

Identification of Cybersecurity Specific Content Using Different Language Models

OTGONPUREV MENDSAIKHAN^{1,a)} HIROKAZU HASEGAWA² YUKIKO YAMAGUCHI³ HAJIME SHIMADA³
ENKHBOLD BATAA⁴

Received: October 31, 2019, Accepted: June 1, 2020

Abstract: Given the sheer amount of digital texts publicly available on the Internet, it becomes more challenging for security analysts to identify cyber threat related content. In this research, we proposed to build an autonomous system to identify cyber threat information from publicly available information sources. We examined different language models to utilize as a cybersecurity-specific filter for the proposed system. Using the domain-specific training data, we trained Doc2Vec and BERT models and compared their performance. According to our evaluation, the BERT-based Natural Language Filter is able to identify and classify cybersecurity-specific natural language text with 90% accuracy.

Keywords: cyber threat, NLP, Text-Classification

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 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. However, due to the increase in the number of information systems used in security operations centers, extraction of the organization-specific cyber threat related information from the public Internet becomes necessary for correlation. For this reason, we are proposing an autonomous system that employs Natural Language Processing techniques to identify the cyber threat related information and filter out the irrelevant content.

In our earlier work, we proposed filtering security related text documents using the keyword generation method [1]. However, after several experiments, we concluded 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, computational linguistics represent text in vector space to capture and calculate the semantic relationships between words [2]. The process of converting text into numerical vectors is called embed-

ding. Earlier embedding techniques such as Bag of Words (BoW) or Term Frequency Inverse Document Frequency (TFIDF) have been preceded by neural network embedding techniques to represent the words in vector space. Mikolov et al. [3] 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 [4] that could represent the documents in vector space using a similar approach. In our previous work [5], we 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 [6], ULMFiT [7] and transformer-based models such as BERT [8] 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.

In this paper we propose to utilize BERT, which achieved state-of-the-art performance on 11 NLP tasks, to classify cybersecurity-specific text documents and compare them with our previous study that used Doc2Vec.

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

The specific contributions of the paper are as follows:

- (1) We propose an architecture of the autonomous system to identify cyber threat related content from the massive amount of publicly available textual documents.
- (2) With the cybersecurity-specific training data and custom pre-processing, the Doc2Vec model has been trained to work as a domain-specific language filter for the autonomous system.
- (3) With the same data, experiments have been conducted to test if the BERT language model can act as a domain-specific language filter.

¹ Nagoya University, Graduate School of Informatics, Nagoya, Aichi 464-8603, Japan

² Nagoya University, Information Security Office, Nagoya, Aichi 464-8601, Japan

³ Nagoya University, Information Technology Center, Nagoya, Aichi 464-8601, Japan

⁴ ExaWizards Inc., Minato, Tokyo 105-0013, Japan

^{a)} ogo@net.itc.nagoya-u.ac.jp

The remainder of this paper is organized as follows. Section 2 will review similar works and how this paper differs in its approach. In Section 3, we will propose an autonomous system to identify cyber threat related information. In Section 4, the datasets to be used as training and the test for the experiments will be discussed. In Sections 5 and 6, the experimental setup and the results of experiments conducted using Doc2Vec and BERT-based Natural Language Filters respectively will be detailed. Section 7 will discuss the differences between both models, and, finally, we will conclude in Section 8 by discussing the future possibilities to extend this research.

2. Related Work

There have been a number of attempts to identify or extract cyber threat related information from the Dark Web or any other publicly available information source automatically. Al-Rowaily et al. [9] 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. Our 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. Mulwad et al. [10] described a prototype system to extract information about security vulnerability from web text using an SVM classifier. We propose a deep learning algorithm using neural embedding for classification to achieve better performance. Joshi et al. [11] proposed a cybersecurity entity and concept spotter that uses the Stanford Named Entity Recognizer (NER), a Conditional Random Field (CRF) algorithm-based NER framework. That research focused on building the cybersecurity-specific vocabulary through linked data, whereas our approach aims to identify the cybersecurity-specific documents using different language models. More et al. [12] proposed a knowledge-based approach to intrusion detection modeling in which the intrusion detection system automatically fetches threat related information from web-based text information and proactively monitors the network to establish situational awareness. Their approach focused mainly on developing a cybersecurity ontology which can be understood by the intrusion-detecting machines. Our research focused on building an autonomous system that assists the human operators in terms of raising situational awareness. Dionísio et al. developed a system to detect cyber threats from Twitter using deep neural networks [13]. Their work has many similarities with our work, in that they collected a relevant threat from a Twitter feed and identified the assets through Named Entity Recognition from specific Twitter feeds. Our approach differs by identifying and extracting cybersecurity related documents from any source and then analyzing the relevance of it. 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 [14]. Their work highlighted the importance of domain specific embedding, which is demonstrated in our 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. [15] developed the Topic2Vec approach, which can learn topic representations in the same semantic vector space as used for words. Dhingra et al. [16] 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. [17] 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.

There have been attempts to utilize a Doc2Vec language model for the domain-specific task as well. Aman et al. [18] 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. [19] proposed a topic recommendation system using a Doc2Vec-based language model. The automated topic recommendation system based on the Doc2Vec model suggests that our proposed autonomous 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. Lee et al. described BioBERT, which was trained for unannotated biomedical text corpora [20]. BioBERT outperformed the generic BERT model on the biomedical text mining tasks. Similar results have been achieved by Beltagy et al. with SciBERT, a pre-trained language model for scientific text [21]. SciBERT has been pre-trained on 1.14M papers of various domains from Semantic Scholar and even outperformed BioBERT on biomedical tasks. Lee et al. described a patent classification system created by fine-tuning the BERT language model [22]. Their model outperformed the state-of-the-art results in classifying the patent claims. Huang et al. pre-trained and fine-tuned BERT on clinical notes to develop ClinicalBERT [23]. ClinicalBERT has outperformed baselines to predict the readmission of patients based on clinical notes. 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 [24]. The proposed VideoBERT model is used in various tasks including action classification and video captioning. Adhikari et al. presented DocBERT for document classification [25]. 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.

3. Proposed System

In our earlier work [1] we proposed a system to mine actionable threat intelligence from publicly available information sources with minimal supervision. This section will review the archi-

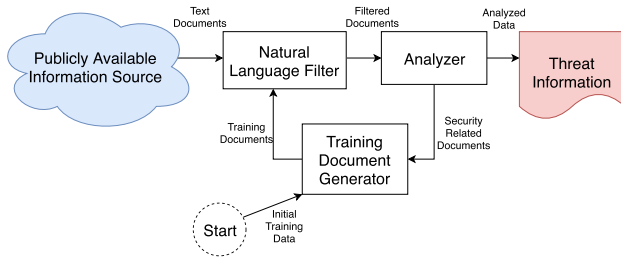


Fig. 1 Proposed system architecture.

ture of the system and our proposed solutions for its Natural Language Filter module.

3.1 Proposed Model of the Autonomous System

We envisioned a system that automatically identifies and analyzes cybersecurity threats to assist security analysts in building situational awareness with three modules including a Natural Language Filter, Analyzer and Training Document Generator. The proposed system architecture is depicted in Fig. 1.

The Natural Language Filter module would identify and filter the security related text documents from publicly available information sources. The organization-specific textual data is collected as initial training data and fed to the Training Document Generator module to prepare the training documents. The Natural Language Filter module is trained by these documents and filters the security related contents. The collected and filtered documents are analyzed by the Analyzer module for second-stage filtering for products, versions specific to the organization that generate meaningful analyzed threat information to the human operators. The documents that have been marked as true positive by the Analyzer are fed back to the Training Document Generator to improve the performance of the Natural Language Filter, and thus the overall system.

3.2 Doc2Vec Based Natural Language Filter

Mikolov et al. [4] 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 a 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 [4]. 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 $X = \text{vector}(\text{"biggest"}) - \text{vector}(\text{"big"}) + \text{vector}(\text{"small"})$ and search in the vector space for the word closest to X measured by its cosine distance [3].

In case of Doc2Vec, the model creates another vector to repre-

sent 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 [14] 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, its probability of getting represented in the vector space decreases. 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 concepts have been included in the Doc2Vec model and proven to be efficient extensions to it according to Mikolov et al. [26].

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 Lau et al. [27], 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 our proposed autonomous system by embedding a document and finding semantic similarity with a cybersecurity-specific text. In Section 5, we will discuss the experiment conducted with the Doc2Vec-based Natural Language Filter. For the purpose of this research, the Gensim^{*1} implementation of the Doc2Vec model has been utilized.

3.3 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 [8]. BERT achieves bidirectionality by using a pre-training 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

^{*1} <https://radimrehurek.com/gensim/>

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.
- (4) Positional embeddings are borrowed from the original transformer paper [27]. 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 (800 M words) and English Wikipedia (2,500 M 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 paper, the BERT model has been trained to classify security related text documents from the non-security related data.

4. 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.

4.1 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, we 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 1**.

Detailed explanations of each data source are as follows:

Table 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

- **Reddit discussions:** Reddit^{*2} is a 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 comments have been downloaded as an individual document for the time period since the subreddit was created to December 2nd 2018. Even though we cannot guarantee the collected subreddits are exhaustive of security related communities at Reddit, we 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 2nd 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, we have manually picked comments on the security related news since the creation of the site to January 1st 2017 as Informal data.
- **Security news outlet RSS feeds** RSS feed summary for select cybersecurity news outlets during the period of November 1st 2018 to December 2nd 2018. News outlets include DarkReading^{*8}, NakedSecurity^{*9}, Security-Magazine^{*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 1st 2015 to September 1st 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 soft-

^{*2} <https://www.reddit.com/>

^{*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/>

ware security vulnerabilities^{*13}. The CVE descriptions that has been accumulated in the NVD during the period from October 1st 1988 to August 1st 2018 has been utilized as the Formal type of data.

4.2 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 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 6th 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 a 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.

5. Experiment Using Doc2Vec Based Natural Language Filter

In our previous work [5] the Doc2Vec-based Natural Language Filter module was introduced. This section reviews the experiment and the results of training the Doc2Vec-based language model.

5.1 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 us 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

Table 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

upon the characteristics of the hashing algorithm is required. Based on the initial analysis, we decided the tokens of less than two characters and more than 40 characters would not contribute positively to the training of the model; hence, we 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.

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

5.2 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. [27] empirically studied these parameters and concluded that for the semantic textual similarity task, the best performing parameter settings are as listed in **Table 2**.

The hyperparameter settings described in Table 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 itera-

^{*13} <http://nvd.nist.org>

^{*14} <http://www.nltk.org/>

Table 3 Classification result of different Doc2Vec models.

Hyperparameter	Setting	Precision	Recall	F1 Score	Accuracy
Baseline	As Table 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

tions 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, we did not tune this setting 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.

5.3 Model Evaluation

In order to evaluate the effectiveness of the model, a similarity test has been performed on the test dataset mentioned in Section 4.2. 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.

The classification results using different models are shown in **Table 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 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

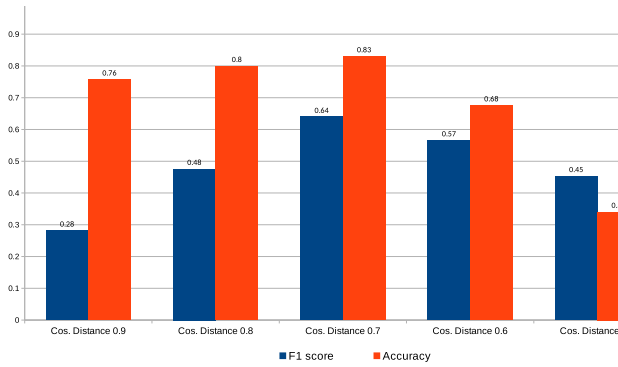


Fig. 2 Performance comparison of Doc2Vec model with different similarity thresholds.

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.

From Fig. 2, the similarity threshold of 0.7 is observed to be the most suitable threshold for the Doc2Vec model to act as Natural Language Filter.

6. 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 4. This section will review the results of the experiment of training and evaluating the BERT-based cybersecurity language model.

6.1 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 [7]. 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 4.

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 4th 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 sen-

Table 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

tence 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 4.2 has been utilized, following the above preprocessing.

6.2 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. We used the same WordPiece tokenizer with a vocabulary size of 30,000 as in the original BERT implementation. 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 our case. BERT-Base has 12 layers and each of them produces a sequence of hidden states. Since our 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, we 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, our dataset contains many examples with much longer sequences. To handle sequences longer than 512 tokens, we use a sliding window 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 5.

6.3 Model Evaluation

We followed the official guide [8] for fine-tuning three epochs on our target dataset; however, we found the reduced epoch yields

^{*15} <https://spacy.io/>

^{*16} <https://github.com/google-research/bert>

Table 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

Table 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 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

better results. The results of the evaluation are shown in **Table 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 7**.

7. Discussion

As shown in Table 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. We 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 the 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 text classification.
- (3) Training the Doc2Vec model used an unsupervised approach without specifying any label, whereas the BERT model has been fine-tuned using a labeled and balanced dataset from which the model could learn better representation.
- (4) As mentioned in Section 6.2 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 a 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

Table 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 dictionary. The Doc2Vec model trained for this research contains a vocabulary size of 530,934 unique tokens which might contain noise that has been found in the training document.

- (5) The training objective of the Doc2Vec model is to represent text documents in vector space accurately such that semantic similarities of documents could be found by cosine distance of the vectors. In comparison, BERT fine-tunes it’s all layer with classification objective.

In order to overcome the limitation mentioned in Point 5, an additional experiment has been conducted to classify cybersecurity related documents using Doc2Vec and logistic regression model. The training objective of this experiment is a 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 the BERT classification process which classifies its internal vector representations based on their feature. For a fair comparison with BERT based Natural Language Filter, we used the same, balanced dataset mentioned in Section 6 to train the 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 8** in comparison with Doc2Vec similarity-based classification and BERT.

As seen in Table 8, the Doc2Vec+LogReg performance is poor with Accuracy of 50% only. Since the test dataset consisted of an imbalanced mix of security and non-security related texts, it shows slightly better performance than a random guess. A possible explanation of such poor performance is that embeddings generated by BERT represent text better in vector space than embeddings generated by Doc2Vec. In order to illustrate this visually, we arbitrarily chose 6 example texts and created a heatmap for their cosine similarity. Each cell in the heatmap represents the cosine similarity of the corresponding texts and the higher the cosine similarity the denser the cell colors. The examples include 2 texts from each of the security and non-security related contexts 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. 3** and **Fig. 4** 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 9**.

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 re-

^{*17} <https://github.com/google-research/bert#tokenization>

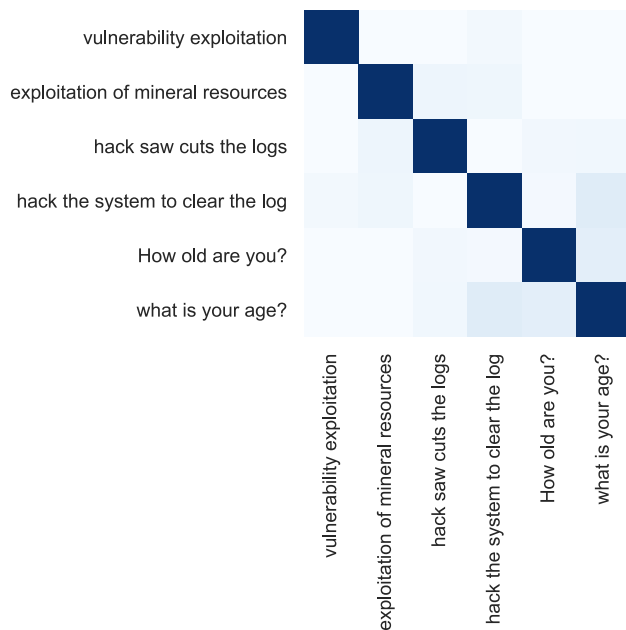


Fig. 3 Semantic similarity visualization of Doc2Vec embeddings.

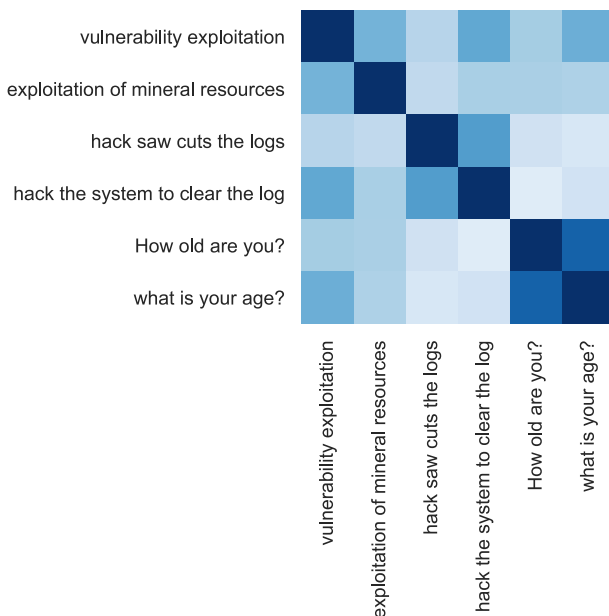


Fig. 4 Semantic similarity visualization of BERT embeddings.

Table 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

sources 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. If the model is used for inference-only tasks, the BERT language model would be suitable, since the costly training will be conducted only once. But if the model is re-trained with consecutive positive results as illustrated in Fig. 1, the BERT-based model would be difficult. However, since BERT requires much less training data to achieve similar results, a Training Document Gen-

erator module and retraining the module might not be required.

8. Conclusion

In this paper we proposed an architecture for a system that could automatically classify and analyze the cyber threat related information to assist human operators. In order to implement the Natural Language Filter module of the proposed system, neural embedding Doc2Vec and pre-trained transformer-based language model BERT were trained with cybersecurity-specific text.

The evaluation results show that the BERT-based Natural Language Filter outperformed Doc2Vec-based Natural Language Filter by 28 points in F1 Score.

From Table 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, we conclude that BERT would be the most suitable language model to implement the Natural Language Filter module for the proposed system.

In future research, we would like to explore the possibilities of retraining the BERT and Doc2Vec language models and determine the necessity of a Training Document Generator module for the proposed system. Also, using Named Entity Recognition methods, we are actively working on the implementation of an Analyzer module for the proposed system.

Acknowledgments We gratefully acknowledge the support of ExaWizards Inc for providing assistance to run computationally expensive experiments in their infrastructure.

References

- [1] Mendsaikhan, O., Hasegawa, H., Yamaguchi, Y. and Shimada, H.: Mining for operation specific actionable cyber threat intelligence in publicly available information source, *Proc. Symposium on Cryptography and Information Security* (2018).
- [2] Turney, P.D. and Pantel, P.: From Frequency to Meaning: Vector Space Models of Semantics, *CoRR*, Vol.abs/1003.1141 (2010) (online), available from <http://arxiv.org/abs/1003.1141>.
- [3] Mikolov, T., Chen, K., Corrado, G. and Dean, J.: Efficient Estimation of Word Representations in Vector Space, *1st International Conference on Learning Representations, ICLR 2013, Workshop Track Proceedings* (2013) (online), available from <http://arxiv.org/abs/1301.3781>.
- [4] Le, Q.V. and Mikolov, T.: Distributed Representations of Sentences and Documents, *CoRR*, Vol.abs/1405.4053 (2014) (online), available from <http://arxiv.org/abs/1405.4053>.
- [5] Mendsaikhan, O., Hasegawa, H., Yamaguchi, Y. and Shimada, H.: Identification of Cybersecurity Specific Content Using the Doc2Vec Language Model, *Proc. IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, pp.396–401, IEEE (online), DOI: 10.1109/COMPSAC.2019.00064 (2019).
- [6] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L.: Deep contextualized word representations, *CoRR*, Vol.abs/1802.05365 (online), available from <http://arxiv.org/abs/1802.05365> (2018).
- [7] Howard, J. and Ruder, S.: Fine-tuned Language Models for Text Classification, *CoRR*, Vol.abs/1801.06146 (online), available from <http://arxiv.org/abs/1801.06146> (2018).
- [8] Devlin, J., Chang, M., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *CoRR*, Vol.abs/1810.04805 (online), available from <http://arxiv.org/abs/1810.04805> (2018).
- [9] Al-Rowaily, K., Abulaish, M., Al-Hasan Haldar, N. and Al-Rubaian, M.: BiSAL - A Bilingual Sentiment Analysis Lexicon to Analyze Dark Web Forums for Cyber Security, *Digit. Investig.*, Vol.14, No.C, pp.53–62 (online), DOI: 10.1016/j.diin.2015.07.006 (2015).
- [10] Mulwad, V., Li, W., Joshi, A., Finin, T. and Viswanathan, K.: Extracting Information About Security Vulnerabilities from Web Text, *Proc.*

2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 03, WI-IAT '11, pp.257–260, IEEE Computer Society (online), DOI: 10.1109/WI-IAT.2011.26 (2011).

- [11] Joshi, A., Lal, R., Finin, T. and Joshi, A.: Extracting Cybersecurity Related Linked Data from Text, *Proc. 2013 IEEE Seventh International Conference on Semantic Computing, ICSC '13*, pp.252–259, IEEE Computer Society (online), DOI: 10.1109/ICSC.2013.50 (2013).
- [12] More, S., Matthews, M., Joshi, A. and Finin, T.: A Knowledge-Based Approach to Intrusion Detection Modeling, *Proc. 2012 IEEE Symposium on Security and Privacy Workshops, SPW '12*, pp.75–81, IEEE Computer Society (online), DOI: 10.1109/SPW.2012.26 (2012).
- [13] Dionísio, N., Alves, F., Ferreira, P.M. and Bessani, A.: Cyberthreat Detection from Twitter using Deep Neural Networks, *CoRR*, Vol.abs/1904.01127 (2019).
- [14] Tavabi, N., Goyal, P., Almukaynizi, M., Shakarian, P. and Lerman, K.: DarkEmbed: Exploit Prediction with Neural Language Models, *Proc. AAAI Conference on Innovative Applications of AI (IAAI2018)* (2018).
- [15] Niu, L. and Dai, X.: Topic2Vec: Learning Distributed Representations of Topics, *CoRR*, Vol.abs/1506.08422 (2015) (online), available from <http://arxiv.org/abs/1506.08422>.
- [16] Dhingra, B., Zhou, Z., Fitzpatrick, D., Muehl, M. and Cohen, W.W.: Tweet2Vec: Character-Based Distributed Representations for Social Media, *CoRR*, Vol.abs/1605.03481 (2016) (online), available from <http://arxiv.org/abs/1605.03481>.
- [17] Choi, E., Bahadori, M.T., Searles, E., Coffey, C. and Sun, J.: Multi-layer Representation Learning for Medical Concepts, *CoRR*, Vol.abs/1602.05568 (2016) (online), available from <http://arxiv.org/abs/1602.05568>.
- [18] Aman, H., Amasaki, S., Yokogawa, T. and Kawahara, M.: A Doc2Vec-Based Assessment of Comments and Its Application to Change-Prone Method Analysis (online), DOI: 10.1109/APSEC.2018.00082 (2018).
- [19] Karvelis, P.S., Gavrilis, D., Georgoulas, G.K. and Stylios, C.D.: Topic recommendation using Doc2Vec, *2018 International Joint Conference on Neural Networks (IJCNN)*, pp.1–6 (2018).
- [20] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H. and Kang, J.: BioBERT: A pre-trained biomedical language representation model for biomedical text mining, *CoRR*, Vol.abs/1901.08746 (2019) (online), available from <http://arxiv.org/abs/1901.08746>.
- [21] Beltagy, I., Cohan, A. and Lo, K.: SciBERT: Pretrained Contextualized Embeddings for Scientific Text, *CoRR*, Vol.abs/1903.10676 (2019) (online), available from <http://arxiv.org/abs/1903.10676>.
- [22] Lee, J. and Hsiang, J.: PatentBERT: Patent Classification with Fine-Tuning a pre-trained BERT Model, *CoRR*, Vol.abs/1906.02124 (2019) (online), available from <http://arxiv.org/abs/1906.02124>.
- [23] Huang, K., Altsaier, J. and Ranganath, R.: ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission, *CoRR*, Vol.abs/1904.05342 (2019) (online), available from <http://arxiv.org/abs/1904.05342>.
- [24] Sun, C., Myers, A., Vondrick, C., Murphy, K. and Schmid, C.: VideoBERT: A Joint Model for Video and Language Representation Learning, *CoRR*, Vol.abs/1904.01766 (2019) (online), available from <http://arxiv.org/abs/1904.01766>.
- [25] Adhikari, A., Ram, A., Tang, R. and Lin, J.: DocBERT: BERT for Document Classification, *CoRR*, Vol.abs/1904.08398 (2019) (online), available from <http://arxiv.org/abs/1904.08398>.
- [26] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J.: Distributed Representations of Words and Phrases and their Compositionality, *CoRR*, Vol.abs/1310.4546 (2013) (online), available from <http://arxiv.org/abs/1310.4546>.
- [27] Lau, J.H. and Baldwin, T.: An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation, *CoRR*, Vol.abs/1607.05368 (2016) (online), available from <http://arxiv.org/abs/1607.05368>.



uational awareness, cyber threat intelligence, and knowledge graph.



Otgonpurev Mendsaikhan received his M.S. degree in Information Security Policy and Management from Carnegie Mellon University, Pittsburgh, PA, USA. He is currently pursuing the Ph.D. degree at Graduate School of Informatics, Nagoya University, Japan. His main research interests include cyber security situational awareness, cyber threat intelligence, and knowledge graph.

Hirokazu Hasegawa received his Ph.D. degree in Information Science from Nagoya University, Japan in 2017. He is currently an Assistant Professor at Information Security Office, Nagoya University, Japan. His research interests include the Internet and network security. He is a member of IPSJ and IEICE.



Yukiko Yamaguchi received her B.S. degree in Information Engineering from Nagoya Institute of Technology in 1983 and the M.S. degree in Information Engineering from Nagoya University. Since then she was affiliated with Fujitsu Lab. Ltd and in April 1991 she joined Nagoya University as an Assistant Professor in the Computer Center. At present, she is an Assistant Prof. in Information Technology Center and engaged in research on network management technology and cyber security. She is a member of IPSJ and IEICE.



Hajime Shimada was born in 1976 and received his B.E., M.E., and D.E. degrees from Nagoya University, Japan in 1998, 2000 and 2004 respectively. He was an assistant professor at Kyoto University from 2005 to 2009 and an associate professor in NAIST from 2009 to 2013. He has been working as an associate professor of Nagoya University, Japan since 2013. He is currently focusing on cyber security, network operation and computer architecture related researches. He is a member of IEEE, IPSJ and IEICE.



Enkhbold Bataa is currently working for ExaWizards Inc as a Machine Learning Engineer. He received his M.S. degree in Information Systems from National Tsing Hua University, Hsinchu, Taiwan. His research focuses on language model, word embeddings and computational linguistics.