NAGOYA UNIVERSITY

DOCTORAL THESIS

Effective application of Natural Language Processing techniques in Automated Cyber Threat Intelligence

Author:
MENDSAIKHAN Otgonpurev

Graduate School of Informatics

January 21, 2021

NAGOYA UNIVERSITY

Abstract

Graduate School of Informatics

Doctor of Philosophy

Effective application of Natural Language Processing techniques in Automated Cyber Threat Intelligence

by MENDSAIKHAN Otgonpurev

The latest advancements of Artificial Intelligence (AI) techniques are complicating the cyber threat landscape. In this arms race, the cybersecurity defenders need to automate their tools to be competent enough against these ever-increasing threats. This thesis proposes to utilize Natural Language Processing techniques for cyber defense, specifically in the Cyber Threat Intelligence process. As a demonstration, I have developed a prototype system that identifies cybersecurity specific text content, analyzes the significance and relevance of it, and enriches it with the existing threat information. The proposed system consists of the following modules.

1. **Natural Language Filter** module classifies and filters the cybersecurity-related text documents from any information source. It has been implemented using Doc2Vec and BERT language models to identify and filter the security-related text documents.

2. **Analyzer** module determines the significance and relevance of the threat information to the user. It has been implemented using a novel approach of engineering the features of the text through Knowledge Graph and Named Entity Recognition methods.

3. **Mapper** module enriches the threat information with the adversarial tactics and techniques. It has been implemented by converting the threat information into its vector representation and applying multi-label classification on it.

Each module has been independently experimented and the individual results support the utilization of particular method. Essentially, it could be inferred that by utilizing various Natural Language Processing techniques in the Cyber Threat Intelligence process the cyber defense could be improved, specifically in situational awareness and security automation operations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The digital age has presented various opportunities to society and to business in general. However, these opportunities also bring with them different kinds of risks such as cyber-attacks, data breaches, loss of intellectual property, financial fraud, etc. To mitigate and minimize these risks the field of cybersecurity has been developing various defense approaches. One such approach is called Cyber Threat Intelligence (CTI) which utilizes existing knowledge about cyber threats to proactively mitigate a cyber-attack before it happens, or resolve the intrusion with minimal damage using the systematic knowledge. Through the CTI, the organizations are able to systematically identify the threat actors, prioritize the defense, share the threat indicators with each other, and mitigate them effectively.

On the other hand, with the growth of the processing power and accumulation of digital data, various forms of machine intelligence are forming that would further complicate the cyber threat landscape. Once a theoretical only concepts such as techniques of Machine Learning (ML), Natural Language Processing (NLP), etc have been applied in the day to day life and it is a matter of time for cyber adversaries to utilize those approaches efficiently. In this arms race, defenders have to embrace the technological shift, so that the latest researches of machine intelligence have to be embedded in the defensive tools. There have been various approaches to utilize ML and other Artificial Intelligence (AI) fields in cyber defense, especially in network security through various Intrusion Detection Systems (IDS), Security Information and Event Management (SIEM) systems, etc. However, I believe that cyber defense could be further improved by utilizing NLP and other AI techniques in the CTI process. To do that, I propose to apply various NLP techniques in the prototype system that could be used for collecting, analyzing, and enriching cyber threat information in text format. Natural Language Processing is a subfield of Artificial Intelligence that processes and analyzes the natural language into machine-understandable form. Hence, the proposed prototype system identifies and extracts the cyber threat-related information from massive textual documents, analyzes their significance and relevance to the user, and finally enriches the document with the more systematic threat information.

The goal of this thesis is to prove the applicability of the various NLP techniques in the CTI process. Therefore, the proposed system uses embedding, transfer learning, text classification, Named Entity Recognition (NER), Knowledge Graph, various algorithms of the single label and multi-label classification methods to demonstrate the effectiveness of such techniques in the CTI process.

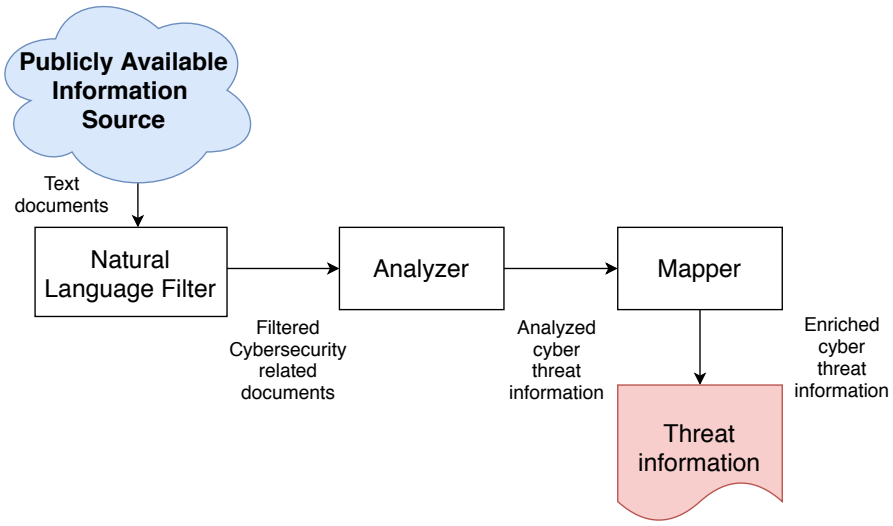

FIGURE 1.1: Proposed system architecture

1.2 Overview of this Dissertation

In order to prove the effective application of Natural Language Processing techniques in the Cyber Threat Intelligence process, this thesis proposes a system that deploys various NLP and ML techniques and assists in the automation of the security analyst.

1.2.1 Model of the Proposed System

I have envisioned a system that automatically identifies, analyzes, and enriches cyber threat information to assist security analysts in building situational awareness with three modules including a Natural Language Filter, Analyzer, and Mapper. The proposed system architecture is depicted in Fig. 1.1.

The proposed system could be used to scan the Internet to create situational awareness and to assist security analysts in identifying risks and threats posed to their organizations. The Natural Language Filter module classifies and filters the cybersecurity-related text documents. The collected and filtered documents are analyzed by an Analyzer module to determine the significance and relevance to the user, thus feeding only the useful threat information. The analyzed documents that are significant and relevant to the user are fed into a Mapper module to enrich with the adversarial tactics and techniques information thus assisting in the threat analysis and CTI process.

1.2.2 Natural Language Filter module

The Natural Language Filter module is a language model that is trained to identify and filter the security-related text documents from publicly available information sources. In [32] I have experimented with the Doc2Vec language model to utilize as Natural Language Filter by training it with over 1 million security-specific text documents. The model would compare the cosine similarity of the vector representation of any incoming text document with its training document and filter out the documents that have less than 70% similarity. With custom preprocessing of the text

documents, it was possible to achieve 83% accuracy. Subsequently, the state-of-the-art model Bidirectional Encoder Representations from Transformer (BERT) has been experimented with and improved this result to 90%. More details regarding this experiment are discussed in Chapter 2. This chapter is an adapted version of the publication [34].

1.2.3 Analyzer module

The collected and filtered documents may contain a lot of irrelevant information. I propose a novel approach to represent the relevance and significance of the cybersecurity text in quantitative numbers. In this approach, a Custom Named Entity Recognition (NER) model has been trained and a Cybersecurity Knowledge Graph (CKG) has been constructed to infer the subjective relevance of the cybersecurity text to the user and to generate correlation features. In addition, the significance of the given text was analyzed in terms of its textual similarity with different repositories of pre-defined "significant" text, and the maximum similarities were computed. These analysis results then act as features of the classifier to generate the significance score. The experimental result showed that the overall system could determine the significance and relevance of the text within a controlled environment with 88% accuracy. More details regarding this experiment are discussed in Chapter 3. This chapter is an adapted version of the publication [33].

1.2.4 Mapper module

In order to automate the tasks of the security analyst the analyzed significant and relevant documents need to be enriched with a common reference. There have been various approaches and threat models developed to generalize the threat landscape and to ease the burden of a security analyst. The most popular threat model MITRE ATT&CK proved to be a very useful tool for the security analyst to perform cyber threat intelligence, red and blue teaming, and so on. However, mapping any threat information to the adversarial techniques listed in MITRE ATT&CK requires a great deal of domain knowledge and analytic experience. Thus, I propose a method to automatically map the software security vulnerability using a multi-label classification approach. The proposed approach took the vector representation of the vulnerability description and classified it with various multi-label classification methods to evaluate in different measures and found out the LabelPowerset method with Multilayer Perceptron as base classifier performs best in this experiment. More details regarding this experiment are discussed in Chapter 4. This chapter is an adapted version of the publication [35].

1.3 Related Work

There have been various approaches to utilize NLP and ML techniques in cyber defense. However, to the best of my knowledge, currently, there are no published works to utilize different NLP techniques in a single system to automate and assist the process of Cyber Threat Intelligence. The works related to this study are categorized as Automated Threat Detection and Language Models, Cybersecurity Knowledge Graph, Cybersecurity Named Entity Recognition, Text classification, MITRE ATT&CK Framework and Vulnerability Classification works. This work could be seen as the amalgamation of these different domains.

1.3.1 Automated Threat Detection and Language Models

There have been a number of attempts to automatically identify or extract cyber-threat-related information from the unstructured text. Mulwad et al. proposed a framework to identify and generate assertions about vulnerabilities, threats, and attacks from web text by using an SVM classifier and Wikitology, an ontology-based on Wikipedia [42]. Joshi et al. proposed an information extraction framework that extracts cybersecurity entities, terms, and concepts to map them to related web resources and create an open ontology [20]. More et al. proposed a knowledge-based approach to intrusion-detection modeling in which the intrusion-detection system automatically fetches threat information from web-based text information and proactively monitors the network to establish situational awareness. Their approach focused mainly on developing a cybersecurity ontology that could be understood by intrusion-detecting machines [41]. Jones et al. proposed a bootstrapping algorithm to extract cybersecurity entities and identify their relationships using Brin's Dual Iterative Pattern Relation Expansion (DIPRE) algorithm, which uses a cyclic process to iteratively build known relation instances and heuristics for finding those instances [41]. Also, Dionísio et al. developed a system to detect cyber-threats from Twitter using deep neural networks [11]. Their work has many similarities with this thesis, e.g. collecting relevant threats from Twitter feeds and identifying the assets with a Named Entity Recognizer. Husari et al. developed a system to automate Cyber Threat Intelligence (CTI) analytics that learns attack patterns [18]. They combined NLP and IR techniques to extract threat actions from threat reports based on semantic relationships. These works focused to extract cyber-threat-related information from a text which is similar to the proposed system. However, the approach I have taken is to first identify the objective relevance (cybersecurity domain topic) and then find the subjective relevance (user-specific threat) from the textual information.

Al-Rowaily et al. [2] developed a Bilingual Sentiment Analysis Lexicon for the cybersecurity domain that can be used to develop opinion mining and sentiment analysis systems for bilingual textual data from Dark Web forums. Proposed approach aims to provide automated identification from a text corpus, whether it is a Dark Web forum or any other publicly available information source, using the semantic representation of the text document. Tavabi et al. developed DarkEmbed, a system to predict the probability of the specific vulnerability getting exploited using neural language model. They have utilized Skip-Gram Word2Vec model to generate low dimensional document embedding and classified them using Support Vector Machine [58]. Their work highlighted the importance of domain specific embedding, which is demonstrated in this research as well.

There have been various approaches to implement domain-specific language models based on Word2Vec or Doc2Vec frameworks and their variations. Niu et al. [44] developed the Topic2Vec approach, which can learn topic representations in the same semantic vector space as used for words. Dhingra et al. [10] described character-based distributed representations for social media by introducing their Tweet2Vec language model. The character-level representations show interesting results for learning informal conversations. Choi et al. [8] proposed a multi-layer representation learning for medical concepts. Med2Vec research has shown that a domain-specific language model could achieve better results than the generic language model.

Some attempts to utilize a Doc2Vec language model for the domain-specific task has been experimented as well. Aman et al. [4] described a system utilizing a

Doc2Vec-based language model to assess comments and its application to change prone method analysis. The research result shows that Doc2Vec could be utilized for domain-specific tasks like comparing source code comments to the programming statements. Karvelis et al. [22] proposed a topic recommendation system using a Doc2Vec-based language model. The automated topic recommendation system based on the Doc2Vec model suggests that the proposed system that classifies cybersecurity related text data could be realized in practice.

Since the inception of the BERT language model, there have been various attempts to train and utilize it for specific domains. Jinhyuk Lee et al. described BioBERT, which was trained for unannotated biomedical text corpora [28]. BioBERT outperformed the generic BERT model on the biomedical text mining tasks. Similar results have been achieved by Iz Beltagy et al. with SciBERT, a pre-trained language model for scientific text [5]. SciBERT has been pre-trained on 1.14M papers of various domains from Semantic Scholar and even outperformed BioBERT on biomedical tasks. Jieh-Sheng Lee et al. described a patent classification system created by fine-tuning the BERT language model [27]. Their model outperformed the state-of-the-art results in classifying the patent claims. Kexin Huang et al. pre-trained and fine-tuned BERT on clinical notes to develop ClinicalBERT [17]. The ClinicalBERT has outperformed baselines to predict readmission of patients based on the clinical notes. Chen Sun et al. proposed a joint visual-linguistic model to learn high-level features by building a model based upon BERT to learn bidirectional joint distributions over sequences of visual and linguistic tokens, derived from vector quantization of video data and off-the-shelf speech recognition outputs, respectively [56]. The proposed VideoBERT model is used in various tasks including action classification and video captioning. Ashutosh Adhikari et al. presented DocBERT for document classification [1]. DocBERT achieved state-of-the-art results across four popular datasets when applied to straightforward document classification task. Although BERT has achieved outstanding results in various domains, its potential has yet to be fully explored in the cybersecurity domain.

1.3.2 Cybersecurity Knowledge Graph

There have been several proposals to extract the relationships of cybersecurity entities and to build cybersecurity Knowledge Graph from unstructured text. Pingle et al. proposed a system called RelExt that would extract possible relationships and create semantic triples over cybersecurity text, using a deep-learning approach [51]. Consequently, Piplai et al. developed a system to extract information from malware After Action Reports (AAR) that can be merged to create a cybersecurity Knowledge Graph using RelExt [52]. In addition, Jia et al. proposed an approach to build a cybersecurity knowledge base and deduction rules based on a quintuple model [19]. Both works focused on developing a comprehensive approach to effectively extract cybersecurity entity relationships and build cybersecurity Knowledge Graph, whereas focus of this research is to utilize existing cybersecurity Knowledge Graph to infer correlations between entities.

1.3.3 Cybersecurity Named Entity Recognition

The latest trends in Named Entity Recognition (NER) has been in deep neural network architecture. Yadav et al. surveyed the recent advances in Named Entity

Recognition focused on neural architectures and compared them to previous feature-based systems [62]. The paper's finding has shown that incorporating the characteristics of feature-engineered models into modern neural network architectures could yield better results. Another development in the NER field is the constituent-based tagging scheme in which a conventional tagging scheme to denote entities is replaced by a more constituent specific tagging scheme. Zhong et al. proposed TOMN and UGTO tagging schemes to better indicate the time expression and compared the performance with state-of-the-art models [65]. The experimental results demonstrated that the proposed models trained with a constituent-based tagging scheme perform equally or more effectively than the representative state-of-the-art models indicating the potential in the approach. These advances have been attracting some research interest in the cybersecurity field, especially to utilize deep-learning architectures. Simran et al. proposed a deep-learning-based framework for NER in cybersecurity and evaluated various deep-learning architectures [21]. Gasmi et al. proposed an LSTM model for NER and Relation Extraction tasks [13]. Even though, not a deep learning approach Yi et al. also proposed cybersecurity NER model based on regular expressions and known-entity dictionary [64]. Their proposed models achieved competitive performance when compared with feature engineered models. Since achieving these state-of-the-art results for NER tasks was not the objective of this work, a simpler, but efficient, Conditional Random Fields (CRF) model have been utilized for the cybersecurity entity identification task.

1.3.4 Text Classification

In general, this work can be viewed as a text classification task by engineering the features of the text. In this regard, there have been numerous works that reviewed and compared the performances of various methods of text classification. Minaee et al. did a comprehensive review of 150 deep learning-based models for the latest state-of-the-art text classification methods [40]. In the paper, the authors reviewed the performance of various language models on different text classification tasks from which the News Categorization task is the most similar task to this approach. In the News Categorization task Transformer based Pre-trained Language Model XLNet has shown the highest performance in the AG News dataset. Yang et al. introduced XLNet in [63] as a generalized autoregressive pre-trained model to overcome the limitations of the state-of-the-art language model BERT. Even though XLNet is a state-of-the-art model, its computationally expensive nature makes it difficult for customization such as identifying the relevant text to the user. Hence, this proposal seeks to develop a domain-specific text classification model that can be easily customized to classify the significant and relevant text to the user.

1.3.5 MITRE ATT&CK Framework and Vulnerability Classification

The Mapper module is intended to associate and enrich the cyber threat information identified through Natural Language Filter and Analyzer modules with its adversarial tactics and techniques enlisted in the MITRE ATT&CK framework. Similar approaches related to MITRE ATT&CK framework and general vulnerability classification works are listed below.

Legoy et al. implemented a tool called rcATT, a system that predicts tactics and techniques related to given cyber threat reports and outputs the results using Structured Threat Information eXpression (STIX) format [29]. They focused to extract MITRE ATT&CK techniques and tactics from cyber threat reports and used simpler

approaches for text representation and classification algorithms, whereas this thesis focused to map the vulnerability description to the same framework, though using more neural and deep learning approaches. Apart from extracting an adversarial technique from textual documents, there have been some studies to directly map the malware behavior to the MITRE ATT&CK framework. Oosthoek et al. did the automated analysis of 951 unique families of Windows malware and mapped them onto the MITRE ATT&CK framework [45]. They generated a behavior signature of the malware in the sandbox and mapped the signature to the corresponding MITRE ATT&CK technique. Their work focused to map the malware based on its behavior to the adversarial techniques defined in MITRE ATT&CK framework whereas focus of this research is to map the vulnerability description that could be exploited by the adversary to the same techniques through its textual representation. Some researchers have been working on the information provided by the MITRE ATT&CK framework to improve the adversarial predictions. Al-Shaer et al. presented their statistical machine learning analysis on Advanced Persistent Threat (APT) and software attack data reported by MITRE ATT&CK to infer and predict the techniques the adversary might use [3]. They associated adversarial techniques using hierarchical clustering with 95% confidence, providing statistically significant and explainable technique correlations. Focus of this research is to correlate individual vulnerability descriptions to the adversarial techniques and create a model that can be used to automatically map new vulnerability to the MITRE ATT&CK framework. There have been also research on classifying the vulnerability information based on its textual description. Huang et al. proposed an automatic vulnerability classification model built on Term Frequency-Inverse Document Frequency (TF-IDF), Information Gain (IG), and deep neural network [16]. They validated their model with CVE descriptions of the National Vulnerability Database and compared them to the performances of SVM, Naive Bayes, and kNN algorithms. This thesis also attempts to classify the vulnerability information based on its textual description, but Huang et al. focused a multi-class classification that each vulnerability belongs to a specific category, whereas this thesis attempt to classify a vulnerability into multiple adversarial techniques at the same time.

Chapter 2

Identifying Cybersecurity Specific Content

2.1 Introduction

In the age of digital information, extracting the relevant content from the massive flow of data becomes a challenge. Recently, with the rise of the social networks as well as ubiquitous computing, the total amount of digital text content is increasing. One of the tasks of a security analyst is to establish the situational awareness, thus identifying cyber threat related information to proactively monitor, prevent the possible intrusion and control the possible risk. For this reason, this chapter is proposing a module of the system that employs Natural Language Processing techniques to identify the cyber threat related information and filter out the irrelevant content.

In [31], I proposed to filter security related text documents using the keyword generation method. However, after several experiments, it became clear that the words should be treated as non-atomic entities in order to preserve their semantic relationship. Fortunately, the field of Natural Language Processing (NLP) has been advancing recently and there are techniques to train machines to understand the semantic relationships between words. Since computers can work only with numbers, the computational linguistics represent text in vector space to capture and calculate the semantic relationships between the words [60]. The process of converting text into numerical vectors is called embedding. Earlier embedding techniques such as Bag of Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF) have been preceded by neural network embedding techniques to represent the words in vector space. Mikolov et al. [37] proposed a groundbreaking method called Word2Vec to represent and compute the semantic similarities of the words in vector space. Given the success of the Word2Vec model, the authors proposed another method called Doc2Vec [26] that could represent the documents in vector space using a similar approach. In [32], I proposed to utilize the Doc2Vec-based language model to identify the cybersecurity related text documents from publicly available information sources. Since the publication of the paper, LSTM-based ELMo [49], ULMFiT[15] and transformer-based models such as BERT [9] have been shifting the paradigm to transfer learning methods. These models have enabled efficient contextual representations of the sentence that overcame the limitations of Doc2Vec representation.

This chapter proposes to utilize BERT, which achieved state-of-the-art performance on 11 NLP tasks, to classify cybersecurity-specific text documents and compare them with the experiment that used Doc2Vec.

The main objective of this chapter is to assess and evaluate the suitability of different language models to act as a Natural Language Filter in the proposed system.

The specific contributions of this chapter are as follows:

1. With the cybersecurity-specific training data and custom preprocessing, the Doc2Vec model has been trained to work as a domain-specific language filter for the proposed system.
2. With the same data, experiments have been conducted to test if the BERT language model can act as a domain-specific language filter.

The remainder of this chapter is organized as follows. In Section 2.2, the overview of the proposed solutions are discussed. In Section 2.3, the datasets to be used as training and the test for the experiments will be discussed. In Sections 2.4 and 2.5, the experimental setup and the results of experiments conducted using Doc2Vec and BERT-based Natural Language Filters respectively will be detailed. Section 2.6 will discuss the differences between both models analyze the evaluation results and conclude with Section 2.7.

2.2 Proposed Module

Chapter 1 proposed a model of the system that utilizes NLP techniques in CTI. This chapter proposes to experiment and compare the two commonly used language models as a Natural Language Filter module that classifies and filters the cybersecurity-related text documents.

2.2.1 Doc2Vec Based Natural Language Filter

Mikolov et al. [26] proposed a paragraph vector, an unsupervised framework that learns continuous distributed vector representations for pieces of texts by extending their previous work on Word2Vec into learning word embeddings. One of the advantages of the Doc2Vec framework is that it can work on variety of different-length texts without losing their order or semantics when representing the documents in continuous vector space. Previous neural network approaches for word embedding consisted of concatenating the several preceding word vectors to form the input of a neural network and tries to predict the next word in sequence. The outcome is that after the model is trained, the word vectors are mapped into a vector space so that semantically similar words have similar vector representations, thus located near to each other [26]. As a result, by performing simple algebraic operations on word vectors, we can easily estimate the word similarity. For example, to find the word that is similar to *small* in the same sense as *biggest* is similar to *big*, we can compute vector *X* = *vector("biggest") - vector("big") + vector("small")* and search in the vector space for the word closest to X measured by its cosine distance [37].

In case of Doc2Vec, the model creates another vector to represent the document itself and combine it with the individual word vectors of the document. Depending upon the working mode, whether it is a distributed memory model of paragraph vector (PV-DM) or a distributed bag-of-words paragraph vector (PV-DBOW), the word ordering is preserved. The PV-DBOW mode ignores the context words in the input, but forces the model to predict words that are randomly sampled from the paragraph in the output, whereas the PV-DM mode concatenates the paragraph vector and the corresponding word vectors to predict the next word in the sequence.

The authors of the Word2Vec model have introduced another two important concepts called subsampling and negative sampling in their subsequent paper [39] in order to improve the performance of the model. Subsampling is the concept in which if the given word under consideration is too frequent in the corpus, higher the probability of it will not be represented in the vector space. This helps to remove the

high frequency words that have minimal influence on the overall performance of the model, thus reducing the model size and improving the accuracy of the learned representations. Negative sampling is the alternative to the output function “hierarchical softmax" in which every training will not adjust all of the neuron weights of the neural network; instead only a small percentage of the weights will be modified, hence improving the computing performance. Both the concepts have been included in the Doc2Vec model and proven to be efficient extensions to it [39].

Thus, the paragraph vectors generated by Doc2Vec represent the document in vector space and inherit the important properties of the word vectors to identify the similarity between the documents. According to the paper by Jey Han Lau et al. [25], Doc2Vec performs better than the alternative document embedding methods for in-domain model training.

Based on the above, Doc2Vec has been tested for its suitability as a Natural Language Filter in the proposed system by embedding a document and finding semantic similarity with a cybersecurity-specific text. In Section 2.4, the experiment conducted with the Doc2Vec-based Natural Language Filter will be discussed. For the purpose of this research, the Gensim[1] implementation of the Doc2Vec model has been utilized.

2.2.2 BERT-Based Natural Language Filter

The latest trend in the Natural Language Processing field includes utilization of Transformer-based deep neural architecture and Transfer learning. Bidirectional Encoder Representations from Transformers (BERT) is a new language representation model that utilizes these approaches to obtain state-of-the-art results on eleven Natural Language Processing tasks. BERT is designed to learn deep bidirectional representations from unlabeled text by jointly conditioning both the left and right contexts in contrast to the previous attempts of predicting a token in a unidirectional (left-to-right, right-to-left) way [9]. BERT achieves bidirectionality by using a pretraining objective called a “Masked Language Model" (MLM). Before feeding the text sequence to a model, BERT replaces 15% of the words in each training example with a [MASK] token. Then, the task of the model is to predict the original token based on the non-masked tokens. In addition to the MLM task, BERT also employs “Next Sentence Prediction" (NSP) task where the model takes a pair of sentences and then tries to predict whether the second sentence is subsequent to the first. During the training, the model is fed 50% of the inputs that are subsequent sentences while the other 50% of sentences are ordered randomly.

In order to successfully train with MLM and NSP tasks, BERT preprocesses the input text according to following steps.

1. A special [CLS] token is placed at the beginning of the first sentence and an [SEP] token is placed right before the second.

2. Token embeddings, where dense embeddings for each token including [CLS] and [SEP] will be learned.

3. Sentence embeddings indicate which tokens belong to which sentence. This is similar to token embeddings; however, vocabulary size here is limited to two only.

[1] https://radimrehurek.com/gensim/

4. Positional embeddings are borrowed from the original transformer paper [61]. Since the transformer is not recurrent, it needs to learn the sequential information with the help of positional embeddings.

BERT has been pre-trained on BookCorpus (800M words) and English Wikipedia (2,500M words) with the goal of minimizing the combined loss of MLM and NSP tasks. Fine-tuning BERT on a classification task is relatively straightforward, involving simply adding a linear layer on top of the transformer output for the first [CLS] token. Applying BERT to downstream tasks involves fine-tuning on the task specific data. For the purpose of this work, the BERT model has been trained to classify security related text documents from the non-security related data.

2.3 Dataset

To test the suitability of Doc2Vec and BERT for the Natural Language Filter module, a significant amount of data has been collected. The details of the data collected for the test and training purposes are explained in this section.

2.3.1 Data Collection

For the model to be trained to understand specific domain contexts, the training data need to include the domain-specific terms and dialogues that occur in casual conversations as well as official statements, respectively. In the real-world scenario, cyber threat information may exist in both the official format, as in news/bulletins, or the casual format, as in conversations on forums, emails and social networks. Since the model needs to be trained to identify the cybersecurity related text whether it is informal conversation or a formal news statement, I wanted to include as diverse range of data sources as possible to minimize the bias. The cybersecurity-specific data sources are categorized as Formal and Informal based on the nature of the conversations.

The data sources and respective number of documents collected are shown in Table 2.1.

TABLE 2.1: Data sources and number of documents

Data class	**Data source**	**No. of documents**
Informal	Reddit discussions	114,391
Informal	StackExchange discussions	841,311
Informal	Hackernews comments	139,946
Formal	Security news outlet RSS feeds	2,077
Formal	Slashdot news archive	7,751
Formal	National Vulnerability Database	99,382
	Total	1,204,858

Detailed explanations of each data source are as follows:

- **Reddit discussions:** Reddit[2] is popular social news aggregation and discussion website. Thirty-two security related subreddit (Reddit communities) discussions have been manually picked, and each of the topics and corresponding

[2]https://www.reddit.com/

comments have been downloaded as an individual document for the time period since the subreddit was created to December 2, 2018. Even though it is not possible to guarantee the collected subreddits are exhaustive of security related communities at Reddit, I believe that it could represent casual and informal conversations around cybersecurity topics.

- **StackExchange discussions** StackExchange[3] is a network of question and answer websites on various topics. The whole discussions of Security[4], Cryptography[5] and Reverse Engineering[6] communities during the time period since the site was created to December 2, 2018 have been collected to represent the Informal type of data.

- **Hackernews comments** Hackernews[7] is a social news website run by Y Combinator. For this research, I have manually picked comments on the security related news since the creation of the site to January 1, 2017 as Informal data.

- **Security news outlet RSS feeds** RSS feed summary for select cybersecurity news outlets during the period of November 1, 2018 to December 2, 2018. News outlets include DarkReading[8], NakedSecurity[9], SecurityMagazine[10] and ThreatPost[11]. The text data that has been collected from these news outlets represent the Formal type of data.

- **Slashdot news** Slashdot[12] is a social news website that features stories on science, technology and politics. Manually picked security related news from the archive of the Slashdot website during the period of January 1, 2015 to September 1, 2018 represents the Formal type of data.

- **National Vulnerability Database** Common Vulnerability and Exposures (CVE) descriptions listed on the National Vulnerability Database (NVD). The NVD is the U.S. government repository of standards-based vulnerability management data and is known as the central database of all software security vulnerabilities[13]. The CVE descriptions that has been accumulated in the NVD during the period from October 1, 1988 to August 1, 2018 has been utilized as the Formal type of data.

2.3.2 Test Data

Upon completion of the data collection, the collected data have been split into test and training datasets. 10% of the total collected data of 1,204,858 documents have been randomly selected and labeled as security related test data.

Reddit categorizes the most popular discussions on its platform at any given instant as Popular subreddit. The discussions on Popular subreddit include varied content including politics, lifestyle, pop culture and everyday news, which would

[3]https://stackexchange.com/
[4]http://security.stackexchange.com/
[5]http://crypto.stackexchange.com/
[6]http://reverseengineering.stackexchange.com/
[7]https://news.ycombinator.com/
[8]https://www.darkreading.com/
[9]https://nakedsecurity.sophos.com/
[10]https://www.securitymagazine.com/
[11]https://threatpost.com/
[12]https://slashdot.org/
[13]http://nvd.nist.org

be suitable to consider as non-security related data. Hence, all of the discussions and corresponding comments of the popular subreddit have been downloaded as of December 6, 2018 as a separate dataset. A total of 294,786 documents have been collected after the preprocessing and marked as non-security related test data.

The security related (120,486 documents) and non-security related (294,786 documents) datasets have been mixed and total of 415,272 documents are prepared for the test purpose. Once the test data has been moved, the total amount of training data reduces to 1,084,372 documents.

2.4 Experiment Using Doc2Vec Based Natural Language Filter

This section reviews the experiment and the results of training the Doc2Vec-based language model.

2.4.1 Preprocessing

In order to improve the efficiency of the Doc2Vec model, the collected data have been preprocessed through the following steps using the standard nltk[14] library of Python.

- **Tokenization:** Since the dataset contains a lot of programming and configuration samples as well as cybersecurity jargon, standard tokenization on the default word boundary has been inefficient. Hence, a custom tokenizer is created using the regular expression that tokenizes as per the non-alphanumeric character. Since some of the training data contains html tags, they have been removed using the regular expressions at this stage.
- **Normalization:** The aim of the model is to identify the cybersecurity-specific text; hence, the normalizing test and training data into lowercase would help to identify the same words in different contexts with different case settings.
- **Token filtering:** Initially, the common English stop words have been removed. During the initial analysis of the collected data, many instances of machine-generated random strings such as hash values or API keys have been found. Hence, identification and removal of those strings based upon the characteristics of the hashing algorithm is required. Based on the initial analysis, it has been decided the tokens of less than two characters and more than 40 characters would not contribute positively to the training of the model; hence, filtered them out. Also, web URLs, email addresses and numerical digits do not contribute to the semantic representation of the language filter and have been removed.
- **Document filtering:** The manual analysis on the dataset revealed that, once the token filter has been applied, many documents were left with less than seven tokens. Since the number of tokens is less than the sliding window of the model, these documents would not positively affect the model representation. Hence, the documents with less than seven tokens have been removed from the dataset. To avoid duplicates, the md5 hash values of the documents have been computed and compared with the rest of the corpus at the end of the process.

[14] http://www.nltk.org/

The above preprocessing has been applied to both the training and test datasets.

2.4.2 Model Training

The Doc2Vec model has been trained with a dataset of 1,084,372 documents. The Gensim implementation of the Doc2Vec model has various hyperparameters to set during the initial training of the model. These parameters affect the model performance in various ways. Lau et al. [25] empirically studied these parameters and concluded that for the semantic textual similarity task, the best performing parameter settings are as listed in Table 2.2.

TABLE 2.2: Baseline hyperparameters of Doc2Vec model

No.	Parameter name	Setting
1	Vector size	300
2	Epochs	400
3	Mode	DBOW
4	Minimum count	1
5	Window	15
6	Subsampling	10^{-5}
7	Negative sampling	5

The hyperparameter settings described in Table 2.2 have been considered as the baseline settings and the initial model has been trained accordingly as the Baseline Model. In order to identify the best performing settings for the Doc2Vec model, each of the hyperparameter settings has been tuned from the Baseline settings and evaluated respectively.

The explanation of the hyperparameters and the respective changes to the Baseline settings are as follows.

- **Vector size or Size** The number of dimensions of the word vector represented. Each dimension represents the specific word in a different context, and semantically similar words have similar vectors in those spaces. The typical dimensions of the Doc2Vec models are set as 100-300 to efficiently represent the semantics of the word. Since the Baseline setting is specified as 300, a model has been trained and evaluated with the reduced value to see how it affects the model performance.

- **Epochs** Also called the training iterations. Training iterations are important to determine the fit of the model. Since Baseline setting for training iteration is 400, a model has been trained with the reduced epochs to observe the fit of the model.

- **Mode** The Doc2Vec model has two modes to work: DBOW stands for distributed bag-of-words mode and DM stands for distributed memory mode. Since the Baseline setting is DBOW, the DM mode is experimented by training a model to determine the performance difference.

- **Minimum count** The minimum frequency threshold of a word in the whole corpus. A model is trained to observe the performance with a higher minimum count threshold than the Baseline.

- **Window** The sliding window size through which the word vector is selected with the neighboring words. Window=15 means that seven words on each side of the selected word will be considered for the analysis. A model is trained with a window size smaller than the Baseline.
- **Subsampling** Threshold to downsample high frequency words. The Baseline value of the subsampling setting is 10^{-5} and the default value for the Gensim implementation of the model is 10^{-3}. The smaller the value is, the less likely it is that frequent words are kept in the model. Hence, a model has been created with an increased subsampling setting.
- **Negative sample** The number of negative word samples are the words to be affected by every training. Since the Baseline setting of the negative sample is the same as a default Doc2Vec setting, it has not been tuned for the experiment.

Note that the initial and minimum learning rates of the Doc2Vec have not been tuned in this experiment and the default values have been used.

2.4.3 Model Evaluation

In order to evaluate the effectiveness of the model, a similarity test has been performed on the test dataset mentioned in Section 2.3.2 Each of the trained models have been used to test if the given test document is similar to any of the training documents of the model. For each test document the vector representation is computed and compared with the stored 1,084,372 document vectors that the model has learned during the training phase. If the cosine distance of the test document representation in vector space is more than 0.7 to any training document vector of the model, the test document is considered as a positive result; if not, it is filtered out from the result set as negative. Using the positive and negative results, the widely used metrics of Precision and Recall have been computed to better visualize the model's performance. Generally, higher precision means more relevant results are found and higher recall means fewer positive results have been missed. Hence, their harmonic mean F1 Score is computed for the consolidated representation and compared with the traditional classification metric of Accuracy.

TABLE 2.3: Classification result of different Doc2Vec models

Hyperparameter	Setting	Precision	Recall	F1 Score	Accuracy
Baseline	As Table 2.2	0.8543	0.4331	0.5748	0.8141
Vector size	300	0.8543	0.4331	0.5748	0.8141
	200	0.7950	0.5287	**0.6351**	**0.8237**
	100	0.8745	0.4503	0.5945	0.8217
Training epoch	400	0.8543	0.4331	0.5748	0.8141
	300	0.8705	0.4392	0.5838	0.8183
	200	0.8834	0.4517	0.5978	0.8236
	100	0.8711	0.4759	**0.6156**	**0.8275**
Mode	DM	0.8722	0.3499	0.4995	0.7965
Min count	10	0.9609	0.3964	0.5613	0.8202
Window	10	0.9271	0.4191	0.5772	0.8219
Subsampling	10^{-3}	0.9814	0.3809	0.5488	0.8182

The classification results using different models are shown in Table 2.3. From the performance comparison, the following observations could be made:

- Only the Vector size and Training epoch hyperparameters have significantly changed the performance of the Baseline model, both in terms of F1 Score and Accuracy.
- The DBOW mode performs better than the DM mode. The model trained with the DM mode has the poorest performance in terms of Accuracy as well as F1 Score.
- The increased minimum word count setting results in better Accuracy but a poorer F1 Score from the Baseline.
- The change in Window size setting seems to affect the document similarity only slightly, increasing Accuracy but reducing the F1 Score.
- Similarly, the reduction in the subsampling rate has very small effect of reducing the F1 Score but increasing the Accuracy of the Baseline model.

Since the Vector size and Training epoch are the only hyperparameters that affect the performance results positively in terms of both F1 Score and Accuracy, the settings have been further tuned to analyze the performance difference. From Table 2.3, it can be seen the performance of the model improves when the Vector size parameter is reduced from 300 to 200 but declines when it is further lowered to 100. Hence Size=200 is picked as the optimal Vector size parameter. Also, since the reduction in training iterations results in better performance, Epoch=100 is picked as the optimal Epoch size. A new model is created with Size=200 and Epoch=100 settings in addition to the Baseline settings. It has proven to be the best performing model with an F1 Score of 0.63 and Accuracy of 0.83.

In order to identify the best working mode for the Natural Language Filter, additional experiments with different similarity thresholds have been performed. For the purpose of this research, a cosine distance of 0.7 has been arbitrarily chosen as a default similarity threshold level. Hence, additional experiments with multiple levels of cosine distance threshold have been done using the best performing model to determine if the arbitrarily chosen threshold of 0.7 is suitable. The results of the experiments are shown in Fig. 2.1.

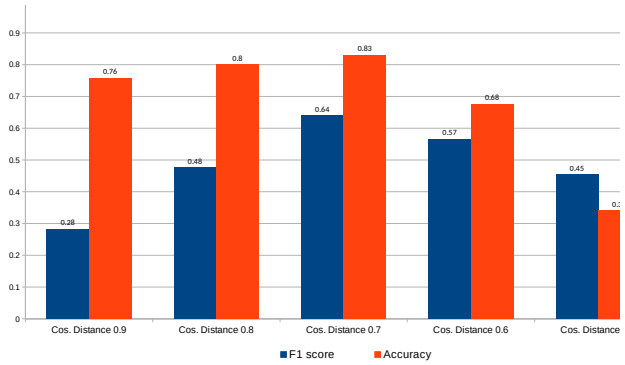

FIGURE 2.1: Performance comparison of Doc2Vec model with different similarity thresholds

From Fig. 2.1, the similarity threshold of 0.7 is observed to be the most suitable threshold for the Doc2Vec model to act as Natural Language Filter.

2.5 Experiment Using BERT-Based Natural Language Filter

The BERT based Natural Language Filter has been trained and evaluated using the subset of the same dataset discussed in Section 2.3. This section will review the results of the experiment of training and evaluating the BERT-based cybersecurity language model.

2.5.1 Dataset and Preprocessing

The advantage of using transfer learning is that it is possible to achieve good performance results with a smaller amount of training data. Jeremy Howard et al. achieved the same training results with only 100 labeled examples in transfer learning as from training from scratch with 100× more data [15]. Hence, to ease the computational burden, the BERT language model has been trained with a reduced dataset. One quarter of the training data used for the Doc2Vec model has been randomly selected from every data source. This would serve as the security related dataset to fine-tune the BERT model. The composition and number of security related training data for the Doc2Vec and BERT models are shown in Table 2.4.

TABLE 2.4: Number of documents used to train the language models

Data source	**Doc2Vec**	**BERT**
Reddit discussions	102,952	25,738
StackExchange discussions	757,180	189,295
Hackernews comments	125,951	31,488
Security news outlet RSS feeds	1,869	467
Slashdot news archive	6,976	1,744
National Vulnerability Database	89,444	22,361
Total	1,084,372	271,093

Also, in the case of the Doc2Vec model the training has been conducted using only security related text data. In order to achieve a better result with BERT, a balanced dataset of security related and non-security related texts are required. Therefore, all of the discussions and corresponding comments on the Reddit's Popular subreddit have been downloaded as of September 4, 2019 to compensate for the non-security related training dataset. From the downloaded Reddit discussions, the same number of documents as security related dataset has been randomly chosen and considered as the non-security related dataset. Finally, both the datasets have been mixed and total training data becomes 542,186 documents.

The BERT model requires the text to be tokenized on the sentence level per line of the text; hence, the spaCy[15] sentence tokenizer has been utilized. As an additional preprocessing step, HTML tags and URLs have been removed and no other preprocessing has been conducted.

For the evaluation of the model, the same test dataset mentioned in Section 2.3.2 has been utilized, following the above preprocessing.

[15]https://spacy.io/

2.5.2 Model Training

BERT-Base model has 110 million parameters, therefore similar hyperparameter optimization as Doc2Vec is not being studied as it becomes another research problem due to its enormous computational cost. Hence the fine-tuning procedure largely followed the hyperparameter settings[16] used in BERT for the text classification task as the original authors recommended. In order to classify security related documents efficiently, BERT-Base Uncased model has been utilized since it lowercases the text before tokenizing. The same WordPiece tokenizer with a vocabulary size of 30,000 as in the original BERT implementation has been used. The benefit of using the WordPiece tokenizer is that instead of naturally splitting English words, they can be divided into smaller sub-word units (e.g., the word “lovely" can be divided into “love" + “ly"). It is more effective for handling unknown words which are to be expected in this case. BERT-Base has 12 layers and each of them produces a sequence of hidden states. Since the task is classification, this sequence needs to be reduced to a single vector in order to represent the whole text. There are multiple ways of reducing the sequence of hidden vectors into a single vector, such as using CNN pooling techniques (e.g., max or mean pooling) or simply applying attention to it. However, it has been decided to go with the simple but effective method of taking the hidden state of the last layer that corresponds to the [CLS] token which happen to be the first in the sequence.

BERT truncates its inputs to a maximum length of 512 tokens; however, the dataset contains many examples with much longer sequences. To handle sequences longer than 512 tokens, a sliding window is used with a sequence length of 256 across the input and take the mean of each representation from the windows. Fine-tuning BERT was performed on 4× NVIDIA Tesla V100 GPUs and was trained with the following hyperparameters as shown in Table 2.5.

TABLE 2.5: BERT training hyperparameters

No.	Parameter name	Setting
1	Batch size	32
2	Max seq length	256
3	Epochs	1, 2, 3
4	Embedding dropout	0.1
5	Attention dropout	0.1
6	Residual dropout	0.1
7	Optimizer	Adam
8	Learning rate	6.25e-5

2.5.3 Model Evaluation

I followed the official guide [9] for fine-tuning three epochs on the target dataset; however, I found that the reduced epoch yields better results. The results of the evaluation are shown in Table 2.6.

To highlight the improvement from the Doc2Vec language model, the results of the best performing experiments for both the language models are shown in Table 2.7.

[16] https://github.com/google-research/bert

TABLE 2.6: BERT Classification result on different epochs

Epoch	Precision	Recall	F1 Score	Accuracy
3	0.91	0.87	0.87	0.87
2	0.91	0.88	0.89	0.88
1	0.92	0.90	0.91	0.90

TABLE 2.7: Performance comparison of both the language models

Model	Precision	Recall	F1 Score	Accuracy
Doc2Vec	0.83	0.51	0.63	0.83
BERT	0.92	0.90	0.91	0.90

2.6 Discussion

As shown in Table 2.7, the BERT-based Natural Language Filter performs better than the Doc2Vec-based Natural Language Filter on the same evaluation dataset, even though it was trained with 4× less security related data. I believe the reasons include the following:

1. Since BERT encodes text in contextual representation, the semantic meanings are better preserved than in Doc2Vec. For example, the word “bank" could mean a financial institution as well as geographical terrain adjacent to the river (as in river bank). In Doc2Vec, the vector representations of both meanings would have same vector, whereas BERT would represent it differently, depending upon the context.

2. The Doc2Vec-based model has been trained only on security related data, whereas BERT was pre-trained with generic knowledge (Wikipedia and books), on top of which the security related data has been fine-tuned. Hence, BERT contains better language representation and benefits the text classification.

3. Training the Doc2Vec model used an unsupervised approach without specifying any label, whereas the BERT model has been fine-tuned using labeled and balanced dataset from which the model could learn better representation.

4. As mentioned in Section 2.5.2 the BERT model splits the natural English words into sub-words using WordPiece tokenizer. This approach lets BERT avoid the Out Of Vocabulary problem by substituting any new word with combination of the sub-words. For example, the input text “John Johanson’s house" would be tokenized as “john johan ##son ’ s house"[17]. In comparison the Doc2Vec model stores internally all the unique tokens and their respective embeddings as dictionary. The Doc2Vec model trained for this research contains vocabulary size of 530,934 unique tokens which might contain noise that has been found in the training document.

5. The training objective of Doc2Vec model is to better represent text document in vector space such that semantic similarities of documents are found by cosine distance of the vectors, whereas BERT fine-tunes it’s all layer with classification objective.

[17] https://github.com/google-research/bert#tokenization

In order to overcome the limitation mentioned in Point 5, additional experiment has been conducted to classify cybersecurity related documents using Doc2Vec and logistic regression model. The training objective of this experiments is classification by introducing logistic regression on top of Doc2Vec model which generates vector embedding features for each sample in training data. Therefore, it is similar to BERT classification process which classifies its internal vector representations based on their feature. For fair comparison with BERT based Natural Language Filter, I used same, balanced dataset mentioned in Section 2.5.1 to train logistic regression model. The Doc2Vec model has been initialized according to the best performing setting of Doc2Vec based Natural Language Filter. The experiment results are shown in Table 2.8 in comparison with Doc2Vec similarity-based classification and BERT.

TABLE 2.8: Performance comparison with Logistic Regression model

Model	Precision	Recall	F1 Score	Accuracy
Doc2Vec	0.83	0.51	0.63	0.83
BERT	0.92	0.90	0.91	0.90
Doc2Vec+LogReg	**0.29**	**0.49**	**0.36**	**0.50**

As seen in the Table 2.8, the Doc2Vec+LogReg performance is poor with Accuracy of 50% only. Since the test dataset consisted of imbalanced mix of security and non-security related texts, it shows slightly better performance than random guess. A possible explanation of such poor performance is that embeddings generated by BERT represents text better in vector space than embeddings generated by Doc2Vec. In order to illustrate this visually, I arbitrarily chose 6 example texts and created heatmap for their cosine similarity. Each cell in the heatmap represents cosine similarity of the corresponding texts and higher the cosine similarity denser the cell colors. The examples include security and non-security related two texts each and also two texts that contain security related words but in non-security related context to emphasize the contextual representation. The heatmap could be seen from Fig. 2.2 and Fig. 2.3 in which the semantic similarity of Doc2Vec and BERT embeddings are illustrated. From the heatmaps it could be seen that BERT embeddings are better representing the semantic similarities of example texts as compared to embeddings generated by Doc2Vec model.

In addition to the model differences mentioned, the technical differences between the experiments conducted using the Doc2Vec and BERT language models are shown in Table 2.9.

TABLE 2.9: Experiment comparisons

Model	Training data	Test data	Resource	Time
Doc2Vec	security related 1,084,372 doc	415,272 mixed doc	8× Intel Xeon E5-2650 CPU	16-18 hours
BERT	balanced set of 542,186 doc	415,272 mixed doc	4× NVIDIA Tesla V100 GPU	6-18 hours

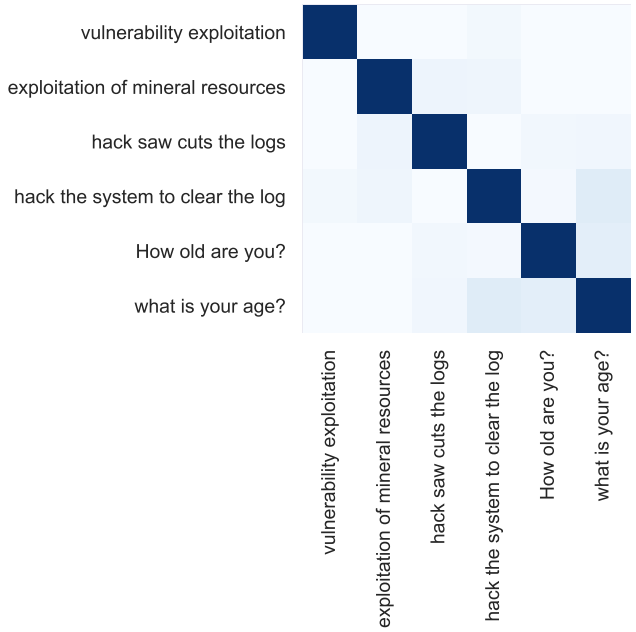

FIGURE 2.2: Semantic similarity visualization of Doc2Vec embeddings

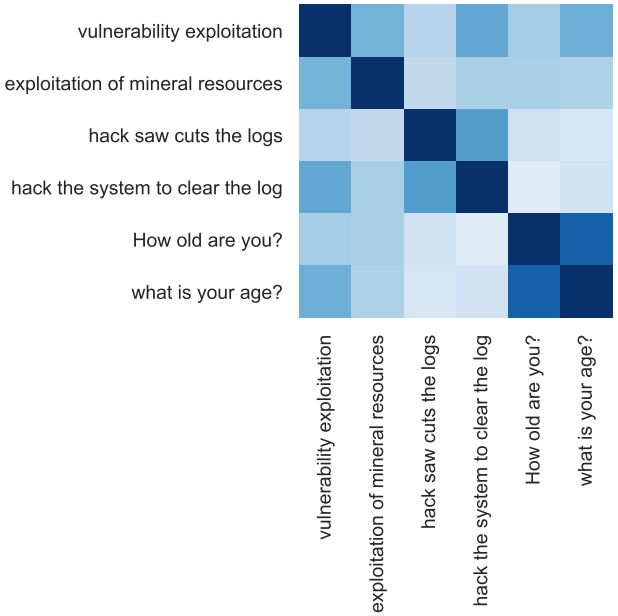

FIGURE 2.3: Semantic similarity visualization of BERT embeddings

2.7 Conclusion

In this chapter I tried to demonstrate the effectiveness of the NLP techniques such as text classification and embedding for cyber defense. For that, two language models have been experimented and evaluated by training with cybersecurity specific training data to deploy as Natural Language Filter to identify cybersecurity specific documents. According to the results of the evaluation, the BERT-based language model outperforms the Doc2Vec-based language model as the Natural Language Filter module for the proposed system. However, there would be considerations regarding computing resources when choosing the right language model. Doc2Vec is a simple and lightweight language model that can be trained with off-the-shelf hardware within a reasonable amount of time, whereas BERT requires higher computing resources to train. However, if the model is used for the inference only tasks, the BERT language model would be suitable, since the costly training will be conducted only once.

Overall, I believe by fine-tuning a pre-trained language model, cybersecurity-related information can be extracted from any unstructured text. This could help to improve situational awareness and assist the security operations. The next chapter will discuss how this extracted cybersecurity information could be turned into cyber threat information by classifying based on its significance and relevance to the user.

Chapter 3

Analyzing the Significance and Relevance of Cybersecurity Text

3.1 Introduction

One approach to mitigate cyber risk is the sharing of threat information via platforms such as the closed and open information-sharing communities as well as the threat feed generating vendors. The idea of sharing threat information stems from the assumption that an adversary that attacks a certain target is also likely to attack similar targets in the near future. While information-sharing platforms have grown in popularity, the amount of shared threat information has grown tremendously, overwhelming human analysts and undermining the efforts to share threat information. In order to identify the significance of the shared information and relevance to their organizations, the analysts have to process considerable amounts of information and separate the actionable threat information from the noise.

Even though there are approaches that automatically share information between machines through structured information sharing such as Structured Threat Information Expression (STIX)[1] and its corresponding protocol Trusted Automated Exchange of Intelligence Information (TAXII), the need to process unstructured text reports that might be shared via email or forums still exists. For example, dark-web forums provide valuable threat information, if the noise can be segregated, with less effort. Also, to establish situational awareness, a security analyst has to be able to identify cyber threat-related information specifically applicable to his environment to proactively monitor and prevent the possible intrusion and control the possible risk. For this reason, this chapter is proposing a module of the system that employs Natural Language Processing techniques to identify the cyber threat-related information specific to the user and filter out the irrelevant content.

Chapter 1 proposed the overall architecture of the proposed system to identify user-specific threat information from publicly available information sources. As a follow-up, this chapter proposes a novel approach to identify the user-specific content from those filtered texts by focusing on to quantify the significance and relevance of the threat information contained in unstructured text by comparing the vector representation of the text with known important text and identifying the cybersecurity entities using a Named Entity Recognizer and by correlating it with an existing Cybersecurity Knowledge Graph (CKG). I have considered the textual similarity of the text and the correlation of the mentioned entities with the CKG as features of the threat information and fed those features through a classifier to generate a score that quantified the significance and relevance of the text.

[1] https://oasis-open.github.io/cti-documentation/

According to Harter, the information could have either objective or subjective relevance to the particular situation [14]. The objective relevance measures how well the topic of the information matches the domain, and subjective relevance deals with user-specific situations. In chapter 2 I attempted to identify the text documents that have objective relevance within the domain of cybersecurity. The goal of this chapter is to seek a way to quantify the subjective relevance of the text documents alongside with its potential significance, which can be customized to meet user-specific needs by utilizing existing Natural Language Processing (NLP) techniques and tools.

Identifying the subjective relevance of entities and concepts is a well-studied field of Information Retrieval (IR), where the search engines provide web-page rankings based on the relevance to the user [54]. However, to the best of my knowledge correlating the extracted entity with an existing knowledge base to determine the subjective relevance has not been attempted in the field of cybersecurity.

The specific contributions of this chapter are as follows:

1. Proposal of a novel approach to analyze text documents to identify the significance and relevance of the text

2. Design for an experiment to prove the viability of this method

The remainder of this chapter is organized as follows. Section 3.2 will briefly discuss the conceptual design of the Analyzer module. In Section 3.3 the implementation of the proposed Analyzer module will be discussed and in Section 3.4 the corresponding experiment to evaluate its viability will be discussed and finally will conclude with Section 3.5.

3.2 Proposed Module

The purpose of the Natural Language Filter module is to identify cybersecurity-related text documents from publicly available information source for further analysis, whereas the purpose of the Analyzer module is to determine the significance and relevance of the text document to the user in order to reduce the workload of the human operators by filtering out information that is insignificant or non-relevant to the user. I believe that a text document's significance and relevance could be determined by identifying textual similarities with pre-defined significant texts and the correlation between the cybersecurity entities mentioned in the text and the user specified entities of interest as shown in Fig. 3.1. These features from the text documents could be used to generate a unique number that could represent the significance and relevance of the text document. With this quantitative score users can calibrate the sensitivity of how much relevant and significant information they want to filter, thus providing a room for optimization.

The Analyzer module could be implemented with the following components as depicted in Fig. 3.1.

1. Similarity Analyzer

2. Cybersecurity Knowledge Graph Analyzer

3. Significance Score Calculator

Each component is discussed in the subsequent section.

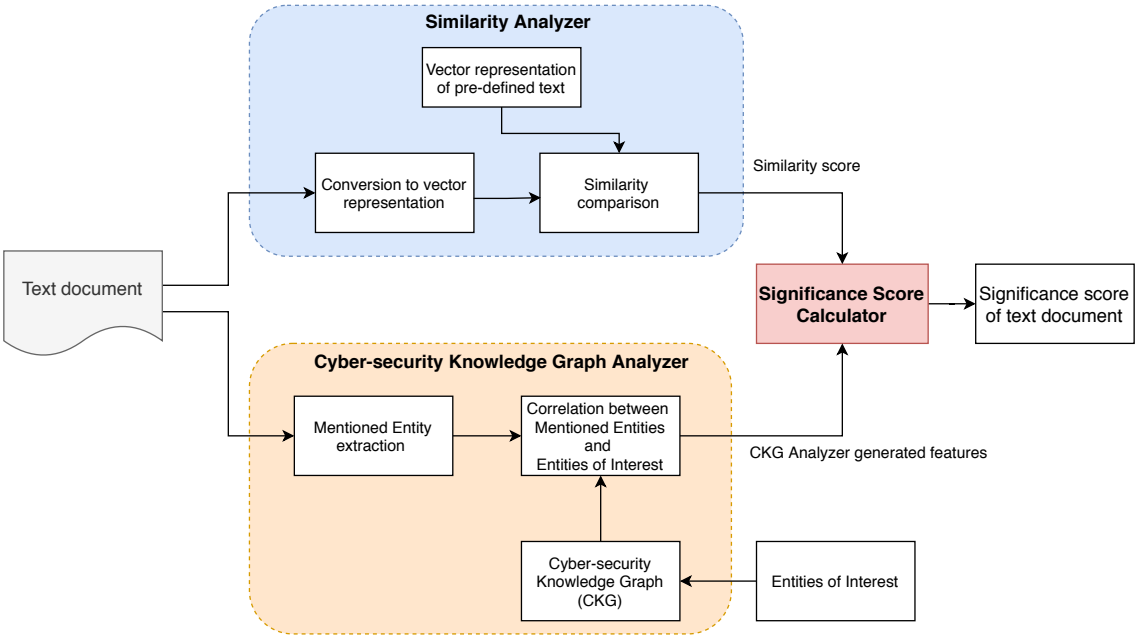

FIGURE 3.1: Overview of proposed Analyzer module.

3.2.1 Similarity Analyzer

The semantic analysis of the text document refers to extracting the lexical meaning of a text independent of its written language. Since computers can work only with numbers, computational linguistics achieves semantic analysis by representing text in vector space and assigning different meanings of the text in different dimensions of the vector. For example, the word "bank" could mean a financial institution as well as geographical terrain adjacent to a river (as in river bank). When the word "bank" is represented in vector space, each meaning would be represented by different components of the same vector, depending upon the context. Once the text is represented in vector space, one way of performing the semantic analysis on the text document is to compare its vector representation with another vector. Comparing the vector representations of different texts is called textual similarity and the distance between the vectors represent the closeness of their semantic meanings.

Kenter et al. demonstrated the effectiveness of computing textual similarity through vector embeddings of short text in [24] and I believe the textual similarity could be used to define the significance of the text by comparing vector representations of the given text with a pre-defined "significant" text.

3.2.2 Cybersecurity Knowledge Graph Analyzer

The Analyzer module proposes to utilize existing knowledge sources to identify the subjective relevance of a text document to the user. A similar approach to utilize external knowledge source to analyze the textual features has been proposed by Nguyen et al. in [43]. They proposed to identify a short-text semantic similarity through word embeddings and external knowledge sources. Their approach of determining the degree of semantic similarity between pairs of a short text by exploiting the semantic relatedness between concepts based on an external source of knowledge and word embeddings has outperformed state-of-the-art systems in short text semantic similarity task on three different datasets. Similarly, I believe with the right

setting, the subjective relevance of the text could be inferred with high confidence using an existing knowledge graph.

In a broad sense, a Cybersecurity Knowledge Graph is a graph representation of a semantic triple that comprises of a pair of cybersecurity entities and the relationship between them. For the purpose of this work, a CKG is used to determine if the given text document has any relevance to the user. To do that, two types of entities, namely Entities of Interest and Mentioned Entities are defined. The Entities of Interest are the user-specific terms that indicate any hardware/software vendors or product names as well as Common Vulnerabilities and Exposures (CVE) ID. The Mentioned Entities are the entities that have been extracted from the given text document through the Named Entity Recognizer.

In [36] I tried to infer the relevance of the text through the number of subjective named entities mentioned in the text document. However, it has been concluded that was not a good approach since the Named Entity Recognizer could not account for the semantically related terms (e.g., "desktop" could mean "computer" depending on the context. But the Named Entity Recognizer would not be able to infer this relationship) unless specifically trained on them. Therefore, to overcome this drawback Cybersecurity Knowledge Graph is being deployed to infer the correlation between the Entities of Interest and Mentioned Entities.

3.2.3 Significance Score Calculator

The Significance Score Calculator (SSC) is a function that outputs a fixed range of numbers based on the given inputs. The inputs consist of the following items.

- Scores with closest similarity to the pre-defined significant text repositories
- Features generated by the correlations between Entities of Interest and Mentioned Entities

These inputs would serve as features to be extracted from the threat-information documents to classify whether the document is significant or not. Ideally, SSC would be a classifier that produces a quantitative number which represents the probability of specific item belonging to a significant class.

3.3 Implementation

To verify the viability of the proposed Analyzer module, I have implemented the proposed components using common open-source libraries. The details of the implementation and experimental environment are discussed in subsequent sections.

3.3.1 Data used

As discussed in Section 2.3.1 a significant amount of cybersecurity-related text data has been collected from various sources to implement Natural Language Filter module. For the purpose of this research, part of it has been re-utilized with additional data source. The data sources include:

- **MalwareTextDB:** Phandi et al. proposed a shared task to classify relevant sentences, predict token labels and relation labels and attribute labels for malware-related text at the International Workshop on Semantic Evaluation 2018 [50]. In the task proposal, they had compiled the largest publicly available dataset of

annotated malware reports, which is called MalwareTextDB and consists of 85 Advanced Persistent Threat (APT) reports that contain 12,918 annotated sentences. For the purpose of this work, this data source will be called MWTDB for short.

- **CVE repository:** Same as in Section 2.3.1. For the purpose of this work, this data source will be called CVE for short.
- **StackExchange discussions:** Same as in Section 2.3.1. For the purpose of this work, this data source will be called SE for short.
- **Security news outlet RSS feeds:** Same as in Section 2.3.1. For the purpose of this work, this data source will be called RSS for short.

From each of the data sources 1,100 text documents have been randomly selected to be utilized for following purposes.

- **Reference text:** 100 documents from each source would act as pre-defined "significant" text to be used in the Similarity Analyzer.
- **Test data:** 1,000 documents from each source makes a total of 4,000 text documents for the evaluation of the overall system.

3.3.2 Similarity Analyzer

In order to perform semantic analysis through textual similarity, the given text is converted into numerical vectors, also known as embeddings. Conventionally, vector embeddings were achieved through shallow algorithms such as Bag of Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF). These approaches have been superseded by predictive representation models such as Word2Vec [38], GloVe [47] etc. Since the utilization of deep neural networks has been proven to be superior in different fields, various studies have adopted deep neural models to embed the text into vector space, such as Facebook's InferSent[2] and Universal Sentence Encoder (USE) from Google Research. Perone et al. evaluated different sentence embeddings and Universal Sentence Encoder outperformed InferSent in terms of semantic relatedness and textual similarity tasks [48]. Therefore, for the purpose of this research, Universal Sentence Encoder has been utilized to generate the vector embeddings of the text.

Universal Sentence Encoder (USE)

In the paper by Cer et al., transformer-based and deep averaging network (DAN)-based models for encoding sentences into embedding vectors have been introduced [7]. The USE models take English sentences of variable lengths as input and produce 512 fixed-dimensional vector representations of the sentences as output. Both the models are pre-trained using Wikipedia, web news, web question-answer pages and discussion forums[3].

Since the sentence embeddings from USE produce good task performance with little task-specific training data, a DAN-based sentence encoder has been employed for this research in order to find textual similarity between the texts in vector space, thereby performing a semantic analysis. The DAN-based sentence encoder model

[2]https://github.com/facebookresearch/InferSent
[3]https://tfhub.dev/google/universal-sentence-encoder/2

makes use of a deep averaging network whereby input embeddings for words and bi-grams are first averaged together and then passed through a feedforward deep neural network to produce sentence embeddings with minimal computing resource requirements.

Implementation of Similarity Analyzer

At first, the USE is utilized to generate vector representations of an initial 256 bytes of every entry in the Reference text and store them in repositories named after the data source. (The choice of the initial 256 bytes is heuristic. Even though it is claimed that USE can work on varying lengths of text, it has been observed that better textual similarity is obtained when texts of the same length are compared.) In order to illustrate the relative differences between the Reference text repositories, the maximum textual similarities have been computed. Table 3.1 shows the maximum cosine similarity between the different reference text repositories.

TABLE 3.1: Reference text similarity.

	MWTDB	**CVE**	**SE**	**RSS**
MWTDB	1.0	0.5696	0.5352	0.5730
CVE		1.0	0.5946	0.6523
SE			1.0	0.5459
RSS				1.0

From Table 3.1 it can be seen that the Reference text repositories are distinct enough to represent different types of cybersecurity text. Since the Significance Score Calculator is a linear classifier, it will represent the bias of the training data in the classification output. Hence, to reflect the significance of the different types of cybersecurity documents, weights have been assigned to the repositories for the composition of the training data. I believe the actions and capabilities of the malware are of utmost importance to the human analysts; therefore, if the given text is semantically similar to pre-defined texts describing malware actions and capabilities as included in the MalwareTextDB, the significance of that text should be considered the highest. The second highest significance is given to a CVE description wherein the details of software vulnerabilities are discussed. The informal discussions that happen around a question-answer system such as StackExchange may be useful when attempting to understand the situation, thus this is third in line. Official news reports have been assigned the lowest significance, based on the assumption that if something is already on a public news outlet, its importance is likely to be outdated. These priorities have been reflected in the test data, as will be discussed in Section 3.4.2.

Similar to the Reference text, 256 characters from the start of the input text have been extracted and converted into a vector representation by the USE. Consequently, cosine similarity is computed with each entry in the Reference text and the highest similarity score for each repository is considered as the output of the Similarity Analyzer. The process is illustrated in Fig. 3.2.

As a result of the process depicted in Fig. 3.2, the Similarity Analyzer feeds following features to the Significance Score Calculator.

1. Maximum similarity score with MWTDB repository
2. Maximum similarity score with CVE repository

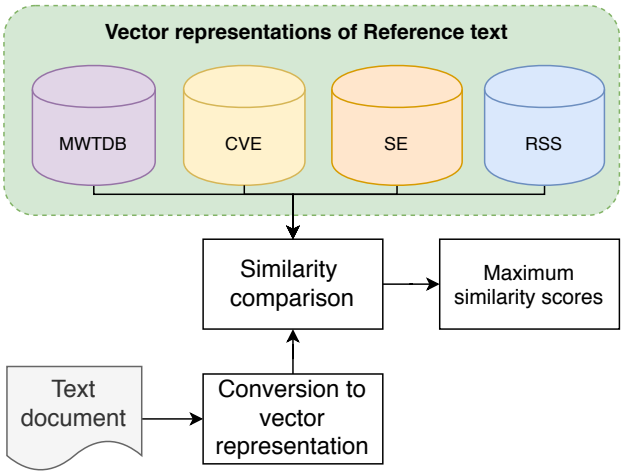

FIGURE 3.2: Similarity analysis process.

3. Maximum similarity score with SE repository
4. Maximum similarity score with RSS repository

3.3.3 Cybersecurity Knowledge Graph Analyzer

Whether a given text document has any subjective relevance to the user has been analyzed using a Cybersecurity Knowledge Graph Analyzer. It uses the custom-trained NER model as a Named Entity Recognizer to extract the Mentioned Entities from a text and the CKG to infer the relationship with the Entities of Interest. The details of these components are discussed in subsequent sections.

Cybersecurity Knowledge Graph

In order to construct a CKG, the NVD repository of CVE descriptions as of March 04, 2020 has been utilized. From each CVE description, a unique semantic tuple is derived that contains the subject, object and their relationship. An arbitrary number as a cost for each type of tuple has been assigned to denote the uniqueness of the relation. The cost has been also used to compute the shortest distance between the nodes. The tuple types and their corresponding costs are shown in Table 3.2.

TABLE 3.2: Semantic tuples derived from NVD.

Subject	Relation	Object	Cost
Vendor	has product	Product	15
Product	product of	Vendor	16
Product	is vulnerable	CVE ID	9
CVE ID	vulnerability of	Product	10

In total 221,202 tuples have been extracted from the NVD repository. The subjects and objects of those tuples are represented as nodes for Vendor, Product, and CVE ID in the graph and the relationship between them serves as the edges of the graph. The total number of unique nodes derived from the semantic tuples is shown in Table 3.3.

TABLE 3.3: Unique nodes in the graph.

Node type	No. of nodes
Vendor	20,190
Product	36,626
CVE ID	115,551
Total	**172,367**

For example, the relationships: {(**Vendor:** Microsoft) (has product) (**Product:** Windows 7)}, and {(**Product:** Windows 7) (is vulnerable) (**CVE ID:** CVE-2015-2519)} is represented in the graph as shown in Fig. 3.3.

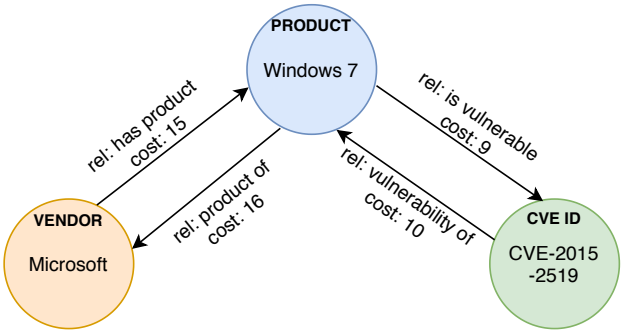

FIGURE 3.3: Cybersecurity Knowledge Graph example.

Named Entity Recognizer

The Cybersecurity Knowledge Graph Analyzer would utilize a Named Entity Recognizer to identify the Mentioned Entities in the text. Hence, it becomes necessary to train the custom NER model to act as a Named Entity Recognizer that can identify the entities of CKG. Fortunately, as pointed out by [6], the structured text data of the NVD repository provides an easier way to automatically label and annotate the large corpus for custom training the NER model. Hence, the CVE descriptions in the NVD repository have been annotated with the vendor and product information. Since it is extremely difficult to distinguish between an IT product and IT vendor from unstructured text (e.g., the entity "debian" could mean the Debian project i.e., **the Vendor** or Debian Linux **the Product** depending upon the context), both the product and vendor have been defined under the same label, **Vendor**. In addition, the structured naming convention of CVE ID (e.g., CVE-YYYY-NNNN) provides an easily identifiable pattern to define a label **CVE ID** from an unstructured CVE description.

For the Named Entity Recognizer the Stanford NER model, also known as CRFClassifier has been utilized. The Stanford NER model is a general implementation of linear-chain Conditional Random Field (CRF) sequence models that is customizable to user-specific data [12]. Even though there are models that utilize deep neural architecture and demonstrate better performance, the Stanford NER model is efficient enough, as an off-the-shelf model that can be easily retrained. Hence, the Stanford NER Model has been trained for the annotated CVE descriptions consisting of 17,113,939 words and tested on 4,658,262 words. The performance of the custom-trained NER model per label is shown in Table 3.4.

TABLE 3.4: Custom trained NER model performance.

Label	Prec	Rec	F1 score
Vendor	0.7473	0.7761	0.7615
CVE ID	1.0	1.0	1.0

From Table 3.4 it could be seen that the Named Entity Recognizer is able to distinguish CVE IDs without any error since it could be identified by a clear pattern.

Implementation of Cybersecurity Knowledge Graph Analyzer

Rada et al. defined a metric of distance in the knowledge base represented as a graph in order to measure the relatedness of the two nodes of a graph [53]. I propose to utilize similar measures such as the number of nodes, number of edges and total cost to define the relatedness or subjective relevance of the cybersecurity entities.

In order to know if the given text has any relevance to the user, the Entities of Interest (EI) have been defined as cybersecurity entities that concern the user. Entities extracted from the given text are marked as Mentioned Entities (ME) and looked up in the CKG to compute the shortest path with every Entity of Interest. The CKG analyzer follows the process outlined below:

1. Define EIs in the CKG.
2. Extract MEs of label **Vendor** and **CVE ID** from input text using Named Entity Recognizer (As mentioned earlier both the Product and Vendor entities are identified under same label as **Vendor**).
3. For every extracted ME, compute the shortest path with every EI using Dijkstra's shortest path algorithm.
4. Consider number of nodes, number of edges and total cost as output between every ME and EI as per the shortest path.
5. If the ME does not exist in CKG or there is no path that connects ME to a specific EI, then assign -1 for output.
6. For every input text, compute the average number of nodes, edges and average cost for each ME with the label **Vendor** and **CVE ID**.

For example, if the user is interested in the node "Debian Linux" and input text contains an entity "Windows 7", then the features of the text are computed as follows. The shortest path between the node "Debian Linux" and the node "Windows 7" according to Dijkstra's algorithm is through nodes "CVE-2018-15473" -> "Enterprise Linux Server" -> "CVE-2019-1125" as shown in Fig. 3.4.

This means Debian Linux was affected by OpenSSH user-enumeration vulnerability CVE-2018-15473 as was the Red Hat Enterprise Linux Server, which also shares Windows Kernel Information Disclosure vulnerability CVE-2019-1125 with Windows 7 OS. This correlation could be represented as the **Vendor** label includes the features of 3 nodes, 4 edges and a total cost of 38. Also, since the text does not contain any **CVE ID** labeled entity, the corresponding features would be all -1. As a result, this input text would generate features as shown in Table 3.5.

These features are fed into the Significance Score Calculator along with its similarity features to generate the significance score of the text.

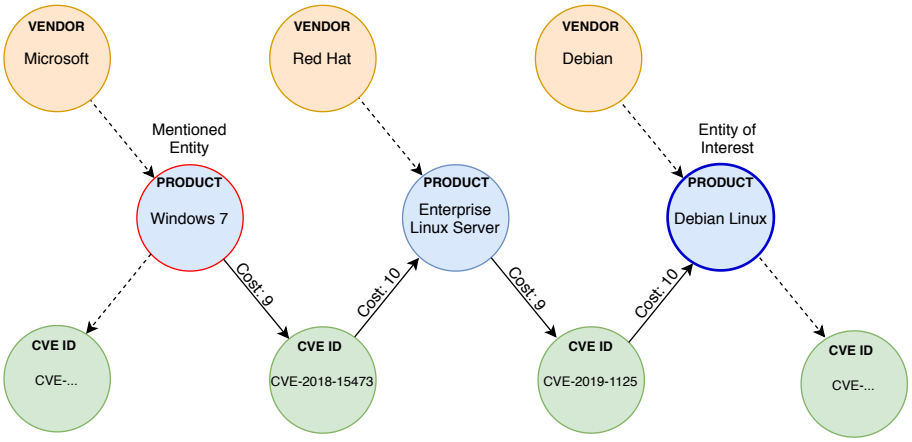

FIGURE 3.4: Cybersecurity Knowledge Graph example: The correlation between node "Debian Linux" and "Windows 7".

TABLE 3.5: Features generated by CKG Analyzer.

Vendor			CVE ID		
Nodes	**Edges**	**Cost**	**Nodes**	**Edges**	**Cost**
3	4	38	-1	-1	-1

3.3.4 Significance Score Calculator

Since the objective of the system is to generate quantitative numbers that represent the significance and relevance of the text, it has been decided to utilize a classifier that could generate numeric output showing the probability of particular element belonging to the significant class. Hence, implementations of the various classification algorithms have been tested using a popular open-source library sklearn[4].

According to the evaluation, the following classifiers could be utilized as SSC since they could generate probability of items belonging to either class:

- Support Vector Machines (SVM)
- K-Nearest Neighbors (KNN)
- Decision Tree Classifiers (Dec.Tree)
- Gaussian Naive Bayes (GNB)
- Logistic Regression (LogReg)
- Multi-layer Perceptron neural model (MLP)

Even though the purpose of the SSC is to generate significance score in a form of probability that particular text is significant and relevant to the user, to evaluate the viability of the proposed method, an experiment has been conducted to generate the performance indicators, such as Accuracy and F1 score. Each output of the Similarity Analyzer and CKG Analyzer is inputted into the Significance Score Calculator to generate classification result that can be used to evaluate the overall performance of the system.

[4]https://scikit-learn.org/stable/

3.4 Experiment and Evaluation

The implementation of the system has been tested by experimenting in different settings. In this section, the experiment and the corresponding evaluation results will be discussed.

3.4.1 Preliminary test

In order to test the functionality of the proposed system, a preliminary test has been conducted on the test data. The randomly selected 4,000 text documents mentioned in Section 3.3.1 have been split into a balanced set of positive and negative datasets. In addition, the Entities of Interest have to be defined in order to generate correlation features from the CKG. As an arbitrary choice, the highest degree 10 nodes of the CKG are defined as Entities of Interest. Table 3.6 lists the nodes chosen as Entities of Interest and the corresponding number of edges connected to them as degree.

TABLE 3.6: Highest degree nodes in CKG.

Node	**Degree**
Debian Linux	3,686
Android	2,592
Linux Kernel	2,462
Ubuntu Linux	2,334
Mac Os X	2,319
Chrome	1,813
iPhone OS	1,802
Firefox	1,758
HP	1,542
Windows Server 2008	1,506

Once the Entities of Interest are defined, the test data is processed by the Similarity and CKG Analyzers, respectively. The generated features are fed to SVM classifier and 70% of the data is used to train and 30% to test the performance of SVM classifier. The evaluation result is as shown in Table 3.7.

TABLE 3.7: Preliminary test results.

Prec	**Rec**	**F1 score**	**Accuracy**
0.5633	0.2649	0.3603	0.5266

From Table 3.7, it can be seen that the proposed system could not work directly, i.e., without a controlled environment it would not be possible to test the full potential of the proposal. Hence, the experimental design has been modified to consider the constraints and simulate a controlled environment. Thus, experimental setup is discussed in the next section.

3.4.2 Experimental setup

Determining the shortest path of each Mentioned Entity to every Entity of Interest using Dijkstra's shortest path algorithm is a computationally expensive task, hence it has been decided to conduct the experiment on a limited set of data to prove the viability of the concept. To ease the computational burden, 4,000 text documents mentioned in Section 3.3.1 have been selected and processed using off-the-shelf hardware of 8× Intel i7-7700 CPU 3.60GHz with 16GB of memory.

As mentioned in Section 3.3.2, the significance weights of the Reference text repository need to be reflected in the test dataset. Therefore, I have constructed the test dataset by allocating more from higher weight repository to the positive dataset. This way, the SSC would be trained to assign higher significance probability to the input text that is similar to the higher weight repository. The dataset composition is shown in Table 3.8.

TABLE 3.8: Dataset composition.

Dataset	**MWTDB**	**CVE**	**SE**	**RSS**	**Total**
Positive	800	600	400	200	2,000
Negative	200	400	600	800	2,000
Total	**1,000**	**1,000**	**1,000**	**1,000**	**4,000**

Also, as discussed in the previous section, the raw textual data cannot be used to test the full potential of the proposed system. I believe the reasons could include the following:

- The randomly selected test data may not include the cybersecurity named entity that could be identified by the NER model.
- The named entities found in the test data may not exist in the CKG; hence, the CKG analyzer is not able to generate correlation data.

Hence, to overcome these constraints, the following assumptions are made to simulate the controlled environment.

- The actual NER model would have perfect performance, so that it can identify any named entity mentioned in the text.
- Every positive data includes at least one named entity that could be identified by the NER model.
- The actual CKG is much larger in scope and contains every entity that is found in the text. In other words, any entity found in the text would give some correlation with the Entities of Interest.

In order to reflect these assumptions, the positive data has been manually manipulated by inserting an entity at the end of the text. In all, 2,000 unique entities with the costliest relationship with chosen Entities of Interest are selected and manually inserted at the end of each of the positive text data.

3.4.3 Final Evaluation

For the final evaluation, the classification experiment is conducted using different classifiers on the final dataset, as explained in the previous section. Since the objective of the experiment is not the classification, but rather the viability of the method of generating the features of the text that could be used by any classifier, the default parameter settings for all the classifiers have been used. Table 3.9 shows the performance differences of using different classifiers.

TABLE 3.9: Different classifier results.

Classifier type	Prec	Rec	F1 score	Accuracy
SVM	0.9938	0.7706	0.8681	0.8792
KNN	0.8767	0.8158	0.8452	0.8458
Dec.Tree	0.8141	0.8207	0.8174	0.8108
GNB	0.9856	0.7722	0.8659	0.8767
Log.Reg	0.9780	0.7916	0.8750	0.8833
MLP	0.9955	0.7092	0.8283	0.8483

From Table 3.9 it could be seen that features generated by the Similarity Analyzer and CKG Analyzer could be used by any classifier to generate significance score. Since the test dataset is small in size, the test result has been validated by conducting 10-fold cross validation. The Logistic Regression classifier shows the highest performance among other classifiers; hence, 10-fold cross validation test has been done using Logistic Regression. The results of the 10-fold cross validation are shown in Table 3.10.

TABLE 3.10: 10-fold cross validation result using Logistic Regression classifier.

Fold	Prec	Rec	F1 score	Accuracy
0	0.9747	0.7857	0.8701	0.8850
1	0.9471	0.7816	0.8564	0.8650
2	0.9808	0.7887	0.8743	0.8900
3	0.9664	0.7461	0.8421	0.8650
4	0.9716	0.8301	0.8953	0.9000
5	0.9770	0.8095	0.8854	0.8900
6	0.9811	0.7393	0.8432	0.8550
7	0.9942	0.8333	0.9067	0.9125
8	0.9615	0.7979	0.8721	0.8900
9	0.9695	0.8281	0.8933	0.9048
Average	**0.9724**	**0.7940**	**0.8739**	**0.8857**

From Table 3.10, it could be seen that the proposed Analyzer module could determine the significance and relevance of cybersecurity text with an average of 88% accuracy under certain assumptions.

3.4.4 Analysis on the evaluation results

Based on the results shown in Table 3.9 and Table 3.10, it can be concluded that the experimental evaluation proves the practical implication of the approach. However, it should be acknowledged that these results came as the outcome of several assumptions and simulated environment. The results of Table 3.8 have shown that the proposed system is not able to function as intended, without the simulated environment.

Also, the poor performance of Named Entity Recognizer influences the final experiment result. Since achieving a state-of-the-art result in domain-specific NER itself is a separate research problem, the off-the-shelf NER model has been utilized that yielded an F1 score of 0.76. I believe by improving the NER performance, not only the experiment result will improve, but also the need to simulate the test data will diminish.

In addition, an analysis has been done to see the contribution of the individual Analyzer modules on the overall classification scheme. The features generated by the CKG Analyzer and Similarity Analyzer are fed into the best performing classifier Logistic Regression separately and classification results are shown in Table 3.11.

TABLE 3.11: Classification results on the individual Analyzers.

Analyzer	Prec	Rec	F1 score	Accuracy
CKG Analyzer	0.9759	0.7851	0.8702	0.8792
Sim. Analyzer	0.6565	0.6236	0.6396	0.6375

From Table 3.11 it could be seen that the features generated by CKG Analyzer produce an almost identical result as Table 3.9, hence majorly contributing to the final system performance, whereas features generated by Similarity Analyzer are less likely to influence the final result. One of the reasons for this outcome is that the CKG Analyzer contributes 6 features and Similarity Analyzer 4 features to the final result. Also, the effect of the Similarity Analyzer in the experiment is through the dataset composition mentioned in Table 3.8. I believe another reason for such asymmetric contributions by the Analyzers is due to the poor distinction between positive and negative examples. Hence such deficiencies should be tackled in the consequent experimental designs.

3.5 Conclusion

As listed in chapter 1, there have been various approaches to utilize NLP techniques in cyber defense. The works discussed in Section 1.3.2 Cybersecurity Knowledge Graph and Section 1.3.3 Cybersecurity Named Entity Recognition are examples of specific use cases of NLP in cybersecurity. The scope of this work was not to achieve the best results in these specific tasks, rather utilize the most optimal implementations of them to prove the viability of the proposed system. The works listed in Section 1.3.1 Automated Threat Detection have some similarities with the proposed autonomous system. But the main difference in this proposal is to narrow down the textual information into the cybersecurity domain and then identify user-specific information by utilizing CKG and NER.

The ultimate goal of the Analyzer module could be seen as a custom version of the text classifier that engineered the textual features to classify the significant

and relevant text to the user. However, the focus is to customize the classification result to the user-specific content whereas the traditional text classification studies mentioned in Section 1.3.4 Text Classification focus on the performance of the classification.

In general, the main aspect of this approach is the attempt to quantify the subjective relevance of the text documents along with its potential significance, which can be customized to meet user-specific needs. To the best of my knowledge, this approach of correlating the extracted entity with an existing knowledge base to determine the subjective relevance has not been attempted in the field of cybersecurity. Given the classification accuracy of 88%, I believe this approach may have a practical application.

By determining the significance and relevance of the text document to the user, this thesis could prove the applicability of the various NLP techniques in the CTI process. The next chapter will discuss how this cyber threat information could be enriched with existing knowledge using other NLP/ML methods.

Chapter 4

Mapping the Vulnerability Information to Adversary Techniques

4.1 Introduction

Software security vulnerability is one of the biggest factors behind the cyber risk challenges of digital age. According to the US National Vulnerability Database (NVD), the total number of reported vulnerability as of June 2020 is 146,000[1] and this number is increasing year by year. In 2019 alone 20,362 vulnerabilities are reported on NVD which is a 17.6% increase from 2018 (17,308) and 44.5% increase from 2017 (14,086) and the trend is likely to be upwards[2].

Given this large number of reported vulnerabilities, tracking individual vulnerabilities is nearly impossible. Hence, there have been various approaches and threat models developed to generalize the threat landscape and to ease the burden of a security analyst. One of the most commonly used approaches is a curated knowledge base called MITRE ATT&CK® that enlists adversary behaviors including their tactics and techniques based on real-world observations. It is a powerful framework commonly used as a threat model in adversary emulation, red and blue teaming, and cyber threat intelligence practices [55]. MITRE ATT&CK generalizes the adversary attack techniques and tactics based on the common weaknesses of the systems without mentioning specific product or vulnerability.

Even though the MITRE ATT&CK proved to be a useful framework, the need to identify the specific threat that individual vulnerability poses in the adversarial landscape still exists. In layperson's terms, MITRE ATT&CK is the playbook of steps that house robber would take in order to rob a house (e.g., find open access) and software security vulnerability is the weaknesses of the house security (e.g., unlocked door or broken window). For the effective defense, the house owner needs to combine this information, the most common approaches that house robbers use and weaknesses of his house, so that he can better understand the situation and prioritize his defenses.

Chapter 1 proposed the overall architecture of the proposed system to identify user-specific threat information from publicly available information sources. As a follow-up, this chapter proposes a method that could be used to enrich the extracted cyber threat information with additional CTI to assist and automate the defensive operation. Specifically, by automatically mapping the software security vulnerability information to the potential adversarial technique and tactics I believe the manual

[1] https://nvd.nist.gov/general/nvd-dashboard
[2] https://www.imperva.com/blog/the-state-of-vulnerabilities-in-2019/

operation could be significantly reduced. Since a specific vulnerability can be used in more than one adversarial technique I believe developing a multi-label classification model that can infer the adversarial techniques to given vulnerability would be suitable. Since every vulnerability has associated textual description, I believe using the features of this text, a classic multi-label classification algorithm could produce a result that could be useful for a practical purpose. Hence, various multi-label classification methods have been experimented to evaluate the performance to automatically map the vector representations of vulnerability description to adversary techniques and tactics as prescribed in the MITRE ATT&CK framework.

The goal of the chapter is to seek the best performing tools and methods to deploy as Mapper module in the proposed system.

The specific contributions of the chapter are as follows:

1. To propose an approach to automate the mapping of vulnerability description to adversarial technique.
2. Explore and experiment with various multi-label classification methods to compare the performance.

The remainder of this chapter is organized as follows. Section 4.2 will review the background information to be used for this research. In Section 4.3, the experiment of the proposed multi-label classification and the corresponding evaluation will be discussed and finally, it will conclude in Section 4.4.

4.2 Proposed Module and Background Information

I believe by enriching identified significant and relevant cyber threat information with the existing knowledge about the related threat, the proposed system would be able to assist the defenders and automate the CTI process. Hence, the Mapper module of the proposed system should input the filtered significant and relevant threat information and output the CTI enriched threat information.

In order to effectively map and enrich the threat information with common reference knowledge of different fields is required. Thus, the theoretical background information is briefly explained in this section.

4.2.1 Vulnerability Modeling

There have been several attempts to standardize the reporting and modeling of software security vulnerabilities or weakness and threat landscape in general. In this section, a few relevant schemes for this study will be discussed.

Common Vulnerabilities and Exposures

Common Vulnerabilities and Exposures (CVE) is a list of entries, each containing an identification number, description, and at least one public reference for publicly known cybersecurity vulnerabilities[3]. CVE was launched in 1999 and now became the standard naming convention to address the interoperability and disparate databases and tools. CVE entries, also called CVEs, CVE IDs, and CVE numbers by the community provide common reference points so that cybersecurity products and

[3]https://cve.mitre.org/

services can speak the same language. CVE is an international cybersecurity community effort and each new CVE entry is assigned by CVE Numbering Authorities (CNAs).

The majority of the disclosed vulnerabilities are stored at the NVD for centralized vulnerability management purposes. The NVD is the U.S. government repository of standards-based vulnerability management data and is known as the de facto central database of software security vulnerabilities[4]. CVEs stored at NVD proved to be a useful resource for vulnerability management and overall cybersecurity-related research.

Common Attack Pattern Enumeration and Classification

Common Attack Pattern Enumeration and Classification (CAPEC) efforts provide a publicly available catalog of common attack patterns that helps users understand how adversaries exploit weaknesses in applications and other cyber-enabled capabilities[5]. CAPEC was established by the U.S Department of Homeland Security in 2007 and continuously evolved to include public participation and contributions. CAPEC defines “Attack Patterns" as descriptions of the common attributes and approaches employed by adversaries to exploit known weaknesses in cyber-enabled capabilities. Each attack pattern captures knowledge about how specific parts of an attack are designed and executed and gives guidance on ways to mitigate the attack’s effectiveness.

CAPEC differs from MITRE ATT&CK framework in a way that it focuses on the application security and enumerates exploits against vulnerable systems, whereas the MITRE ATT&CK framework focuses on network defense and provides a contextual understanding of malicious behavior. CAPEC is mainly used for application threat modeling and developer training and education, whereas ATT&CK is used for comparing network defense capabilities and hunting new threats.

MITRE ATT&CK framework

Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) was created at MITRE corporation to systemically categorize adversary behavior in September 2013 [55]. It was originally designed as a project to document and categorize post-compromise adversary TTPs against Microsoft Windows systems and later added other platforms and called ATT&CK for Enterprise and publicly released in May 2015. Subsequently, complementary models such as PRE-ATT&CK, ATT&CK for Mobile, and ATT&CK for ICS has been published in 2017 and 2020. The ATT&CK framework consists of the following components:

- **Adversary group**: Known adversaries that are tracked and reported in threat intelligence reports.
- **Tactics**: Tactics represent the adversary’s tactical objective: the reason for performing an action.
- **Technique/Sub-Technique**: Techniques represent “how" an adversary achieves its tactic, whereas Sub-technique further breaks down techniques into more specific descriptions of actions to reach the goal.

[4]http://nvd.nist.org
[5]https://capec.mitre.org/about/index.html

- **Software**: Software represents an instantiation of a technique or sub-technique at the software level.
- **Mitigation**: Mitigation represents security concepts and technologies to prevent a technique or sub-technique from being successfully executed.

The relationship between the components is visualized in Fig. 4.1.

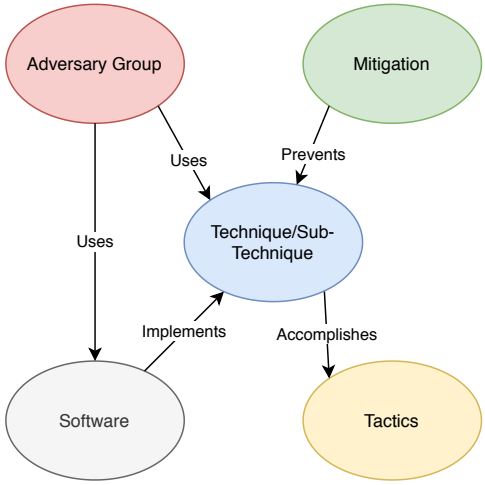

FIGURE 4.1: MITRE ATT&CK components and their relationship

The MITRE ATT&CK framework is constantly enriched with techniques and sub-techniques. At the time of writing, there are 266 techniques/sub-techniques of 12 tactics in the MITRE ATT&CK Enterprise model, 174 techniques of 15 tactics in the PRE-ATT&CK model and 79 techniques of 13 tactics in ATT&CK for Mobile model.

4.2.2 Multi-label classification

Classification is the task of learning to classify the set of examples that are from a set of disjoint labels L, $|L| > 1$. If $|L| = 2$, then the learning problem is called a binary or single-label classification and if $|L| > 2$, it is a multi-class classification. In the case of multi-class classification, the example should correspond to a single class or label whereas multi-label classification the examples are associated with a set of labels $Y \subseteq L$ [59]. According to Madjarov et al. the multi-label classification methods could be of the following categories [30].

1. **Algorithm adaptation methods:** The existing machine learning algorithms that are adapted, extended, and customized for multi-label classification problem. The examples include: boosting, k-nearest neighbors, decision trees, and neural networks.
2. **Problem transformation methods:** This method transforms the multi-label classification into one or more single-label classification or regression problems. It is further divided into categories as binary relevance, label power-set, and pair-wise methods.
3. **Ensemble classification:** The ensemble methods are developed on top of existing problem transformation or algorithm adaptation methods. The examples include Random k-label sets (RAkEL) and ensembles of pruned sets (EPS) etc.

4.2.3 Evaluation measures of multi-label classification

Since the multi-label classification task is different from the traditional binary classification, the evaluation metrics to measure the performance of the method also differs. The multi-label classification measures generally fall into the following categories according to [30].

1. Example based measures
2. Label based measures
3. Ranking based measures

The evaluation measures used in this study are briefly discussed below. In the definitions, y_i denotes the set of true labels for example x_i and $h(x_i)$ denotes the set of predicted labels for the same examples. N is the number of examples and Q denotes the total number of possible class labels.

Subset Accuracy

Subset Accuracy, also called as Exact Match Ratio is the most strict metric, indicating the percentage of samples that have all their labels classified correctly. It can be calculated as shown in (1):

$$Accuracy(h) = \frac{1}{N}\sum_{i=1}^{N} I(h(x_i) = y_i) \tag{4.1}$$

where $I(true) = 1$ and $I(false) = 0$.

Micro averaged F1 score

Since the classification is on multiple labels the results have to be averaged out. Micro-precision and micro-recall are the measures averaged over all the example/label pair. In the definitions below TP_j, TN_j denote the number of True Positive and True Negative, FP_j, FN_j denote the number of False Positive and False Negative examples per label λ_j when considered as binary classification.

$$Precision = \frac{\sum_{j=1}^{Q} TP_j}{\sum_{j=1}^{Q} TP_j + \sum_{j=1}^{Q} FP_j} \tag{4.2}$$

$$Recall = \frac{\sum_{j=1}^{Q} TP_j}{\sum_{j=1}^{Q} TP_j + \sum_{j=1}^{Q} FN_j} \tag{4.3}$$

Micro averaged F1 Score is the harmonic mean between micro-precision and micro-recall.

$$F1 = \frac{2 \times microPrecision \times microRecall}{microPrecision + microRecall} \tag{4.4}$$

Macro averaged F1 score

Macro-precision and macro-recall are the measures averaged across all labels and defined as shown in (5) and (6).

$$Precision = \frac{1}{Q}\sum_{j=1}^{Q}\frac{TP_j}{TP_j + FP_j} \tag{4.5}$$

$$Recall = \frac{1}{Q}\sum_{j=1}^{Q}\frac{TP_j}{TP_j + FN_j} \tag{4.6}$$

Macro-F1 is the harmonic mean between precision and recall where the average is calculated per label and then averaged across all labels. If P_j and R_j are the precision and recall for all $\lambda_j \in h(x_i)$ from $\lambda_j \in y_i$ then Macro F1 is defined as in (7):

$$F1 = \frac{1}{Q}\sum_{j=1}^{Q}\frac{2 \times P_j \times R_j}{P_j + R_j} \tag{4.7}$$

Hamming loss

Hamming loss evaluates how many times an example-label pair is misclassified i.e., fraction of labels that are incorrectly predicted.

$$HammingLoss(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Q} \mid h(x_i)\Delta y_i \mid \tag{4.8}$$

where Δ stands for the symmetric difference between two sets. The smaller the Hamming loss better the model performance.

Ranking loss

Ranking loss evaluates the average fraction of label pairs that are reversely ordered for the particular example.

$$RankingLoss(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{\mid D_i \mid}{\mid y_i \mid \mid \bar{y}_i \mid} \tag{4.9}$$

where

$D_i = \{(\lambda_m, \lambda_n) \mid f(x_i, \lambda_m) \leq f(x_i, \lambda_n), (\lambda_m, \lambda_n) \in y_i \times \bar{y}_i)\}$, while $\bar{y}_i)\}$ denotes the complementary set of y in L. The smaller the Ranking loss better the model performance.

4.3 Experiment

In order to test the suitability for Mapper module the experiment on the multi-label classification of vulnerability information has been conducted using the background information of Section 4.2.

4.3.1 Experimental Dataset

The European Union Agency for Cybersecurity (ENISA) published a report in December 2019 titled State of Vulnerabilities 2018/2019 [23]. The report aimed to provide an insight into both the opportunities and limitations of the vulnerability ecosystem. They collected in total 27,471 vulnerability information published during January 1, 2018 to September 30, 2019 from various data sources. As part of the analysis of the collected data, authors mapped the CVEs to the MITRE ATT&CK technique using the common CAPEC information found in both NVD and ATT&CK. The authors generously made available the dataset they've analyzed[6] and this study utilized the CVE information mapped to MITRE ATT&CK tactics and techniques for training and testing the multi-label classification model.

The ENISA report dataset contained 8,077 CVEs that are mapped to 52 unique MITRE ATT&CK techniques or in this instance labels. The dataset cardinality (mean of the number of labels of the instances) is 9.43 and density (mean of the number of labels of the instances that belong to the dataset divided by the number of dataset labels) is 0.18. The dataset CVEs are distributed into 7 discrete buckets of technique combinations. For example, there are 668 CVEs that have a single label or technique associated with it and 1,891 CVEs that have 19 labels assigned to them. Fig. 4.2 shows this distribution.

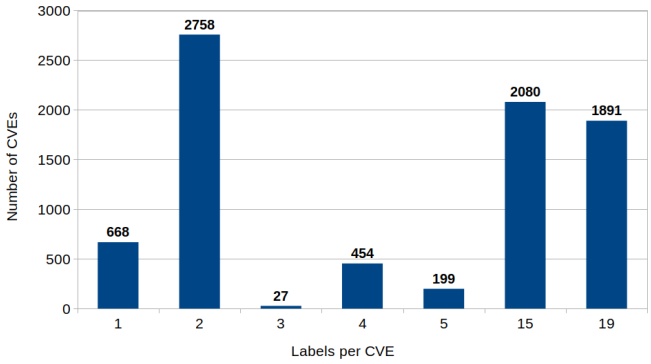

FIGURE 4.2: CVE to Label mapping of the dataset

From the ENISA report dataset of 8,077 examples, 200 examples have been held out to validate and analyze the trained model. The remaining 7,877 examples were used to train and evaluate the various multi-label classification methods.

This study has been focused to do multi-label classification based on the textual features of the CVE descriptions. The mean length of the CVE description is 368 characters and minimum/maximum lengths are 40 and 3,655 characters long. The oldest CVE updated during the data collection period is CVE-2007-6763 and the newest is CVE-2019-9975.

4.3.2 Text representation

In order to conduct the multi-label classification, the given text needs to be converted into numerical vectors, also known as embeddings. Conventionally, vector embeddings were achieved through shallow algorithms such as Bag of Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF). These approaches have been superseded by predictive representation models such as Word2Vec [38], GloVe

[6] https://github.com/enisaeu/vuln-report/

[47], and so on. Since the utilization of deep neural networks has been proven to be superior in different fields, various studies have adopted deep neural models to embed the text into vector space, such as Facebook's InferSent[7] and Universal Sentence Encoder (USE) from Google Research. Perone et al. evaluated different sentence embeddings and Universal Sentence Encoder outperformed InferSent in terms of semantic relatedness and textual similarity tasks [48]. Therefore, for the purpose of this research, Universal Sentence Encoder has been utilized to generate the vector embeddings of the text.

Since the sentence embeddings from USE produce good task performance with little task-specific training data, a Deep Averaging Network (DAN)-based USE model introduced in [7] has been used to represent the CVE descriptions in numerical vectors, so that multi-label classification methods could be applied. The model takes English sentences of variable lengths as input and produces 512 fixed-dimensional vector representations of the sentences as output[8].

4.3.3 Model selection

The CVE descriptions of the dataset have been initially converted into numerical vectors using USE. Every CVE description becomes a 512 fixed-dimensional vector that can be treated as features for the classifier. To determine the suitable model to map the vulnerability description to MITRE ATT&CK techniques 1 Algorithm Adaptation, 3 Problem Transformation, and 1 Ensemble multi-label classification methods have been experimented using the open-source library scikit-multilearn[9]. The experimented methods are listed below.

- **Multi-label k-nearest neighbors (MlkNN)** is the adaptation of the popular k-nearest neighbors (kNN) algorithm to the multi-label classification task and an example of Algorithm adaptation method. It has been estimated the number of neighbors k to be most optimal when $k = 3$ where $1 \leq k \leq 30$ when optimized for macro-average F1 measure.

- **LabelPowerset** is a Problem Transformation method that transforms a multi-label problem to a multi-class problem with 1 multi-class classifier trained on all unique label combinations. It maps each combination to a unique id number and performs multi-class classification using the classifier as a multi-class classifier and combination ids as classes.

- **ClassifierChain** is also a Problem Transformation method. The classifiers are linked along a chain where the i-th classifier deals with the binary relevance problem associated with its label. The feature space of each link in the chain is extended with the $0/1$ label associations of all previous links.

- **BinaryRelevance** is the well known one-against-all method. It learns one classifier for each label using all the examples labeled with that label as positive and remaining as negative. And while making a prediction each binary classifier predicts whether its label is relevant for the given example or not. It is an example of the Problem Transformation method.

[7] https://github.com/facebookresearch/InferSent
[8] https://tfhub.dev/google/universal-sentence-encoder/2
[9] http://scikit.ml/

- **RAndom k-labELsets multi-label classifier (RAkELd)** is an Ensemble method that divides the label space into equal partitions of size k, trains a LabelPowerset classifier per partition and predicts by summing the result of all trained classifiers.

The above-mentioned methods use traditional classification algorithms for the multi-label classification task. Since the utilization of neural networks has been proven to be superior in almost every task more neural approaches has been also experimented as multi-label classification algorithms. Since the multi-label classification task of the experiment doesn't require the sequential input or memory state of the input a simple Multilayer Perceptron (MLP) neural model has been tested to conduct the classification. Szymański et al. included a wrapper in a scikit-multilearn library that allows any Keras or PyTorch compatible backend to be used to solve multi-label problems through problem-transformation methods [57]. It has been utilized to conduct the same experiment with neural methods. The following lists the neural methods used along with their basic parameters.

- **LabelPowerset (neural)** LabelPowerset method with the Multilayer Perceptron as the base classifier. It has 2 hidden layers and the softmax function is used for activation.
- **BinaryRelevance (neural)** BinaryRelevance method with the Multilayer Perceptron as base classifier. It has 2 hidden layers and the sigmoid function is used for activation.

Next section will discuss the evaluation results of these different methods.

4.3.4 Model Evaluation

Since the dataset is limited in size the models have been evaluated with 10-fold cross validation method. The evaluations results as the average of 10-fold validation is listed in Table 4.1.

TABLE 4.1: Experiment Result

Algorithm	Acc. score	Micro F1	Macro F1	Hamm. loss	Rank. loss
MlkNN	0.6138	0.6740	0.5576	0.1079	0.3595
LabelPowerset	0.6133	0.6369	0.5174	0.1157	0.3654
ClassifierChain	0.5036	0.6089	0.4243	0.1427	0.4298
BinaryRelevance	0.3744	0.6158	0.4907	0.1471	0.3263
RakelD	0.4237	0.6230	0.5024	0.1340	0.3411
LabelPowerset (n)	**0.7432**	**0.7452**	**0.6396**	0.0911	**0.2448**
BinaryRelevance (n)	0.5538	0.7426	0.6279	**0.0883**	0.2885

From the results listed in Table 4.1, it could be seen that LabelPowerset using the neural model as base classifier has the best results in all except one evaluation measures selected. In terms of Hamming loss, the neural BinaryRelevance has a better score, but neural LabelPowerset is second in place and outperforms in every other measure. Hence, neural LabelPowerset model has been chosen as the best performing model.

There is no rule-of-thumb for "good" multi-label classification result and it depends upon various factors such as classification domain, dataset, and evaluation measures. However, in order to understand the efficiency of the model the results

have been compared with the results published in [46] in which Pakrashi et al. did a benchmarking study on various multi-label classification algorithms using eleven different datasets. When compared with their results it has revealed that the least performance in this experiment is better than the best performing results of 4 of the 11 datasets in the same measure as Macro Averaged F1 score. Hence, it has been concluded that the experimental results are good enough to be considered for discussion.

4.3.5 Model analysis

After training and testing the best performing model an in-depth analysis of the model has been conducted using the held-out validation data. A total of 200 examples were held out from the training dataset to validate and analyze the best performing model. Neural LabelPowerset model trained and tested with 7,877 examples is used to predict the labels for the previously unseen 200 examples. Fig. 4.3 depicts the prediction result in terms of distributions of CVEs per label. In this experiment, it has been considered the correct prediction only if all the expected labels are correctly predicted.

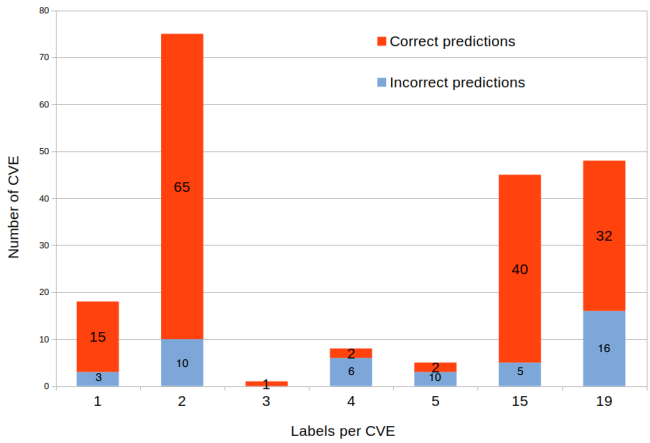

FIGURE 4.3: CVE to Label mapping of validation dataset of 200 examples

From Fig. 4.3, it could be seen that when the number of labels to be predicted increases, the chances of incorrect prediction also increases. I believe such bias could be as a result of the skewed training data. Since the experiment dataset is small in size and limited to only 52 adversarial techniques of 266 techniques/sub-techniques of MITRE ATT&CK Enterprise model and distributed to only 7 different buckets of the number of labels (see Fig. 4.2), the model tends to inadequately classify the examples. However, based on this analysis, I believe with a comprehensive training dataset, the multi-label classification method could be applied to the mapping of vulnerabilities to the adversarial techniques.

4.4 Conclusion

This chapter explored the potential tools and methods to utilize as the Mapper module of the proposed system. Various multi-label classification approaches have been tested on the vector representation of the vulnerability information to map to adversarial techniques prescribed in the MITRE ATT&CK framework.

From the experimental results, it could be seen that the neural LabelPowerset model with the Multilayer Perceptron as a base classifier could work as a Mapper module for the proposed system when trained with the multi-label classification task for mapping the vulnerability information to known adversarial techniques and tactics.

I believe using similar approaches, the cyber threat information could be enriched with the existing knowledge base and assist in the automation of the cyber defense operation.

Chapter 5

Conclusion

In this thesis, I propose to utilize various Natural Language Processing techniques for cyber defense, specifically in the Cyber Threat Intelligence process. To demonstrate it, I developed a prototype system that identifies cybersecurity specific text content, analyzes its significance and relevance, and enriches it with existing knowledge. The proposed system consists of the following modules.

1. **Natural Language Filter** module has been experimented by comparing Doc2Vec and BERT language models. The evaluation results show that the BERT-based Natural Language Filter outperformed Doc2Vec-based Natural Language Filter by 28 points in the F1 Score. From Table 2.7, it can be seen that a language model pre-trained with generic knowledge (Wikipedia and books) performs better when fine-tuned with few domain-specific data as compared to a language model that has been trained on a large amount of domain-specific data. Hence, it has been concluded that BERT would be the most suitable language model to implement the Natural Language Filter module for the proposed system.

2. **Analyzer** module has been experimented with a novel approach to quantify the relevance and significance of unstructured text that represents subjective importance to the user. I believe this approach could potentially address the problem of processing a massive amount of unstructured text for cybersecurity situational awareness. I propose to do it, through textual similarity with pre-defined important documents that the significance of the text can be determined and that by utilizing the existing Cybersecurity Knowledge Graph to correlate the named entities, the subjective relevance of the cybersecurity text could be found. For that, I trained a custom Named Entity Recognition model using over 17 million words and constructed a Cybersecurity Knowledge Graph with 221,202 semantic tuples in order to generate features that would represent their significance and subjective relevance. Combining these features, the significance and relevance of the text document could be represented in quantitative numbers. Due to the constraints such as a probable lack of identifiable cybersecurity named entity in test data and the uncertainty of identified Mentioned Entities to exist in CKG the effectiveness of the proposed architecture could not be proven directly on the raw test documents; however, by simulating the controlled environment by manipulating the test document achieved a classification accuracy of 88% using the logistic regression classifier. Since it is impossible to expect the controlled environment in a real-life situation, the experiment has to be improved to reconcile the simulated dataset with real-life data. I believe by improving the NER performance and extending the scope of CKG the experiment would come closer to producing production-grade results.

3. **Mapper** module uses an approach to automatically map the vulnerability information to adversary techniques in the cybersecurity context. Vulnerability descriptions have been converted into vector space and experimented with various multi-label classification methods to identify the most suitable method to map the vulnerability into MITRE ATT&CK adversarial techniques. 8,077 examples from open datasets prepared by ENISA have been used for this experiment, of which 7,877 have been used to train and test 7 multi-label classification methods in 9 evaluation measures. A comprehensive analysis of the remaining 200 examples as a prediction only task has been also performed using the best performing neural LabelPowerset model. Due to the partial nature of the experimental dataset, the experimental result could not be fully tested in real-life scenarios. However, in the given dataset, the chosen methods show good performance, indicating a comprehensive dataset may yield a production-ready system that could be used to automate and prioritize the cyber defense operations.

The individual results of the independent experiments support the utilization of particular method for that problem. Essentially, it could be inferred that by utilizing various Natural Language Processing techniques in the Cyber Threat Intelligence process the cyber defense could be improved, specifically in situational awareness and security automation operations.

Acknowledgements

First of all, I would like to thank my primary supervisor, Prof. Hajime Shimada, and supervisors, Prof. Yukiko Yamaguchi and Prof. Hirokazu Hasegawa who guided me throughout this project. Besides my supervisors, I would like to thank the examiners of my thesis, Prof. Tsutomu Murase and Prof. Yuichi Kaji for examining this thesis.

I would also like to express my sincere gratitude to MEXT Scholarships, the Japanese Government scholarship program, for providing me the opportunity to pursue a higher academic degree.

Last but not the least, I would like to thank my family and my friends, without whom I couldn't have written this thesis.

Appendix A

List of Publications

Below are the list of publications related to this doctoral thesis.

A.1 Journal Papers

1. O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi, H. Shimada, and E. Bataa, “Identification of cybersecurity specific content using different language models", Journal of Information Processing, vol. 28, pp. 623-632, 2020. DOI: 10.2197/ipsjjip.28.623.

2. O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi and H. Shimada, “Quantifying the Significance and Relevance of Cyber-Security Text Through Textual Similarity and Cyber-Security Knowledge Graph", IEEE Access, vol. 8, pp. 177041-177052, 2020. DOI: 10.1109/ACCESS.2020.3027321.

A.2 Peer Reviewed International Conference Papers

1. O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi and H. Shimada, “Identification of cybersecurity specific content using the doc2vec language model", in 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), vol. 1, 2019, pp. 396-401. DOI: 10.1109/COMPSAC.2019.00064.

2. O. Mendsaikhan., H. Hasegawa., Y. Yukiko., and H. Shimada., “Quantifying the significance of cybersecurity text through semantic similarity and named entity recognition", in Proceedings of the 6th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,, INSTICC, SciTePress, 2020, pp. 325-332, ISBN: 978-989-758-399-5. DOI: 10.5220/0008913003250332.

3. O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi and H. Shimada, “Automatic Mapping of Vulnerability Information to Adversary Techniques", in Proceedings of The Fourteenth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE) 2020, pp. 53-59, ISBN: 978-1-61208-821-1.

A.3 Domestic Conference Papers

1. O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi, and H. Shimada, “Mining for operation specific actionable cyber threat intelligence in publicly available information source", in Proceedings of Symposium on Cryptography and Information Security (SCIS) 2018, Niigata Japan, 2018.

2. O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi and H. Shimada, “Quantifying the Significance of Cybersecurity Related Text Documents by Analyzing IoC and Named Entities", in Proceedings of Computer Security Symposium (CSS) 2019, Nagasaki, Japan, 2019, pp. 1378-1383.

Bibliography

[1] A. Adhikari, A. Ram, R. Tang, and J. Lin, "Docbert: BERT for document classification," *CoRR*, vol. abs/1904.08398, 2019. arXiv: 1904.08398. [Online]. Available: http://arxiv.org/abs/1904.08398.

[2] K. Al-Rowaily, M. Abulaish, N. Al-Hasan Haldar, and M. Al-Rubaian, "Bisal - a bilingual sentiment analysis lexicon to analyze dark web forums for cyber security," *Digit. Investig.*, vol. 14, no. C, pp. 53–62, Sep. 2015, ISSN: 1742-2876. DOI: 10.1016/j.diin.2015.07.006. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2015.07.006.

[3] R. Al-Shaer, J. M. Spring, and E. Christou, *Learning the associations of mitre att&ck adversarial techniques*, 2020. arXiv: 2005.01654 [cs.CR].

[4] H. Aman, S. Amasaki, T. Yokogawa, and M. Kawahara, "A doc2vec-based assessment of comments and its application to change-prone method analysis," Dec. 2018. DOI: 10.1109/APSEC.2018.00082.

[5] I. Beltagy, A. Cohan, and K. Lo, "Scibert: Pretrained contextualized embeddings for scientific text," *CoRR*, vol. abs/1903.10676, 2019. arXiv: 1903.10676. [Online]. Available: http://arxiv.org/abs/1903.10676.

[6] R. A. Bridges, C. L. Jones, M. D. Iannacone, K. M. Testa, and J. R. Goodall, *Automatic labeling for entity extraction in cyber security*, 2014. arXiv: 1308.4941 [cs.IR].

[7] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil, "Universal sentence encoder for English," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 169–174. DOI: 10.18653/v1/D18-2029. [Online]. Available: https://www.aclweb.org/anthology/D18-2029.

[8] E. Choi, M. T. Bahadori, E. Searles, C. Coffey, and J. Sun, "Multi-layer representation learning for medical concepts," *CoRR*, vol. abs/1602.05568, 2016. arXiv: 1602.05568. [Online]. Available: http://arxiv.org/abs/1602.05568.

[9] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. arXiv: 1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805.

[10] B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. W. Cohen, "Tweet2vec: Character-based distributed representations for social media," *CoRR*, vol. abs/1605.03481, 2016. arXiv: 1605.03481. [Online]. Available: http://arxiv.org/abs/1605.03481.

[11] N. Dionísio, F. Alves, P. M. Ferreira, and A. Bessani, "Cyberthreat detection from twitter using deep neural networks," *CoRR*, vol. abs/1904.01127, 2019.

[12] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ser. ACL '05, Ann Arbor, Michigan: Association for Computational Linguistics, 2005, 363–370. DOI: 10.3115/1219840.1219885. [Online]. Available: https://doi.org/10.3115/1219840.1219885.

[13] H. Gasmi, J. Laval, and A. Bouras, "Information extraction of cybersecurity concepts: An lstm approach," *Applied Sciences*, vol. 9, p. 3945, Sep. 2019. DOI: 10.3390/app9193945.

[14] S. P. Harter, "Psychological relevance and information science," *J. Am. Soc. Inf. Sci.*, vol. 43, pp. 602–615, 1992.

[15] J. Howard and S. Ruder, "Fine-tuned language models for text classification," *CoRR*, vol. abs/1801.06146, 2018. arXiv: 1801.06146. [Online]. Available: http://arxiv.org/abs/1801.06146.

[16] G. Huang, Y. Li, Q. Wang, J. Ren, Y. Cheng, and X. Zhao, "Automatic classification method for software vulnerability based on deep neural network," *IEEE Access*, vol. 7, pp. 28 291–28 298, 2019.

[17] K. Huang, J. Altosaar, and R. Ranganath, "Clinicalbert: Modeling clinical notes and predicting hospital readmission," *CoRR*, vol. abs/1904.05342, 2019. arXiv: 1904.05342. [Online]. Available: http://arxiv.org/abs/1904.05342.

[18] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017.

[19] Y. Jia, Y. Qi, H. Shang, R. Jiang, and A. Li, "A practical approach to constructing a knowledge graph for cybersecurity," *Engineering*, vol. 4, no. 1, pp. 53 –60, 2018, Cybersecurity, ISSN: 2095-8099. DOI: https://doi.org/10.1016/j.eng.2018.01.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2095809918301097.

[20] A. Joshi, R. Lal, T. Finin, and A. Joshi, "Extracting cybersecurity related linked data from text," in *2013 IEEE Seventh International Conference on Semantic Computing*, 2013, pp. 252–259. DOI: 10.1109/ICSC.2013.50.

[21] S. K, S. S, V. R, and S. KP, *Deep learning approach for intelligent named entity recognition of cyber security*, 2020. arXiv: 2004.00502 [cs.CL].

[22] P. S. Karvelis, D. Gavrilis, G. K. Georgoulas, and C. D. Stylios, "Topic recommendation using doc2vec," *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, 2018.

[23] V. Katos, S. Rostami, P. Bellonias, N. Davies, A. Kleszcz, S. Faily, A. Spyros, A. Papanikolaou, C. Ilioudis, and K. Rantos, "State of vulnerabilities 2018/2019," European Union Agency for Cybersecurity (ENISA), Tech Report, 2019.

[24] T. Kenter and M. de Rijke, "Short text similarity with word embeddings," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15, Melbourne, Australia: Association for Computing Machinery, 2015, 1411–1420, ISBN: 9781450337946. DOI: 10.1145/2806416.2806475. [Online]. Available: https://doi.org/10.1145/2806416.2806475.

[25] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," *CoRR*, vol. abs/1607.05368, 2016. arXiv: 1607.05368. [Online]. Available: http://arxiv.org/abs/1607.05368.

[26] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014. arXiv: 1405.4053. [Online]. Available: http://arxiv.org/abs/1405.4053.

[27] J. Lee and J. Hsiang, "Patentbert: Patent classification with fine-tuning a pre-trained BERT model," *CoRR*, vol. abs/1906.02124, 2019. arXiv: 1906.02124. [Online]. Available: http://arxiv.org/abs/1906.02124.

[28] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: A pre-trained biomedical language representation model for biomedical text mining," *CoRR*, vol. abs/1901.08746, 2019. arXiv: 1901.08746. [Online]. Available: http://arxiv.org/abs/1901.08746.

[29] V. Legoy, M. Caselli, C. Seifert, and A. Peter, "Automated retrieval of att&ck tactics and techniques for cyber threat reports," *ArXiv*, vol. abs/2004.14322, 2020.

[30] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Deroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recogn.*, vol. 45, no. 9, 3084–3104, Sep. 2012, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2012.03.004. [Online]. Available: https://doi.org/10.1016/j.patcog.2012.03.004.

[31] O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "Mining for operation specific actionable cyber threat intelligence in publicly available information source," in *Proceedings of Symposium on Cryptography and Information Security (SCIS)*, 2018.

[32] ——, "Identification of cybersecurity specific content using the doc2vec language model," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, 2019, pp. 396–401. DOI: 10.1109/COMPSAC.2019.00064.

[33] ——, "Quantifying the significance and relevance of cyber-security text through textual similarity and cyber-security knowledge graph," *IEEE Access*, vol. 8, pp. 177 041–177 052, 2020. DOI: 10.1109/ACCESS.2020.3027321.

[34] O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi, H. Shimada, and E. Bataa, "Identification of cybersecurity specific content using different language models," *Journal of Information Processing*, vol. 28, pp. 623–632, 2020. DOI: 10.2197/ipsjjip.28.623.

[35] O. Mendsaikhan., H. Hasegawa., Y. Yukiko., and H. Shimada., "Automatic mapping of vulnerability information to adversary techniques," in *Proceedings of the SECURWARE 2020, The Fourteenth International Conference on Emerging Security Information, Systems and Technologies*, IARIA, ThinkMind, 2020, pp. 53–59, ISBN: 978-1-61208-821-1.

[36] ——, "Quantifying the significance of cybersecurity text through semantic similarity and named entity recognition," in *Proceedings of the 6th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, INSTICC, SciTePress, 2020, pp. 325–332, ISBN: 978-989-758-399-5. DOI: 10.5220/0008913003250332.

[37] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781.

[38] ——, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations*, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781.

[39] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. arXiv: 1310.4546. [Online]. Available: http://arxiv.org/abs/1310.4546.

[40] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, *Deep learning based text classification: A comprehensive review*, 2020. arXiv: 2004.03705 [cs.CL].

[41] S. More, M. Matthews, A. Joshi, and T. Finin, "A knowledge-based approach to intrusion detection modeling," in *2012 IEEE Symposium on Security and Privacy Workshops*, Los Alamitos, CA, USA: IEEE Computer Society, 2012, pp. 75–81. DOI: 10.1109/SPW.2012.26. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SPW.2012.26.

[42] V. Mulwad, W. Li, A. Joshi, T. Finin, and K. Viswanathan, "Extracting information about security vulnerabilities from web text," in *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 03*, ser. WI-IAT '11, USA: IEEE Computer Society, 2011, 257–260, ISBN: 9780769545134. DOI: 10.1109/WI-IAT.2011.26. [Online]. Available: https://doi.org/10.1109/WI-IAT.2011.26.

[43] H. T. Nguyen, P. H. Duong, and E. Cambria, "Learning short-text semantic similarity with word embeddings and external knowledge sources," *Knowl. Based Syst.*, vol. 182, 2019.

[44] L. Niu and X. Dai, "Topic2vec: Learning distributed representations of topics," *CoRR*, vol. abs/1506.08422, 2015. arXiv: 1506.08422. [Online]. Available: http://arxiv.org/abs/1506.08422.

[45] K. Oosthoek and C. Doerr, "Sok: Att&ck techniques and trends in windows malware," in. Dec. 2019, pp. 406–425, ISBN: 978-3-030-37227-9. DOI: 10.1007/978-3-030-37228-6_20.

[46] A. Pakrashi, D. Greene, and B. MacNamee, "Benchmarking multi-label classification algorithms," in *24th Irish Conference on Artificial Intelligence and Cognitive Science (AICS'16), Dublin, Ireland, 20-21 September 2016*, CEUR Workshop Proceedings, 2016.

[47] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: https://www.aclweb.org/anthology/D14-1162.

[48] C. S. Perone, R. Silveira, and T. S. Paula, "Evaluation of sentence embeddings in downstream and linguistic probing tasks," *arXiv preprint arXiv:1806.06259*, 2018.

[49] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *CoRR*, vol. abs/1802.05365, 2018. arXiv: 1802.05365. [Online]. Available: http://arxiv.org/abs/1802.05365.

[50] P. Phandi, A. Silva, and W. Lu, "SemEval-2018 task 8: Semantic extraction from CybersecUrity REports using natural language processing (SecureNLP)," in *Proceedings of The 12th International Workshop on Semantic Evaluation*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 697–706. DOI: 10.18653/v1/S18-1113. [Online]. Available: https://www.aclweb.org/anthology/S18-1113.

[51] A. Pingle, A. Piplai, S. Mittal, A. Joshi, J. Holt, and R. Zak, "Relext: Relation extraction using deep learning approaches for cybersecurity knowledge graph improvement," *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 879–886, 2019.

[52] A. Piplai, S. Mittal, A. Joshi, T. Finin, J. Holt, and R. Zak, "Creating cybersecurity knowledge graphs from malware after action reports," *UMBC Faculty Collection*, 2019.

[53] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 1, pp. 17–30, 1989. DOI: 10.1109/21.24528.

[54] A. M. Rinaldi, "An ontology-driven approach for semantic information retrieval on the web," *ACM Trans. Internet Technol.*, vol. 9, no. 3, Jul. 2009, ISSN: 1533-5399. DOI: 10.1145/1552291.1552293. [Online]. Available: https://doi.org/10.1145/1552291.1552293.

[55] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK®: Design and philosophy," The MITRE Corporation, Tech Report, 2020.

[56] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, "Videobert: A joint model for video and language representation learning," *CoRR*, vol. abs/1904.01766, 2019. arXiv: 1904.01766. [Online]. Available: http://arxiv.org/abs/1904.01766.

[57] P. Szymański and T. Kajdanowicz, "A scikit-based Python environment for performing multi-label classification," *ArXiv e-prints*, Feb. 2017. arXiv: 1702.01460 [cs.LG].

[58] N. Tavabi, P. Goyal, M. Almukaynizi, P. Shakarian, and K. Lerman, "Darkembed: Exploit prediction with neural language models," in *Proceedings of AAAI Conference on Innovative Applications of AI (IAAI2018)*, 2018.

[59] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *IJDWM*, vol. 3, pp. 1–13, 2007.

[60] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *CoRR*, vol. abs/1003.1141, 2010. arXiv: 1003.1141. [Online]. Available: http://arxiv.org/abs/1003.1141.

[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[62] V. Yadav and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models," in *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2145–2158. [Online]. Available: https://www.aclweb.org/anthology/C18-1182.

[63] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5753–5763.

[64] F. Yi, B. Jiang, L. Wang, and J. Wu, "Cybersecurity named entity recognition using multi-modal ensemble learning," *IEEE Access*, vol. 8, pp. 63 214–63 224, 2020. DOI: 10.1109/ACCESS.2020.2984582.

[65] X. Zhong, E. Cambria, and A. Hussain, "Extracting time expressions and named entities with constituent-based tagging schemes," *Cognitive Computation*, vol. 12, pp. 844–862, 2020, ISSN: 1866-9956. DOI: 10.1007/s12559-020-09714-8. [Online]. Available: http://researchrepository.napier.ac.uk/Output/2665369.