values over each node's neighborhood, the coefficients are normalized as follows:

$$a_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\Sigma_{q\in\mathcal{N}} exp(e_{iq})}. \tag{3.10}$$

Edge attributes do not convolve with a parametric kernel, so the filters are not applied to edge attributes to project fixed-length transformed signals.

Attending the combined information of node features and edge attributes (without convolving with the parametric kernel) results in an imperfect learning representation for a central node because the linear combination of the two sources of information (node features and edge attributes) generates a scalar value that does not effectively project into a transformed representation according to the specified number of kernels. In contrast, parametric kernel-based approaches do not take attention into account. Therefore, GNN architecture must improve representation learning by adopting the useful features of these two techniques. Moreover, unlike parametric kernel techniques, the attention model uses node features for computing similarity without considering node structural properties. This reveals a research gap: not knowing the graph structure upfront for the classification task.

3.4 Implemented Technique

We will first describe our APKGNN convolutional layer, the layer structure used to implement the GNN model to generate node representation learning. We have two different inputs to this layer:

1. A set of node features, X = $(\{\overrightarrow{x_1}, \overrightarrow{x_2}, ..., \overrightarrow{x_F}\},\ \overrightarrow{x_i} \in R^N)$
2. A set of neighboring node d-dimensional pseudo-coordinates $u(x, y)$ defined as $u \in \sum_1^M e$, where e is a single edge connection.

From these two sets of inputs, the edge attribute vectors are represented as $E = 2\sum_1^M e \times d$, where M is the number of total edge connections, d is the dimension of edge attributes, N is the number of nodes, F is the number of node features, x is the given node, and $y \in \mathcal{N}(x)$ is the neighboring node of x.

We first self-attend neighboring node features (features of local patches) X of a central node x, so we can define which features contribute more in learning a better graph representation. The formulation can be written as follows:

$$\begin{aligned} att\_vec_k &= softmax(LeakyReLU([\theta_k \overrightarrow{X}_i || \theta_k \overrightarrow{X}_j])), where K \in 1, 2, 3, ..., k \\ &= \frac{exp(LeakyReLU([\theta_k \overrightarrow{X}_i || \theta_k \overrightarrow{X}_j]))}{\sum_{l \in \mathcal{N}(x)} exp(LeakyReLU([\theta_k \overrightarrow{X}_i || \theta_k \overrightarrow{X}_l]))} \end{aligned} \tag{3.11}$$

where $K \in 1, 2, 3, ..., k$, att_vec_k is the attention vector for nodes in their corresponding neighborhoods, dimension is defined according to the number of specified kernels k, and $||$ is the concatenation operation.

We adopted the GMM-based parametric kernel function to extract patches from neighboring edge attributes (pseudo coordinates) in a local graph structure where the kernel function (weight function) maps relative positions (pseudo coordinates) to weight vectors. The size vectors are defined according to the number of kernels K. Therefore, the weight function applied to the edge attributes (or the relative positions or pseudo coordinates) is defined as follows:

$$w_\theta(u) = (w_1(u), ..., w_k(u)) \tag{3.12}$$

where θ refers to the kernel learnable parameters used to extract patches P from edge attributes E. The following equation presents the GMM operation on edge attributes to extract patches P:

$$\overrightarrow{P} = exp(\frac{-1}{2}(u - \mu_k)^T \Sigma_k^{-1}(u - \mu_k))). \tag{3.13}$$

The set of attention vectors is aggregated with the set of extracted patches after mapping these with respect to the number of kernels K. The resulting feature attended patches are then transformed by a parameter vector $\overrightarrow{q}$.

$$\overrightarrow{S} = \overrightarrow{q}(att\_vec_k + \overrightarrow{P}) \tag{3.14}$$

Finally, the set of updated patch vectors is convolved with the neighboring node features to compute node representations as the final output, followed by a linear activation as follows:

$$X^k = \sigma(\sum_{x \in \mathcal{N}(x)} \overrightarrow{S} * WX^{k-1}). \tag{3.15}$$

A two-layer APKGNN model is implemented for semi-supervised node classification on graph data sets. After applying all these steps, the forward model expression can be defined as follows:

$$Z = f(E, \theta, X) = softmax(W_\theta \odot ELU(W_\theta \odot X)) \tag{3.16}$$

where $\mathcal{Z}$ is the final convolved feature matrix.

Softmax activation applied to the final layer output can be defined as follows:

$softmax(x_i) = \frac{1}{Z} exp(x_i)$.

Therefore, $\mathcal{Z} = \sum_i exp(x_i)$.

For semi-supervised multi-class classification, the cross-entropy loss over the labeled example is defined as follows:

$$\mathcal{L} = -\sum_{l \in Y_L} \sum_{f=1}^{C} Y_{lf} ln Z_{lf} \tag{3.17}$$

where $\mathcal{Y}_L$ is the set of nodes that have labels and $\mathcal{C}$ is the set of output classes. The implemented GNN model is trained by mini-batch gradient descent on a full graph data set for some training epochs. The mini-batches are sampled by the neighborhood node sampler technique that was proposed in the GraphSage[69] algorithm. Figure 3.3 shows the operational steps of node classification using a graph neural network constructed by a novel APKGNN layer. This figure shows neighborhood sampling is applied to make a partition of an input graph into multiple subgraphs which are then processed in multi-processing settings. Finally, each process is executed by a prototype of an APKGNN network. 3.4 displays construction of the APKGNN layer by applying a parametric kernel with vector attention on edge attributes. For an input graph with node features, this figure shows how different parameters of the parametric kernel (μ, Σ) are learned and node features are transformed according to the number of kernels to produce patch vectors. Figure 3.5 shows node feature update operation with APKGNN layer output multiplied by current node features.

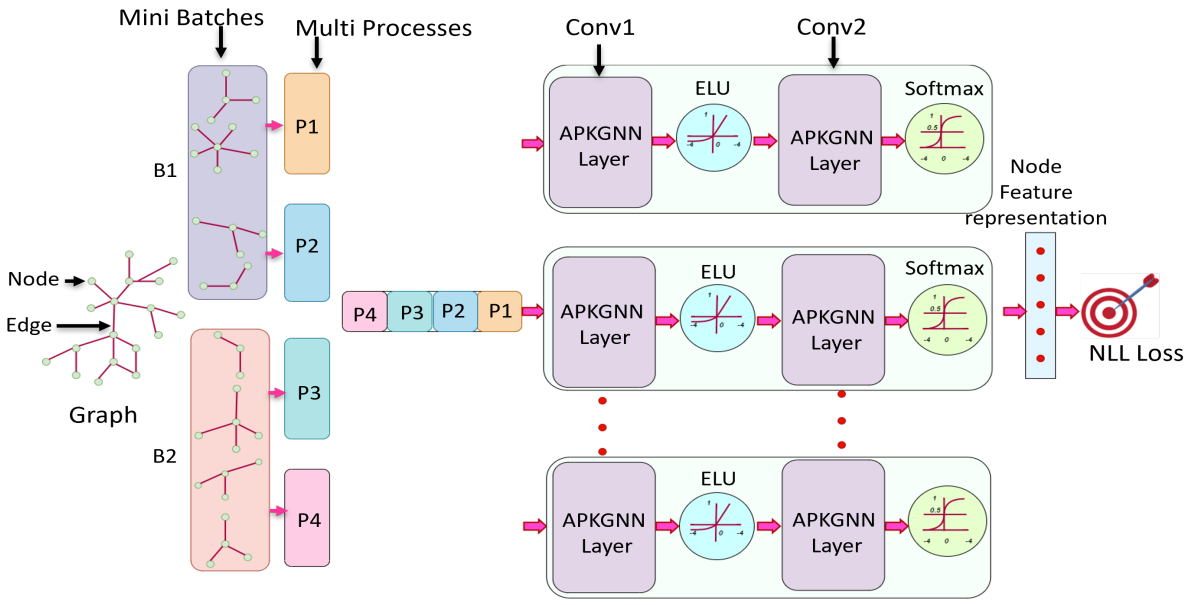

Figure 3.3: *Operational steps of node classification using novel implemented APKGNN layer)*

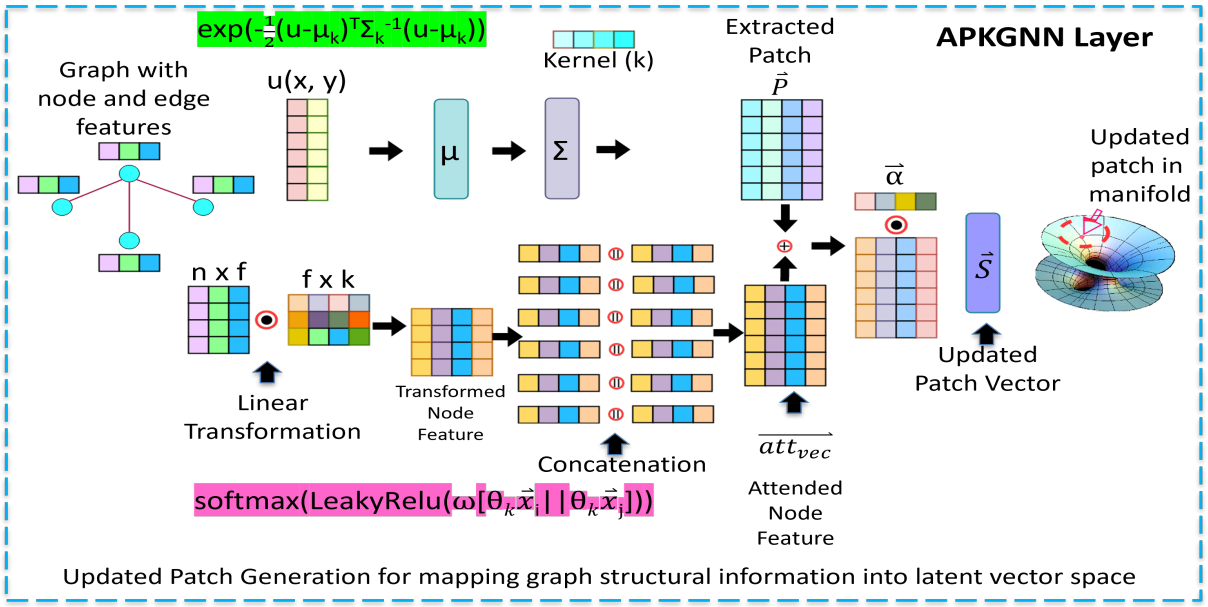

Figure 3.4: *Construction of APKGNN layer)*

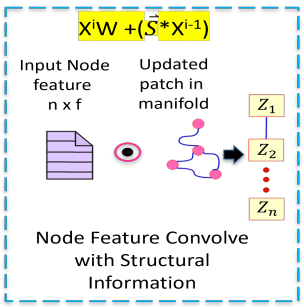

Figure 3.5: *Updating node representation using novel APKGNN layer)*

3.5 Experiments

We experimented with our implemented model on several graph benchmark data sets (citation networks) and one CTI text data set for semi-supervised node classification.

3.5.1 Data sets

We followed the Yang et al.[27] experimental setup for benchmark data sets (see Table 3.1) where the properties of the data sets are nodes, classes, number of features, and labels. Train mask, test mask, and validation mask from Table 3.2 denote the number of nodes used for training, testing, and validation, respectively.

Table 3.1: *Data set statistics for benchmark data sets and CTI data*

Dataset	Type	Nodes	Classes	Features	Label
Cora	Citation network	2,708	5,429	7	1,433
CiteSeer	Citation network	3327	4732	6	3703
Pubmed	Citation network	19,717	44,338	3	500
CTI	Text graph	23113	303074	10	8114

Table 3.2: *Training statistics for benchmark data sets and CTI data*

Dataset	Training Mask	Test Mask	Validation Mask
Cora	140	1000	500
CiteSeer	120	1000	500
Pubmed	60	1000	500
CTI	5000	2500	614

3.5.2 Experimental Setup

We conducted the experiments as transductive learning tasks where we trained a two-layer APKGNN as described in Section 3.3. We evaluated prediction accuracy on a test mask of 1,000 labeled nodes, with a validation mask of 500 labeled nodes for hyperparameter optimization, but we did not use the validation mask for training. We kept the dropout rate for all layers at 0.5 and weight decay at 0.0005. The first layer of the implemented GNN network was followed by an exponential linear unit (ELU)[76] nonlinearity whereas the second layer followed by a softmax activation was used for node classification that computes C features (C is the number of classes). The hidden layer unit size is 16, and the kernel size is 4.

For all the citation network data sets (Cora, Pubmed, and Citeseer), we used the same hyperparameter settings and trained all models for 50 epochs (training iterations). We used the Adam optimizer[77] with a learning rate of 0.01 and an early stop with a window size of 10. All the model and network parameters, including transformation weight matrices, were initialized using the method described in this work[78]. For the CTI data set, the experiment settings required slight changes because the data set was derived from raw tweets and the text graph was generated following the method described in TextGCN[79]. In this data set, document nodes did not have any direct connection; document nodes were instead connected by common word nodes shared by two different documents. In addition, only document nodes have labels (cyber-threat types) because words cannot have text categories or more specifically, cyber-threat types.

3.6 Results and Performance Evaluation

The experimental results are summarized in Table 3.4 with reported mean test and validation accuracies of different SOTA GNN models on benchmark data sets and our CTI data set. All models were run for 50 epochs with random mini-batch on training data samples using a neighborhood batch sampler adopted from the GraphSage[69] algorithm. For each citation benchmark data set (Cora, CiteSeer, and Pubmed) and for the CTI data set, mean test accuracy, mean validation accuracy, and mean training accuracy curves were plotted with SOTA GNN models compared to our newly introduced GNN model. Three seed numbers were used for three different simulations for each data set, and an average of three accuracy scores were plotted with corresponding standard deviations. We found that the proposed model outperformed every model in all combinations by a significant margin and converged in fewer epochs. Figures 3.6, 3.7, and 3.8 illustrate that the APKGNN model outperformed all SOTA models on test, training, and validation data samples of the Cora citation data set.

Table 3.3: *Performance of APKGNN compared to SOTA methods on validation and test accuracy for benchmark and CTI data sets based on implemented experimental set up*

Method	Cora		Pubmed		Citeseer		CTI	
	Test	Val	Test	Val	Test	Val	Test	Val
GCN[3]	0.8400	0.8560	0.8200	0.8000	0.7260	0.7260	0.42	0.43
GAT[23]	0.8460	0.8550	0.8140	0.7940	0.7280	0.7350	0.32	0.33
GMM[59]	0.8260	0.8360	0.8180	0.8090	0.7300	0.7180	0.24	0.23
SPLINECNN[80]	0.8400	0.8580	0.8120	0.8000	0.7320	**0.7600**	0.40	0.41
APKGNN	**0.8620**	**0.8750**	**0.8280**	**0.8240**	**0.7400**	0.7540	**0.44**	**0.45**

Table 3.4: *Performance of APKGNN compared to SPLINECNN on validation and test accuracy for benchmark data sets based on SPLINECNN experimental set up*

Method	Cora		Pubmed		Citeseer	
	Test	Val	Test	Val	Test	Val
SPLINECNN[80]	0.88.80	0.8780	0.8820	**0.8760**	0.7320	0.7850
APKGNN	**0.8920**	**0.8850**	**0.8880**	0.8740	**0.7400**	**0.7960**

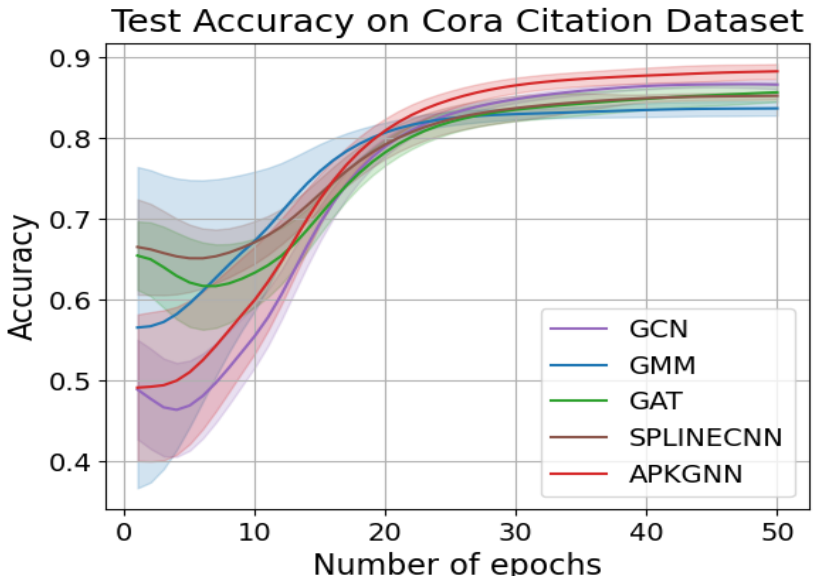

Figure 3.6: *Test accuracy curve up to fifty epochs for the CORA data set for all GNN layers*

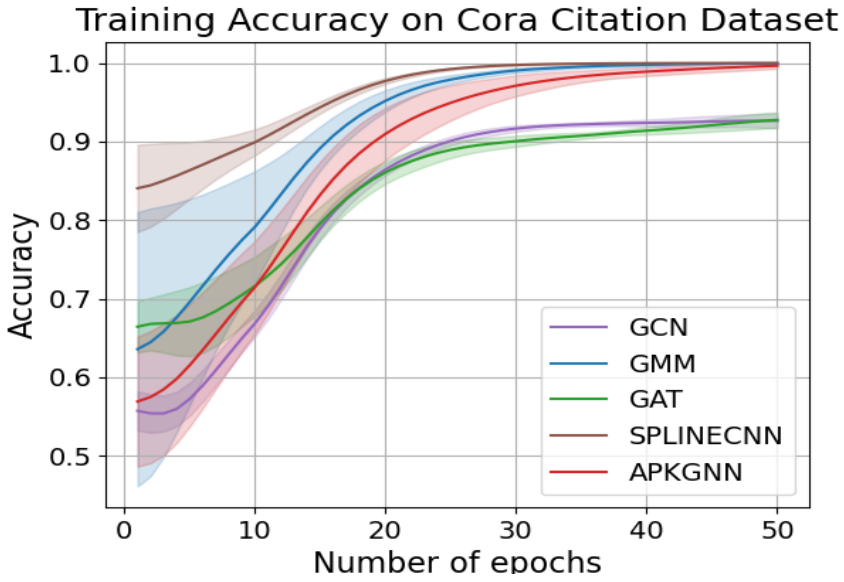

Figure 3.7: *Training accuracy curve up to fifty epochs for the CORA data set for all GNN layers*

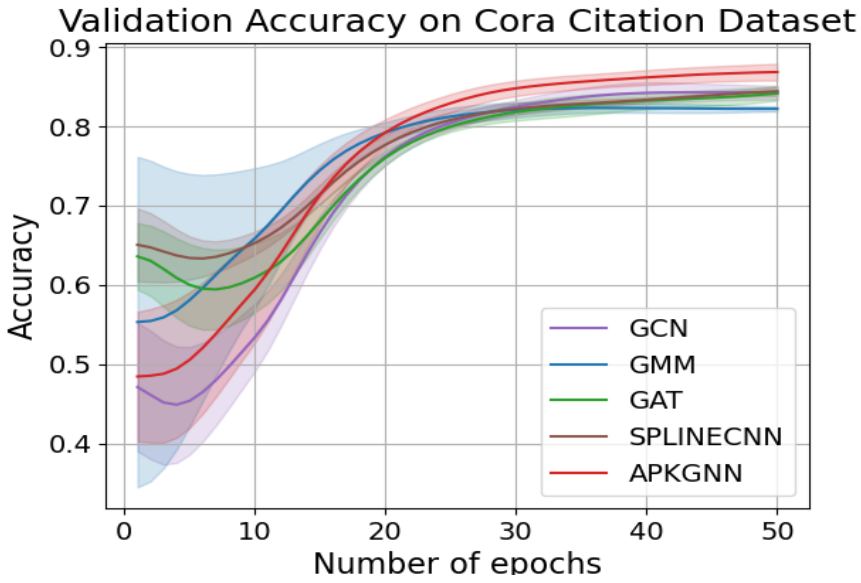

Figure 3.8: *Validation accuracy curve up to fifty epochs for the CORA data set for all GNN layers*

Figures 3.9, 3.10, and 3.11 show the same trend, with the APKGNN model outperforming

all SOTA models on test, training, and validation data samples from the CiteSeer citation data set.

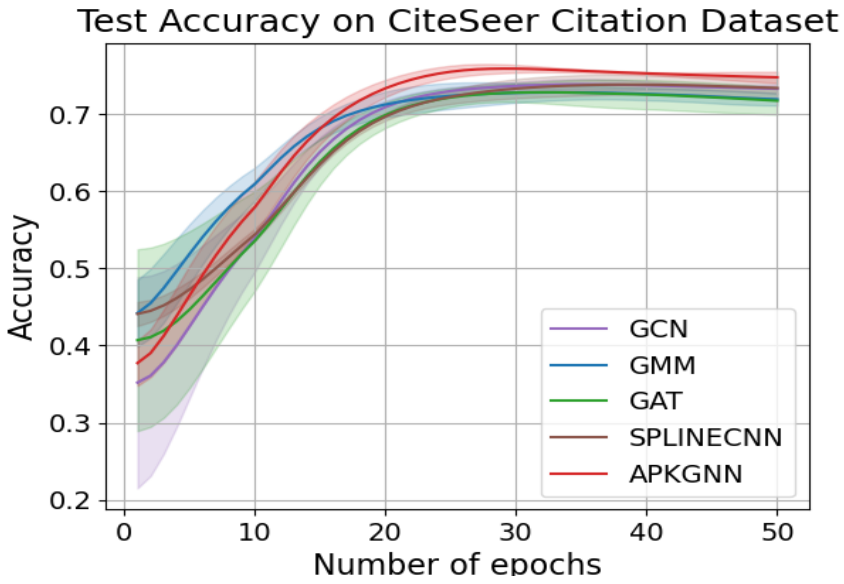

Figure 3.9: *Testing accuracy curve up to fifty epochs for the CITESEER data set for all GNN layers*

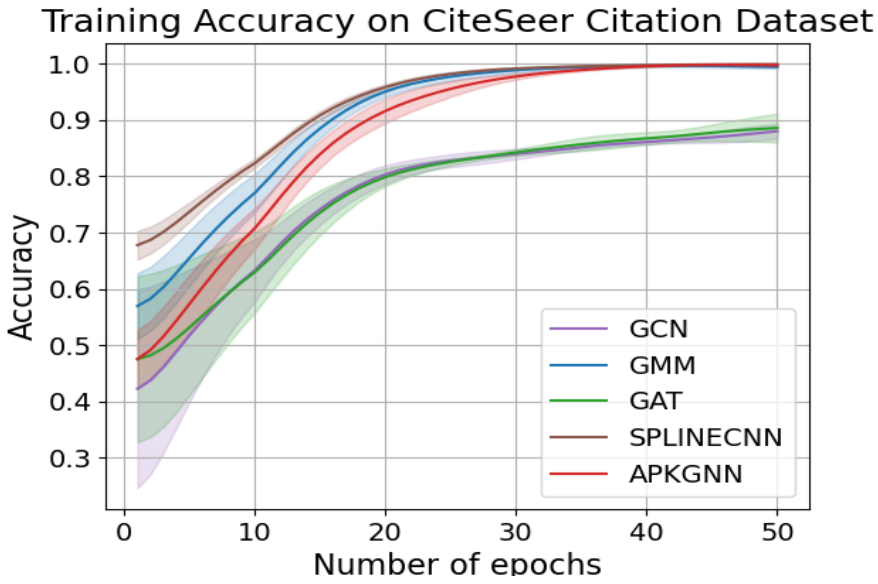

Figure 3.10: *Training accuracy curve up to fifty epochs for the CITESEER data set for all GNN layers*

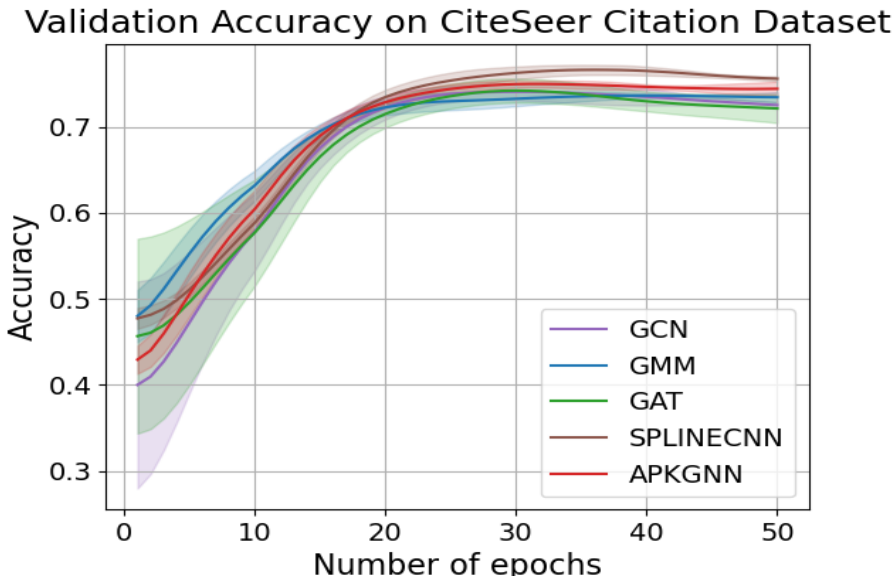

Figure 3.11: *Validation accuracy curve up to fifty epochs for the CITESEER data set for all GNN layers*

Figures 3.12, 3.13, and 3.14 also illustrate the consistent performance of the new model on testing, training, and validation data samples of the CiteSeer citation data set as opposed to the SOTA models.

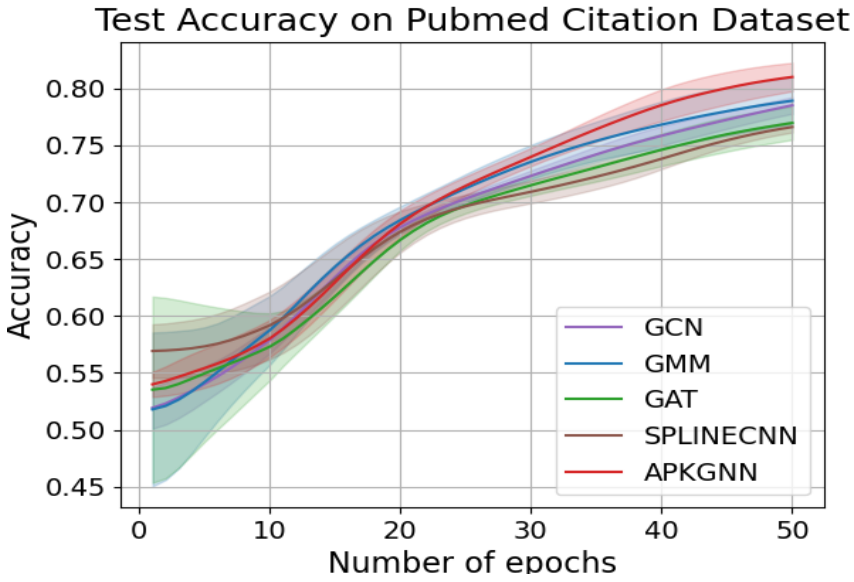

Figure 3.12: *Testing accuracy curve up to fifty epochs for the PUBMED data set for all GNN layers*

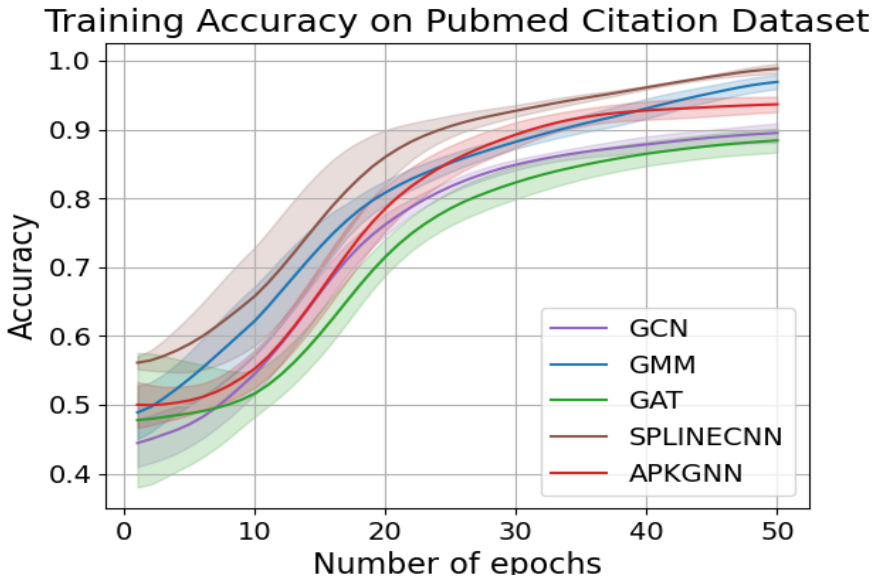

Figure 3.13: *Training accuracy curve up to fifty epochs for the PUBMED data set for all GNN layers*

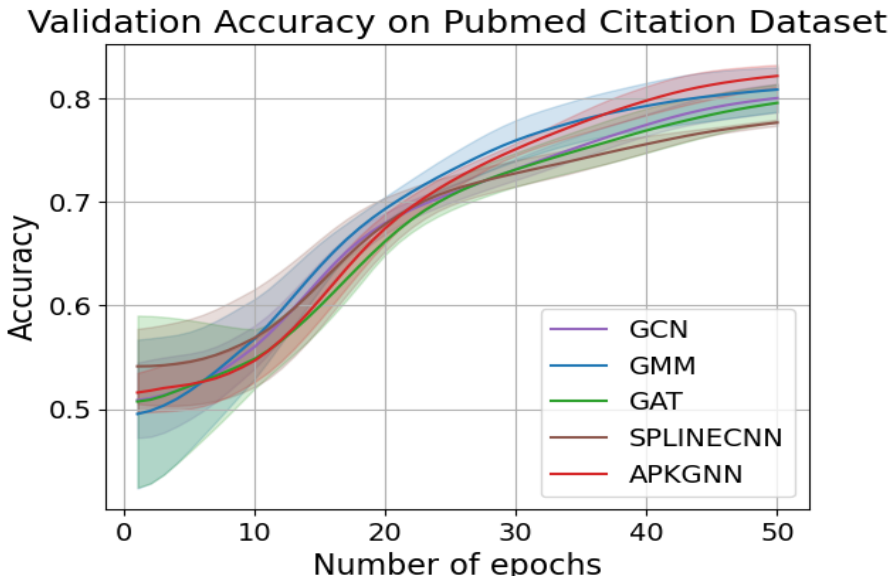

Figure 3.14: *Validation accuracy curve up to fifty epochs for the PUBMED data set for all GNN layers*

Figures 3.15, 3.16, and 3.17 also follow the same trend where the experimental model outperforms the SOTA models in testing, training, and validation data samples of the CTI data set. This particular result is significant because the CTI data set is generated and then transformed into graph data from raw tweets. This raw data is highly complex in terms of its sparsity, poor input features, and fewer numbers of labeled nodes because word nodes are mapped with any labels.

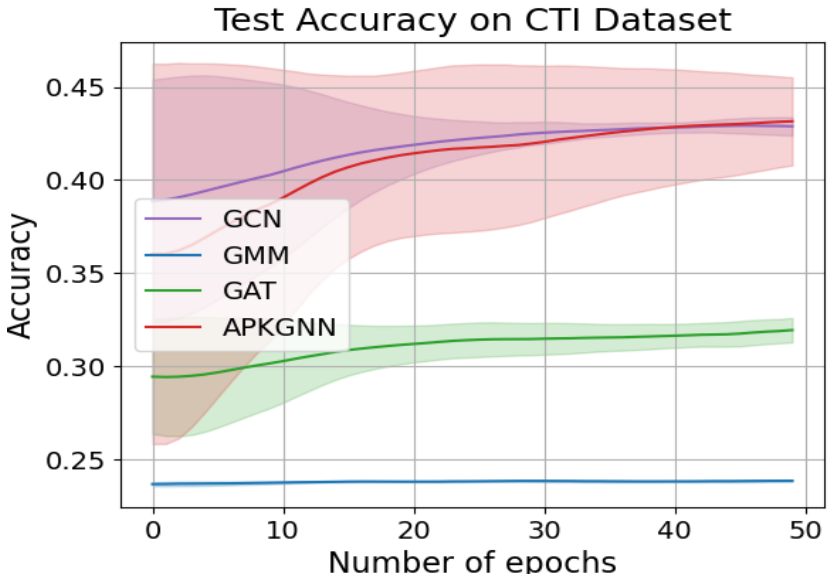

Figure 3.15: *Testing accuracy curve up to fifty epochs for the CTI data set for all GNN layers*

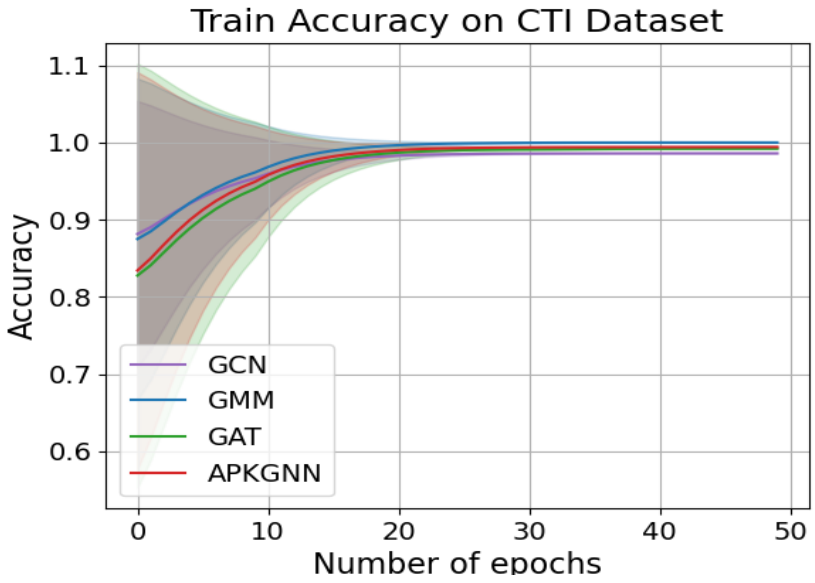

Figure 3.16: *Training accuracy curve up to fifty epochs for the CTI data set for all GNN layers*

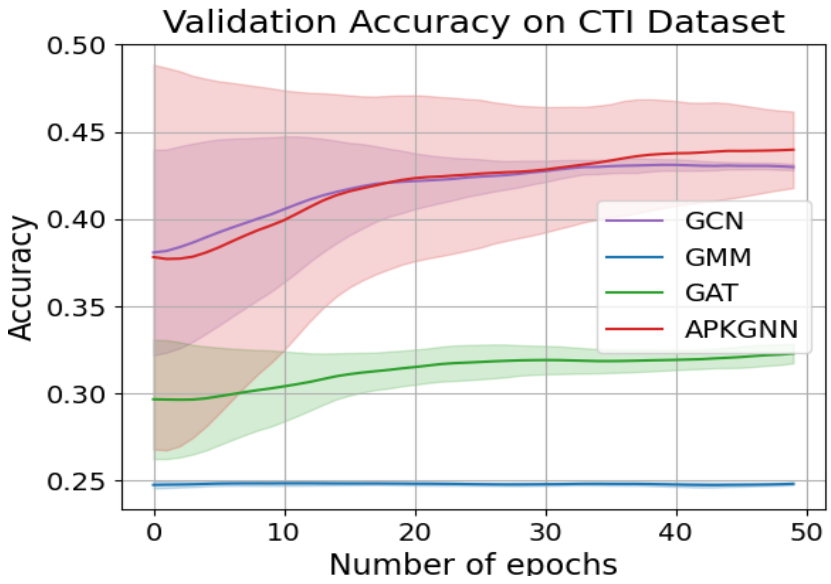

Figure 3.17: *Validation accuracy curve up to fifty epochs for the CTI data set for all GNN layers*

3.6.1 Discussion

Implementing the APKGNN model was rooted in the theory that constructing a GNN layer to produce efficient learning representations supported better experimental results. The layer-wise computation of APKGNN includes two individual modules executed together. The first module calculates attention vectors of neighboring node feature similarity, and the second extracts patches by applying weight function on local graph structure components called pseudo coordinates. A linear combination of the two modules followed by learning an attribute vector produces better learning representations than SOTA GNN models in node classification tasks. The running average of accuracy plotted to different curves indicate that the performance of the models is consistent across epochs. Moreover, while at the initial stage of the simulation, the new model does perform well, once the simulation proceeds toward completion, the model outperforms all the other models. The newly implemented APKGNN model learns more parameter vectors than other models, which is why each epoch for the new model takes longer than the other models. The research, when extended to

short text multi-class classification tasks, also exhibits remarkable performance compared to others. This result indicates that learning additional parameters helps the model to learn highly complex graph data sets.

3.7 Conclusion and Future Work

In this research, we improved node representation learning by introducing a GNN layer architecture for the node classification task. The GNN model constructed by the introduced layer outperformed other models on all the data sets in all experimental settings. The model was also applied to the CTI data set as a semi-supervised learning approach to classify different cyber-threat types. The experiments were done in multi-processing settings. In the future, I will use multi-GPU training, to parallelize the process over multiple GPUs.

Chapter 4

Event Detection and Ranking of Cyber-Threat Events

4.1 Introduction

This chapter presents a new methodology for recognizing potential cyber-threats using passive filtering and ranking in social text streams, particularly Twitter streams. *Passive filtering* also known as *passive monitoring* here refers to collecting intelligence and solutions of different cyber-threats from different platforms using only text corpora and lists of named entities or keywords (e.g., gazetteers) rather than direct background knowledge of threats. Twitter was used because it is a high-bandwidth platform where actors from both sides of cyberdefense, both attackers and security professionals, post cybersecurity-related messages[25]. The overall goal of this work is to analyze these messages collectively to obtain actionable insights and collect intelligence on emergent cyber-threat events. Detecting events from social media includes a) novel event detection, including first stories or tweets about previously non-extant topics; and developing events (especially for bursty topics, but also for non-bursty topics for which volume and aggregate importance build up gradually). In this research, we treat novel events from the time they begin (emergence) as **orthogonal** properties. This allows us to track novel events that have not yet attained trending status or viral propagation,

while still incorporating traditional trend detection methods.

Recent research includes some work on detecting both novel and developing events in Twitter streams (e.g.,[4 81]), especially where emergence is defined as trending. However, only a few studies have further focused on detecting cyber-threat events in Twitter streams. Furthermore, we propose an approach to ranking events for their significance. While such a ranking generally depends on both the application domain and user objectives, the relative importance to a general community of interest, like the cybersecurity community, can be imputed based on pervasiveness, spread rate, and novelty. In this study, we also ranked the two types of events based on the order of their corresponding importance score to show how much more important a particular event is than proximate events within a user-specified range of time for a reference tweet. In contrast with large document corpora, analyzing short documents like tweets presents some specific semantic challenges for extracting terms, relationships, patterns, and actionable insights in general. For example, terms mentioned in a short tweet lack context, with less co-occurrence data in the entire corpus on which to base expressible relationships between named entities or terms.

Our system took as input a user-specified maximum interval of detection for related cyber-threat events within an original tweet that was deemed relevant. The full text of this tweet, or quote part of a retweet, was captured. Social network parameters such as indegree (number of followers) were calculated and normalized by range. The text bodies of tweets are vectorized using the term frequency-inverse document frequency (TFIDF)[82], and the resulting TFIDF vectors were clustered using the *DBSCAN*[83] density-based clustering algorithm. Noise points were discarded, and the concatenated text contents of each cluster were ranked using the *TextRank* algorithm[84] to obtain representative keywords and named entities that represent potential events. We then identified different scenarios: a) novel and developing story; b) novel story only; c) developing story only; d) an event not based on heuristics as described in Section 4.4. Additionally, we also calculated an importance score for each event based on the heuristics presented in Section 4.4. Finally, we tagged each event according to the descriptive features and provided a rank based on the importance scores.

Key novel contributions of this work are as follows:

1. We detected both trendy and novel types of events related to cybersecurity from Twitter streams.

2. We provided a method for ranking potential cyber-threat events according to their importance score based on keywords, as well as their named entity confidence and user influence scores.

3. The proposed method could be tuned to capture important cybersecurity events based on user-specified parameters.

4.2 Related Approaches

This section briefly summarizes key methodologies for cyber-threat detection from text corpora, particularly social media.

Dabiri et al.[6] analyzed traffic-related tweets for detecting traffic events by applying deep learning models, including convolutional and recurrent neural networks incorporating a `word2vec`-based word embedding layer to represent terms. This approach performed well but is domain-dependent and costly in terms of manual annotation for high-throughput sources of training data such as Twitter. In contrast, TwitInfo[5] incorporated a new streaming algorithm that automatically discovers peaks of event-related tweets and labels them from the tweet texts. This approach, however, focused only on the burstiness of tweets and ignores both user influence and the novelty of developing events.

Rupinder et al.[9] also proposed a framework based on deep learning for extracting cyber-threat and security-related insights from Twitter, categorizing three types of threats (examples of which are Distributed Denial of Service (DDoS) attacks, data breaches, and account hijacking). From text documents, events were extracted using a) target domain generation; b) dynamically-typed query expansion; and c) event extraction. This approach uses both syntactic and semantic analysis with dependency tree graphs and convolutional kernels but is highly computationally intensive because of the cost of autoencoder training.

Sceller et al.[12] used unsupervised learning to detect and categorize cybersecurity events

by analyzing cybersecurity-related Twitter posts based on a set of seed keywords specified for each level taxonomy. This algorithm is prone to false negatives because it may not detect potential cyber-threat events as events.

Mittal et al.[7] proposed a method for processing threat-related tweets using the Security Vulnerability Concept Extractor (SVCE), which generates tags for cybersecurity threats or vulnerabilities such as means of attack, consequences of the attack, and affected software, hardware, and vendors. This approach does not generalize to user communities because it is personalized for individual user system profiles.

Edouard[8] proposed a framework that used Named Entity Recognition (NER) and ontology reasoning (using *DBPedia*), along with classification learning approaches such as Naive Bayes, SVM, and a deep neural network (Long Short Term Memory/Recurrent Neural Network or LSTM-RNN), for imputing category tags. The graph algorithm *PageRank* ranked candidate items for retrieving information.

The approach of Lee et al.[10] focused on community communication and influence to detect cyber-threats by grouping high contributing Twitter users and scoring them as an expert community to get information to explore and then efficiently exploit. This framework incorporated four components: a) an interface to the Twitter social media platform; b) a flexible machine learning system interface for document categorization; c) a mixture-of-experts weighting and extraction scheme; and d) a new topic detector. This framework depends heavily on expertise and data quality.

Sapienza et al.[11] considered various web data sources to generate warnings of potential upcoming cyber-threats. While potentially extensible to named entities discoverable by set expansion, this approach focused on detecting "novel words" and does not incorporate a full contextualized topic model, feature weighting model, or method of user influence.

Finally, the research of Alan et al.[85] used a supervised learning approach to train an extractor for extracting new categories of cybersecurity events by seeding a small number of positive event examples over a significant amount of unlabeled data. As with previous approaches, it does not incorporate full NER nor allow for entity set expansion.

4.3 Background

This section presents a brief review of the key technologies that were adopted in our proposed framework for detecting threat events in tweets.

4.3.1 Named Entity Recognition (NER)

In general, NER is an information extraction task for locating and classifying the names of specific entities such as persons, organizations, and locations, based on analysis of text units such as n-grams and noun phrases. Generic entities such as numerical quantities are sometimes also included. In our analysis, NER was used to discover the names of entities in reported cyber-threats. Key objectives of using machine learning to improve NER were a) set expansion to broaden the set of cyber-threats based on synonymy and other relationships that can be inferred by text pattern analysis; b) feature weighting for relevance or salience; c) relationships discoverable from data; and d) confidence scoring.

4.3.2 TextRank

The TextRank algorithm[84] is an extended version of the Google PageRank[86] algorithm that determines keywords by generating a word graph from a given text document instead of determining highly ranked webpages using the PageRank algorithm. The TextRank score calculates the importance of a word from a given text, which works like PageRank score for webpages. The importance associated with a vertex, however, is determined from votes cast for it and the score of the vertices casting these votes.

4.3.3 TFIDF

TFIDF[82] is an information retrieval method used for such purposes as word co-occurence based document vectorization, word ranking, and document similarity calculation. In information retrieval, TF (term frequency) refers to the term frequency of a particular word that occurs in a document, while IDF (inverse document frequency) refers to the inverse

document frequency of a word that occurs in the whole corpus of documents.

4.3.4 DBSCAN

DBSCAN[83] is a density-based clustering approach that operates by enforcing a minimum number of data points (*MinPts*) inside a specified-radius neighborhood (*Eps*) of a data point to make a density-reachable cluster; this process continues until no points on the frontier are density-reachable, then begins again with a new initial point.

4.4 Methods

Analyzing Twitter texts for valuable insights has always been problematic because the writing lacks structure and tweets are short. In this section, we describe the steps we took for analysis.

4.4.1 Tweet Collection and Early Annotation

For our analysis, we collected Twitter data for four days between September 6^{th} to September 9^{th}, 2018; a small portion of data was collected on August 30^{th} and 31^{st}, 2018, which is stored in MongoDB database. Our main focus was to gain insight into cybersecurity-related events and rank their scores, so we crawled Twitter data using the Twitter API based on some security related keywords. Without applying security related keywords, the crawled Twitter data would be generalized to all domains, and the results would be biased to detecting general events. This data set was manually annotated by four annotators for our earlier work[25] to find tweets relevant to cybersecurity. Although we used security related keywords for crawling the Twitter data, many tweets were irrelevant or promotional. That is why annotation is crucial here; the resulting data set of this process is available at[25]. In this study, we initially had 21368 tweets. After annotation, we found 11111 tweets related to cybersecurity. We applied our algorithm to the cybersecurity related tweets and the entire tweet data set individually. We took the full text of each tweet if the tweet is not quoted

or retweeted by any other users. If any tweet is retweeted or quoted by any other user, we take the original retweeted or quoted tweet full text. Then, we let a user give a numeric value input as the number of time intervals based on tweet occurrence. This process divides the whole time period of tweet occurrence into equal time chunks based on the number we took from the user. Thus, for each time interval, a number of tweets were aggregated into a chunk based on their corresponding time of occurrence.

4.4.2 Tweet Pre-processing and Cleaning

As we noted, tweet text is very unstructured and contains many misspelled words. Sometimes the text is not a complete sentence, so we applied SymSpell[87] a tool that corrects misspelled words. Then we took only alphanumeric characters from the tweet text and removed all punctuation characters. We then removed all stopwords from the text because they occur so frequently over the entire data set that stopwords could reduce the analysis performance. We then cleaned the tweet texts either by applying stemming or without stemming. We removed all the words or tokens with lengths of only one.

4.4.3 Influential Twitter User Impact

Influential user tweets are valuable for detecting important cybersecurity related events, which is why we keep records of each Twitter user and their followers corresponding to their posted cybersecurity related tweets. The number of followers is directly proportional to the influence of the user, so their used words in cybersecurity related tweets are also important. So for each time interval, we normalized the values of follower numbers for each user with Min-Max normalization. This normalization process normalizes each value between 0 and 1. We then assigned the normalized value of follower numbers to each used noun phrase in the used posted tweets. Here, we used Python nltk library to extract noun phrases from tweets. If similar words were used by several Twitter users, we kept the highest normalized value of a user for each word used in a tweet. Now, for each time interval, for all tweets, each noun phrase had a corresponding value that represented its weight as inherited by its user. This

value was also used to calculate event the score.

4.4.4 Determining Algorithm Design Architecture

In this study, we applied a very popular word vectorization method in the NLP domain named TFIDF[82] that is based on word co-occurrence in documents to make a word vector for each tweet from the data set. After doing this for all tweets in a time interval, we generated a TFIDF matrix. We found this method performs better than the word semantic relation based approaches. Then, we applied the DBSCAN[83] density based clustering algorithm using the TFIDF matrix to find tweet clusters with similar meanings. These clusters may represent potential events. However, we did not apply the K-means clustering algorithm because we wanted to limit the number of events found in our analysis for each time interval. For this analysis, we ignored the noise points generated by applying DBSCAN because in our observation of the data set we found that the noise points are conveying very little impact to find cybersecurity related events.

4.4.5 Event Detection Heuristics and Scoring

First, we aggregated all tweet texts in a cluster into a single text. Then we simulated the named entity recognition process using the TextRazor online NER API and the keyword identification process on the aggregated text with the TextRank[84] algorithm from the Gensim library. Additionally, TextRazor provided a confidence score for each named entity and TextRank[88] provided a score for each keyword based on the word graph. We applied two different set rules to determine the type of events and their corresponding score, which was then used to rank the cybersecurity related events. To test our idea for implementation, we produced four different sets of tokens:

1. *commonSet* refers to the set of words common to both named entity and keyword. Additionally, we took some higher-scored named entities and keywords from both *namedEntitySet* and *keywordSet* to add tokens to the *commonSet*.

2. *keywordSet* refers to the set of words that appear only in the keyword set but not the named entity set.

3. *namedEntitySet* refers to the set of words that appear only in the named entity set but not the keyword set.

4. *unionSet* kept all named entities and keywords.

Figure 4.1 depicts the graphical illustration of the four token sets. Here *commonSet* is represented by $k \cup (K \cap N) \cup n$, *keywordSet* is represented by K-N, *namedEntitySet* is represented by N-K, and *unionSet* is represented by $K \cup (K \cap N) \cup N$. Here, k and n are the sets of keywords with a high score and named entities that can be represented as $k \in K$ and $n \in N$, respectively.

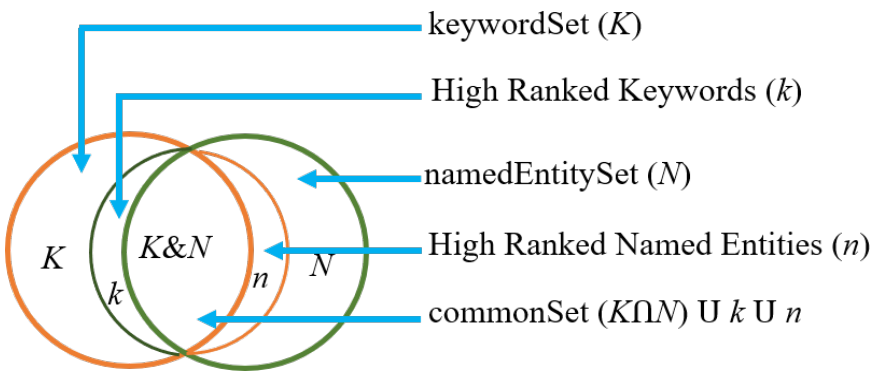

Figure 4.1: *Graphical representation of* commonSet, keywordSet *and* namedEntitySet

Determine Event Novelty

We stored all the tokens of the set *namedEntitySet* and *commonSet* into another set of tokens named *noveltyCheckerSet* for all clusters generated for all time intervals. We have stored all these tokens because we are checking the similarity of tokens from the set of a subsequently generated cluster to the stored token set *noveltyCheckerSet* to determine whether the newly generated cluster has some novelty or not based on a cosine similarity threshold value determined empirically.

Determining Trendiness

If the similarity score reached the defined threshold *cosineThresh*, the working cluster was "Just Trendy" except for the very first cluster because no other set was available to compare the similarity of this first cluster. We also took a user defined threshold of numbers of tweets *tweetThresh* to determine which cluster was trendy. Thus, if the the number of tweets did not reach the value mentioned by the user, it was an unnoticeable event. However, if tweets from a cluster satisfied the cosine similarity threshold *cosineThresh* as well as the threshold *tweetThresh* for number of tweets, it still may not represent a noticeable event because it may be a spammed banal topic. We applied a different heuristic if the length of the *commonSet* set is greater than the one fifth(0.20) times of the *namedEntitySet* set; only then will the cluster be counted as trendy. We are checking this because a big cluster of tweet texts will have so many named entities that might mean a variety of topics; a single cluster should be biased towards a single topic described in it.

Determining Novelty

If the cosine similarity of a stored token set *noveltyCheckerSet* and working cluster token set is less than the threshold value *cosineThresh*, the working cluster could be a potential novel event. However, we set a threshold value minimum of three tweets in a cluster to call it an event. If the number of tweets was greater than or equal to the user-defined threshold value of trendiness *tweetThresh*, the cluster was called'Novel and Trendy", but if the number of tweets in the cluster is less than the number of users defined threshold value, it was called "Just Novel". That was how we defined a type of a generated cluster of tweets as an event type.

Event Score Calculation Process

The ranking of cyber-threat related events was also important, so we were motivated to calculate score of each event if an event finds an event type based on our empirically defined heuristics. We calculated scores for each defined event individually by applying different

heuristics. The confidence score of a named entity in TextRazor and the score of a keyword in the TextRank algorithm are different in scale, so we applied sigmoid function to normalize each score. Every token is stored in a dictionary for each generated cluster, but we updated the score of each token if it was both a named entity and a keyword by adding the two scores after normalizing.

Score Calculation for Trendy Events

For a "Just Trendy" event, we first calculated the entity score of the event by adding the scores of each token in the *commonSet* and then multiplied the total added value by the value of total number of tweets that makes an event trendy. Second, we added the value of each noun phrase corresponding to event tweets where the noun phrases are inherited from the value of influential users' followers. This score was then added to the initially calculated entity score for the aggregated tweet texts to get the total score. We calculated the entity score in this way because it assigned a higher score to a trendy event if tweets in a same topic appear so many times in a cluster or if the number of tokens in the *commonSet* is higher. This heuristic assumes that even if the event topic does not appear many times in corresponding tweets like other highly tweeted event topics, because of the number of common tokens in both name entity and keywords, the heuristic identifies those tokens as important.

Score Calculation for Trendy and Novel Event

For an event that is both "Novel and Trendy", we added the values of all the tokens of the set generated by the union operation of *keywordSet* and *commonSet*. Afterward, we multiplied the added value by the value of the total number of tweets representing the event to calculate the entity score. Then we added the value of each noun phrase corresponding to event tweets where the noun phrases were inherited from the value of followers of influential users. This score was then added to the entity score as discussed earlier to get the total score. We calculated the entity score like this because such an event was already proved

a novel event, and we needed to know whether it had tokens common in both the named entity and keyword sets to check the main topic of this event. Additionally, we considered the keywords appearing in this event to check which topics were also mentioned in the tweet texts of the novel event. That means a novel and trendy event will get a much higher score than the other events.

Score Calculation for Just Novel Events

For the first story event (also known as "Just Novel"), we kept a set of tokens generated by differentiating *keywordSet* from *unionSet* and then did a union operation with the *commonSet*. The resulting set stored all the named entities with keywords, which have very high score. Then, we added all the values of the corresponding tokens in the resulting set and multiplied the added value by the user defined threshold value *tweetThresh* because a novel event is sufficiently trendy to get the entity score. Then we repeat the procedure of adding the value of noun phrases to the entity score of the working event. We calculated the entity score this way because if an event is only novel, it will not appear in many other tweets, which may cause the event to lose its importance. So to recognize the importance of this kind of event, we multiplied the total score from the set of tokens by the value of *tweetThresh*. Thus, a novel event can get a score at least as high as any trendy event. We chose this resulting set because the novelty of an event had to be considered based on the the confidence score of the named entity and some highly ranked keywords. We had no way to consider the whole keyword set because, in this case, it was useless to other discussed trendy topics except the common ones with named entity set.

Ranking Scores of Events

Our proposed approach repeated condition checking for each cluster to determine which were events and calculated a score for each cluster in each time interval that were considered events. Finally, we ranked each event by ordering their total score for each time interval. The flow chart of our proposed approach is depicted in Figure 4.2.

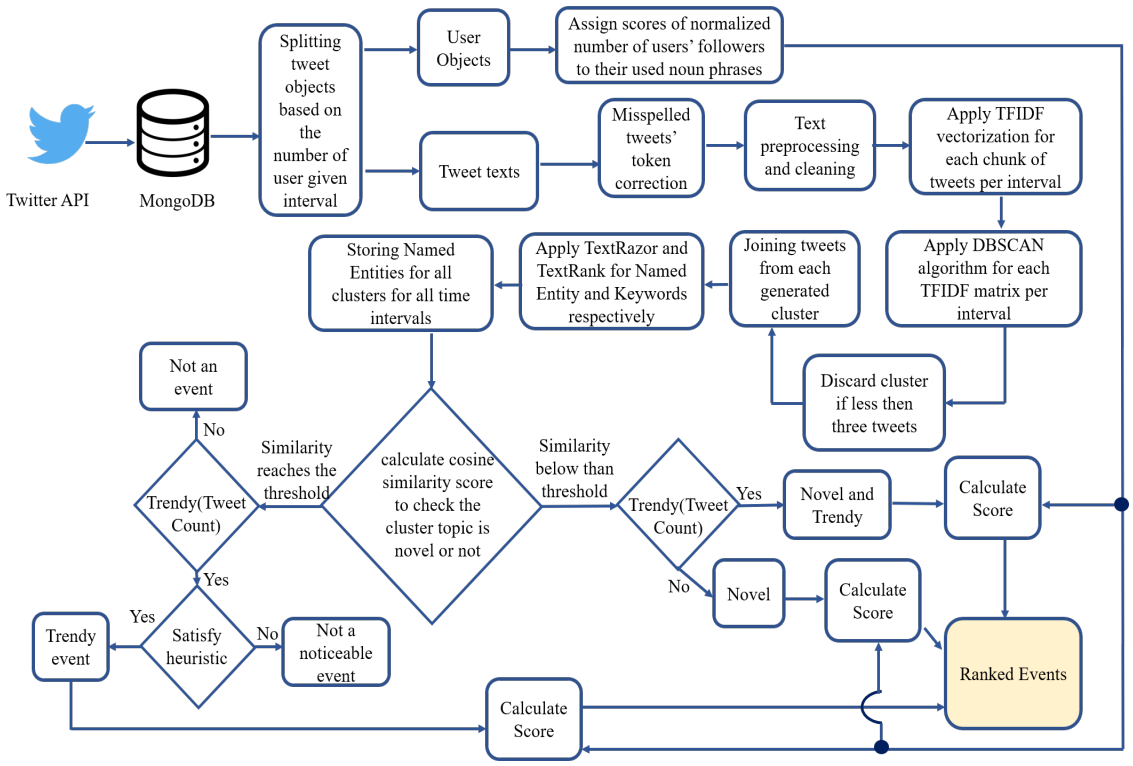

Figure 4.2: *Flowchart of the proposed approach*

4.4.6 Annotation Approach

To evaluate the performance of our approach, we compared the results of our method to a manually annotated list of events. A subset of 301 tweets collected in sequence from 2018-08-30 23:00:08 CST to 2018-09-02 10:50:19 CST was manually annotated according to a. impact, b. tweet count, and c. worldwide effect to be considered as an event. We also consider three categories whether they are a. first story or novel, b. trending or developing, and iii. novel and trending. For validation, we checked the ratio of correctly detected events in that window to the total number of relevant events, and the ratio of correctly detected events to the total number of detected events.

4.5 Experimental Results

4.5.1 Simulation

For our analysis, we used scikit learn[89] to calculate the TFIDF matrix[82], to apply the DBSCAN[83] algorithm and cosine similarity. The parameter assignments for making the TFIDF matrix were $max\_df = 0.90$, $max\_features = 200000$, $min\_df = 0.01$ and $ngram\_range = (1,1)$. Then, parameter assignments for the DBSCAN algorithm are $eps = 1$, and $min\_samples = 3$. Again, we used the cosine similarity threshold of 0.5 for similarity checking for trendiness. Table 4.1 shows the results of five time intervals collectively from 2018-08-30 23:00:08 to 2018-09-02 10:50:19.200000, from 2018-09-02 10:50:19.200000 to 2018-09-04 22:40:30.400000, from 2018-09-04 22:40:30.400000 to 2018-09-07 10:30:41.600000, from 2018-09-07 10:30:41.600000 to 2018-09-09 22:20:52.800000, and from 2018-09-09 22:20:52.800000 to 2018-09-12 10:11:04 (intervals 1, 2, 3, 4, and 5). We kept only detected True Positive events in Table 4.1. Table 4.1 shows for the first interval, we have 145 tweets and 15 events. Moreover, out of 15 events, we had no "Trendy Event", one "Novel and Trendy Event", and 14 "Novel Events". The rest of the intervals are entered similarly. Table 4.2 shows the extracted keywords for each event for the first time interval. The keywords mentioned in the table were used to detect cybersecurity related events for the first time interval. For better representation, we show only the plot of the 2^{nd} time interval in Fig. 4.3. This figure depicts found events on the x-axis, the number of tweets on the left side of in y-axis, and the event score on the right side of the figure in the y-axis. The red vertical bar represents the number of tweets and the blue vertical bar represents the event score for each detected event.

Table 4.1: *Summary results of five time intervals; NT:Number of Tweets; JT: Just Trendy; TN: Trendy and Novel; FS: First Story; TE: Total Number of Events*

Interval	NT	JT	TN	FS	TE
1	145	0	1	14	15
2	314	0	0	50	50
3	812	1	7	37	45
4	1239	0	9	18	27
5	297	4	0	5	11

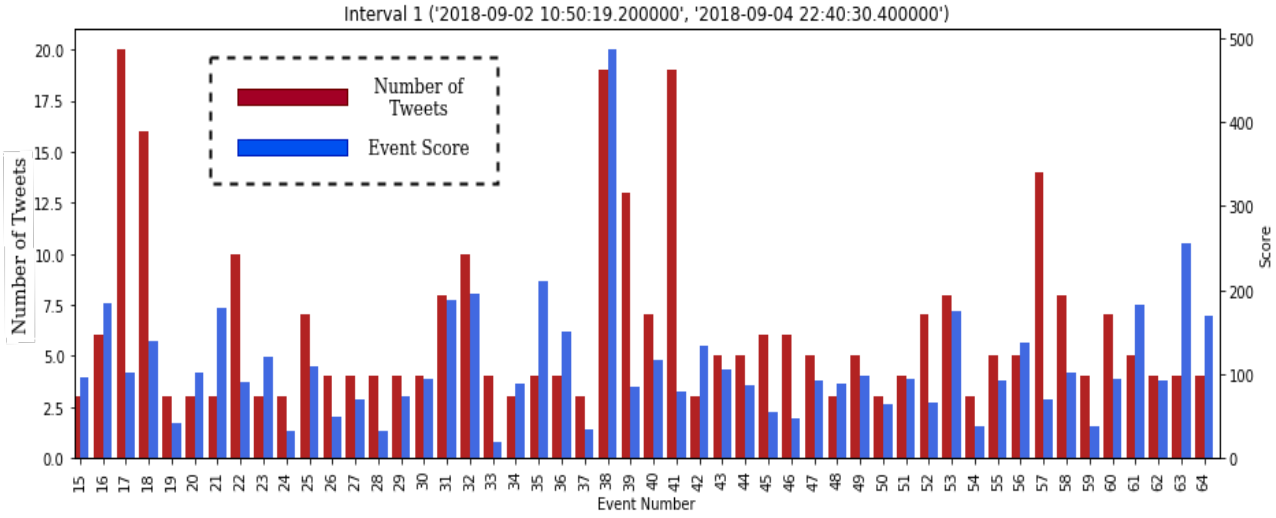

Figure 4.3: *Event plot of the second time interval proposed approach*

Table 4.2: *Summary results of time interval 1*

Event Number	Keywords
0	'security', 'android (operating system)', 'android', 'wi-fi', 'privacy'
1	'microsoft', 'disclosed', 'twitter', 'windows','microsoft windows', 'hacker discloses'
2	'website', 'catalonia', 'spain', 'banking', 'bank', 'inf'
3	'based', 'huff', 'buffer overflow'
4	'vulnerability (computing)', 'security', 'repository', 'critical vulnerability', 'apache', 'inf'
5	'vulnerability', 'resource consumption', 'prior', 'resource', 'rsa', 'bleach'
6	'task','windows','patch,'scheduler'
7	'security', 'android (operating system)', 'android', 'data', 'privacy', 'tracking'
8	'cracking ransom', 'coin', 'free', 'ransom', 'cybersex', 'net'
9	'vulnerability (computing)', 'patch', 'spyware', 'phishing', 'inf sec cube security', 'patched', 'malware'
10	'security', 'website'
11	'plus', 'pump'
12	'version', 'web server', 'debugger', 'skype', 'update', 'denial service', 'exploit (computer security)'
13	'cisco systems', 'service', 'cisco'
14	security', 'photo','service'

4.5.2 Annotation-Based Validation

Design Selection Approach

The design architecture of this algorithm required a few decisions. First, we wanted to analyze the semantic relationships between the words of each tweet text to get insight into cybersecurity events. Thus, we previously applied doc2Vec[90], an extended application of word2vec[91] for grouping tweets with similar meanings to find events from the data sets. We could not, however, get satisfactory results because shallow neural network model text domain tools like doc2Vec[90] works based on word vector embeddings that do not perform well in a data set with short and noisy text. Embedding methods do not work properly on short texts because tokens in a short text have thin contextual relationships, and this gets worse when tokens are misspelled and incomprehensible. Table 4.3 shows a sample result of doc2Vec applying hypermeter values $vector\_size = 300$, $min\_count = 2$ and $epochs = 45$ that exhibits the most similar tweet and second most similar tweet of a particular tweet has no noticeable similarity with the original tweet document. A document must show similarity at least to itself. Similarity scores of each tweet to the particular tweet are in parentheses in the first column. We decided to apply LDA[92] to find topics that may represent events. Because the tweet text is very short, almost always a tweet represents no more than one event. Therefore, we applied LDA[92] on the aggregated tweet texts from corresponding time intervals. This approach, however, also fails to show the expected results because of the incoherence of tweet text. Thus, we decided to apply the very popular TFIDF vectorization because words in a tweet text have few semantic relationships and word co-occurrence is a better option to use in this domain. We found this provided better performance compared to the results from previously applied approaches.

Validation of the Approach

Table 4.4 shows the performance results of our approach according to the evaluation methodology described in Section 4.4.6. The annotators annotated 301 tweets and found 20 events and six tweet clusters that are not events. Our algorithm found 16 events. Out of 16

Table 4.3: *Sample results of doc2Vec*

Terms	***Texts***
Document	guides on fixing sql injections vulnerabilities sql injection technique exploits security vulnerability occurring database layer application the vulnerability present user input either
Most Similar (0.8027611970901489)	free vps server ddos protected hosting
Second Most Similar (0.6457577347755432)	cvnway just ddos server
Median (0.21316465735435486)	minibb bbfuncsearchphp table sql injection
Least (-0.4030833840370178)	ransomware weapon used cyber attacks elixir ng news source trust

events, 15 are real events (True Positive) with one false positive. So, the True Positive, False Positive, False Negative, and True Negative rates are 75%, 16.67%, 25%, and 83.33%; our precision value is good at 93.75%. We could stream only a very small number of tweets per millisecond, approximately 1% of the total tweets posted; this issue is addressed in the linked web article[93]. The Twitter data itself is insufficient to detect all ongoing cybersecurity events. Thus, we limited ourselves to calculating recall scores by keeping track of published cybersecurity events online. Table 4.5 presents the 15 true positive events, along with their tweet count and their corresponding scores. We ordered the events by their corresponding scores and matched the scores with the annotators' annotations. The 4^{th} column of Table 4.5 presents our approach's event ranking, and the 7^{th} column shows annotators' rankings. Comparing the 4^{th} and the 7^{th} column of the table shows the annotator predictions are quite similar to our approach results for event detection and event ranking. The validity of the detection approach can be checked by clicking the link in the 6^{th} column to see reports published in authentic blogs and newspapers. The 5^{th} column provides the types of events detected by our algorithm. The sum squared error (SSE) of the event ranking of our approach and annotator rankings is 86 by calculating the difference mentioned in the 8^{th} column of the table.

Table 4.4: *Confusion matrix of the algorithm's generated result*

Total Population	***Ground Truth positive***	***Ground Truth negative***
Derived positive	True Positive=75%	False Positive=16.67%
Derived negative	False Negative=25%	True Negative=83.33%

Table 4.5: *Summary results of time interval 1; EN: Event Number;EL: Event Link; TC:Tweets Count*

NESR:Normalized Event Score Rank;ET: Event Type; AER: Annotator Event Ranking; DBR: Difference between Rankings; FS: First Story ("Just Novel"); FST: First Story and Trendy (developing)

EN	TC	Event Score	NESR	ET	EL	AER	DBR
0	5	167.6084	5	FS	link1	4	1
1	7	211.033	3	FS	Link2	2	1
2	21	190.5950	4	FS	Link3	3	1
3	3	55.3226	10	FS	NA	13	3
4	12	110.6048	9	FS	Link4	7	2
5	4	130.4169	8	FS	Link5	8	0
6	3	17.2225	14	FS	Link6	10	4
7	6	145.7938	7	FS	Link7	5	2
8	8	154.3115	6	FS	Link8	6	0
9	5	389.7082	2	FS	Link9	9	7
10	4	40.0639	13	FS	NA	14	1
11	7	46.4706	12	FS	Link10	12	0
12	51	391.3391	1	FST	Link11	1	0
13	5	52.8906	11	FS	Link12	11	0
14	4	16.9345	15	FS	NA	15	0

4.6 Conclusion

We presented a novel machine learning and text information extraction method for detecting cyber-threat events from tweets. We considered two types of such events: those that are novel, and those that are further developments of previously detected tweets. Furthermore, we proposed an approach for ranking cyber-threat events based on an importance score computed using the named entities and keywords in the text of tweets. We also impute influence to users in order to assign a weighted score to noun phrases in proportion to user influence and the corresponding event scores for named entities and keywords. To evaluate the performance of our approach, we measured the efficiency and detection error rate for events from a specified time interval relative to human annotator ground truth and demonstrated the feasibility of applying our approach to detect cyber-threat events in tweets. Future research should extend our method for detecting and ranking sub-events in each cyber-threat event. Moreover, the heuristics applied in this work provide proof of concept, while leaving room for further enhancement and customization as users require. Further, future research can apply the method for measuring the influence of users can be extended with meta-network modeling and link extraction of the dynamic social network of users active in the cybersecurity domain.

Chapter 5

Tracing Relevant Twitter Accounts in CTI Domain

5.1 Introduction

Analyzing Twitter data streams as an open source[94] for cyber-threat monitoring is a key research topic in Open Source Intelligence (OSINT). Although the feasibility and significance of Twitter as a streaming data source for tracing Cyber-Threat Intelligence (CTI) have already been established in earlier research[5,7], the task of tracing user accounts as a potential source of information on threats[25] and then extracting pertinent information from the accounts have not yet been fully explored.

In Twitter, user domain-specific posts and shared content over the social network tend to have more importance/influence on followers if they structurally belongs to the community of interest. CTI is such a community. Thus, detecting who belong to a structural community is necessary before beginning to trace accounts of interest. User importance in a domain of interest can, however, be evaluated by an automated ranking mechanism by considering relevant domain experience and activities. Therefore, many accounts belonging to a community may not be domain relevant if a ranking measure filters user accounts into appropriate categories. Our interest lies with certain target topics, so topic similarity calculation of

user tweets is an obvious downstream process for tracing user accounts as CTI instances. Tracing and monitoring groups of users is more effective than attempting to find a single source because this can help i) identify emerging cyber-threats, ii) measure the credibility of the information, and iii) find Twitter users who share similar or paraphrased texts on a cyber-threat related topic. This research scope has high potential but is less explored and inspired us to propose an approach to extract Twitter user accounts actively involved in pursuing and propagating CTI on Twitter.

Many previous studies traced relevant instances of CTI information using text analysis only, which is not efficient because such approaches account for neither the underlying topology structure of Twitter nor the huge volume of data content and the speed with which data are produced. To address this issue, we leveraged both the "followers" and "following" relationships among users to formulate a homogeneous, directed user graph network. Then, we detected user communities from the Twitter graph network, using the topological information in the graph to trace CTI-relevant user accounts in the form of extracting graph nodes. One regression model was trained on tweets and account descriptions of user accounts that have numeric ratings (from pre-tag labels) to compute a weight for each user account by adding the predicted score from the regression model of each inputting text. The prediction scores for users in a community are then added to generate an overall weight for a community.

We found similarities between the TF-IDF feature vectors of the community text comprising all user account texts; the feature vectors were obtained from previously selected relevant nodes (i.e., seed nodes). We kept a record of the similarity scores calculated for each community for input seed accounts, and then we added each score to the corresponding community weight to generate a community ranking. To reduce computational complexity, we kept only top-p (where p$\in \mathbb{N}$) communities, and for each community. We also calculated the similarity between the text feature vectors of all users in the current community and the text feature vector of the seed users. Based on the aggregated values of similarity scores and respective user account weights, we ranked user accounts where the resulting top-k (where k$\in \mathbb{N}$) user nodes were considered traced nodes in the CTI domain. In this research[95], we

ran and evaluated our approach on a representative data set sampled from an original larger data set of user nodes on Twitter. The result demonstrates that our approach performs well in tracing user accounts that frequently feature tweets about CTI-relevant information.

The main contributions of this paper are as follows:

- To the best of our knowledge, this is the first approach[95] to trace relevant user accounts as instances of CTI-related information while using both graph structure and user account content.
- To generalize our approach, we applied a regression model for predicting scores of all posted tweets and the account description of each user account to rank and select user accounts according to their relevance to CTI.
- We considered central user influence on other users in a community while ranking and selecting user accounts/nodes.
- We also answered two relevant questions that can help explain our methodology.

5.2 Related Work

User account tracing comprised several methodological steps, a hybrid task that is theoretically different from a user recommendation system. Although the purpose of this study was different from the more extensively researched topic of user recommendation or friend recommendation in social networks, the low-level working procedures of user account tracing and user account recommendation is similar with many common terms, backgrounds, and ideas. Our research was compatible with the current trend of research on the recommendation system, so we have provided references to earlier research.

A recommender system can be implemented in two mutually exclusive processes: one using graph structure and one without. However, for a user recommender system in a social network platform, the first outperforms the second[96] in recommendations using missing link prediction and identification. The working principle of recommendation models has three

main approaches for recommending users: i) Network Structure-based[97], ii) Content-based[19], ii) Hybrid[20]. These approaches encompass link prediction to efficiently find potential future links or missing links between user nodes in social network graphs. Several research reports indicate that neither network structure nor content-based alone can produce efficient user recommendations; on the other hand, a hybrid approach that fuses network structure and content performs better.

Again, friend recommendation systems using Latent Dirichlet allocation (LDA)[21] also have limitations because most of the time Twitter's short text content only covers a single topic. Research[22] includes structural and community information but lacks content information in the analysis of effective link prediction.

Our research is the first that focuses on tracing specifically CTI user accounts in the Twitter data stream. Our goal is to build a cost-effective approach by incorporating both graph-structural and content information.

5.3 Methods

This section describes the techniques and methods used in our approach with the steps illustrated in Figure 5.1. Our work is divided into two parts: (1) user account tracing, and (2) result validation. This approach is formally depicted in algorithm 1 where 'U_acc' represents a single user account and 'U_accs' multiple user accounts.

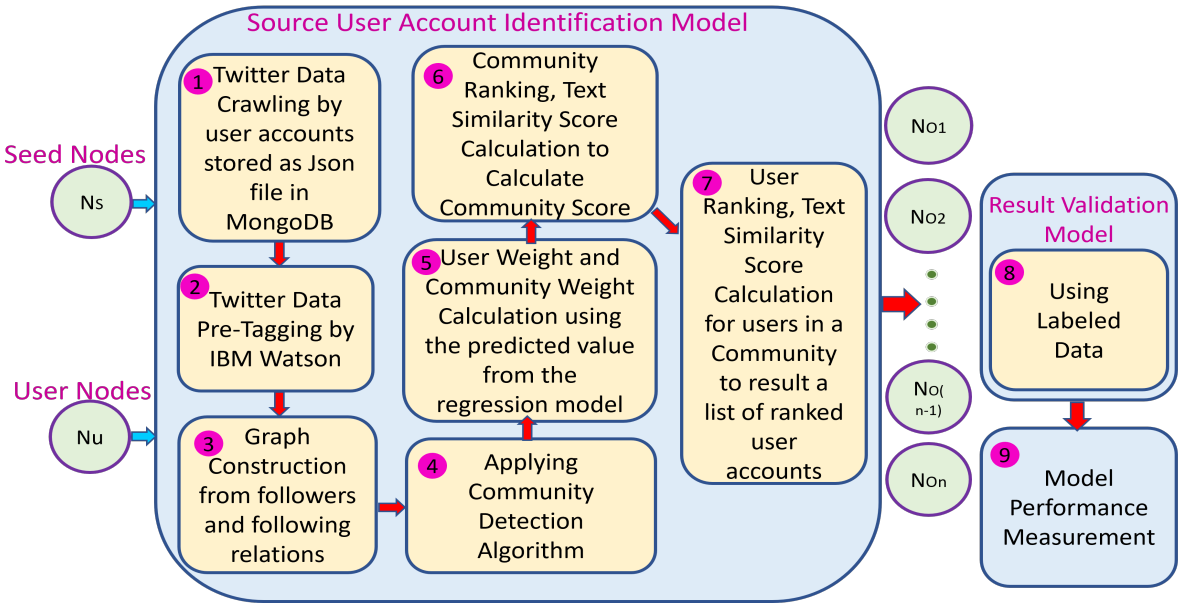

Figure 5.1: *Flow diagram of our proposed approach*

5.3.1 Graph Construction

Because we wanted to construct a directed user graph with outgoing edges in a "follower" relationship, we used a current Twitter user account as a source node (n_{s_i}) (where i$\in \mathbb{N}$), and all the Twitter user accounts in the "follower" list of the account as destination nodes (n_{d_j}) (where j$\in \mathbb{N}$). Similarly, for incoming edges in the "following" relationship, we considered all user accounts in the "following" list of the Twitter account as source nodes (n_{s_i}) and the Twitter account itself as a destination node (n_{d_j}). We then used NetworkX to build a directed graph $\overrightarrow{G}$. Figure 5.1 shows this step in sub-block 3.

5.3.2 Text Pre-processing

By nature, Twitter texts are often not grammatically correct, and the words in a tweet can be misspelled or abbreviated because of Twitter's promptness and short text structure. To extract usable information from the text, we modularized the steps of the text pre-processing task using i) contraction mapping: a contraction dictionary used for generating expanded words, ii) misspelled word correction with SymSpell[98], and iii) token removal to remove, for

example, punctuation, non-alphanumeric tokens, stopwords, and token length shorter than one. For each Twitter account in the data set, we pre-processed the most recent 50 tweets, the account description, and a combined text produced by concatenating the 50 tweets. We then applied TF-IDF to generate feature vectors for the textual content of each account.

5.3.3 Community Detection

After constructing the directed homogeneous user graph $\overrightarrow{G}$, we applied the Leiden[99] community detection algorithm to find communities $\cup_{i=1}^{i=\mathbb{N}} C_i = \mathbf{N}C$ in the user graph where $\mathbf{N}$ is the number of communities found by the algorithms. This detection algorithm extracts communities by optimizing modularity that compares the relative density of edges inside the community to edges outside the community. The underlying principle of the Leiden algorithm[99] follows the basic principle of the well-known Louvain algorithm[100], which generates communities that are connected, and converge to a locally optimally assigned partition. However, the Leiden algorithm runs far faster than the Louvain algorithm. Figure 5.2 illustrates an abstract structure of a community, as detected by the community detection algorithm, where the three bubble notes and the table of node contents are displayed to simply represent how the proposed approach works on a community. The Look-Up table in the figure keeps the required information needed for each node in a community for future analysis and computation. The process flow diagram in Figure 5.1 shows this step in sub-block 4.

5.3.4 Calculating Community Weight and User Weight

In tracing CTI relevant user accounts U_r as outputs, we calculated the weight of each community after calculating the weights of their corresponding user accounts. We have two different text entity groups: i) all 50 tweets and ii) the description of each Twitter account; these were stored in a new data set, although the tweets and descriptions from the sample user account data set were omitted (discussed in sections 5.4.2 and 5.4.3). Each text from the text data set was analyzed and pre-tagged by the IBM Watson text categorization tool

(see sub-block 2 in Figure 5.1). We applied an efficient semi-automated strategy to rate all tweets and the descriptions from the text data set, assigning a numeric score to texts depending on the pre-tag category generated by IBM Watson NLU. The pre-tag category scores in the data set were specified distinctively by two cybersecurity expert raters (see Section 5.4.1). To show the performance of our proposed approach we used only the sample data set to evaluate the resulting outputs.

To generalize the process, we used a ridge regression model trained on the newly created data set of tweets and descriptions of accounts with their corresponding scores. We fed all tweets and the descriptions of each sample data set user account into the regression model to predict each text's CTI relevance score, which could be used to calculate the weight for each user account. To obtain the weight of a user account, the score for each tweet was added to the score of the account's description of a user account. We used this regression model for generalization because we wanted the proposed approach to work even on different data sets or in different domains not specific to CTI.

For a community C_i detected, we then had each user weight W_{n_i}; we combined all user weights $\sum_{i=1}^{i=\mathbf{M}} W_{n_i} = W_{C_i}$ ($\mathbf{M}$ indicates number of users where $\mathbf{M} \in \mathbb{N}$) into a community to calculate the corresponding community weight W_{C_i}. Very likely, bigger communities will have higher scores than smaller communities because of the large number of users, but a smaller community can also be vital because of the specific shared content and user expertise. To normalize the community weight, therefore, we divided the community weight value by the number of user accounts in a community ($W_{C_i} / \sum_{i=1}^{i=\mathbf{M}} n_i$). At this point in our analysis, we selected only top-p (where p$\in \mathbb{N}$) filtered communities $\bigcup_{i=1}^{|F|} C_i^F$ (where $F \subset \mathbb{N}$) for the next step in the process to minimize computational overhead. Figure 5.1 shows this step in sub-block 5, and Figure 5.3 shows the details of user weight calculation by predicting values from the regression model.

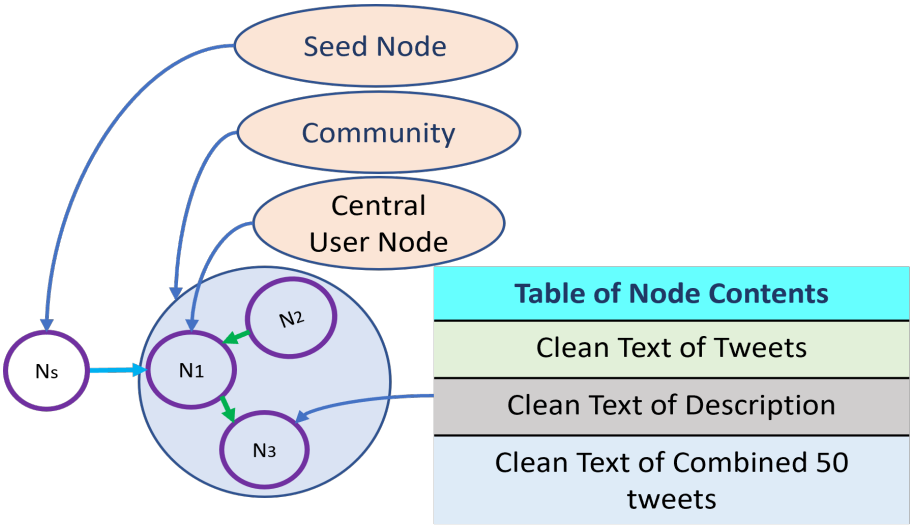

Figure 5.2: *High level structure of a user community in our analysis*

5.3.5 Calculating Text Similarity and Community Ranking

After calculating all community weights $\bigcup_{i=1}^{i=\mathbf{N}} W_{C_i}$ using predicted values from the regression model, we formulated text vectors for each community $\overrightarrow{T}_{C_i}$ using TF-IDF vectorization. However, before doing so, we combined all 50 tweets for each user account into a community to make a single long text T_{n_i} for each community. We also added **Q** input seed node **Q**S_i (where i$\in$ $\mathbb{N}$) texts in text vectorization. Then we calculated cosine similarity *community_cosim_score* = $cosim(\overrightarrow{T}_{C_i}, \overrightarrow{T}_{S_i})$ between the text vector of each seed node T_{S_i} and combined text vector for each community T_{C_i}. At this point, we had top-p communities already ranked based on their weights. For each seed input S_i, we then added each community weight W_{C_i} to the corresponding score of cosine similarity $cosim(\overrightarrow{T}_{S_i}, \overrightarrow{T}_{C_i})$ between the community combined text and seed node text vectors. The community score V_{C_i} could then be calculated using the following equation. See Figure 5.1 sub-block 6.

$$V_{C_i} = W_{C_i} + cosim(\overrightarrow{T}_{S_i}, \overrightarrow{T}_{C_i}). \tag{5.1}$$

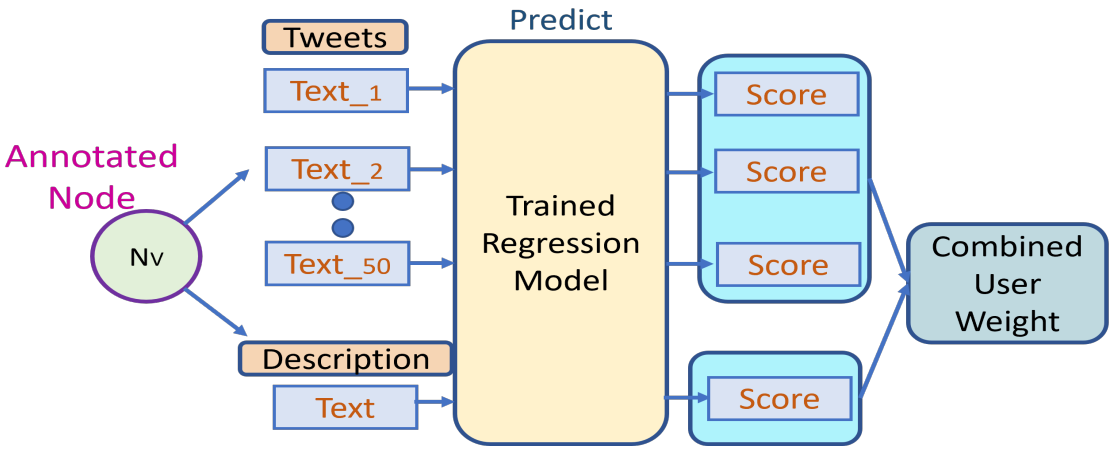

Figure 5.3: *User weight calculation using the predicted values of account tweets and descriptions*

5.3.6 Scoring and Ranking User Nodes

After community ranking ($rank(\bigcup_{i=1}^{i=\mathbb{N}} C_i)$), we started consecutively taking each user n_i from the top p-communities C_i^F based on the community ranking because we wanted to minimize computational complexity by reducing the number of communities $\mathbf{N}C$. Then, we extracted the combined text of the top fifty tweets T_{n_i} for each user n_i and used TF-IDF vectorizing on each user's texts in a community with the seed node tweet texts $T_{\bigcup_{i=1}^{i=\mathbb{N}} S_i}$. We then calculated cosine similarity and added the similarity score $user\_cosim\_score = cosim(\overrightarrow{T}_{n_i}, \overrightarrow{T}_{S_i})$ to the user weight (see subsection 5.3.4). At this point, we already had user accounts with their scores, so the user account indicates output U_r. We ranked the user accounts based on the score V_{U_r} reached by adding user weight W_{n_i} and similarity score $cosim(\overrightarrow{T}_{n_i}, \overrightarrow{T}_{S_i})$ and ranked user accounts by using only the filtered communities $|F|C$ and all seed nodes $\forall S$. The central user in a community may have influenced other members in the community. So we added a partial weight to other user nodes of the community by applying the following formula:

$$U_r - max(U_r) = U_r - maxmal(U_r) + maxmal(U_r) * \beta. \tag{5.2}$$

where [β is the influence factor of the central user account in a community, and *maxmal* is the user account with maximum value].

We then re-ranked the list of user nodes again based on their updated combined scores to produce a user node output list in descending order of priority. Figure 5.1 shows this in sub-block 7.

5.3.7 Validating Model Output

Social network data annotation is an expensive task; thus, we were inspired to develop our approach in a semi-supervised manner. Instead of implementing our framework using the original 50K data set, we extracted a representative sample data set. Then, we ran and validated our framework on the sample data set. To do this, we annotated the sample data set and proposed a novel validation process. User nodes in the sample data set were labeled as "relevant" and "not relevant" based on their relevance to CTI. We calculated the **Precision** value by finding the ratio between the number of resulting "relevant" user accounts and the total number of obtained user accounts. **Recall** value was calculated by finding the ratio between obtained "relevant" user count and the total number of "relevant" users in the sampled annotated data set. We calculated **F1-**β where a user can give input β. We fed the seed accounts as seed nodes into the framework, and the resulting nodes were fed again into the framework to find cyber-threat related nodes. This process iterated for a certain user-defined number of epochs, and after completing the process, we have results to validate performance. We implemented different validation cases according to different percentages of user nodes, so a certain percentage of ranked user nodes was selected to feed into the framework again for a certain number of epochs. For any particular set of seed nodes, and for all different percentages of node selection, we ran this process for both

the proposed method and all other methods. As we executed each method with different sets of seed nodes, we finally averaged all calculated performance metrics values for each node obtained from each set of seed nodes. The final step in the proposed work includes **Precision**, **Recall**, and **F1-β** values for each selection percentage of user nodes. Figure 5.1 shows this process in sub-block 9.

Algorithm 1: Tracing User Accounts

input : seedNodeList
output: User NodeIDs
Edge generation $e(n_{s_i}, n_{d_j})$
Directed graph $\overrightarrow{G}$ construction
Community Detection
for *each community* C_i *generated in* $\cup_{i=1}^{i=\mathbb{N}} C_i$ **do**
  for *each userNode* n_i *in Community* C_i **do**
    $\{n_i : W_{n_i}\} \leftarrow$ regression(U_acc n_i)
  end
  $\{n_i : W_{C_i}\} \leftarrow W_{C_i}$ (where $W_{C_i} = \sum W_{n_i} / \sum n_{c_i}$)
end
$\bigcup_{i=1}^{|F|} C_i^F :=$ Filter_community(top-p)
$\overrightarrow{T}_{S_i} :=$ Text_Vector(seedNodes' Texts T_{S_i})
$\overrightarrow{T}_{C_i^F}, :=$ Text_Vector(filteredCommunity.Texts $T_{C_i^F}$)
$\bigcup_{i=1}^{i=|F|} \{C_i^F : V_{C_i^F}\} := W_{C_i^F} + \text{cosim}(\overrightarrow{T}_{S_i}, \overrightarrow{T}_{C_i^F})$
for *each community in ranked*$(\bigcup_{i=1}^{|F|} C_i^F)$ **do**
  $\overrightarrow{T}^F_{n_i} :=$ Text_Vector($T^F_{n_i}$)
  $\bigcup_{r=1}^{r=|u_r^C|} \{u_r : V_{u_r}\} := W_{n_i} + \text{cosim}(\overrightarrow{T}_{S_i}, \overrightarrow{T}_{n_i^F})$
end
comMap := rank($\bigcup_{i=1}^{i=|F|} \{C_i^F : V_{C_i^F}\}$)
usrMap := rank($\bigcup_{r=1}^{r=|u_r^C|} \{u_r : V_{u_r}\}$)
traced.U_accs := score_distribution(comMap, usrMap)
if *traced.U_accs.scores <= Threashold K* **then**
  validate_result(traced.U_accs u_r)
else
  skip U_acc u_r
end

5.4 Collecting and Preparing Data

To obtain instances of CTI in the form of tracing Twitter user accounts, we wanted to crawl through Twitter user accounts using a Twitter API called Tweepy[101]. We began by scraping specific user details (account descriptions, the contents of the most recent 50 tweets, and lists of "followers" and "followings" U_accs) using a colleague's account. Our colleague is a passionate cyber threat enthusiast and an active Twitter user who follows expert cyber threat professionals and is also followed by other cyber threat enthusiasts. We collected tweet objects up to three levels of the follower and following lists from our colleague's account and collected 50K Twitter user accounts. For each user account, we collected their follower and following lists, account description, and most recent 50 tweets. Then we created a new entity namely "AllText" by concatenating all these recent tweets from each user as well as the corresponding account's description. Figure 5.1 illustrates this in sub-block 1.

5.4.1 Text Rating

The text data set derived by combining tweets and the descriptions (except sample data set users) were rated to train the regression model. We used the scikit-learn ridge regression library where we assigned alpha=1.0 and random_state=241 with default settings for the rest of the parameters. Raters assigned a score to each text of the derived text data set based on text relevance to categories such as "CTI", "Technology" and "Computer". To facilitate and speed up the rating process, raters used a text analytic feature of IBM's Watson Natural Language Understanding (NLU) service[102] called 'Categories' to tag each text by category. This feature of Watson NLU returned a five-level taxonomy of each text, and from them, we used three categories: "antivirus and malware", which was mapped to the CTI; "technology and computing", mapped to technology; and "computer science" mapped to the computer. The categories assigned were limited to the top three of the highest score orders given by the Watson NLU. Raters considered the category of "antivirus and malware" had precedence over "technology and computing", and "Technology and Computing" had precedence over "computer science" for any particular text. If raters agreed with the Watson tagging of a

text, they put 1 for "CTI", 0.5 for "Technology", and 0.1 for "Computer". If raters did not agree on a tweet category that falls in any of the three categories, they put 0 as a score for that tweet. We tagged tweets and the descriptions of 50K user accounts (other than the sample data set users), making up 1802852 tweets and 16596 account descriptions. To generalize our proposed approach, only the text rating process should be modified by assigning a different set of scores to different texts. The scores of texts and their mapping with corresponding tagged categories from IBM Watson would be determined by domain experts.

5.4.2 Extracting Sample Network

To validate the results produced by the model, we extracted a sample data set from the 50K labeled Twitter user accounts and created a sample network from the sample data set. We tried to retain the properties of the original network for creating this sample data set where the root of the sample network remained the same. Starting from the root node, we randomly chose 75 user accounts from each of the Followers and Following lists for a total of 150 user accounts. We found a chance of duplication of IDs because of the network structure. Before moving on to the next step, duplicate user IDs were removed from the collection but not from the network, giving us 148 unique user IDs at this level. In the next stage of creating a sample network, we collected three user accounts from Follower and three from the following lists of these 148 user accounts, giving us 789 unique user accounts. In the end, we had a sample network with 938 unique user accounts.

5.4.3 Annotation of Sample Network

The Twitter accounts in the sample network were manually annotated by two human cyber threat experts. The accounts were classified as either **"relevant"** or **"irrelevant"** with respect to CTI. Accounts were considered relevant to CTI if the description of the account was related to cyber threats and at least two of the collected tweets were about cyber threat incidents. In case no description was present in the account details, but the account has

three tweets related to CTI, the account was labeled as **"relevant"**. Keywords such as "vulnerabilities", "zero-day", "malware attacks", "Phishing", "APT groups", and "cyber espionage" are good indicators of CTI relevance to label text category. The rest of the data in the sample network were labeled as **"irrelevant"**. After taking into account all these specifications to annotate the sampled data set, we ended up with 199 CTI **"relevant"** Twitter accounts out of 938 accounts in the sample network. After that, the text processing steps described previously were applied to the text data before running the validation process on the sample network. Figure 5.1 shows the sample network extraction and annotation steps in sub-block 8 where it was used to validate the proposed method's output. The Inter Annotator Agreement (IAA) of the annotated sample data set by the two annotators is 0.9617 using the Cohen Kappa method.

5.5 Experimental Setup

In this study, we applied the Leiden community detection algorithm for the 50k tagged data set, but we evaluated our results on the sample annotated data set. The standard Python library of this algorithm does not provide any hyperparameter tuning option, so this worked as a black-box method. We evaluated the performance of our results against two relevant user recommendation methods[103], and we applied the Leiden algorithm on the 938 annotated user nodes; the Leiden algorithm found 31 communities out of 938 user nodes. We experimented with three different sets of seed nodes where each set included three "relevant" user nodes that were randomly chosen from the annotated sample data set. Each time, we selected a percentage (from 10% to 100%) of the resulting nodes that were outputted using the given seed nodes. The selected nodes were then fed into all the methods (our proposed method and two other methods for comparison) for 5 epochs. For each method, for one set of seed nodes, we got 10 different values of a performance metric according to the percentage selection (from 10% to 100%) of resulting nodes. Finally, we averaged each performance metric output of all the seed node sets for each method. We used the F1-β performance metric where the value of β was 0.5 and the influence factor was 20%. To trace user nodes,

we retrieved two maximally similar communities from the similarity matrix for each seed user node. We then selected the top 20 users (k=20) from filtered communities ranked by adding the community weight score to its corresponding similarity score. Similarly, user nodes were ranked based on similarity scores added to the corresponding user weights.

5.6 Results and Discussion

To the best of our knowledge, this is the first research to trace CTI related user accounts in the Twitter data stream that considered both structure and content in the network. Our approach is specifically a case of user recommendation, so we adopted two relevant user recommendation methods to compare with our approach.

Friend of Friend (FoF) is a well known user recommendation method[103] in the social network domain that we used to compare to our proposed approach. This approach results in a list of recommended users where a user connected to another user is connected to the target user.

Content-plus-link (CplusL)[103] is a user recommendation method that incorporates content matching and social link information obtained from the underlying structure of a social network. This method emphasizes exposing a network path to a weak tie or an unknown user. The recipient user could thus accept the recommendation.

Table 5.1 shows that the precision of our approach outperformed the two comparable approaches for all the selection threshold percentages. However, the recall values were lower for the selection threshold percentages than either comparable approach. The reason behind this paradigm is that our method results in fewer user nodes whereas the two other methods have more user nodes. Because the number of traced user nodes from our proposed method is not analogous to the other two approaches, the selection percentage cannot make any difference. On the other hand, we value precision more than recall because the purpose of our work is to find user nodes highly relevant to CTI rather than getting more relevant nodes. This realization inspired us to use $F_1 - \beta$ score where the *beta* is 0.5. We can see noticeably improved performance in the $F_1 - \beta$ score. We also calculated the Pearson

Table 5.1: *Performance comparisons with different methods on the sampled annotated data set*

Selection Threshold Percentage	Proposed			FoF			CplusL		
	Prec	Rec	$F_1\beta$	Prec	Rec	$F_1\beta$	Prec	Rec	$F_1\beta$
10%	0.55	0.08	0.26	0.27	0.68	0.32	0.29	0.33	0.30
20%	0.58	0.11	0.32	0.27	0.70	0.32	0.29	0.34	0.29
30%	0.60	0.12	0.33	0.27	0.71	0.32	0.28	0.35	0.29
40%	0.56	0.11	0.31	0.27	0.73	0.31	0.28	0.37	0.29
50%	0.61	0.12	0.34	0.26	0.74	0.31	0.27	0.38	0.28
60%	0.64	0.12	0.35	0.26	0.74	0.31	0.28	0.42	0.30
70%	0.65	0.13	0.36	0.26	0.74	0.31	0.28	0.46	0.30
80%	0.66	0.13	0.37	0.26	0.74	0.31	0.28	0.49	0.31
90%	0.66	0.13	0.37	0.26	0.75	0.30	0.28	0.53	0.30
100%	0.67	0.13	0.37	0.26	0.75	0.30	0.27	0.57	0.30

Correlation Coefficient (R-value) between the predicted values of tweets and descriptions of the sample user network from the regression model and the generated ratings from the semi-automated process. We conducted this analysis to evaluate whether our regression model was effective in predicting a text score regarding CTI where text scores were added for calculating user weights; we found an R-value of 0.7167. In this analysis, user accounts in the same community tend to share similar and relevant content, which helped us find specific information by identifying underlying communities in a social network. Figure 5.4 shows plots of precision, recall, and $F_1 - \beta$ of the introduced approach against comparable approaches on user account tracing for different selection percentages.

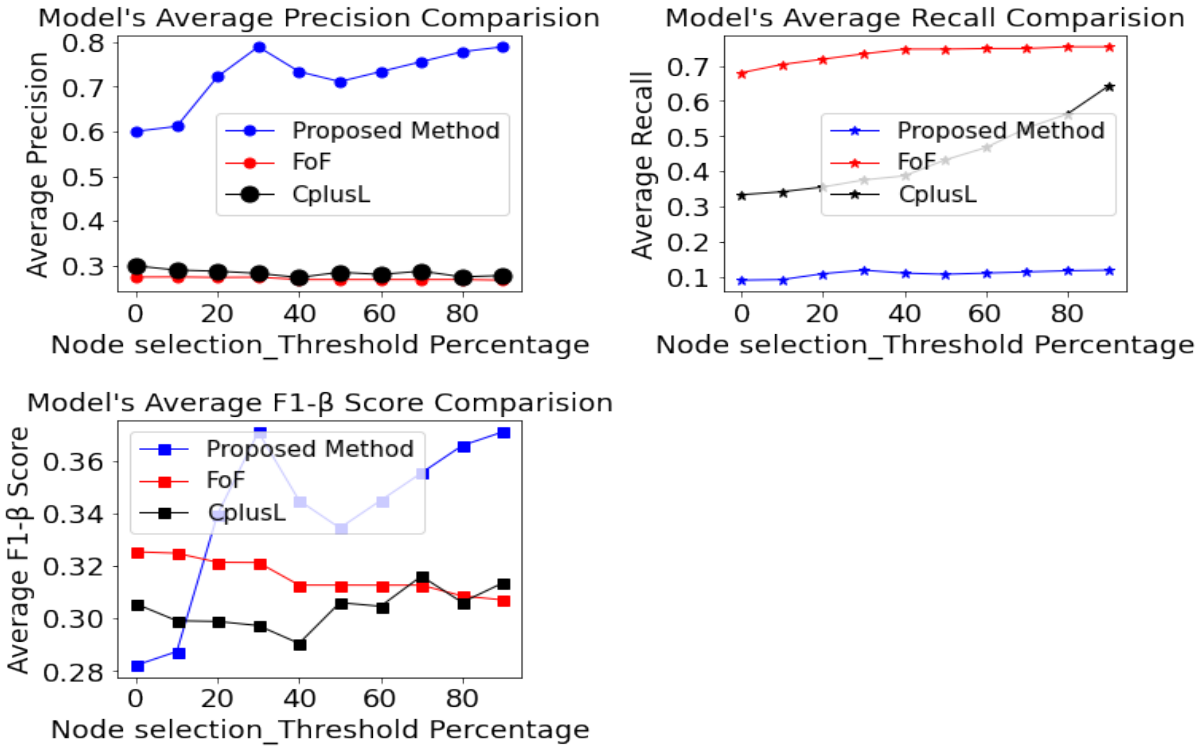

Figure 5.4: *Precision, Recall, and $F_1\beta$ comparison of our approach against two relevant user recommendation approaches*

Question 1: Why is the predictive analysis performed on sample annotated user accounts necessary?

Answer: We mentioned in subsection 5.3.4 that our approach computed the user weight with predicted scores from the regression model. The predictive analysis works for the sample data set users whose texts were unknown to the regression model when the model was trained on the original 50K user data sets. We annotated only the sample user network to evaluate the performance of our approach because user account annotation is an expensive task. To present practically our approach and show its performance, we used the new validation technique after experimenting with our approach on the sampled data set.

Question 2: Why did we develop an iterative validation process?

Answer: One reason for validating iteratively was that we intended to generate an evolving list of traced CTI user accounts. Feeding fixed seed accounts can cause a bias

toward user preference, which should not be encouraged. However, selecting a new set of seed accounts each time from the results can certainly address the bias issues of seed node list selection. A second reason is that iterative filtering of user accounts also increased the probability of tracing more relevant CTI user accounts.

5.7 Conclusion and Future Work

In this semi-supervised approach, we traced and recommended Twitter user accounts that can serve as instances of CTI related information according to a set of given seed nodes. We introduced new methods for accomplishing the task: predicting a text score using the regression model, community formation, user weight calculation, and central user influence measurement. Moreover, our proposed approach can be easily adapted to different domains of interest by training the regression model on a rated data set specific to that domain using the semi-automated rating process. We found that applying different regression algorithms or distinct regression models for individual tweets and descriptions does not make any noticeable difference in performance. We also provided a method of validating and evaluating the results of our approach where the method feeds resulting user nodes as seed nodes iteratively into the process with different selection percentages. However, our approach should be validated with a larger sample of annotated user data sets. An interesting future direction would be to apply a newly introduced tensor-based Graph Convolutional Network on an adequately labeled Twitter user data set.

Chapter 6

Conclusions and Future Research

This chapter provides a summary of the key contributions of the dissertation, focusing on theory, applications, and claims later justified by the experimental results. Finally, I provide suggestions for future research.

6.1 Summary of Contributions

This dissertation presented techniques to improve representation learning on short text corpora for downstream deep learning tasks; the goal was to reduce human effort in data annotation and augmentation. These new techniques were applied to different components of CTI. The novel contributions in this dissertation included theoretical advances for CTI-relevant key phrase extraction using transformer model performance augmentation and threat type classification using a novel GNN. Moreover, I implemented a technique for actively tracing CTI user accounts on the Twitter network and detecting cyber-threat events as integral components for constructing a potential cyber-threat awareness system. In this dissertation, I focused on addressing the following fundamental problems:

- Exploring semi-supervised and unsupervised machine learning techniques to reduce human effort in data labeling and annotation tasks;
- Research direction to improve transformer model performance on raw text data for

extraction;

- Improving node feature learning in graph data and adapting it for text classification;
- Pointing out key components of cyber-threat awareness systems and improving them individually against baselines and SOTAs.

Chapter 2 addressed contextual limitations of current deep learning-based and heuristic key phrase extraction tools as applied in cybersecurity. To address these limitations, I developed a hybrid system that augmented SOTA transformers for labeling key phrase sequences using a novel set of part-of-speech and role-aware tagging rules to generate fine-grained tag sequences from short text corpora. Next, I fine-tuned several SOTA deep learning language model architectures to these transformed sequences. I then evaluated the architectures by measuring the outcomes from the language models to select the best underlying transformers for extracting cybersecurity key phrases. This new ensemble achieves significantly better predictions over SOTA baselines on general cybersecurity corpora. The F1 scores, for instance, are at least 25% higher than hybrid SOTA transformers fine-tuned using baseline tagging rules on the generic corpus, with a significantly lower trade-off on a vulnerability-specific corpus.

Chapter 3 introduced the APKGNN model for short text/node classification tasks. I adopted the attention mechanism and Gaussian mixture model-based parametric kernel as a function of the local graph to extract patches. This combination process resulted in augmented local patches that ultimately convolved with node features to generate updated node representation. The updated learning representation benefits the performance of short text classification. I also applied neighbor sampling with a multi-processing setup that improves even the performance of SOTA models. The implemented GNN convolution layer outperforms SOTA algorithms by a significant margin with better training, validation, and test accuracy for three standard benchmark data sets and one cyber-threat text data set.

Chapter 4 presented a new machine learning and text information extraction approach to detect cyber-threat events in Twitter that were both novel (previously non-extant) and developing (significantly similar to previously detected events). Furthermore, the implemented

approach allows us to rank cyber-threat events using an importance score; we extracted the tweet terms characterized as named entities, keywords, or both. Influence is also imputed to users according to their followers and following relationships to rank both types of events. Finally, the performance of the implemented approach was evaluated, measuring the efficiency and detection error rate for events over a specified time interval that is relative to human annotator ground truth.

Chapter 5 introduced a way to trace user accounts to monitor as instances of CTI. In this chapter, a novel approach was implemented to trace cyber-threat associated user accounts in the Twitter data stream: ranking users were more accurate with contextual relevance and topological information extracted by finding user communities on the Twitter network. I used structural information in the graph network and tweet content from user accounts to find relevant user accounts through previously identified seed user accounts. The implemented method outperformed two baseline methods using annotated data of CTI-related Twitter user accounts to trace accounts as instances of CTI.

6.2 Future Research

In the future, researchers may want to focus on the following

CTI Application

- User account tracing and recommendations using GNN

  User account tracing and recommendations from social networks can be considered as an application of user community detection. Traditional modularity optimization-based research is a black box method, whereas deep learning approaches detect community by learning static node features and network structural information. Applying GNNs could be an appropriate approach to community detection tasks because they have multi-dimensional nodes and connection features. This leaves a wide scope for research that applies efficient GNN models to identify cyber-threat relevant user communities from social networks.

- Dynamic Event Detection and Ranking with Time Series Analysis

  Social networks generate a massive amount of data within minutes, posing challenges for all downstream analytical tasks such as classification and detection. Dynamic or run-time versions of these processes directly acquire data from streams and create additional overhead in the processing pipeline. This causes scalability issues. Therefore, to implement real-life cyber-threat awareness applications, scalability issues must be addressed.

- Define trade off between generalizability and specificity

  Defining a margin between generic and domain-specific methods of key phrase identification is a significant challenge. Domain-specific methods can distinguish specific features from the text but do not transfer or adapt well to other domains like different social networks, news articles, and online blogs. Generic methods, on the other hand, face the problem of identifying important domain-specific key phrases. This complexity and unknown nature of cyber-threat indicators mean that specificity requirements remain crucial. Policies thus must be determined and proper measures taken to define the level of specificity and validate experimental results.

- Improve the GNN model by minimizing number of parameters

  The novel GNN model introduced in this dissertation does outperform SOTA models in node classification, but it also requires learning more parameter vectors than other models. The overhead of learning additional parameters means the model requires more time to train. The direction of this continuing research would be to reduce the number of parameters.

- Apply GNNs to Parallalize on Multiple GPUs Efficiently

  GNN model training is more computationally expensive than comparable CNN models because of necessary additional matrix multiplications. The current GNN library has insufficient support to parallelize model training on multiple GPUs, which is crucial

for real-life data sets. The next step in the research would be to investigate how to efficiently parallelize the GNN model training.

Bibliography

[1] Huichen Yang and William H Hsu. Named entity recognition from synthesis procedural text in materials science domain with attention-based approach. In *SDU@ AAAI*, 2021.

[2] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[3] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[4] Quanzhi Li, Armineh Nourbakhsh, Sameena Shah, and Xiaomo Liu. Real-time novel event detection from social media. In *2017 IEEE 33Rd international conference on data engineering (ICDE)*, pages 1129–1139. IEEE, 2017.

[5] Adam Marcus, Michael S Bernstein, Osama Badar, David R Karger, Samuel Madden, and Robert C Miller. Twitinfo: aggregating and visualizing microblogs for event exploration. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 227–236, 2011.

[6] Sina Dabiri and Kevin Heaslip. Developing a twitter-based traffic event detection model using deep learning architectures. *Expert systems with applications*, 118:425–439, 2019.

[7] Sudip Mittal, Prajit Kumar Das, Varish Mulwad, Anupam Joshi, and Tim Finin. Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 860–867. IEEE, 2016.

[8] Amosse Edouard. *Event detection and analysis on short text messages*. PhD thesis, Université Côte D'Azur, 2017.

[9] Rupinder Paul Khandpur, Taoran Ji, Steve Jan, Gang Wang, Chang-Tien Lu, and Naren Ramakrishnan. Crowdsourcing cybersecurity: Cyber attack detection using social media. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1049–1057, 2017.

[10] Kuo-Chan Lee, Chih-Hung Hsieh, Li-Jia Wei, Ching-Hao Mao, Jyun-Han Dai, and Yu-Ting Kuang. Sec-buzzer: cyber security emerging topic mining with open threat intelligence retrieval and timeline event annotation. *Soft Computing*, 21(11):2883–2896, 2017.

[11] Anna Sapienza, Sindhu Kiranmai Ernala, Alessandro Bessi, Kristina Lerman, and Emilio Ferrara. Discover: Mining online chatter for emerging cyber threats. In *Companion Proceedings of the The Web Conference 2018*, pages 983–990, 2018.

[12] Quentin Le Sceller, ElMouatez Billah Karbab, Mourad Debbabi, and Farkhund Iqbal. Sonar: Automatic detection of cyber security events over the twitter stream. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–11, 2017.

[13] Aditya Pingle, Aritran Piplai, Sudip Mittal, Anupam Joshi, James Holt, and Richard Zak. Relext: Relation extraction using deep learning approaches for cybersecurity knowledge graph improvement. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 879–886, 2019.

[14] Aritran Piplai, Sudip Mittal, Anupam Joshi, Tim Finin, James Holt, and Richard Zak. Creating cybersecurity knowledge graphs from malware after action reports. *IEEE Access*, 8:211691–211703, 2020.

[15] Anastasiia Sirotina and Natalia Loukachevitch. Named entity recognition in information security domain for russian. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1114–1120, 2019.

[16] Varish Mulwad, Wenjia Li, Anupam Joshi, Tim Finin, and Krishnamurthy Viswanathan. Extracting information about security vulnerabilities from web text. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 257–260. IEEE, 2011.

[17] Arnav Joshi, Ravendar Lal, Tim Finin, and Anupam Joshi. Extracting cybersecurity related linked data from text. In *2013 IEEE Seventh International Conference on Semantic Computing*, pages 252–259. IEEE, 2013.

[18] Sachini Weerawardhana, Subhojeet Mukherjee, Indrajit Ray, and Adele Howe. Automated extraction of vulnerability information for home computer security. In *International Symposium on Foundations and Practice of Security*, pages 356–366. Springer, 2014.

[19] RH Nidhi and B Annappa. Twitter-user recommender system using tweets: A content-based approach. In *2017 International Conference on Computational Intelligence in Data Science (ICCIDS)*, pages 1–6. IEEE, 2017.

[20] Shulin Cheng, Bofeng Zhang, Guobing Zou, Mingqing Huang, and Zhu Zhang. Friend recommendation in social networks based on multi-source information fusion. *International Journal of Machine Learning and Cybernetics*, 10(5):1003–1024, 2019.

[21] Gang Zhao, Mong Li Lee, Wynne Hsu, Wei Chen, and Haoji Hu. Community-based user recommendation in uni-directional social networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 189–198, 2013.

[22] Jorge Valverde-Rebaza and Alneu de Andrade Lopes. Exploiting behaviors of communities of twitter users for link prediction. *Social Network Analysis and Mining*, 3(4): 1063–1074, 2013.

[23] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[24] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[25] V. Behzadan, C. Aguirre, A. Bose, and W. Hsu. Corpus and deep learning classifier for collection of cyber threat indicators in twitter stream. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 5002–5007, 2018. doi: 10.1109/BigData.2018.8622506.

[26] Robert A Bridges, Corinne L Jones, Michael D Iannacone, Kelly M Testa, and John R Goodall. Automatic labeling for entity extraction in cyber security. *arXiv preprint arXiv:1308.4941*, 2013.

[27] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[28] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.

[29] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[30] Abdulkareem Alsudais and Hovig Tchalian. Clustering prominent named entities in topic-specific text corpora. 2019.

[31] Angli Liu, Jingfei Du, and Veselin Stoyanov. Knowledge-augmented language model and its application to unsupervised named-entity recognition. *arXiv preprint arXiv:1904.04458*, 2019.

[32] Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. Bond: Bert-assisted open-domain named entity recognition with distant

supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1054–1064, 2020.

[33] J Devlin M Chang K Lee and K Toutanova. Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[34] S Thenmalar, J Balaji, and TV Geetha. Semi-supervised bootstrapping approach for named entity recognition. *arXiv preprint arXiv:1511.06833*, 2015.

[35] Atefeh Zafarian, Ali Rokni, Shahram Khadivi, and Sonia Ghiasifard. Semi-supervised learning for named entity recognition using weakly labeled training data. In *2015 the international symposium on artificial intelligence and signal processing (AISP)*, pages 129–135. IEEE, 2015.

[36] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[37] Houssem Gasmi, Abdelaziz Bouras, and Jannik Laval. Lstm recurrent neural networks for cybersecurity named entity recognition. *ICSEA*, 11:2018, 2018.

[38] Ivan Mazharov and Boris V Dobrov. Named entity recognition for information security domain. In *DAMDID/RCDL*, pages 200–207, 2018.

[39] Tiberiu-Marian Georgescu, Bogdan Iancu, and Madalina Zurini. Named-entity-recognition-based automated system for diagnosing cybersecurity situations in iot networks. *Sensors*, 19(15):3380, 2019.

[40] Ya Qin, Guo-wei Shen, Wen-bo Zhao, Yan-ping Chen, Miao Yu, and Xin Jin. A network security entity recognition method based on feature template and cnn-bilstm-crf. *Frontiers of Information Technology & Electronic Engineering*, 20(6):872–884, 2019.

[41] Tao Li, Yuanbo Guo, and Ankang Ju. A self-attention-based approach for named entity

recognition in cybersecurity. In *2019 15th International Conference on Computational Intelligence and Security (CIS)*, pages 147–150. IEEE, 2019.

[42] Han Zhang, Yuanbo Guo, and Tao Li. Multifeature named entity recognition in information security based on adversarial learning. *Security and Communication Networks*, 2019, 2019.

[43] Shengping Zhou, Zi Long, Lianzhi Tan, and Hao Guo. Automatic identification of indicators of compromise using neural-based sequence labelling. *arXiv preprint arXiv:1810.10156*, 2018.

[44] Han Wu, Xiaoyong Li, and Yali Gao. An effective approach of named entity recognition for cyber threat intelligence. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 1370–1374. IEEE, 2020.

[45] Sudip Mittal, Anupam Joshi, and Tim Finin. Thinking, fast and slow: Combining vector spaces and knowledge graphs. *arXiv preprint arXiv:1708.03310*, 2017.

[46] Houssem Gasmi, Jannik Laval, and Abdelaziz Bouras. Information extraction of cybersecurity concepts: an lstm approach. *Applied Sciences*, 9(19):3945, 2019.

[47] Valerie Mozharova and Natalia Loukachevitch. Two-stage approach in russian named entity recognition. In *2016 International FRUCT Conference on Intelligence, Social Media and Web (ISMW FRUCT)*, pages 1–6. IEEE, 2016.

[48] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[49] Nasik Muhammad Nafi, Avishek Bose, Sarthak Khanal, Doina Caragea, and William H Hsu. Abstractive text summarization of disaster-related document. In *ISCRAM 2020 Conference Proceedings–17th International Conference on Information Systems for Crisis Response and Management*, 2020.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[51] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[52] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[53] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[54] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[55] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[56] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[57] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[58] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[59] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[60] Paolo Frasconi, Marco Gori, and Alessandro Sperduti. A general framework for adaptive processing of data structures. *IEEE transactions on Neural Networks*, 9(5):768–786, 1998.

[61] Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

[62] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks*, volume 2, pages 729–734, 2005.

[63] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks, 2017.

[64] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[65] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

[66] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(11), 2006.

[67] Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Proceedings of the 25th international conference on Machine learning*, pages 1168–1175, 2008.

[68] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[69] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[70] James Atwood and Don Towsley. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[71] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023. PMLR, 2016.

[72] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[73] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[74] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[75] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[76] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[77] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[78] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[79] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377, 2019.

[80] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877, 2018.

[81] Georgiana Ifrim, Bichen Shi, and Igor Brigadir. Event detection in twitter using aggressive filtering and hierarchical tweet clustering. In *Second Workshop on Social News on the Web (SNOW), Seoul, Korea, 8 April 2014*. ACM, 2014.

[82] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):1–37, 2008.

[83] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[84] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

[85] Alan Ritter, Evan Wright, William Casey, and Tom Mitchell. Weakly supervised extraction of computer security events from twitter. In *Proceedings of the 24th international conference on world wide web*, pages 896–905, 2015.

[86] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[87] Wolf Garbe. *¡wolf.garbe@faroo.com¿*, ("SymSpell 6.4").

[88] Radim Rehurek and Petr Sojka. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks*. Citeseer, 2010.

[89] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[90] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[91] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[92] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[93] Andy Piper. Potential adjustments to streaming api sample volumes. *Retrieved August*, 24:2019, 2015.

[94] A. Bose, V. Behzadan, C. Aguirre, and W. H. Hsu. A novel approach for detection and ranking of trendy and emerging cyber threat events in twitter streams. In *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 871–878, 2019. doi: 10.1145/3341161.3344379.

[95] Avishek Bose, Shreya Gopal Sundari, Vahid Behzadan, and William H Hsu. Tracing relevant twitter accounts active in cyber threat intelligence domain by exploiting

content and structure of twitter network. In *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE, 2021.

[96] Liming Pan, Tao Zhou, Linyuan Lü, and Chin-Kun Hu. Predicting missing links and identifying spurious links via likelihood analysis. *Scientific reports*, 6(1):1–10, 2016.

[97] Iftikhar Ahmad, Muhammad Usman Akhtar, Salma Noor, and Ambreen Shahnaz. Missing link prediction using common neighbor and centrality based parameterized algorithm. *Scientific reports*, 10(1):1–9, 2020.

[98] Wolf Garbe. Symspell 6.4, 2020. URL https://github.com/wolfgarbe/symspell.

[99] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

[100] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[101] Joshua Roesslein. Tweepy documentation, 2020. URL https://docs.tweepy.org/en/v3.10.0/.

[102] IBM. Ibm watson natural language understanding documentation, 2021. URL https://cloud.ibm.com/apidocs/natural-language-understanding?code=python.

[103] Jilin Chen, Werner Geyer, Casey Dugan, Michael Muller, and Ido Guy. Make new friends, but keep the old: Recommending people on social networking sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, page 201–210, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605582467. doi: 10.1145/1518701.1518735. URL https://doi.org/10.1145/1518701.1518735.