Constructing Knowledge Graph for Cybersecurity Education

by

Fanjie Lin

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2018 by the
Graduate Supervisory Committee:

Dijiang Huang, Chair
I-Han Hsiao
Yinong Chen

ARIZONA STATE UNIVERSITY

December 2018

ABSTRACT

There currently exist various challenges in learning cybersecuirty knowledge, along with a shortage of experts in the related areas, while the demand for such talents keeps growing. Unlike other topics related to the computer system such as computer architecture and computer network, cybersecurity is a multidisciplinary topic involving scattered technologies, which yet remains blurry for its future direction. Constructing a knowledge graph (KG) in cybersecurity education is a first step to address the challenges and improve the academic learning efficiency.

With the advancement of big data and Natural Language Processing (NLP) technologies, constructing large KGs and mining concepts, from unstructured text by using learning methodologies, become possible. The NLP-based KG with the semantic similarity between concepts has brought inspiration to different industrial applications, yet far from completeness in the domain expertise, including education in computer science related fields.

In this research work, a KG in cybersecurity area has been constructed using machine-learning-based word embedding (i.e., mapping a word or phrase onto a vector of low dimensions) and hyperlink-based concept mining from the full dataset of words available using the latest Wikipedia dump. The different approaches in corpus training are compared and the performance based on different similarity tasks is evaluated. As a result, the best performance of trained word vectors has been applied, which is obtained by using Skip-Gram model of Word2Vec, to construct the needed KG. In order to improve the efficiency of knowledge learning, a web-based front-end is constructed to visualize the KG, which provides the convenience in browsing related materials and searching for cybersecurity-related concepts and independence relations.

DEDICATION

Dedicated to my beloved family.

ACKNOWLEDGMENTS

First and foremost, I am grateful to Dr. Dijiang Huang for giving me the opportunity to work with him and his group. He has guided me patiently through challenges in the research for completion of my thesis work. I would not be able to achieve it without the insightful suggestions from him. It is my great honor to be your master student.

In addition, I would like to thank Dr. I-Han Hsiao and Dr. Yinong Chen for being my committee members and provide valuable feedback and suggestions to my research.

I would express my gratitude to the School of Computing, Informatics, and Decision Systems Engineering for providing me the good learning opportunities and supporting my internships during my graduate study.

I would also like to acknowledge all SNAC lab members and my friends who helped me and guided me a lot in my study. Specially, I would like to express my sincere thanks to Yuli Deng, a doctoral student at ASU, for spending much of his valuable time to provide technical support in machine installation and upgrade, as well as guidance in my daily work. I would like to extend my thanks to Dr. Stanford Ho at ASU Health Services, Dr. Michael Whitaker at Mayo Clinic for taking care of me in these years.

Finally, I would like to thank my loved parents, Yuepeng Lin and Dr. Aixi Zheng who love me and support me during the entire period of my study. Last but not the least, I would like to express my sincere thanks to my husband Dr. Yang Feng for his encouragement and shaping my writing skills in my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Nowadays, cybersecurity has become one of the most-discussed fields in technology and there is a high demand for cybersecurity professionals in the industry with constant growth within the past few years. According to a Peninsula Press analysis of the Bureau of Labor Statistics report [1], there are more than 209,000 unfilled cybersecurity jobs in United States alone in 2016 and this demand is expected to grow by 53 percent through 2018. Existing learning materials on cybersecurity education are mainly managed in a problem-centric fashion, in which instructors arrange learning and corresponding materials based on a specific security issue or solution, e.g., firewall configuration, IDS deployment, buffer overflow attack, etc. However, the concepts of this materials' dependencies are usually complicated and unclear, which hinders both students and instructors to manage the learning materials in a coherent way. Due to the lack of requirements for students and limited qualifying teaching resources, the talent pool of cybersecurity professionals is yet to keep up.

To close the cybersecurity professionals demanding gap, we need to greatly improve existing cybersecurity education, and in particular, providing self-guided learning modules is an effective approach not only allowing students and researchers to follow the well-structured curriculum but also giving them the opportunity to learn at their own pace. Knowledge Graph (KG) is one of the effective solutions to organize, index, recommend reading materials for learners. In this thesis, We have presented a research work on designing and developing the cybersecurity KG which includes both learning-related and domain-content-related knowledge to improve the learning efficiency of cybersecurity knowledge.

1.1 Problem Statements

Unlike other Computer Science (CS) topics such as computer architecture and programming languages where well-structured curriculum has been adopted by most of the existing institutions, cybersecurity is a multidisciplinary area that focuses more on loosely connected problems or solutions. As a result, the learning materials vary based on the preference of individual instructors and learning outcomes are difficult to measure. Usually, practitioners are only required to tackle the real-world problem in a few specific areas. Currently, using hands-on exercises is a critical and efficient learning approach for cybersecurity education since it focuses more on practical problem-solving skills instead of theory and models. However, most of the existing learning materials lack a coherent way to manage and provide a more productive learning plan for cybersecurity learners.

1.2 Research Questions

The goal of the research is to the issues in 1.1, more precisely to answer the research questions as follows:

- How to provide more effective learning for students currently cybersecurity education?
- Is there any solution that is easier for rookies to understand in the cybersecurity area?
- Can we recommend more reasonable reading materials/hands-on labs for lecturers and students?
- Instead of learning specific knowledge required by the courses/instructors, can

we have a general guidance to preview the knowledge structure?

- Can students learn the knowledge more clearly?

1.3 Contributions

To address the above-described cybersecurity education issues, we managed to design a new learning solution that can provide the knowledge graph and guidance to effectively organize, index, recommend reading materials and hands-on labs for learners.

In this presented work, we have deployed Natural Language Processing (NLP) as the foundation to build the corresponding KG. NLP is a multidisciplinary field, which covers CS, artificial intelligence (AI), and linguistics, aiming to improve the interactions between human and computer. In the past, lots of NLP tasks, i.e., chunking and document classification, are based on supervised learning methods which require manual annotation and feature selection in data training. Such annotated data usually has the limited size and it is hard to perform operations (e.g., annotation and feature selection) efficiently on new dataset. With the dramatic development of deep learning and representation of learning emerging in various domains, more and more research works have proven the unsupervised learning capabilities for NLP tasks in analyzing the massive corpus of text. Word embedding is one of the unsupervised learning abilities of NLP tasks, which is also known as the distributed representation of words by representing words with much lower dimensional numerical vectors[2]. Word embedding, with its advantage on converting human-readable words into machine-readable numeric, has been gaining extensive attention in recent years. Embedded words are learned as vectors [3] to represent the semantic similarities among different words. With the numerical representation of similarities available to machine, NLP-based

KG built from word embedding, which largely reduce the search scope for people in learning, has brought inspiration to different industrial applications.

The contributions of this research work is as follows below:

(1) Use unsupervised learning-based word embedding to convert the full dataset of words available at Wikipedia [4] into its corresponding numerical vector. We have processed Wikipedia as the corpus in which has the wide coverage of words and concepts. The latest version of English Wikipedia, which was used in this work, has over one billion concepts (terms or phrases that possess a page/definition in the knowledge base, in our case Wikipedia) for the machine to learn the meaning of around two billion words without supervision and embedding. Due to the ambiguity in present existing word embedding approaches, we have followed the Sherkat's method [5] to learn the embedding of Wikipedia Entity and Concept separately. We also used the Stanford CoreNLP [6] which is known as the 'Name Entity Recognition' to preprocess the contexts as the alternative approach. These two different approaches to word embedding have been deployed for comparison purposes in terms of word similarity.

(2) Develop the toolkit to collect concepts from computer security category in Wikipedia which was contributed by numerous Wikipedia users as the baseline test corpus and collect research articles in the cybersecurity domain from the IEEE Xplore Digital library [7]. In addition, we have developed a toolkit to preprocess selected ebooks and lab descriptions as part of training data;

(3) Compare the different methodologies in word embedding and modify the current Word2Vec method by proposing the use of dynamically allocated window size to improve the accuracy in word embedding training;

(4) Given the numerical vector representation of every single word, extract the semantic similarity between different terminologies (may be made up by multiple words) within the cybersecurity area of Wikipedia, by using a modified Word2Vec Skip-gram model [8] (and it will be discussed in detail in later chapters);

(5) Based on the given categorization from Wikipedia page, each term being trained in (4) will be used to find the most related (in terms of key-word similarity) research articles to improve the efficiency of knowledge learning and the quality of research outcome in the cybersecurity area. It will help students and researchers to explore the materials and develop skills in the specific area;

(6) Construct a visualized KG of concepts and terminologies of cybersecurity based on a huge amount of public cybersecurity contents as mentioned above (Wikipedia, cybersecurity research articles, ebooks and cybersecurity lab descriptions). The nodes of the KG and their dependency relationship (similarity) are obtained by learning public cybersecurity corpus which is fine-tuned with research articles and hands-on online lab - ThothLab [9], [10];

(7) Develop a web-based front end to visualize KG to fill the knowledge base gap as well as the talent gap: CyberKG-Research (KG with Research Articles Recommendation)and CyberKG-Lab (KG for Cybersecurity Online Lab - ThothLab) for easy browsing and search cybersecurity related concepts, as well as their interdependence relations.

1.4 Outline

The remainder of this thesis report is organized as follows. Chapter 2 presents the related background, literature and works on word embedding, as well as on knowledge graph. Chapter 3 explains the data preparing, processing, and training in our experiment implementation, discusses models(results) comparison by using obtained previous training, introduces how we recommend related research articles and construct the knowledge graph. Chapter 4 shows the visualization of our knowledge graphs with the explanation. Discussions and conclusions are presented in Chapter 5. Finally, we provide the future work of this research in Chapter 6.

Chapter 2

RELATED WORK

In this chapter, we will discuss the related research works on the KG construction in recent years. We have analyzed different methodologies for the word representations and word similarities. In addition, we will discuss and analyze the related existing works on implementation of word embedding by using the massive size of data (e.g. Wikipedia corpus), as well as the possibilities to construct the NLP-based KG on this approach.

Building a KG can be difficult although related works have been done in this area recently. According to research conducted, there are two approaches to develop the knowledge bases in education: the first approach is primarily relying on the professional experts, which involves manual work to certain extent in order to decide the discrepancies from different professionals and then generate a corresponding graph. The outcome graph is usually small due to different viewpoints of human experts and it is limited for learning purposes since most useful KG's are larger in terms of comprehensiveness. Another approach is the automated data from web pages and online books which are retrieved by the computers rather than humans, i.e., Wikimindmap. There are various solutions been proposed within the last decade of research on building the KG: Mahdisoltani, Biega and Suschanek [11] have shown how to construct a KG from Wikipedia in multiple languages and extended existing YAGO knowledge bases; Punuru and Chen [12] presented their work on automatic ontology extraction from domain texts using different machine learning techniques; Nickel, Murphy, Tresp and Gabrilovich [13] gave a comprehensive review on training statistical models for large KG's, and further used them to predict new edges in the graph. In 2003, J.

Undercoffer, A. Joshi and J. Pinkston[14] developed the most significant ontology on model attacks and related entities. Since then this ontology has been extended to the cybersecurity area which acquires U.S. National Vulnerability Database (NVD) [15] by A. Joshi et al. [16] in 2013. After a few years, M. Iannacone et al. [17] developed and created an ontology for cybersecurity knowledge graphs which included 15 entity types and 115 properties in total. The common things on these works are focused on the specific cybersecurity categories that are not working well for rookies in the cybersecurity area, and it is not for the educational purpose. To compare with traditional ontology which describes the types, properties, and interrelations between entities as part of knowledge representation for big data processing [18], the KG, as a graph form, is the learning repository for the collection of entities, instances and relationships to capture the data used by problem-solving.

Word embedding has recently been drawn attention to various learning tasks. It is about mapping words or phrases to a low dimensional numerical vector. For computer to understand natural languages and the knowledge within, we need a way to represents words efficiently. Traditionally, natural language processing systems treat words as discrete symbols which lead to data sparsity which commonly means that more data is needed in order to successfully train the statistical models. While a word can be understood by the human when it appears in the context, its numerical model has to be constructed based on the complex contexts using the neural network. Using vector representations makes natural language computer-readable which allows us to perform powerful mathematical operations on words to detect their similarities. The distributed representation of words as vectors has significant advantages over the traditional words representation, e.g., higher accuracy of finding the similarity between words.

GloVe (Global Vector) [19] and Word2Vec [20] both are pathfinder methodologies with unsupervised learning for word embedding. GloVe is the count-based model by using matrix factorization and local context window size to train on the co-occurrence matrix, producing word embedding. This approach may cause overfitting which decreases the accuracy during evaluating the massive size of data (due to the unacceptable low accuracy compared to Word2Vec methodology in our generated data, this approach has been withdrawn in our research). In 2013, Tomas Mikolov et al. [8], [2] proposed Word2Vec, a two-layer neural network that embeds text by using two possible models: CBOW (Continuous Bag of Words) and Skip-Gram to minimize the complexity in computation on continuous vector representation. It either uses the context to predict a target word or, vice versa, uses a word to predict the target context. Its input is a text corpus and the output are a set of vectors: feature vectors for words in that corpus. These two models generate word embedding by capturing many syntactic and semantic relations between words where a relation is also represented as the translation between two different word embedding vectors. With sufficient data, context and use cases, Word2Vec is able to make highly accurate estimates regarding the meaning of a word based on its occurrences in history. Word2vec trains words against other neighboring words fast and efficient, in addition it has shown the state-of-art performance in similar tasks calculating similarities. The KG-based similarity of terminologies can be then constructed using Word2Vec by joining vectors for similar words into the same vector space, which helps to connect highly related words in the constructed KG.

According to the previous work done by Milne and Witten [21], two pages from Wikipedia are said to be more similar when they have more common information being shared. Other research, e.g., Tsai and Roth [22] showed that using the Anchor texts of Wikipedia led to good performance for learning the phrase vectors. Grefenstette and Muchemi [23] represented their work on constructing the specialized dictionary by using Word2vec to train the Wikipedia. Speer, Chin, and Havasi [24] represented a KG - ConceptNet5.5 which combines several sources to acquire word embeddings by using distributional semantics, e.g. Word2vec. On the other hand, Musto, Semeraro-Marco, Gemmis, and Lops [25] showed their work on learning word embedding from Wikipedia dataset to construct the content-based recommendation system (CBRS) to learn user profiles.

Chapter 3

METHODOLOGIES AND IMPLEMENTATION

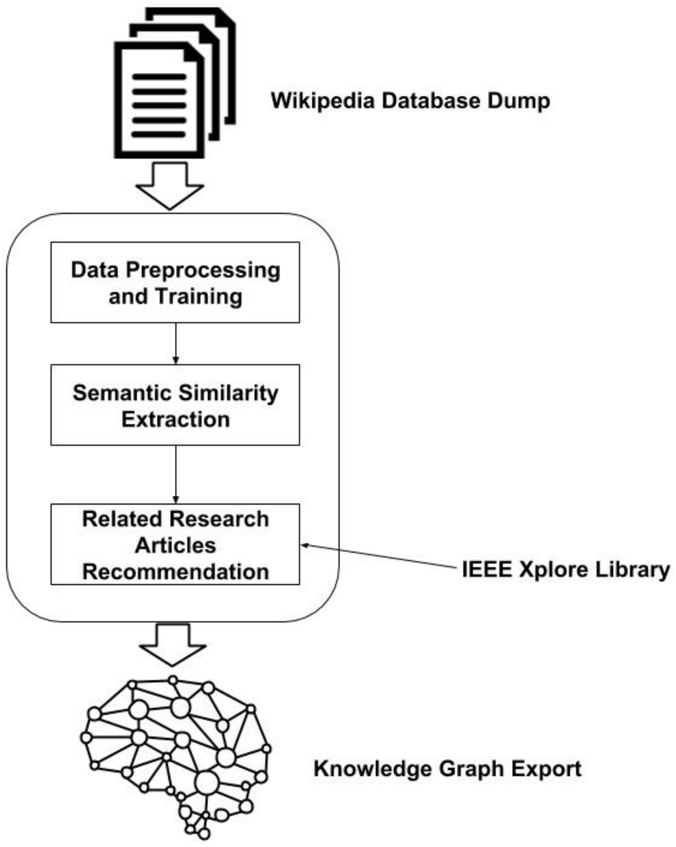

Wikipedia Database Dump
Data Preprocessing and Training
Semantic Similarity Extraction
Related Research Articles Recommendation
IEEE Xplore Library
Knowledge Graph Export

Figure 3.1: KG Construction Flowchart

In this chapter, we will present our research work of building KG for cybersecurity education. The goal of this research work is to automatically construct a cybersecurity area KG based on the similarity between concepts. The overall work flowchart is given in Figure 3.1, that we describe the steps of dataset preparing and processing at the beginning; then we explain how we apply different methods to train the words and terminologies vectors. The detailed evaluation and comparison will also be discussed.

After that, each term being trained in previous tasks' categorization of Wikipedia Pages will be used to find the most related research article from IEEE Xplore Digital library. In the end, the constructed KG with related research paper recommendation will be discussed.

3.1 Dataset Preparing and Processing

3.1.1 Full Wikipedia

In this research, the English version of Wikipedia database dump has been used which was dumped by on February 1st, 2018 from https://dumps.wikimedia.org. The detailed data preprocess with two approaches flowchart is given in 3.2.

After gathering the database dump, the toolkit has been designed and developed by using Python together with its several open source libraries to extract 18, 213, 244 pages from the XML files in English Wikipedia by that date. The extraction took about 6 hours to complete on a single PC running Linux OS, and the processed wiki data is divided into many parts as individual text files, while each file contains several Wikipedia pages (plain text without graphs). Wikipedia contains different types of pages, e.g., a page in Wikipedia can be reclassified by its function, for instance, redirect, help, category and etc. The full list of types and corresponding descriptions has been shown in Appendix A as reference [5], [26]. These pages have been removed by using the toolkit that we developed. As a result, there are 5,415,342 unique Wikipedia pages been acquired and the corresponding classified type map generated for each selected page. In the meantime, each page's information, including the plain text of concepts, internal links, and external links, will be extracted.

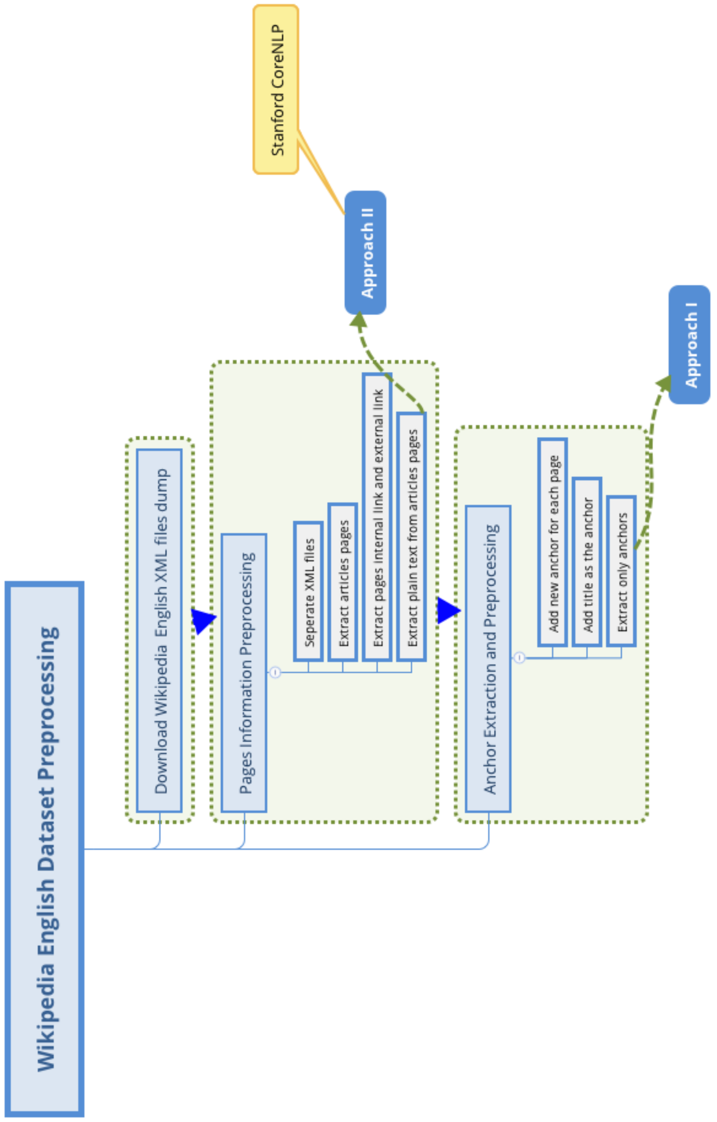

Wikipedia English Dataset Preprocessing
Download Wikipedia English XML files dump
Pages Information Preprocessing
Seperate XML files
Extract articles pages
Extract pages internal link and external link
Extract plain text from articles pages
Approach II
Stanford CoreNLP
Anchor Extraction and Preprocessing
Add new anchor for each page
Add title as the anchor
Extract only anchors
Approach I

Figure 3.2: Full Wikipedia Preprocess Flowchart

Data Processing

We have proposed two approaches to process the full Wikipedia dataset as shown in Figure 3.2 and the detailed descriptions as follows.

Approach I In this type of experimental approach, the pages which have at least one external link are applied. The basic tasks for text preprocessing, for instance, removing all punctuation characters, converting texts to lower case, are included. Then we have matched the title of internal links as initial anchors with their corresponding Page ID's which are marked by the Wikipedia users. To make the anchors more comprehensive, we have added the new anchors to the page if there is at least one anchor on the page. For instance, if token 'malware' is given in the page list then 'malware' will be added as anchor of that page. In addition, there is no self-link (no page links to the page title itself) in the page by Wikipedia policy. However, we added the title of the page as anchor as well since it is common that the title of the page itself is repeated on that page. As a result, there are 4,724,129 entries found after removing all other texts and deduplication by using the toolkit in this approach.

Approach II The Stanford CoreNLP toolkit 3.9.1 [6] has been used for sentence tokenization to preprocess Wikipedia pages' plain text only (process Wikipedia article pages only) as another approach. As a linguistic analysis tool, it helps to simplify the analyses of a bunch of texts. The basic tasks, for instance, split sentence, remove sentence which is less than five tokens and convert numbers to zero, have been included. The total number of tokens generated from the given dataset by this approach is over two billion.

With these two approaches demonstrated that we can compare the word embedding results with different size of the corpus. The data samples generated from these two approaches are been shown in Appendix B.

3.1.2 Wikiepdia Category

We have designed and developed a toolkit by using Python to scrape Wikipedia pages for the category in computer security section to acquire more accurate related information as the test dataset. The tool that we developed iterates through categories and stores a list of the corresponding information. All pages in computer security and pages in the first level, second level and third level of subcategories have been scrapped. There are 303 pages from the first level, 2,173 pages from the second level and 5,265 pages from the third level are obtained in computer security category after deduplication in this experiment as the demo.

3.1.3 IEEE Xplore Library

According to the data accessible by the end of February 2018, there are 4,529,571 articles in IEEE Xplore Library. In this research, we have selected three content types of articles from the online library: Conferences (3,172,527), Early Access Articles (15,840) and Journals & Magazines (1,296,339). We also designed and developed toolkits by using Python and Selenium [27] to scrape the related information from the website as needed. As a result of scrapping a few frequently used terms from the SANS Glossary of Security Terms [28] as sample test, we have acquired information for 25,867 unique articles (includes article title, authors, abstract and references) as well as extracted 146,080 keyword terms among the given available terms from IEEE Xplore Digital Library. The obtained keyword terms, which include IEEE Keywords, INSPEC: Controlled Indexing, INSPEC: Non-Controlled Indexing and Author Keywords, were then used to get the related Wikipedia terms.

In addition, we have used the toolkits which we developed to download 882 full

PDF format papers automatically from this online library.

3.1.4 Other Resources

To obtain more data in the cybersecurity area, we have designed and developed a toolkit by using Python to process 71 PDF format ebooks which cover network security, hacking area, and 882 full PDF format papers from IEEE Xplore Library to plain text format. These data sources and 5,265 Wikipedia plain text pages obtained in 3.1.2 are combined as the cybersecurity domain training data. Then we have applied the Stanford CoreNLP toolkit to process the above plaintext data, with Approach II described, as CyberData 1. In addition, we have extracted nouns only as a subset of CyberData 1 to be CyberData 2.

3.2 Training Methods Comparison and Discussion

3.2.1 Background

A Word is the basic unit in linguistic analysis. Unlike weather, temperature, length and others of which the characteristic can be measured with meaningful numerical values, a word is simply represented as string (a sequence of letters) in computer coding which is meaningless to computer. The representation of words in forms of numerical vectors is the foundation task to help computer in learning and understanding human languages that has a high impact on the performance of a learning system.

One-Hot Representation

The legacy representation, e.g., one-hot representation, to represent the word as vectors, is the symbolic representation and does not include any semantic information.

For instance, we have created a vocabulary V consists three words with a unique sequence number like Dog is 1, House is 2 and Animal is 3. Each word in this vocabulary will be represented as a vector and the size of the vector is the same as the size of the vocabulary set. The vector contains all zeros except for a single one at the position associated with the corresponding word index as shown below:

$$V = \{Dog, House, Animal\}$$

The one-hot vector representation is

$$Dog = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad House = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad Animal = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

As the localist representation with one dimension, there is no natural notion of similarity between with two similar words' vectors are orthogonal, e.g. $(Dog)^T(Animal) = 0$. In addition, this kind of representation computation is very expensive and overfitting while training with massive data set given the fact that a extremely high dimensional vector will be used (e.g. if there are billions of words in a data set, a vector with the length of billions will be needed).

Distributional Representation

In 1954, Z. S. Harris standardized distributional hypothesis as the theoretical basis for embedding semantic into word representations: Words that occur in similar contexts tend to have similar meanings [29]. A few years later, J. R. Firth phrased this theory in 1957 that a word is characterized by the company (context) it keeps [30]. Assuming that each word is represented as the size of vocabulary $|V|$ dimensional vector, we searched for the occurrence of the target word $|T|$ and its neighborhood words with a defined the context $|c|$ in the entire corpus as the vector. Then we can normalize the vector by using the updated frequency of the target word and its neighborhood words to get the probability distribution. To compare with one-hot representation, the distributional representations can be effectively applied to reduce the dimensionality to the word co-occurrence matrix of size $|V|$ divided by $|T|$.

For several years development of this theory, there are three major types of word-representation-based models: the distributional-semantic models which are based on the matrix, the distributional-clustering-based model and the distributed-representation-based model using neural network. With the upgrade of hardware performance and algorithms optimization breakthrough, Great advantages have been shown for neural network applied in various areas. By using neural network to construct effective word representation with contexts becomes the mainstream approach in these years.

Word Embedding

Neural-network-based distribution representations are known as word embedding or distributed representation [31]. It is the task of mapping words or phrases to a low-dimensional (the dimension size range is typically between 50 to 1000 which is significantly smaller than the vocabulary size) numerical vector. With the lower dimension representation, the semantics can be preserved by this embedding, for instance, two words that are semantically close if they are close in the mapped vector space.

Word embedding deploys neural network method to construct the model from contexts and the relationship between contexts and target words. Unlike the counting-based approach to simply find the occurrence of the neighborhood words, word embedding is learned from a given text corpus without supervision, but only by predicting the context of each word or, vice versa, predicting the current word given its context, yielding that the vectors will be learned in here [32], [33]. By starting with the random vector for each word, the target word and its neighborhood words will be updated until they become closest in the vector space. For example, given the trained network and an input word 'Food', the possibilities of output are much higher for words like 'Cooking' or 'Fruit' than unrelated words like 'Airplane' or 'Zoo'. The neural network is trained by the feed of word pairs found in our input text. The network then learns the statistics according to the number of times that each pairing is found. Eventually the network will get greatly more training samples between 'Food' and 'Fruit' than the samples between 'Food' and 'Zoo'. Once the training is finished, using the word 'Food' as input, the trained network will output a much higher probability for 'Fruit' than 'Zoo' based on the semantic similarity calculated from the network.

As for the high flexibility in neural network, word embedding can therefore represent complex contexts. In addition, when the neural network expresses n grams, n words can be combined by certain patterns which limits the number of parameters growing only at a linear speed to avoid dimensionality catastrophes. With this advantage, neural network models can be used to model contexts with more complexity that contain richer semantic information in word vectors.

3.2.2 Training Experiment

In this training experiment, we will train both CBOW and Skip-Gram model by using the Python implementation of Word2Vec in the Gensim package which is the open source tool for computing the distributed representation of words [34]. Word2Vec trains words against other neighboring words, the main idea for word embedding is that the words occur in similar contexts can be represented by vectors in close nearness to each other [8] then estimated the probability prediction of the central word in CBOW model/context words in Skip-Gram model.

We will first analyze the structure of these two models and the training algorithms in this section to understand the theory of acquiring effective word embedding and efficient computation of the output probabilities. The performance of word embedding training depends on the various training parameters. To investigate and analyze the effects of word embedding generation, the varied dimensionality of word vectors, context-window size will be applied in this training section. Then the same experiment setting will be used on the Approach I and Approach II dataset to compare the accuracy and impartiality. The detailed experiment settings and parameters settings have been given in Table 3.1.

Table 3.1: Experiment Settings

Category	**Experiment Settings**
Model	CBOW, Skip-Gram
Datasets	Approach I
	Approach II
	CyberData 1
	CyberData 2
Parameter Settings	Dimension size: 50, 100, 200, 300, 400, 500, 1000
	Window size: 10(fixed), sentence size(dynamic)
	Fixed # of minimum count: 5

Continuous Bags of Words (CBOW)

The CBOW model is a simplified model as a combination of the Neural Network Language Model(NMLM) [32] and C&W model [35], to predict the suitable word given the surrounding context as shown in Figure 3.3. By given only one context word x which will be calculated from the average context words c as input $\omega_{i-(n-1)}, ..., \omega_i$

in 3.1, then the model has to learn (predict) the target word ω.

$$\boldsymbol{x} = \frac{1}{n-1} \sum_{\boldsymbol{\omega} \in c} \boldsymbol{e(\omega_j)} \tag{3.1}$$

As one type of Neural Network, the CBOW model contains only input, projection, and output layer by removing the hidden layer in a typical Neural Network. Due to the removal of hidden layers, the context words as the input layer in this model will be used to predict the target word.

$$P(\omega|c) = \frac{exp(e^{'}(\omega)^T x))}{\sum_{\omega' \in v} exp(e^{'}(\omega')^T x))} \tag{3.2}$$

The CBOW model has been translated into linear 'log' function and the training speed of model has been increased significantly. Similar to other NMLMs, the likelihood of the CBOW model can be maximized by:

$$\sum_{(\omega,c) \in D} logP(\omega|c) \tag{3.3}$$

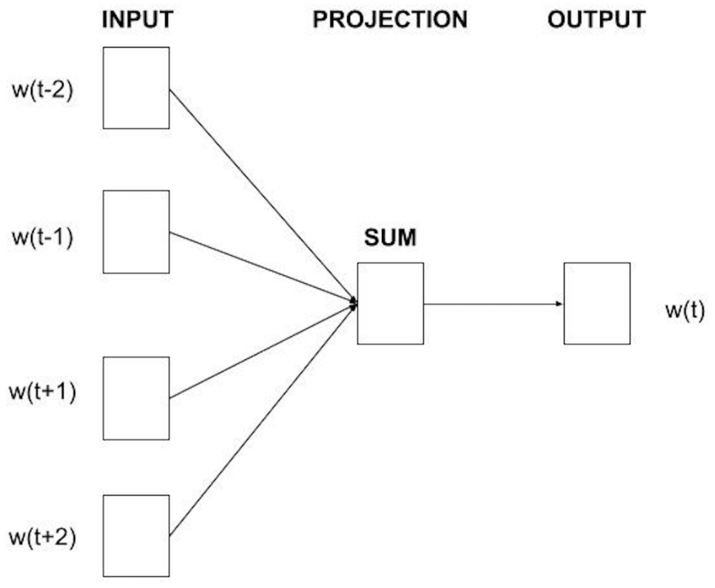


Figure 3.3: CBOW Model Structure

Skip-Gram

The Skip-Gram model has a structure which is similar to the CBOW model, except for the difference that the Skip-Gram model used one word to predict its contexts (a reverse logic compare to the CBOW model which is predicting word based on the given context) as shown in Figure 3.4. By given a target word ω to find the similar word from one of the context words c. The prediction of this model can be represented as

$$P(\omega|\omega_j) = \frac{exp(e^{'}(\omega)^T e(\omega_j)))}{\sum_{\omega' \in v} exp(e^{'}(\omega')^T e(\omega_j)))} \tag{3.4}$$

and the likelihood of this model can be maximized by 3.5.

$$\sum_{(\omega,c)\in D} \sum_{(\omega,j)\in c} logP(\omega|\omega_j) \tag{3.5}$$

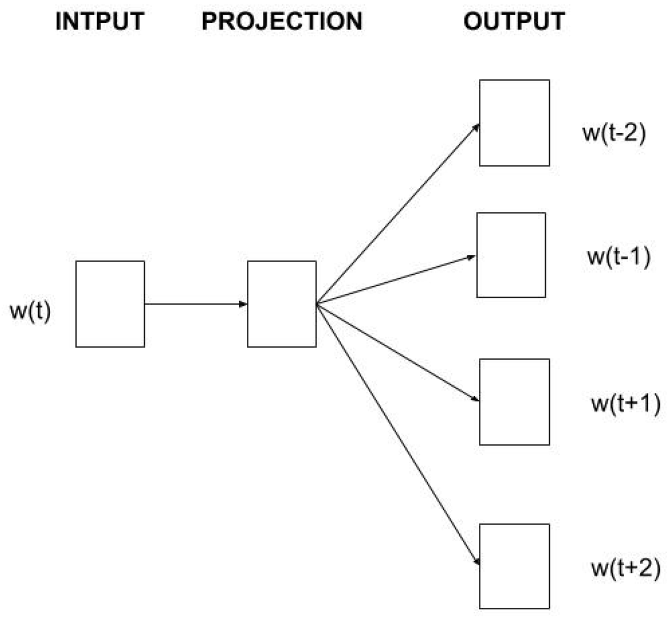


Figure 3.4: Skip-Gram Model Structure

According to the existing research works [8], it is shown that the Skip-Gram model performs better than CBOW model. We will verified this argument in detail in the later section of this chapter .

3.2.3 Window Sizes

The training patterns in Word2Vec are extracted from the window of words in sentences of training files. In the original Word2Vec implementation, we need to set a fixed window size for training. For example, we have a sentence ”I drink milk everyday” and we set the window size is 2, the steps to achieve training patterns will be shown as follows:

(1) Take ’I’ as the target word, the right side neighborhood words are ’drink’ and ’milk’;

(2) The ’drink’ as the target word, left side word ’I’ and two right side words ’milk’ and ’everyday’ are the neighborhood words;

(3) The third-word ’milk’ as the target word, the left side word ’I’ and ’drink’ and the right side word ’everyday’ are the neighborhood words of the target word;

(4) Take the last word ’everyday’ as the target word, the neighborhood words are ’drink’ and ’milk’

In conclusion, we have (I, drink, milk), (drink, I, milk, everyday), (milk, I drink, everyday), (everyday, drink, milk) as the training patterns from the given sentence. As we can see that the fixed window size limit the neighborhood words in the original Word2Vec method and then leads to the insufficient of the training patterns during analyzing large size data. Sometimes, the target word may be related to the neighborhood sentences not only in the own sentence.

To verify the effect of using different window size in Word2Vec training results, we have trained the Approach II data with Skip-Gram model, dimensionality of 400 and set window size from 5 to 50 in the original proposed Word2Vec method. The results are shown in Figure 3.5. From Figure 3.5, we can find that the top two highest accuracy range is between window size 20 and window size 30 for this dataset.

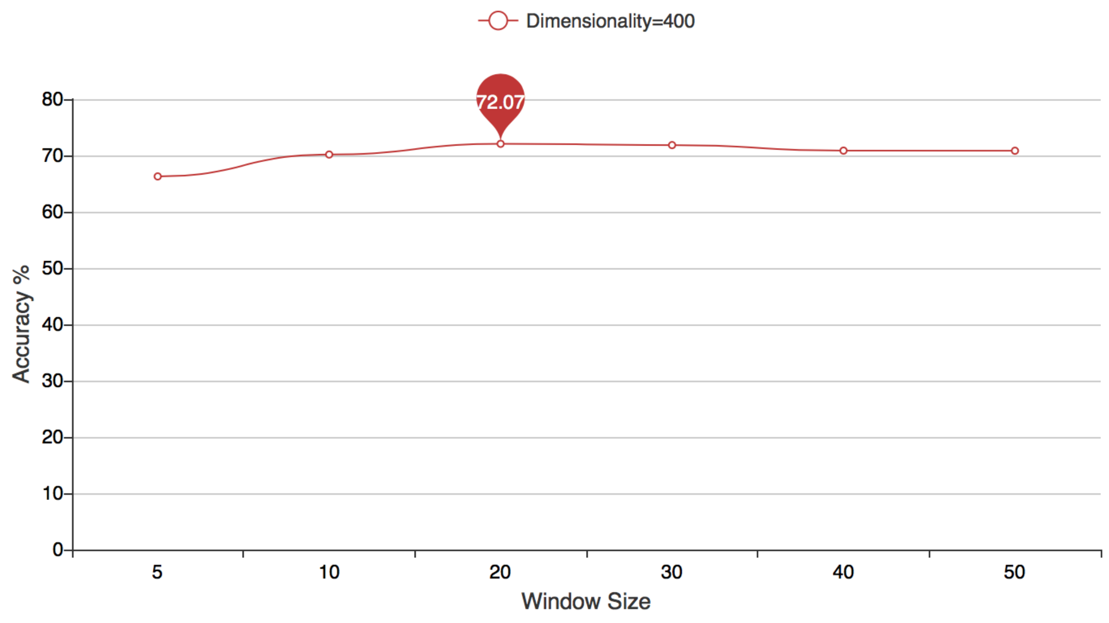


Figure 3.5: Accuracy comparison for different window size with a dimension size of 400

According to the calculation, we find that the average length of sentences in Approach II data is around 25 which is in the top two highest accuracies range's window sizes.

In addition, to verify the impact of fixed window size and dynamic window size with different dimensionalities, we have used the fixed window size (window of 10 has been set in this verification) and the dynamics window size which varies based on the length of sentences in Approach II data, trained from the Skip-Gram model.

The comparison results are shown in Figure 3.6. From the results, we can see that the proposed method, by using dynamic window, can reach the highest accuracy - 72.97% among all other methods.

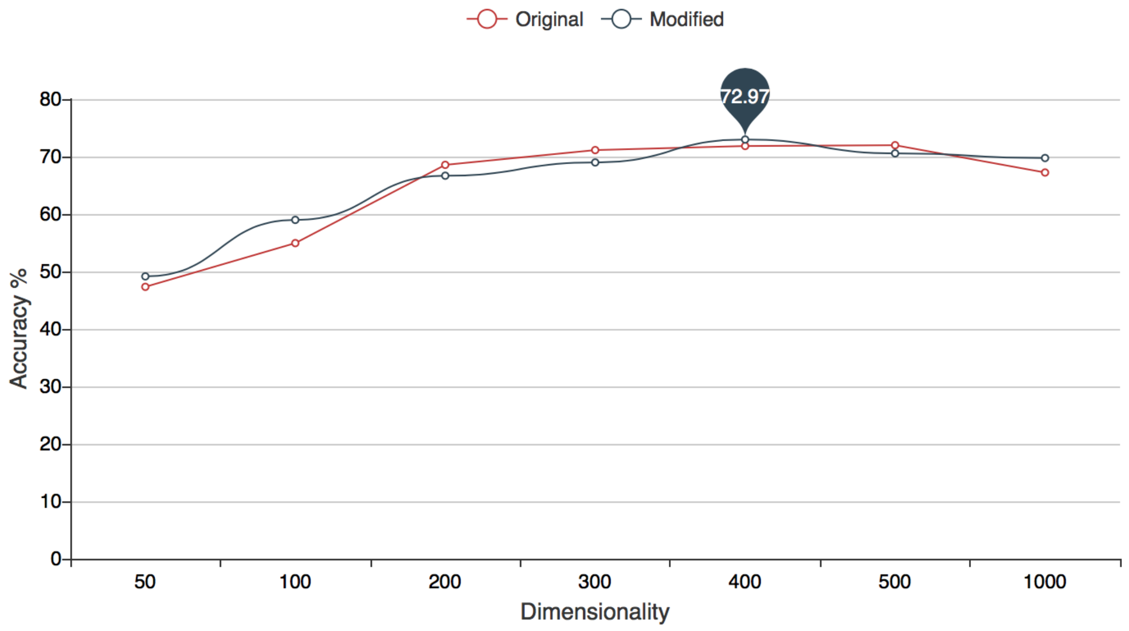


Figure 3.6: Accuracy comparison of fixed window size(original) and dynamic window size(modified) by using Skip-Gram Model

3.3 Evaluation

3.3.1 Comparison of Similarities

In this experiment, we have used the same parameters to train two types of pre-processed Wikipedia datasets as comparison. From S. Lai's research work result, large dimensions of word vectors performed better on word embedding training [36]. 50, 100, 200, 300, 400, 500, 1000 are used as the dimension size input for the user-defined Word2Vec parameter, for performance comparison purposes. As tokens reach over 2 billion for Approach II containing plural words, we have removed these plural forms of the word in the Approach II trained result to get a clear comparison.

In Word2vec, the similarity distance is calculated from the cosine distance of two vectors. The similarity is calculated by the dot product of two words' numeric vectors represented as x and y, and it is normalized by the product of the vector lengths, yielding that if two words are close to each other, the distance tends to positive and close to 1.0 with a indication of high similarity. The formula is shown as below 3.6:

$$cosineSimilarity(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{y}||} \tag{3.6}$$

Approach I and Approach II

The test results of top 5 similarities as sample are given for CBOW and Skip-gram model from Table 3.2 to Table 3.5 for the two different datasets with dimension size of 400.

Table 3.2: Top5 similarities for "Botnet" by using CBOW model to train Approach I

Most similar word	Similarity Distance
Zero-day (computing)	0.978682
Operation Tovar	0.975276
Srizbi botnet	0.972704
Antivirus software	0.971838
Conficker	0.971023

Table 3.3: Top5 similarities for "Botnet" by using Skip-Gram model to train Approach I

Most similar word	Similarity Distance
Operation: Bot Roast	0.884675
ZeroAccess botnet	0.874679
Cutwail botnet	0.873807
MoColo	0.873506
DarkOde	0.872759

Table 3.4: Top5 similarities for "Botnet" by using CBOW model to train Approach II

Most similar word	Similarity Distance
honeypot	0.851286
cryptolocker	0.767310
malware	0.765936
ransomware	0.751337
ddos	0.732009

Table 3.5: Top5 similarities for “Botnet” by using Skip-Gram model to train Approach II

Most similar word	Similarity Distance
honeypot	0.825247
malware	0.765143
ddos	0.755441
phisihing	0.727613
spamming	0.722274

From these test samples, we found that the closest terminologies of ’Botnet’ (given in 3.3) are almost interrelation from each other which are defined in the Wikipedia Pages. Given in 3.2 are the descriptions of the closest terminologies from their Wikipedia Pages, which all contains the word ’Botnet’. In Table 3.4 and 3.5, the frequent words that appear with ’Botnet’ at the same time are given.

Phrase similarity is one of the tasks used to evaluate different word embeddings. To evaluate the quality of vectors in cybersecurity area, we have designed and developed a phrase similarity dataset of cybersecurity which is based on the SANS Glossary of Security Terms. As a result, we have set 224 pairs in this dataset which contains two words with high similarity assigned by the cybersecurity expert, for instances, ’Botnet’ and ’Malware’ as one of the pairs from the dataset. We have applied this dataset onto two approaches with different dimension size setting and calculate the average similarity distances by using the formulation in 3.7. The results is shown in Table 3.6 and Table 3.7.

$$\text{Average Similarity Distance} = \frac{1}{N}\sum_{i=1}^{N} SimilarityDistance_i \text{ (N=224)} \tag{3.7}$$

Table 3.6: Average Similarity distances for using Approach I

Model	CBOW	Skip-Gram
Size 100	0.5718	0.6573
Size 200	0.6613	0.6921
Size 300	0.6823	0.7118
Size 400	0.6896	0.7209

Table 3.7: Average Similarity distances for using Approach II

Model	CBOW	Skip-Gram
Size 100	0.4795	0.5397
Size 200	0.4920	0.5728
Size 300	0.5239	0.5927
Size 400	0.5702	0.5969

We found that using a window size of 400, both Approach I and Approach II reach their highest similarity in average with Skip-Gram model which aligns with the conclusion from previous research work.

CyberData 1 and CyberData 2

According to the previous comparison between Approach I and Approach II, we verified that the Skip-Gram model works better than CBOW model. The test results of top 5 similarities for "botnet" are given as an example for Skip-gram model from Table 3.8 to Table 3.9 for the two different datasets with dimension size of 100 and dynamic window size that we proposed.

Table 3.8: Top5 similarities for “botnet” by using Skip-Gram model to train CyberData 1

Most similar word	Similarity Distance
bot	0.804090
infection	0.755443
backdoor	0.672041
detect	0.668269
herder	0.657947

Table 3.9: Top5 similarities for “botnet” by using Skip-Gram model to train CyberData 2

Most similar word	Similarity Distance
bot	0.857493
zombie	0.779758
worm	0.749846
crimeware	0.683296
trojan	0.678619

From the above tables, we found that the quality of the training results from these two datasets is not as expected by comparing with the ones obtained from Approach I and Approach II. We have verified that sufficient data is required in order to archive good word embeddings, given the following analysis: the size of the data is too small and not sufficient enough for training purposes (does not include irrelevant data for the machine to learn due to ambiguity in the data).

3.3.2 Similarity Accuracy

In order to select a good model for cybersecurity area training, terms in 'Computer security' category 1st level and terms in its 1st level subcategories are used as test dataset by using the toolkit developed previously: there are 303 terms in 'Computer security' category 1st level and 25 subcategories with 2,165 terms in total being scrapped. The accuracy of the trained similarity in each category can be evaluated by using the top number of terms in each category. For instance, there are 303 terms in 'Computer security' category, the top 303 similarities for 'Computer security' will be evaluated. The formulation to calculate the accuracy is given in 3.8. And the average accuracy from these categories will be calculated as the final result.

$$\text{Accuracy} = \frac{\text{Number of similarities for 'Computer security' in category}}{\text{Number of total terms in category}} \tag{3.8}$$

The test results of the accuracy are as shown in Table 3.10, 3.11.

Model	CBOW	Skip-Gram
Size 100	25.18%	31.68%
Size 200	25.84%	34.32%
Size 300	37.16%	47.52%

Table 3.10: Accuracy of training results for using Approach I

Model	CBOW	Skip-Gram
Size 100	16.81%	24.19%
Size 200	17.02%	26.21%
Size 300	28.72%	34.13%

Table 3.11: Accuracy of training results for using Approach II

From the above results, the model trained with Skip-Gram model with high dimension yields better performance than others. We can find that the Skip-Gram model trained in approach I has the best result among multiple different experiments.

3.3.3 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a procedure to reduce the high dimensional data to low dimensional space linearly by using Singular Value Decomposition (SVD). As being used in previous research work, this method was to deal with high dimensional data [37], [38]. In our experiment, we implemented two types of data to test the trained Skip-Gram Model in Approach I with dimension size 300:

(1) Set 20 groups of cybersecurity terms with its top 5 similarities. For instance, top 5 similarities for 'Botnet' and its related terms. In SANS glossary of security terms, the 'Botnet' definition related to the 'Spamming', 'Malware' and 'Denial-of-service attack'. We have used this example result as a sample.

(2) Terms in 1st layer of 'Computer security' in Wikipedia category from the previous experiment.

To interpret the data clustering of models, we have applied PCA method to check the vectors of these terms distributed status which among the complete trained model. The 2D diagrams are given in Figure 3.7 to Figure 3.8, respectively.

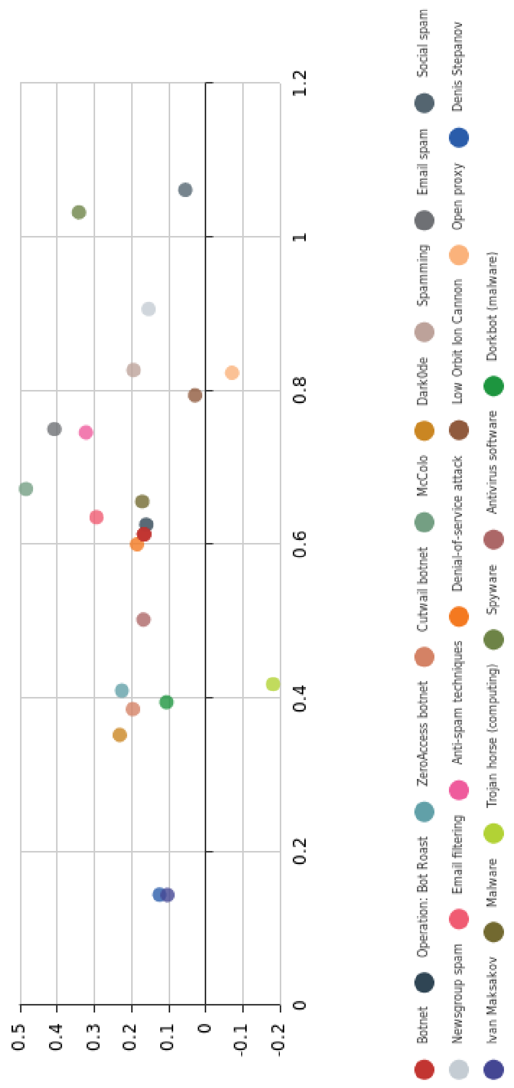


Figure 3.7: Top 5 similarities for 'botnet' and its related terms

From Figure 3.7, we can find that the 'Botnet' is close with 'Denial-of-service attack' and 'Malware' as expected. In addition, the same category is close to each other, for instance, Denis Stepanov and Ivan Makskov are the people under the same category.

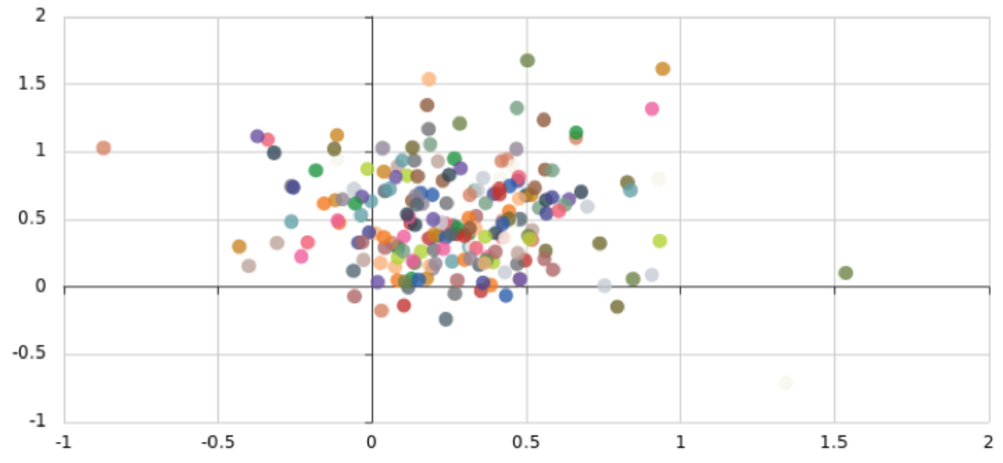


Figure 3.8: Terms in 1st layer of 'Computer security' in Wikipedia category

3.4 Research Articles Recommendation

By using the right keywords in the article, it becomes easier to know the major contributions of the article that can be reliably discovered. There are four types of keywords in IEEE Xplore Library for an article and they are described as below:

(1) Author Keywords: assigned by authors manually in their article.

(2) IEEE Keywords: automatically assigned to the research articles from the controlled vocabulary which created by IEEE.

(3) INSPEC: Controlled Indexing: assigned to articles from a controlled vocabulary of over 10,000 scientific terms created by INSPEC.

(4) INSPEC: Non-Controlled Indexing: This type of keywords are not the part of (3) assigned to articles may with new concepts which describe the topics or subjects of a document.

In this research articles recommendation part, we have developed a toolkit to compare the terminologies categorization in Wikipedia pages to match at least three keywords (include the term itself) in a combination of the four types of keywords for each article. We will only use the highest matched results as the most related research paper recommended by the built KG.

3.5 Knowledge Graph Construction

KG can be viewed as a special kind of semantic network. It is a way of representing knowledge by labeled nodes and links between these nodes. The construction of a KG starts with the extraction of information from texts. Originally, these tasks are handled by human experts. They did text analysis and get a list of concepts, represented as labeled points, and a list of links between these nodes. A small graph from a single author is called author graph. The next step is to combine graphs from various authors into one large graph by identifying points with each other. When the texts of the nodes deal with the same subject, points with the same label are identified first. Also, when an author used synonyms for a concept, points are also connected even with different labels. In addition, there is a way to compare neighborhoods of points to identify identical points. Then, similarity index is introduced to measure the similarity between two sets of points. It is then used to decide upon identification of two concepts, which can eventually help us to detect homonyms (the same label but referring to different contents).

After data gathering in previous sections, we constructed a basic KG from Wikipedia dataset. As to know that each Wikipedia page represents a concept and its explanation (which contains knowledge). There are also links within each Wikipedia page that links to other concepts. By analyzing the URL links within one Wikipedia page, we get a simple author graph, e.g., on the 'DDoS' page, there are hyperlinks that are linked to 'Exploit', 'Trojan Horse', 'IDS', 'IPS', 'Computer Fraud', 'Botnet', 'Firewall' and 'Computer Virus'. With 2,173 pages under the second level of computer security category in Wikipedia as a demo, we now have 2,173 single author graph ready to be merged together. Then we utilizes the similarity data obtained from Word2Vec training results (Approach I with modified Skip-Gram model with a dimension size of 400 been used) to further connect these author graphs. Figure 3.9 shows how we merge graphs of 'Firewall' and 'DDoS' graph into a single graph. Words paired like 'Antivirus', 'Computer virus', 'Spyware', 'Trojan Horse' are connected together as a result of their high similarity calculated based on word embedding. We set the similarity lower bound to 0.8 (while 0 means no relationship and 1 means the two concepts share the same embedding) and connect all node pairs over this similarity threshold together. After that, we get one unified and meshed connected KG ready for further utilization. We have also tried to use more author graphs (5,265 pages under the third level of computer security category) with the same threshold (0.8), resulting in more than thousands of nodes not being connected, which means that these concepts are not close enough under computer security category. As a result of this demo, we have obtained 2,095 nodes connected in KG. In other words, the deeper levels of the category is, the higher possibilities that two terms are not close enough.

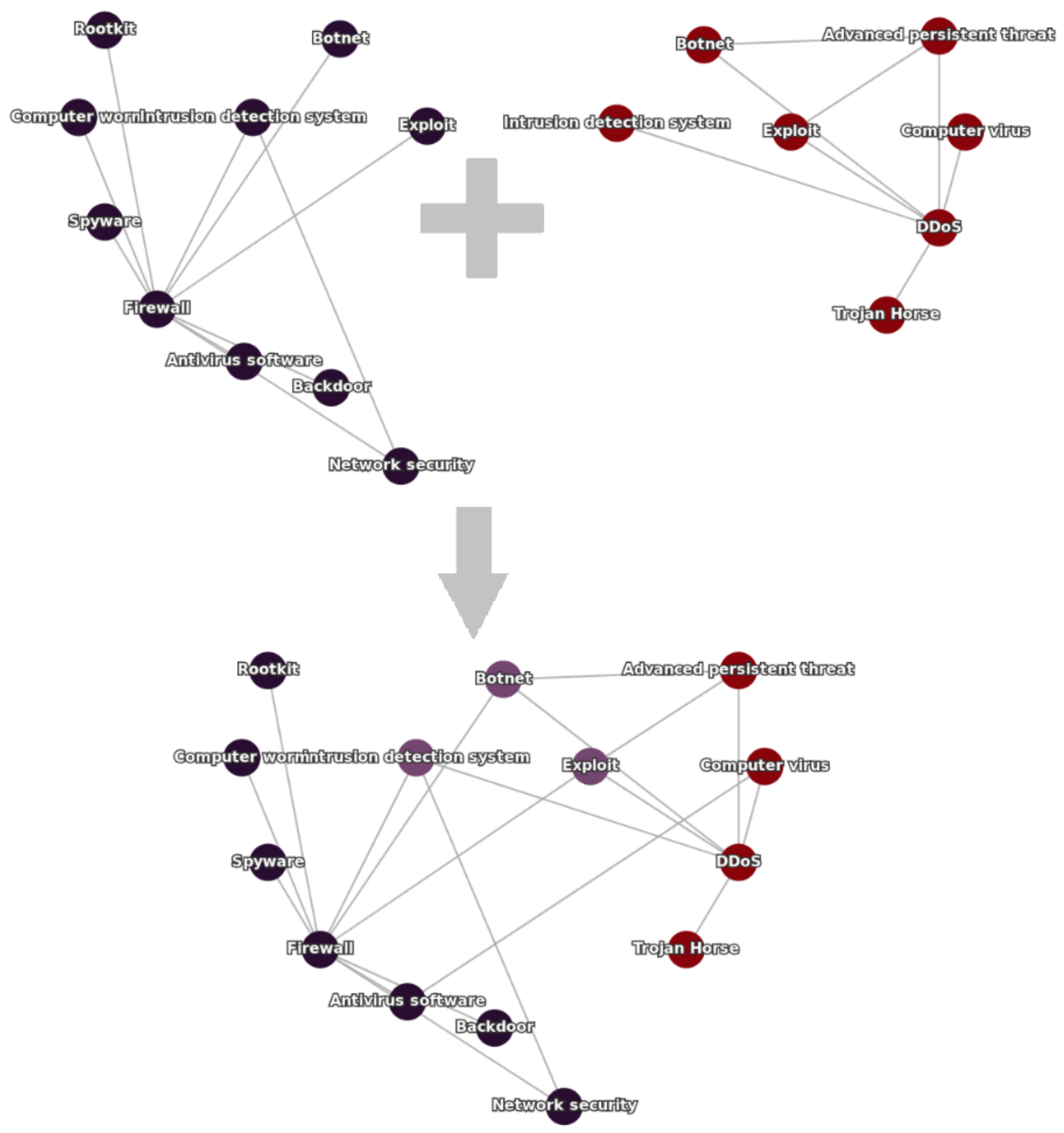


Figure 3.9: Merge two small graph together base on overlap and word embedding similarity.(Firewall Graph on the left, DDoS graph in the middle and merged result on the right.)

Chapter 4

KNOWLEDGE GRAPH VISUALIZATION

In this chapter, we will represent the KG's - CyberKG-Research and CyberKG-Lab in an interactive Graphical User Interface (GUI) to empower learning and details of the visualization.

4.1 CyberKG-Research (KG with Research Articles Recommendation)

After data gathering and preprocessing from previous tasks, we have developed a toolkit by using mixed-language programming, including Python, JavaScript, HTML and open source visualization libraries, to visualize the terminologies of the relationships under cybersecurity area by using semantic similarity measurement, with research articles recommendation embedded. At first, we used the developed toolkit to process and generate new data (which includes the node descriptions, relations and related article recommendation) with JSON format as data representation obtained from previous sections (section 3.4 and section 3.5) results. Then we generated the KG by integrating the related files developed to the web framework.

The built KG is an abundant graph model, the entity of which can be represented as a node and the link can be represented by the relationships between nodes. The attribute can be also represented in each node and its link. Figure 4.1 shows the processed result by using the developed toolkit for part of terminologies graph to present the relations in terms of similarities among the terminologies captured in the knowledge base (which was already set up in the previous tasks).

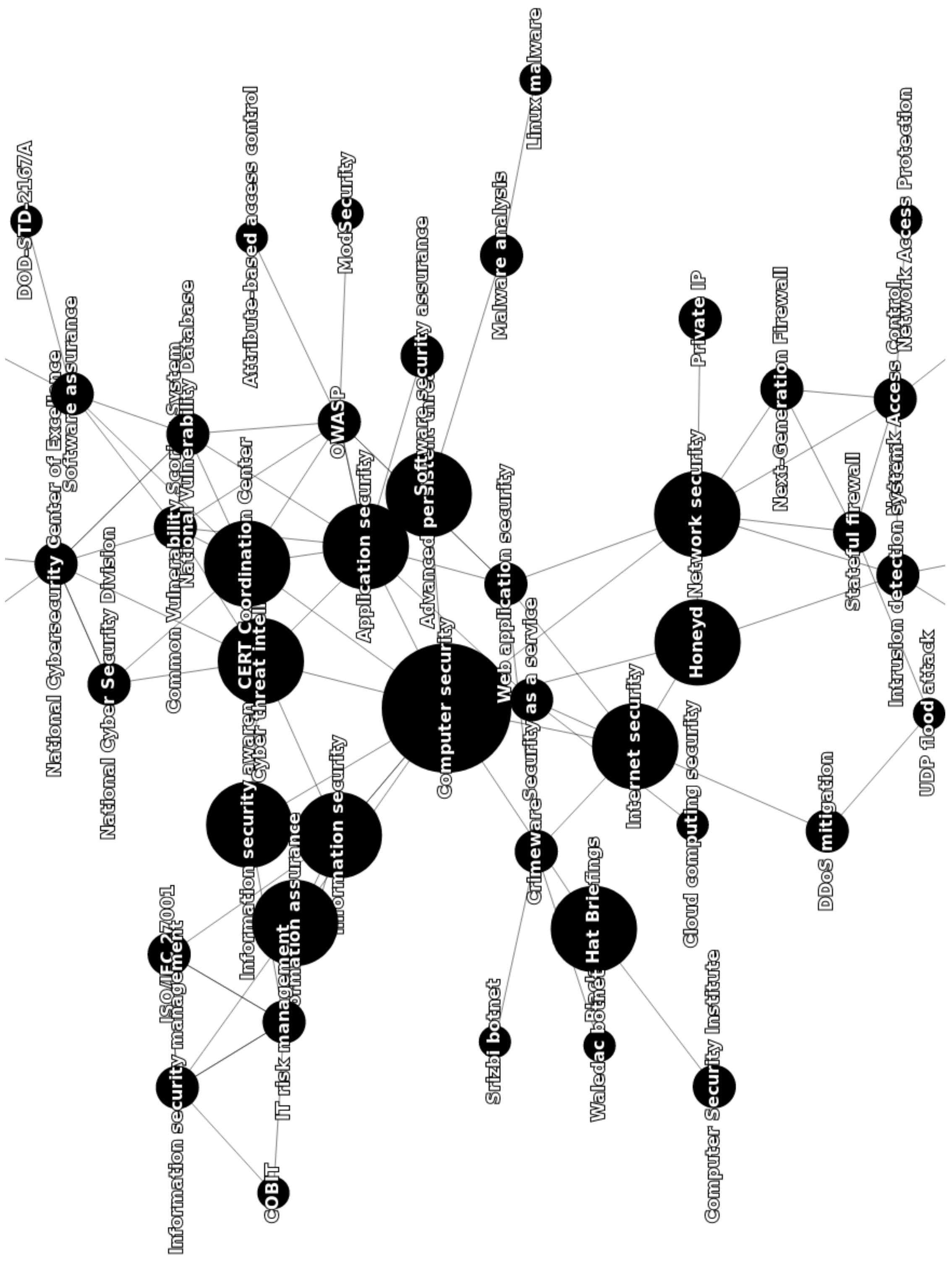


Figure 4.1: CyberKG-Research: Sample Visualized Knowledge Graph

In this graph, each circle represents a cybersecurity term from Wikipedia pages, and each link represents a relation with highest similarities (only show up when the similarity distance is higher than a user-defined threshold, e.g., 0.8). The graph represents the relations among different terminologies with ordered level, i.e., the node size from big to small indicates the level from up to down, respectively, as shown in Figure 4.2. Take 'Network security' as an example, it is related to 'Stateful firewall' and further 'Stateful firewall' links to 'Network Access Control', 'Next-Generation Firewall' and 'UDP flood attack'.The color of the node is made random, however, the size of each node is decided based on the count of directly connected nodes. Each link is associated with the cosine similar distance (though not visible in the graph), e.g., cosine similar distance between 'Stateful firewall' and 'Network security' is 0.8192, which represents relation discovered in terms of similarity between the two terminologies.

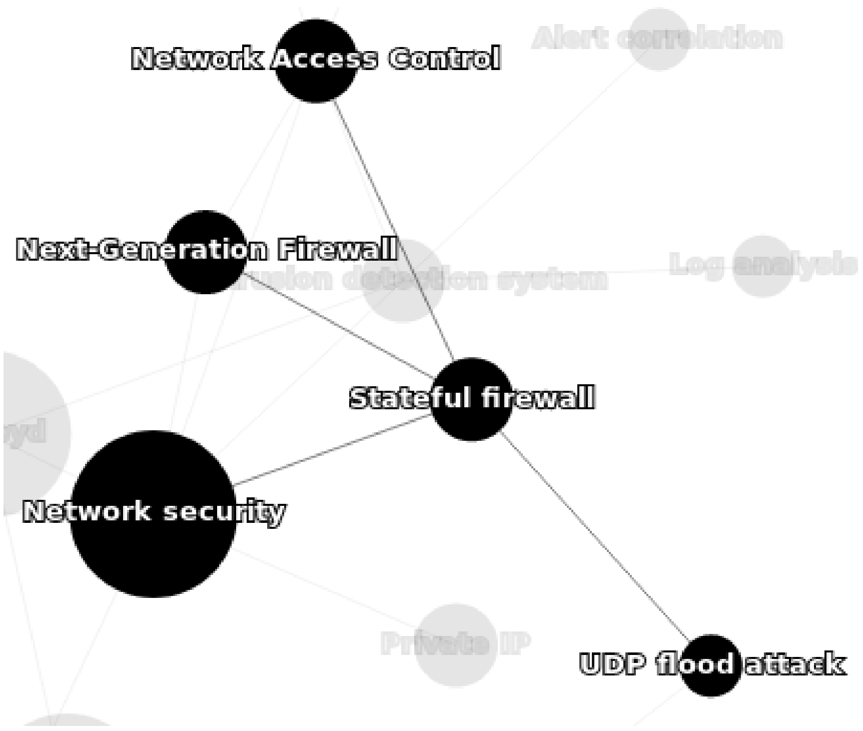


Figure 4.2: CyberKG-Research: Sample graph showing the relation among terminologies

In addition, we represented our graph data as a force-directed graph, then add more interactive functions:

1. Mouse Hangover function to show each node's terminology and gray out unlined nodes. The sample graph is given in Figure 4.3.

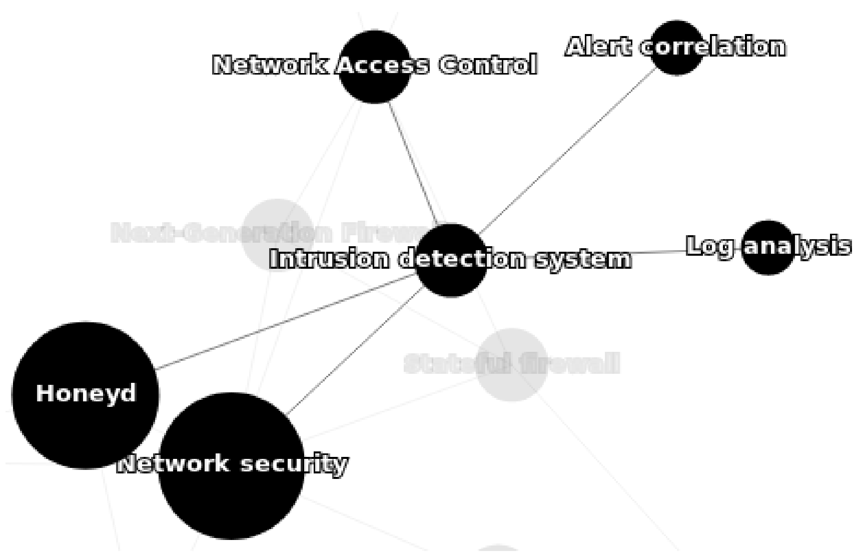
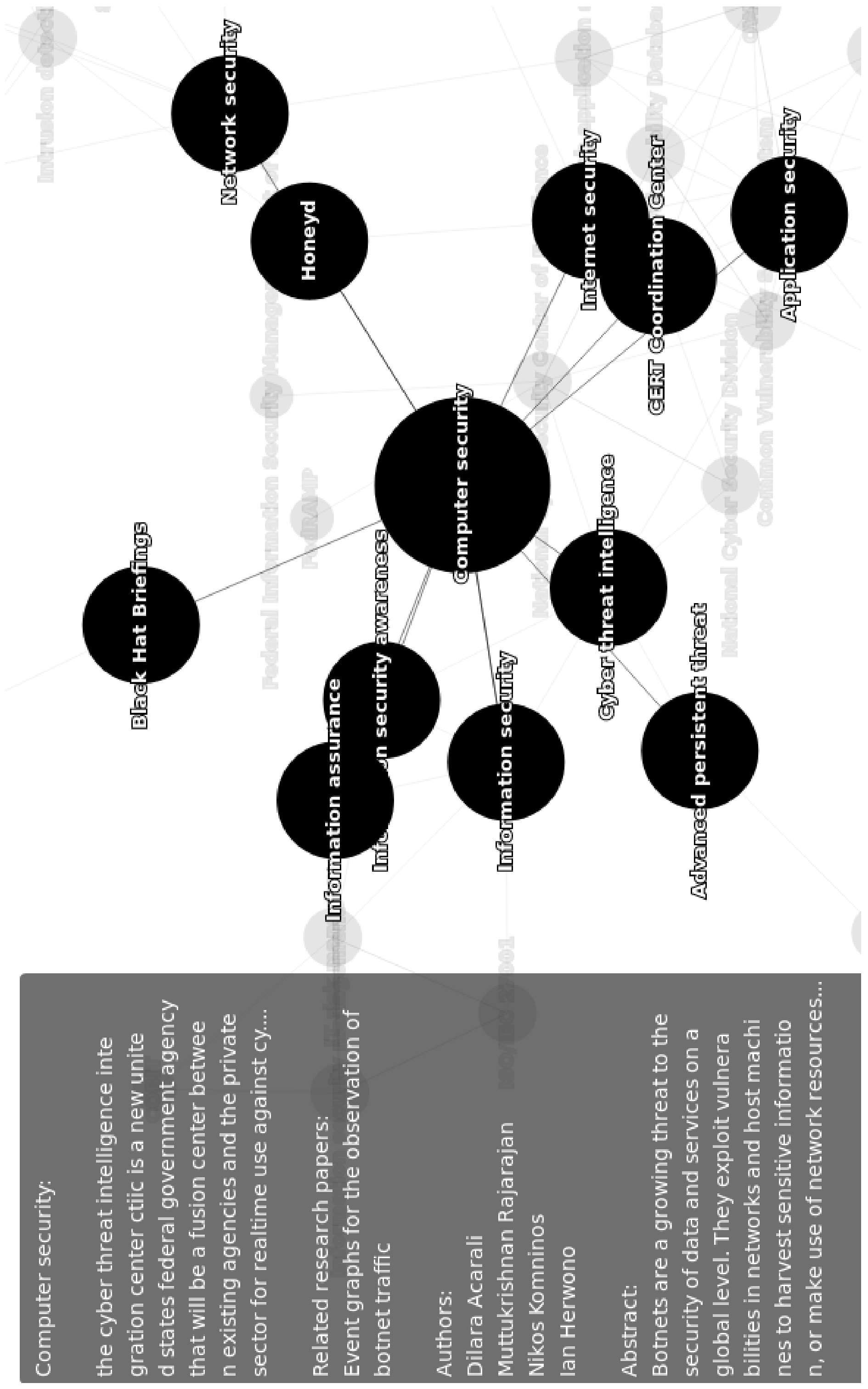


Figure 4.3: CyberKG-Research: A sample of the KG showing linked nodes

2. Mouse Click function to redirect to the corresponding Wikipedia page of the node to browse more detailed information.

3. Drag and drop function to re-arrange/resize the graph based on the user preference.



Network security
Honeyd
Internet security
CERT Coordination Center
Application security
Computer security
Black Hat Briefings
Information security awareness
Information assurance
Information security
Cyber threat intelligence
Advanced persistent threat
Intrusion detection
Federal Information Security Management
National Cyber Security Division
Common Vulnerability Scoring

Computer security:

the cyber threat intelligence inte
gration center ctiic is a new unite
d states federal government agency
that will be a fusion center betwee
n existing agencies and the private
sector for realtime use against cy...

Related research papers:
Event graphs for the observation of
botnet traffic

Authors:
Dilara Acarali
Muttukrishnan Rajarajan
Nikos Komninos
Ian Herwono

Abstract:
Botnets are a growing threat to the
security of data and services on a
global level. They exploit vulnera
bilities in networks and host machi
nes to harvest sensitive informatio
n, or make use of network resources...

Figure 4.4: CyberKG-Research: A subset of our knowledge graph showing most related reading research article.

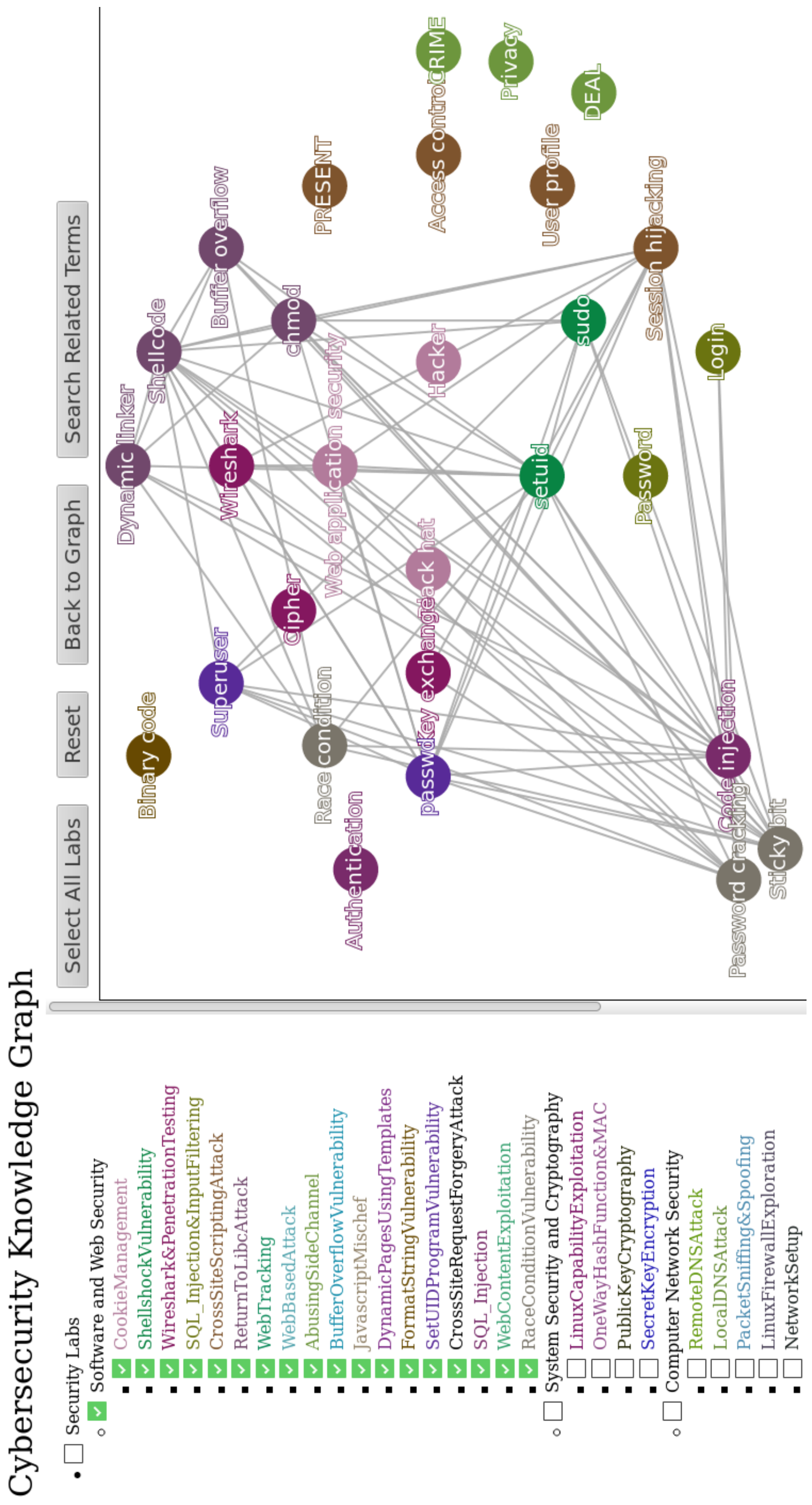
Figure 4.4 shows the visualization result when a user enlarges the graph and hover mouse over 'Computer security' nodes. The first ordered nodes related to 'Computer security' have been connected and shown with different colors in the graph. The combined information - part of terminology descriptions from Wikipedia and the related research article title, authors and truncated article abstract will be shown on the left side of the node for users to briefly go through the basic information about 'Computer Security'.

4.2 CyberKG-Lab (KG for Cybersecurity Online Lab - ThothLab)

To build the KG for ThtothLab, we have designed and developed a toolkit by using Python to extract keywords for each lab by matching the lab descriptions with extracted concepts from the computer security category (5,265 concepts from the third level of computer security category are used) at first. For example, the keywords extracted from the "Buffer Overflow Vulnerability Lab" (one of the labs available in the ThothLab repository) include "Dynamic linker", "Shellcode", "Buffer overflow", "chmod", "setuid" and "Password". Some of these concepts, like Password, setuid are not directly related to "Buffer Overflow Vulnerability", but are necessary knowledge for each student to finish the lab successfully. Instructors may also edit these concepts before adding them to the lab repository if they think some important concepts were skipped by our system.

With the KG represents in a graph data structured, the next step is to represent the graph in an interactive GUI to empower instructor and students who are actively using it. Since the virtual lab platform itself is a pure web-based lab environment, we want to integrate our CyberKG system into the Web UI seamlessly. We utilized JavaScripts and Echarts (a web-based visualization library that features a plethora of API's) to create interactive and dynamic contents on the web. In this KG, we applied threshold 0.8 (only show up when the similarity distance of two keywords is higher than 0.8) and visualized our graph using different ways as described below.

1. There are 48 labs in the current ThothLab system and can be classified by Security Labs, Software Labs, Network Labs, System Labs and Cloud Computing Labs [39]. We randomly set 48 colors for each lab and the color as well as the corresponding nodes (keywords) in each lab. The graph section is blank at the beginning. Users can select labs to learn the key concepts which are covered in the corresponding labs. The CyberKG-Lab will be presented to the user as shown in Figure 4.5.



Cybersecurity Knowledge Graph

- ☐ Security Labs
  - ✓ Software and Web Security
    - ✓ CookieManagement
    - ✓ ShellshockVulnerability
    - ✓ Wireshark&PenetrationTesting
    - ✓ SQL_Injection&InputFiltering
    - ✓ CrossSiteScriptingAttack
    - ✓ ReturnToLibcAttack
    - ✓ WebTracking
    - ✓ WebBasedAttack
    - ✓ AbusingSideChannel
    - ✓ BufferOverflowVulnerability
    - ✓ JavascriptMischef
    - ✓ DynamicPagesUsingTemplates
    - ✓ FormatStringVulnerability
    - ✓ SetUIDProgramVulnerability
    - ✓ CrossSiteRequestForgeryAttack
    - ✓ SQL_Injection
    - ✓ WebContentExploitation
    - ✓ RaceConditionVulnerability
  - ☐ System Security and Cryptography
    - ☐ LinuxCapabilityExploitation
    - ☐ OneWayHashFunction&MAC
    - ☐ PublicKeyCryptography
    - ☐ SecretKeyEncryption
  - ☐ Computer Network Security
    - ☐ RemoteDNSAttack
    - ☐ LocalDNSAttack
    - ☐ PacketSniffing&Spoofing
    - ☐ LinuxFirewallExploration
    - ☐ NetworkSetup

Figure 4.5: Sample Visualized CyberKG-Lab: User select all labs under Software and Web Security category

In addition, the shortcut key button - ”Select All Labs” has been created for selecting all labs in ThothLab system. When the user clicks this button, all keywords from 48 labs will be shown in Figure 4.6. Note that if labs contain common keywords, the color of the first lab will be applied to the corresponding node. The graph will be updated based on the user selection of labs (select more or unselect selected labs). By clicking on the ”Reset” button, current selections will be cleared and graph section is back to blank as initialized.

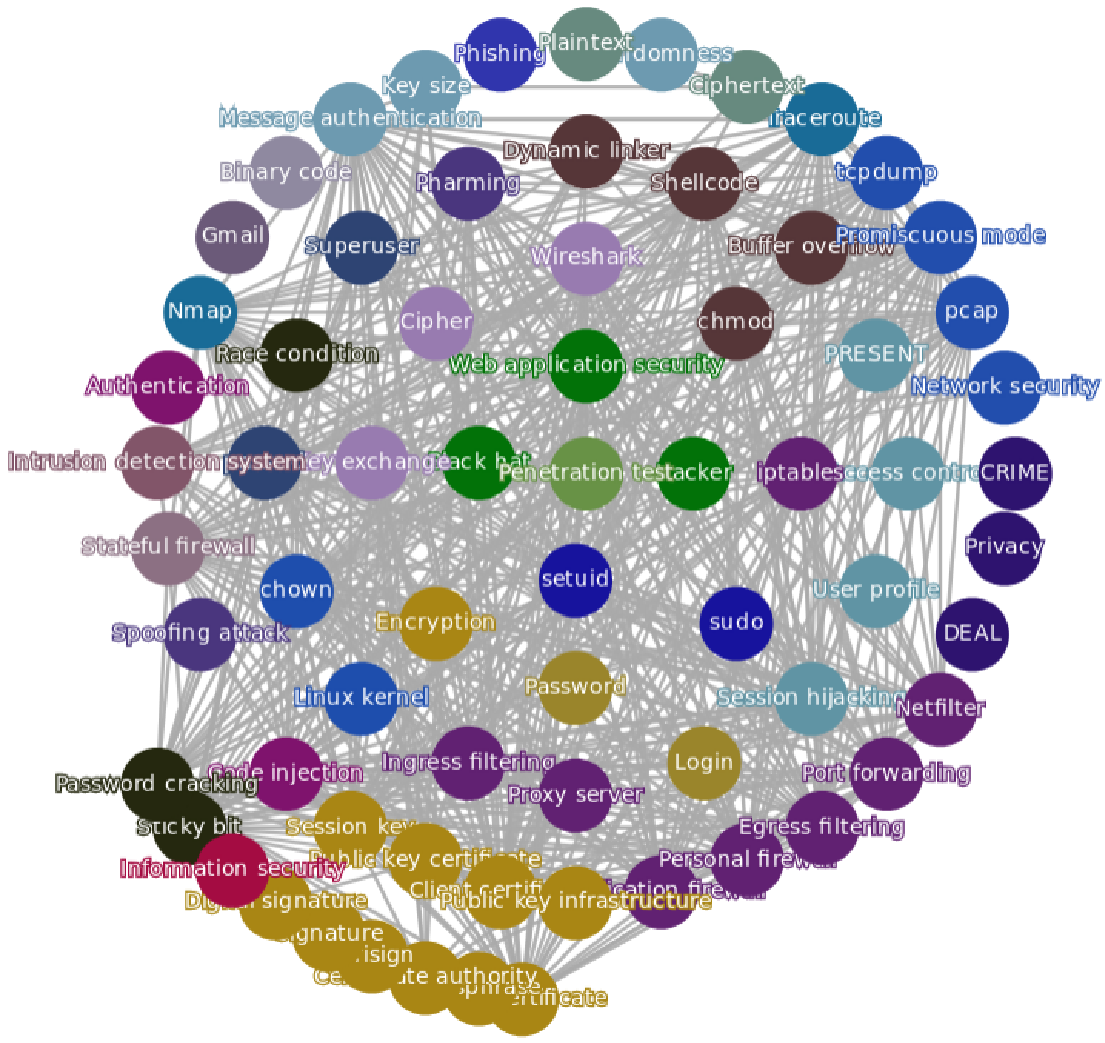


Figure 4.6: Sample Visualized CyberKG-Lab: User select all labs in ThothLabs System

2. The user may zoom in/out the graph and hover mouse onto nodes in the graph to highlight nodes' neighborhood (as well as the first level children nodes) and gray out unconnected nodes, as shown in Figure 4.7. At the same time, the similarity between two words will be shown on the corresponding links.

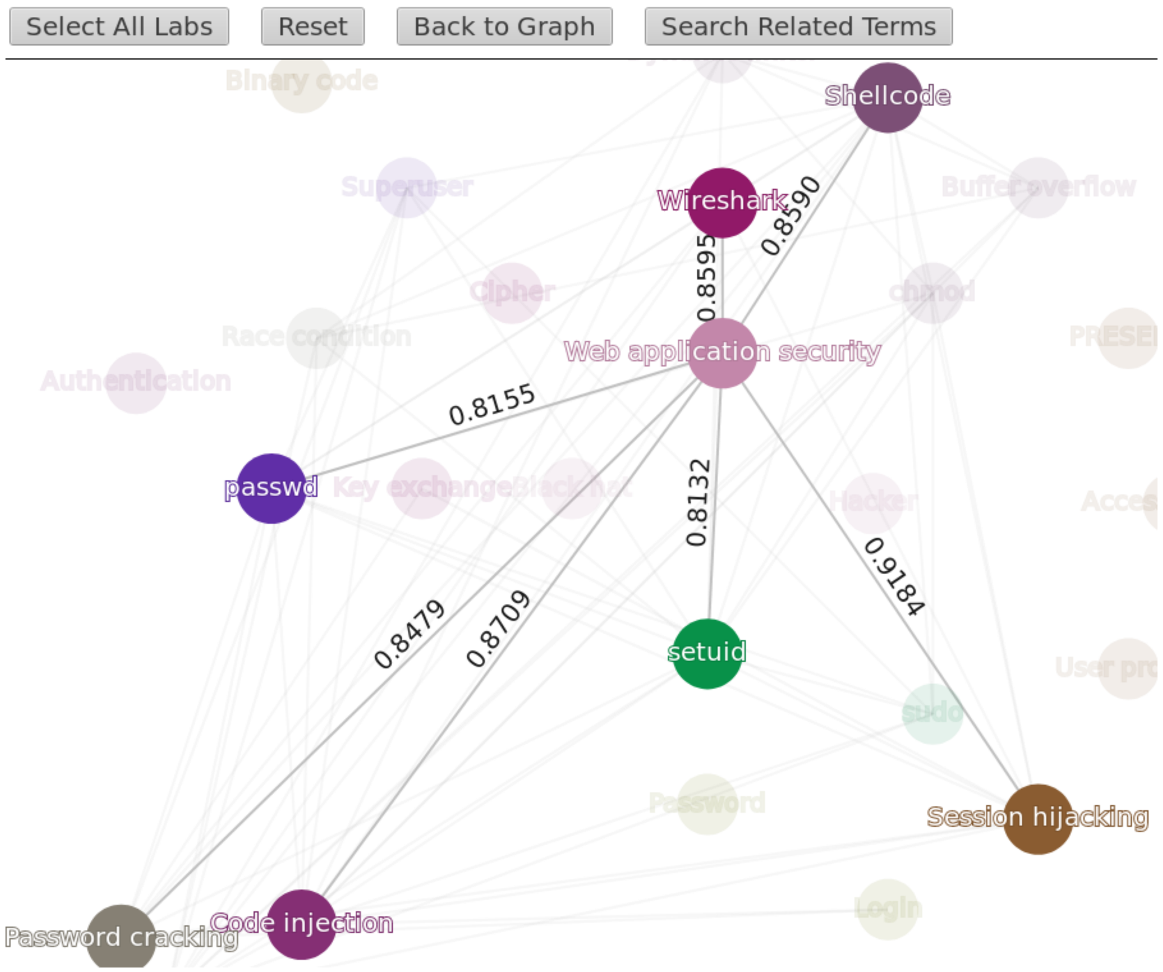
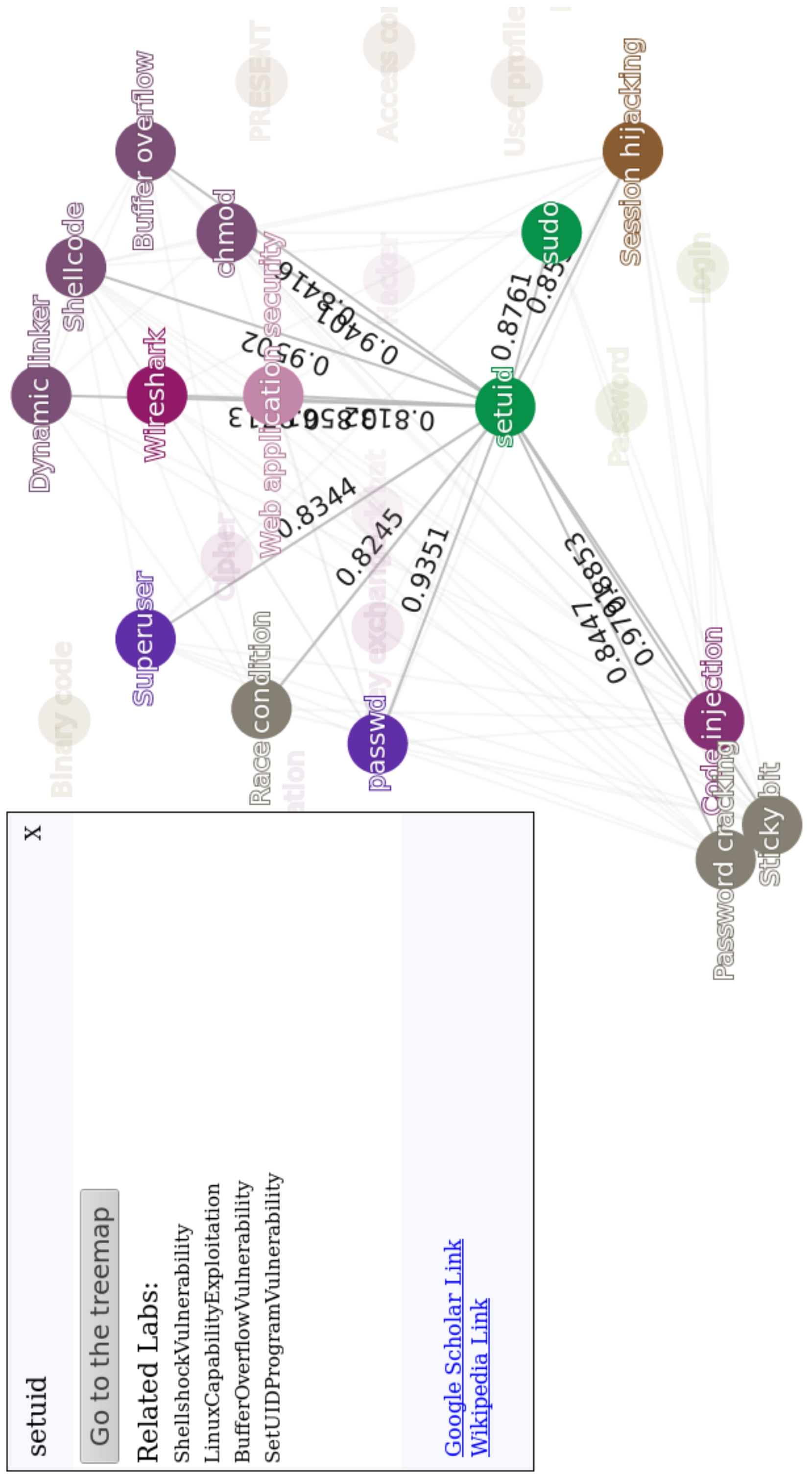


Figure 4.7: Sample Visualized CyberKG-Lab: Hang Over 'Web application security' node and highlight its neighborhood nodes

3. The user can click on one node to learn more details of the node. The related labs of the node, responding Google Scholar and Wikipedia link will be shown in Figure 4.8. By clicking on the link, the web page will be redirected to the corresponding Google Scholar Page or Wikipedia page of the node to browse more detailed information. The node's information window can be dragged or closed.



X

setuid

Go to the treemap

Related Labs:
ShellshockVulnerability
LinuxCapabilityExploitation
BufferOverflowVulnerability
SetUIDProgramVulnerability

Google Scholar Link
Wikipedia Link

Figure 4.8: Sample Visualized CyberKG-Lab: Click 'setuid' node to display corresponding infomration

4. After clicking on one node, there is a button "Go to the treemap's" shown in Figure 4.8. By clicking on this button, the selected node will become the root to generate an updated treemap as shown in Figure 4.9.

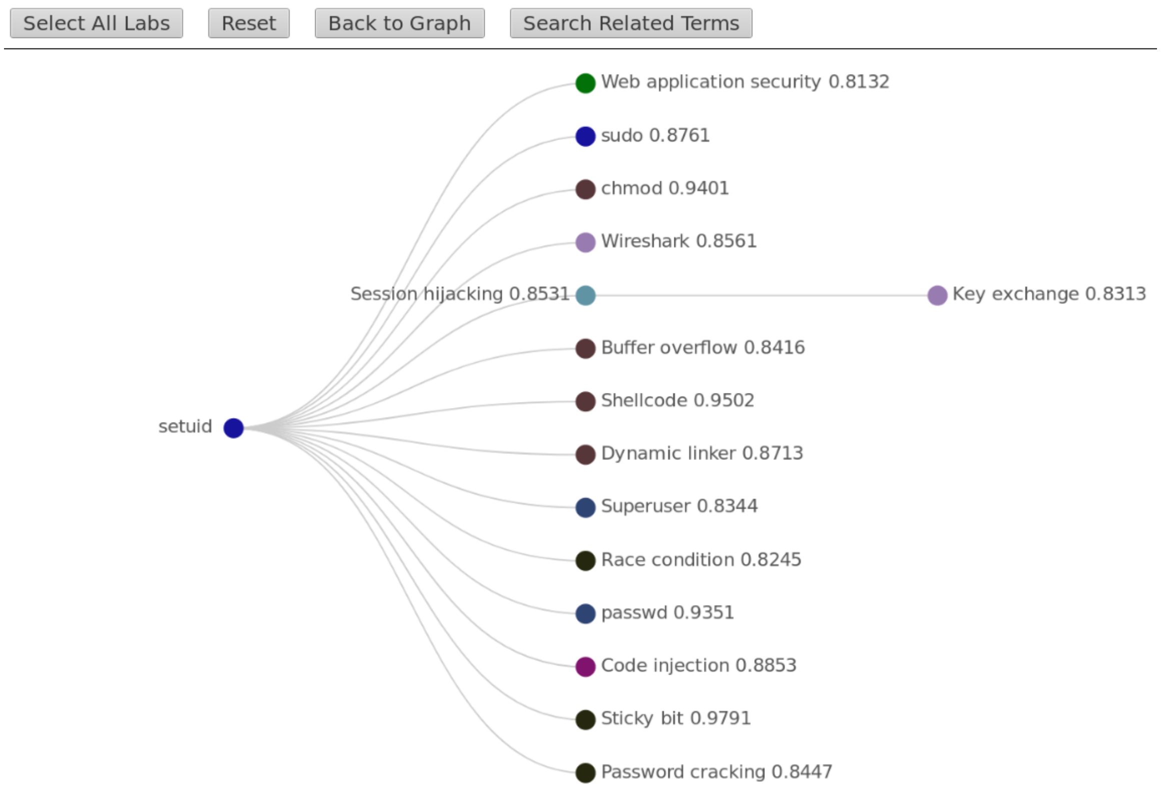


Figure 4.9: Sample Visualized CyberKG-Lab: Tree map of node 'setuid'

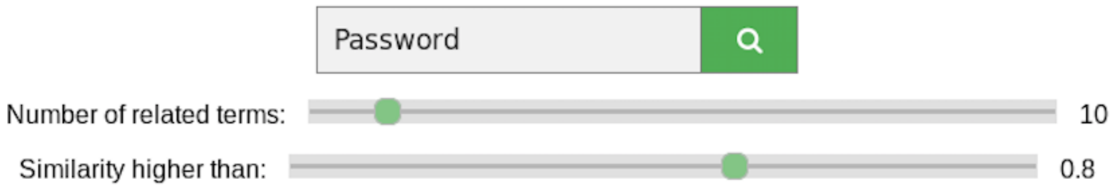
To avoiding a child node being displayed for more than once, this treegraph generated by using Breadth First Search (BFS) algorithm as shown in 1 on the previous undirected graph (data) which only contains nodes and relations.

Algorithm 1 Breadth First Search(BFS) algorithm

```
1: procedure BFS($G, s$)        ▷ G is the undirected graph and s is the source node
2:     create a queue Q
3:     Q.enqueue(s)                                          ▷ Adding s in queue
4:     mark s as visited
5:     while Q is non-empty do
6:         u = Q.dequeue()                           ▷ removing the head u from Q
7:         for all neighbours n of u in Graph G do                ▷ using all edges
8:             if n is not visited then
9:                 Q.enqueue(n)        ▷ adding unvisited n in Q to further visits its
   neighbour
10:                mark n as visited
```

The leaves (children nodes) in this treemap can be expanded or collapsed. The treemap also will be updated based on the user selection of labs like the graph (Note that the treemap may be blank for unselecting the labs of current root exist). Click on the ”Back to Graph” button to redirect to the previous graph.

5. By clicking on the ”Search Related Terms” button, user will be directed to the new page for search and learn any term’s relation. In this page as shown in Figure 4.10, enter a term ’Password’ in the search box and drag the slider to set 10 as the number of related terms(range: 1 to 100) and set 0.8 as the expect similarity (similarity range from 0.5 to 1).



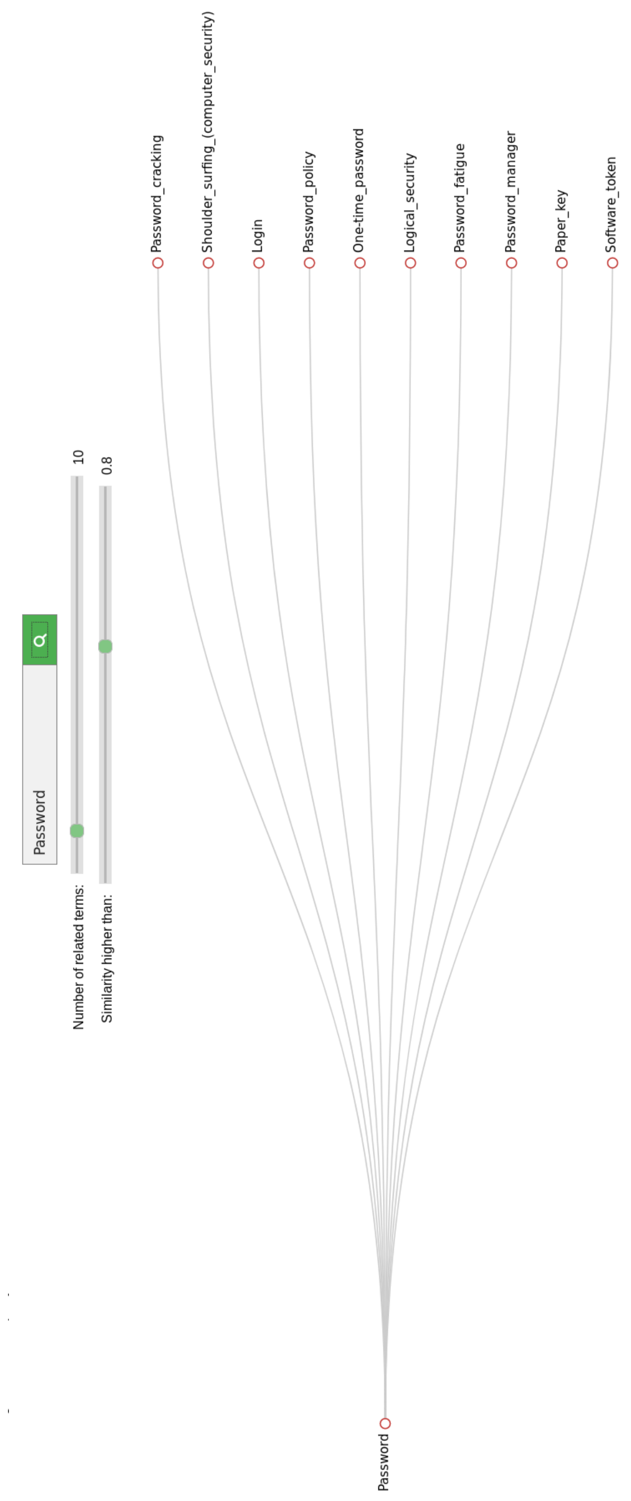
Password

Number of related terms: 10

Similarity higher than: 0.8

Figure 4.10: Sample Visualized CyberKG-Lab: Searching relate terms

After finishing the above settings then click on search button, the result will be displayed as shown in the Figure 4.11.


Password

Number of related terms: 10

Similarity higher than: 0.8

Figure 4.11: Sample Visualized CyberKG-Lab: Searching 'Password' related terms

Same as the previous treemap, the related nodes in this search result tree can be expanded or collapsed based on the user preference. Expanding one of the nodes is shown in Figure 4.12.

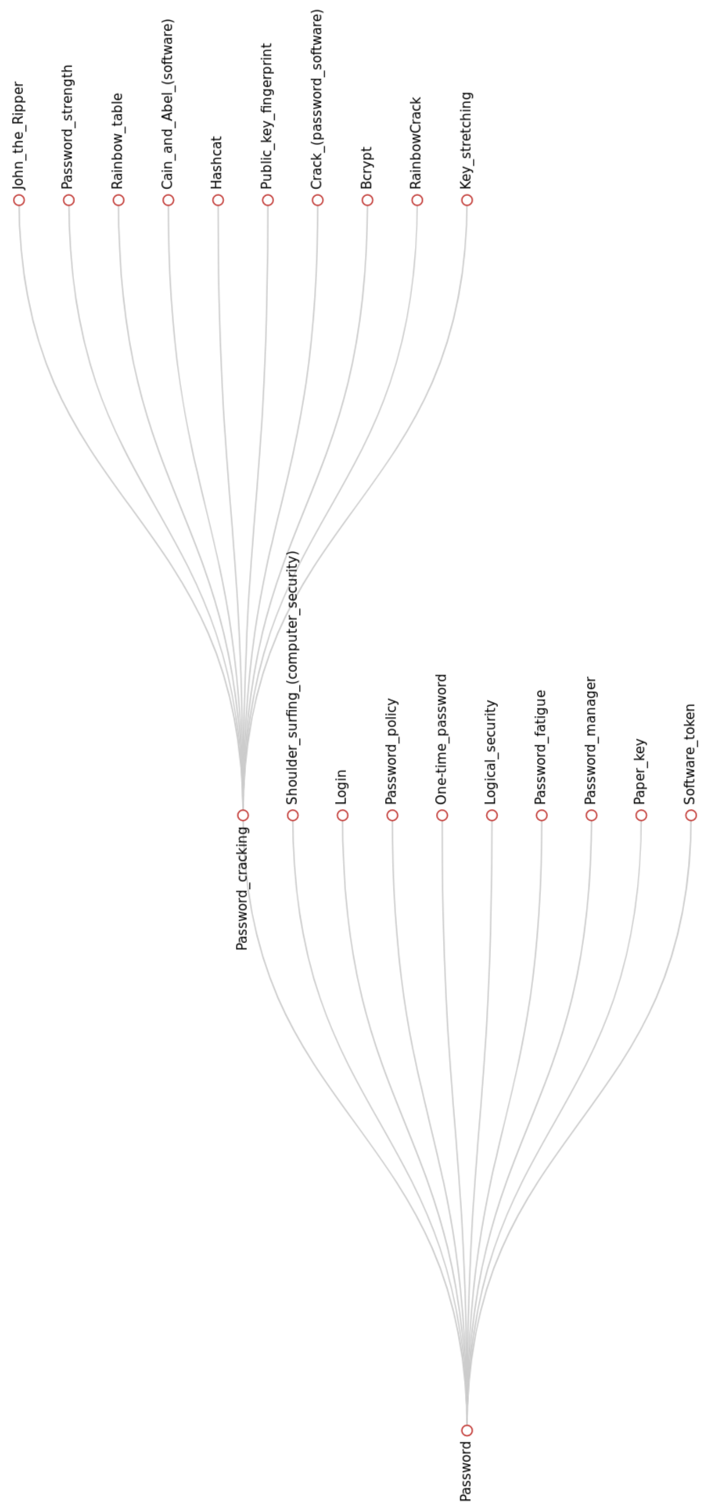


Figure 4.12: Sample Visualized CyberKG-Lab: Expanding related node 'Password cracking' in 'Passowrd' related terms search result

In addition, the relationship of the nodes (root and children nodes) similarity will be shown by hovering mouse on the node as shown in Figure 4.13. Note that the result of related node similarity is sorted from high to low as well as from up to down while generating the search treemap.

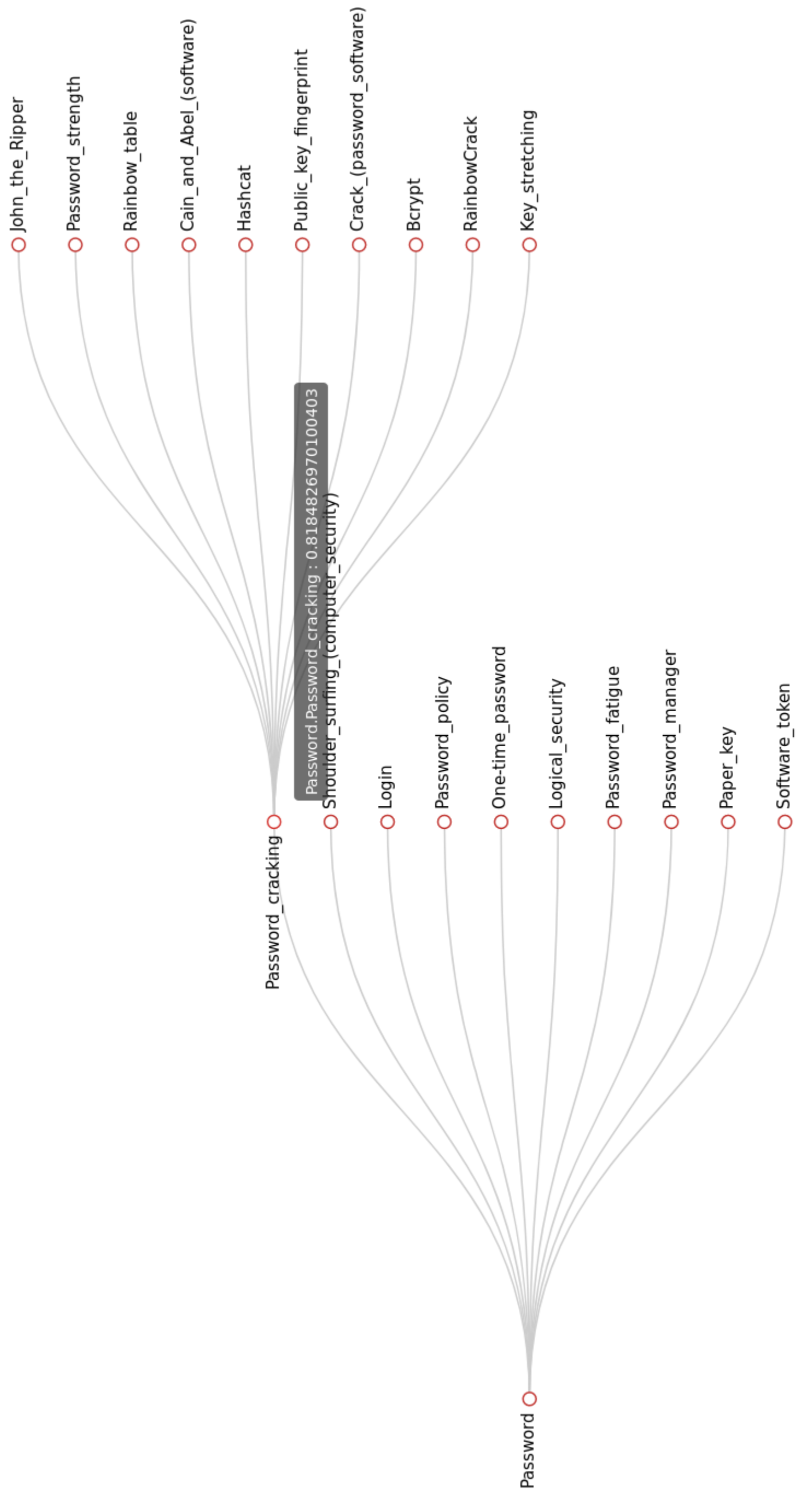


Figure 4.13: Sample Visualized CyberKG-Lab: Hovering mouse on the 'Password cracking' node

Chapter 5

CONCLUSIONS AND DISCUSSIONS

5.1 Conclusions

In this research work, we described and discussed in details how we construct the KG in cybersecurity education to represent the concepts and their relationships, in addition, to recommend related research articles. We have applied two datasets by using different approaches to process the Wikipedia database dump - (1) only use the anchors from Wikipedia and (2) use full article contexts in Wikipedia for concepts embedding. We implemented CBOW model and Skip-Gram model which are from Word2Vec to train above approaches. We have verified that large corpus size does not always have better performance in some tasks. The completed results have been shown and discussed in section 3.3 of Chapter 3. Several flexible toolkits have been developed for data scraping, data preprocessing and graph construction. In addition, we have defined several test datasets based on our own judgment which contain similar word pairs of cybersecurity domain for comparing and evaluating different models. Besides constructing the cybersecurity KG's, we have visualized the generated graph with its most related articles recommendation by using semantic similarity and a visualized graph for cybersecurity online learning lab.

By these cybersecurity knowledge graphs developed, students and lecturers are able to review the specific knowledge structure without learning required courses or working on hands-on labs. They have a general guidance and are able to dig more related topics/ materials according to their interests.

5.2 Discussions

Our current KG generation module relies on Word2Vec model to represent terminologies with vectors and uses these vectors to calculate the cosine similarity. However, there are known limitations in this task: due to the hidden layer being removed in the Word2Vec, the word order information in CBOW model and Skip-Gram model is mostly neglected. This simplification adopted by these models leads to more efficient in computational time, but may reduce the ability of semantics capture. Another challenge in this research is to evaluate and validate the training results in cybersecurity area. In English language domain, there are several datasets are defined by human experts for evaluating phrase/word similarity and phrase/word analogy, e.g. phrase analogy dataset which contains 3,218 analogy questions defined in [2], Rubenstein and Goodenough dataset(RG-65) as the classical word similarity dataset [40] and WordSim353 as the test collection for measuring word similarity or relatedness [41]. These datasets are widely used as evaluation baseline for NLP processing modules in English language domain but no work exists for such in cybersecurity domain. Although we have assigned several datasets based on own judgments for models evaluation in cybersecurity area, it is still far from completion to serve as baseline purpose.

Chapter 6

FUTURE WORK

In this section, we will discuss the possibility future works on current work improvement and extension.

Improvement

In future work, we plan to study how to incorporate more unstructured data into our system, including but not limited to textbooks, internet web pages, and online video transcripts. We also plan to incorporate cybersecurity ontology which is intended to support our knowledge graph generation. By adding ontology in CyberKGs, edges in our knowledge graph will get the semantic definition which is much more useful than the similarity value we currently used. Our ultimate goal is to build a knowledge base that will serve as the core of the cybersecurity domain, which would evolve and grow with additional cybersecurity data sets as they become available.

Extension

Multiple flexible toolkits have been designed and developed in this research work, we can base our current work then improve extend to other areas for improving the knowledge learning.

REFERENCES

[1] Demand to fill cybersecurity jobs booming. URl: http://peninsulapress.com/2015/03/31/cybersecurity-jobs-growth/.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[3] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.

[4] Wikipedia. URl: https://www.wikipedia.org.

[5] Ehsan Sherkat and Evangelos E Milios. Vector embedding of wikipedia concepts and entities. In *International Conference on Applications of Natural Language to Information Systems*, pages 418–428. Springer, 2017.

[6] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[7] Ieee xplore digital library. URl:http://ieeexplore.ieee.org/Xplore/home.jsp.

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

[9] C.J. Chung D. Huang Y. Deng, D. Lu and Z. Zeng. Personalized learning in a virtual hands-on lab platform for computer science education. In *2018 IEEE Frontiers in Education Conference (FIE) Proceedings*, Oct 2018.

[10] I. Hsiao D. Huang Z. Zeng, Y. Deng and C.J. Chung. Improving student learning performance in a virtual hands-on lab system in cybersecurity education", in proceedings of ieee frontiers in education. In *2018 IEEE Frontiers in Education Conference (FIE) Proceedings*, Oct 2018.

[11] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*, 2013.

[12] Machine learning techniques for automatic ontology extraction from domain texts. URl: http://csc.lsu.edu/ jianhua/Ontology-learning.ppt.

[13] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[14] Jeffrey Undercoffer, Anupam Joshi, and John Pinkston. Modeling computer attacks: An ontology for intrusion detection. In *International Workshop on Recent Advances in Intrusion Detection*, pages 113–135. Springer, 2003.

[15] Nvd.nist.gov. URl: https://nvd.nist.gov/.

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

[17] Michael Iannacone, Shawn Bohn, Grant Nakamura, John Gerth, Kelly Huffer, Robert Bridges, Erik Ferragut, and John Goodall. Developing an ontology for cyber security knowledge graphs. In *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, page 12. ACM, 2015.

[18] Chen Yinong. Service-oriented computing and system integration: Software, iot, big data, and ai as services. 2017.

[19] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[20] word2vec. URl: [15] https://code.google.com/archive/p/word2vec/.

[21] David Milne and Ian H Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM, 2008.

[22] Chen-Tse Tsai and Dan Roth. Cross-lingual wikification using multilingual embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 589–598, 2016.

[23] Gregory Grefenstette and Lawrence Muchemi. Determining the characteristic vocabulary for a specialized dictionary using word2vec and a directed crawler. *arXiv preprint arXiv:1605.09564*, 2016.

[24] Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451, 2017.

[25] Cataldo Musto, Giovanni Semeraro, Marco de Gemmis, and Pasquale Lops. Learning word embeddings from wikipedia for content-based recommender systems. In *European Conference on Information Retrieval*, pages 729–734. Springer, 2016.

[26] URl: https://en.wikipedia.org/wiki/Special:Export/FULLPAGENAME.

[27] Selenium. URl: https://www.seleniumhq.org/.

[28] Sans glossary security of terms. URl: https://www.sans.org/security-resources/glossary-of-terms/.

[29] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[30] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London:, 2014.

[31] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedi ngs of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[32] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[33] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 238–247, 2014.

[34] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[35] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[36] Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14, 2016.

[37] Charles Bouveyron and Camille Brunet-Saumard. Model-based clustering of high-dimensional data: A review. *Computational Statistics & Data Analysis*, 71:52–78, 2014.

[38] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1, 2009.

[39] Yuli Deng, Dijiang Huang, and Chun-Jen Chung. Thoth lab: A personalized learning framework for cs hands-on projects (abstract only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 706–706, New York, NY, USA, 2017. ACM.

[40] Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October 1965.

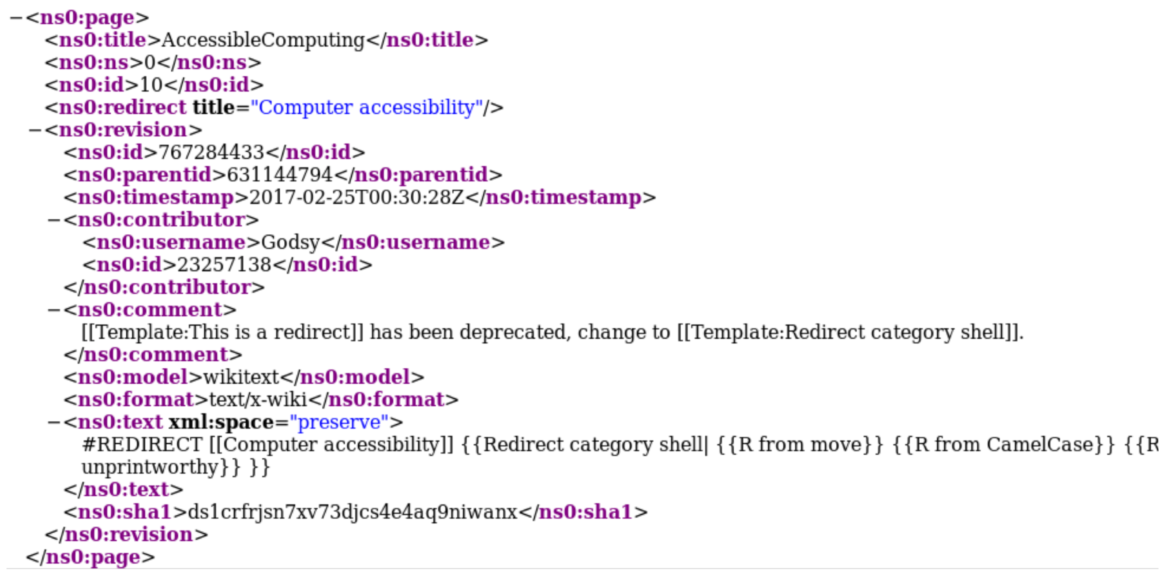
[41] The wordsimilarity-353 test collection. URl: http://www.cs.technion.ac.il/ gabr/resources/data/wordsim353/.

[42] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[43] Deqing Li, Honghui Mei, Yi Shen, Shuang Su, Wenli Zhang, Junting Wang, Ming Zu, and Wei Chen. Echarts: A declarative framework for rapid construction of web-based visualization. *Visual Informatics*, 2018.

APPENDIX A

TYPE OF WIKIPEDIA PAGES



```
−<ns0:page>
   <ns0:title>AccessibleComputing</ns0:title>
   <ns0:ns>0</ns0:ns>
   <ns0:id>10</ns0:id>
   <ns0:redirect title="Computer accessibility"/>
 −<ns0:revision>
     <ns0:id>767284433</ns0:id>
     <ns0:parentid>631144794</ns0:parentid>
     <ns0:timestamp>2017-02-25T00:30:28Z</ns0:timestamp>
   −<ns0:contributor>
       <ns0:username>Godsy</ns0:username>
       <ns0:id>23257138</ns0:id>
     </ns0:contributor>
   −<ns0:comment>
       [[Template:This is a redirect]] has been deprecated, change to [[Template:Redirect category shell]].
     </ns0:comment>
     <ns0:model>wikitext</ns0:model>
     <ns0:format>text/x-wiki</ns0:format>
   −<ns0:text xml:space="preserve">
       #REDIRECT [[Computer accessibility]] {{Redirect category shell| {{R from move}} {{R from CamelCase}} {{R
       unprintworthy}} }}
     </ns0:text>
     <ns0:sha1>ds1crfrjsn7xv73djcs4e4aq9niwanx</ns0:sha1>
   </ns0:revision>
 </ns0:page>
```

Figure A.1: Sample XML files in Wikipedia dump

List of types in Wikipedia dump has been removed in this work to acquire unique article pages as shown below:

1. Redirect: with <'ns0:redirect> tag in the xml file.
2. Help/Help talk: 'Help:'/'Help talk:' in the first part of page name
3. Category/Category talk: 'Category:'/'Category talk:' in the first part of page name
4. Template/Template talk: 'Template:'/'Template talk:' in the first part of page name
5. File/File talk: 'File:'/'File talk:' in the first part of page name
6. Topic/Topic talk: 'Topic:'/'Topic talk:' in the first part of page name
7. Mediwiki/Mediwiki talk: 'Mediwiki:'/'Mediwiki talk:' in the first part of page name

8. Portal/Portal talk: 'Portal:'/'Portal talk:' in the first part of page name
9. ListPage: with 'List of' in the first part of the page name
10. Wikipedia/Wikipediat talk: 'Wikipedia:'/'Wikipedia talk:' in the first part of page name
11. Book/book talk: 'Book:'/'Book talk:' in the first part of page name
12. Draft/Draft talk: 'Draft:'/'Draft talk:' in the first part of page name
13. Timetext/Timetext talk: 'Timetext:'/'Timetext talk:' in the first part of page name
14. Module/Module talk: 'Module:'/'Module talk:' in the first part of page name
15. Disambiguation: 1) with '(disambiguation)' in the page name 2)with 'may refer to:' or 'may also refer to' in the text file.
16. SmallPage: The Pages which have incoming links lower than 5 threshold.
17. NoneEnglishTitle: without redirect or alias
18. Education Program/Education Program talk: 'Education Program:'/Education Program talk:' in the first part of page name

APPENDIX B

DATA SAMPLES

Table B.1: Dataset Sample I: Concept pages from Wikipedia dataset with corresponding ID and Title

	FolderID:PageID,Title	PageID	Title
0	1:12,Anarchism	12	Anarchism
1	1:25,Autism	25	Autism
2	1:39,Albedo	39	Albedo
3	1:290,A	290	A
4	1:303,Alabama	303	Alabama

Table B.2: Dataset Sample II: Replace PageIDs for matched titles Wikipedia concept pages' description

	Format
0	anarchism is a 23040 that advocates 191161 soc...
1	autism is a 536032 characterized by troubles w...
2	albedo 1007667 albedo whiteness is the measure...
3	a 378194 plural as a s a s a s or aes is the f...
4	alabama is a 18618239 in the 179553 of the 343 ...

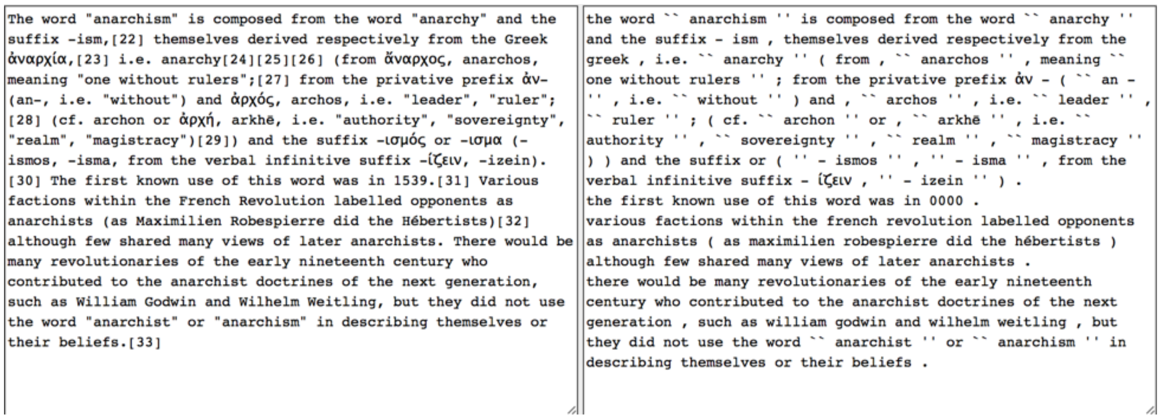
Table B.3: Dataset Sample III: Extract PageIDs only from Dataset Sample II

	Format
0	23040 191161 4228181 13993 26271818 28151 4558...
1	536032 161744 5177 4475349 2092692 3232713 177...
2	1007667 51331 44364 35553026 41644 35553026 277...
3	378194 3675310 32693 21440570 929 265914 17803...
4	18618239 179553 3434750 30395 48830 18933066 2...

Table B.4: Dataset Sample IV: Convert PageIDs in Dataset Sample III to corresponding titles

	Format
0	Political_philosophy Self_governance Stateless...
1	Developmental_disorder Interpersonal_relations...
2	Hispanic_and_Latino_Americans Dimensionless_qu...
3	English_alphabet Letter_(alphabet) Vowel Iso_b...
4	U.S._state Southern_United_States United_State...

Figure B.1: Data Sample V: Processed with Stanford CoreNLP



```
The word "anarchism" is composed from the word "anarchy" and the
suffix -ism,[22] themselves derived respectively from the Greek
ἀναρχία,[23] i.e. anarchy[24][25][26] (from ἄναρχος, anarchos,
meaning "one without rulers";[27] from the privative prefix ἀν-
(an-, i.e. "without") and ἀρχός, archos, i.e. "leader", "ruler";
[28] (cf. archon or ἀρχή, arkhē, i.e. "authority", "sovereignty",
"realm", "magistracy")[29]) and the suffix -ισμός or -ισμα (-
ismos, -isma, from the verbal infinitive suffix -ίζειν, -izein).
[30] The first known use of this word was in 1539.[31] Various
factions within the French Revolution labelled opponents as
anarchists (as Maximilien Robespierre did the Hébertists)[32]
although few shared many views of later anarchists. There would be
many revolutionaries of the early nineteenth century who
contributed to the anarchist doctrines of the next generation,
such as William Godwin and Wilhelm Weitling, but they did not use
the word "anarchist" or "anarchism" in describing themselves or
their beliefs.[33]
```

```
the word `` anarchism '' is composed from the word `` anarchy ''
and the suffix - ism , themselves derived respectively from the
greek , i.e. `` anarchy '' ( from , `` anarchos '' , meaning ``
one without rulers '' ; from the privative prefix ἀν - ( `` an -
'' , i.e. `` without '' ) and , `` archos '' , i.e. `` leader '' ,
`` ruler '' ; ( cf. `` archon '' or , `` arkhē '' , i.e. ``
authority '' , `` sovereignty '' , `` realm '' , `` magistracy ''
) ) and the suffix or ( '' - ismos '' , '' - isma '' , from the
verbal infinitive suffix - ίζειν , '' - izein '' ) .
the first known use of this word was in 0000 .
various factions within the french revolution labelled opponents
as anarchists ( as maximilien robespierre did the hébertists )
although few shared many views of later anarchists .
there would be many revolutionaries of the early nineteenth
century who contributed to the anarchist doctrines of the next
generation , such as william godwin and wilhelm weitling , but
they did not use the word `` anarchist '' or `` anarchism '' in
describing themselves or their beliefs .
```

APPENDIX C

OPEN SOURCE LIBRARIES

The following open source libraries are applied in this research work:

- Wikiextractor: www.github.com/attardi/wikiextractor
- Mwparserfromhell: www.github.com/earwig/mwparserfromhell
- Gensim[34]: https://radimrehurek.com/gensim
- Matplotlib[42]: https://matplotlib.org
- ECHARTS[43]: https://ecomfe.github.io/echarts-doc/public/en/index.html