Semantics-Aware Representation Learning for Malware Retrieval

Anonymous Author(s)

ABSTRACT

We consider the method for creating semantics-aware malware retrieval system. Existing malware detection systems may cope with only simple malware variants. To be able to retrieve malwares which are semantically similar but structually dissimilar, systems need to understand semantics of malwares. In order to make the system have the semantic understanding attribute, we supervise the multiple labels about malicious behaviors. Also we approximate representation vector to target semantic vectors using multilabel centerloss. Experimental results show that our system has capability of retrieving semantically similar samples. Also the qualitative tests show that this model works well for following three types of queries: malware samples, semantic components and a combination of malware samples and semantic components.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**;

KEYWORDS

Malware, Information Retrieval, Semantic Space, Deep Learning, Multilabel

1 INTRODUCTION

Hundreds of thousands of malwares are generated daily, and this number continues to grow because of the development of malware obfuscation tools that leverage differing hash values. However, the time expended by security experts has not increased. Security experts perform two tasks: categorize malicious code that not been filtered by a hash value, and research obfuscation techniques or malicious behaviors to cope with new malware and their variants.

Generally, between these two tasks, the first can be handled by machine learning algorithms. It is impossible for security experts to categorize numerous new malware programs via static or dynamic analysis. Therefore, systems that automatically classify malware have been created. Malware classification models, such as support vector machines, random forests, and deep neural networks (DNN), having static features such as PE structure or API-call-graphs, have been studied[2, 3, 28, 33, 45].

If the accuracy of the classifier is low, or if the classifier does not cope with new malware, security experts should add a verification process to check whether the machine learning model works well. The verification process requires a system that enables experts to retrieve and compare samples similar to the input samples. Therefore, there is an increasing need for a malware retrieval (MR) system that can validate the classification model while helping with the second task: the deep dive.

MR systems semantically retrieve and rank similar malware samples like natural language or image information retrieval systems. Thus, we want to build a MR system to capture the malware's "meaning.''

A semantics-aware system is one that has a semantic relationship with malware. Even if they are structurally different, one can be regarded as a sample having the same meaning if their behaviors coincide. Likewise, if the behavior of malware differs, even if it is structurally nearly the same, the two samples will retain different meanings. In any case, if the semantic differences between the samples can be considered, it is a semantics-aware system.

There was an attempt to build a semantics system using traditional semantic-aware features. First, dynamic features represented the behavior of malware, but they had the disadvantage of being difficult to obtain, because they were searched in all behavior cases[14, 30, 42]. In a second method, a syntactic feature obtained by static analysis could not represent the meaning of the malware sample[4, 46]. To overcome this, hand-crafted features were used to better detect the common patterns of malicious activity detected by security experts[13]. However, to create a machine learning model that expresses complex semantic relationships, it is necessary to consider the application of feature expression learning techniques, such as deep learning.

Deep Learning can hierarchically train common latent features to represent malware samples that have the same meaning from not only hand-crafted features engineered by security experts, but also non-engineered inputs, such as malware binaries, source code, metadata, resource files, dynamic behavior logs, and screenshots. Therefore, it is possible to learn abundant complex semantic relationships, which is advantageous for generalizing to samples that we have not seen before.

Thus, if there is a good input feature and a deep learning ability to train latent features, one learn the semantic relationships of malware. In the natural language domain, the self-supervised method is used (e.g., the front-to-back relationship of a sentence) to learn the semantic relationship [20], or by using metrics to stochastically express the co-occurrence based on context[29].

However, because we do not have a sufficient information for unsupervised learning with in malware domains, we should make model learn semantic relations through supervised learning. Most deep learning-based malware classifiers learn to classify single label of malware family. In this case, the representation vectors obtained from learning show only that malwares are similar when distance between those vectors is small and dissimilar when the distance is large. But the distance does not guarantee a difference in meanings.

Given the aforementioned point, how can model learn the semantics of malware? Malware can display multiple malicious acts or attributes. Thus, the semantics of malware can be considered a combination of malicious actions or semantic components implied by attributes. We refined the malware tags, offering them to the supervised deep learning model, which learns the semantics of the malware. During the learning process, the malware's representation vector is trained via a linear combination of the semantic component vectors to have a specific meaning. We taught our model with the proposed metric learning method in which the

distance between malware samples in the vector space equaled the difference in meaning. Thus, we obtained a malware semantic space representing the semantics of malware samples as vectors.

The metric learning method we proposed uses the multi-label centerloss objective function designed by augmenting centerloss[40]. This loss function causes the malware representation vector to approach its target semantic vector. This can be used with other loss functions as constraints and is not difficult to implement. Additionally, as with the scoring technique used in the information retrieval task, we assign a weight to a specific semantic component to provide a constraint that plays a bigger role in ranking than in other components.

Also we introduce a flexible MR system that retrieves malwares from semantically similar malware, semantics of malware or those combinations. Then, we check how well the MR system is created through quantitative and qualitative evaluations.

Our main contributions can be summarized as follows.

- We model the semantics of malwares as vectors and represent the latent features of the malwares as linear combinations of semantic components. We then embed them on the semantics space. We propose a structure of malware classifier based on deep learning to learn the semantic space.
- We propose a metric learning method based on multi-label centerloss to learn the difference of semantics in the semantic space.
- We present a flexible MR system that retrieves malware via malware, semantics, and their combination for malware sample vectors in the semantic space, and we evaluate its performance.

The composition of our paper is as follows. Background which contains Deep learning, IR, metric learning, and semantic space for natural language is described in Section 2. Section 3 defines the structure and desired properties for the malware IR system. Section 4 explains the malware semantic space and shows our suggestions for learning it. Section 5 describes the data set used in the experiment, the neural network hyperparameter settings, and the quantitative and qualitative test results. In Section 6, we discuss related works. In section 7, we discuss future works. In Section 8, we present our conclusion.

2 BACKGROUND

2.1 Deep Learning

Neural networks are machine learning implementations that extract features based on statistical characteristics of training data. DNNs also provide hierarchical feature representations by configuring the network into several hidden layers. Generally, a DNN consists of an input layer, several hidden layers, and output layer. For a classification task, the input layer receives a feature vector representation of the target sample of the task as input. The output layer outputs the probability of a class related to a given input vector as a vector. This allows the DNN to predict the class per the input vector.

The image classification task can improve the performance of the model by using the convolutional neural network (CNN)[17], consisting of convolutional layers, pooling layers, and general neural network layers. For the convolutional layer, we extract features by applying a convolutional filter to the input. The pooling layer has the effect of down-sampling the features giving less effect.

2.2 Information Retrieval

Information retrieval was introduced in natural language processing as a task to find related documents from a string query. However, it has been extended to other domains and is now used in various fields (e.g., image[6, 44], music[34], medical care[8, 23], and malware[32]).

Boolean model. A Boolean model is a classical information retrieval model. It is a search model that considers documents and user queries as set from the background of Boolean logic and set theory. Whereas its ease-of-implementation and intuitive use is advantageous, it is limited because the similarity is calculated discretely, and the retrieval result becomes very large or cannot be retrieved if it does not exactly match. Furthermore, all terms have the same weight[18] .

Vector space model. Another method is the vector space model. This model represents documents and user queries in a vector space so that the distance between vectors can become similar. This requires a simple linear algebra model. It improves the retrieval result by giving different weights to the terms; it shows the similarity of the query as a continuous value. There are various models according to how to make the query of the document and the user into the vectors[9, 19] and how to measure the distance between the vectors[7, 11, 22, 31].

Learn vector representations using deep learning. As mentioned, using deep learning to perform machine learning tasks results in better generalization. This is because it learns via a hierarchical representation. Therefore, the combination of various tasks using the vector space model and deep learning results in performance breakthroughs for many tasks[10, 17, 20]. Additionally, the IR field using the vector space model also achieves a breakthrough with deep learning[11, 35, 39].

There are supervised and unsupervised methods for learning vector representations. A typical unsupervised learning method uses an autoencoder. Attempts have been made to encode malware using the autoencoder. Supervised learning occurs when a person provides an answer label, and a classification or regression model performs the task of matching the label. The output of the bottleneck layer obtained from the learning process as a representation vector of the corresponding sample contains the most compressed information. This allows some control over the position of the vector representation. For example, one can use the objective function for metric learning to determine which samples to locate near and which samples to locate far. This allows one-shot or few-shot learning tasks, improving the performance of information retrieval and determining rankings based on the distance between vectors and respond to fewer samples.

2.3 Metric Learning

Metric learning is a technique for learning distance functions. More specifically, through metric learning, one can train a distance function to determine which samples will be close to each other, which samples will be farther, or, to some extent, how far the distance will be.

A typical example of metric learning using deep learning is the Siamese network. The Siamese network receives pairs of samples as input data, teaching that the distance difference of the feature vector is 0 if the pairs are the same label, and that the distance difference is 1 or more if the labels are different.

Another example is centerloss[40]. This creates a template vector for each class in the single label classification task and adds constraints to the existing cross-entropy loss so that the representation vectors of the samples belonging to the same class become the same as the template vector of the corresponding class. This reduces the intra-class variance so that representation vectors can be used for tasks like information retrieval.

The distance function learned by metric learning is used in many real-word tasks to find similar data, such as face recognition, image retrieval, and text retrieval[11, 36, 39].

2.4 Semantic Spaces for Natural Language

The purpose of creating semantic spaces is to represent a natural language that captures meaning in the natural language domain. The original motivation of semantic spaces was to address two challenges in the natural language domain. The first is to make words with similar syntactics but different meanings to different representations, and the other is to make words with the same meaning but the different syntactics to the same representation.

Recently, neural networks have made major advances in combination with other new approaches (e.g., word2vec[20], glove[29] and fastText[15]). If we can express the natural language well in semantic spaces, we can use it for various tasks, such as document classification, document search, question and answering, speech recognition, and translation.

3 MALWARE INFORMATION RETRIEVAL SYSTEM

In this section, we describe the behavior, configuration, and properties of the proposed malware retrieval system. Our system is divided into two phases: training and retrieval. During the training phase, we train the representation vectors of the training data that reflect the semantics. During the retrieval phase, the ranking result is output from the malware sample input from the user, the tag query, or the previously stored representation vectors. The following subsections define the notations to be used in the two phases and use them to describe in detail which tasks to perform in each phase.

3.1 Notations

We have $X = \{x_1, x_2, ..., x_N\}$ as a training malware set. We also have a multi-label set, $Y = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$, corresponding to malware labels. Then, $V = \{v_1, v_2, ..., v_n\}$ is defined as a set of hand-crafted features extracted from malware.

MR system is defined as tuple with $[u, h, E, d, R]$ elements. u is a feature extractor module that extracts the hand-crafted features from malware sample. h is an embedding function that obtains a vector representations from malware features. $E$is the embedded vector set of malwares to be retrieved. d is a function that calculates the distance between the input query and the embedded vector. R is a ranking module. During the training phase, it learns the appropriate h and gets E from V. During the retrieval phase, the malware sample query, q, is entered, and the nearest k neighbors are returned via the ranking module, R.

Table 1: List of Symbols

Symbol	Meaning
X	Set of malwares
x_i	i-th malware sample
v_i	Extracted features of i-th malware
V	Set of extracted features
Y	Set of multilabels
$\mathbf{y}_i$	Multilabels of i-th malware
u	Feature extractor
h	Neural embedder
θ	Parameters of neural embedder
g	Classifier
W, $\mathbf{b}$	Parameters of classifier
E	Set of embedded malware vectors
$\mathbf{e}$	Embedded malware vector
$\mathbf{c}$	Importance coefficient
d	Distance measuring function
$\mathbf{s}$	Semantic component vector
R	Ranking module
$\mathbf{e}_q$	Embedded vector of queried sample

3.2 Tasks

During the training phase, we perform an auxiliary task to obtain the vector representations used in the MR system. The auxiliary task can be a single label classification or a multi-label classification. $f : V \rightarrow Y$ f is a function that estimates a label from malware features. The function, f, can be defined as a composite function of g and h. h is an embedding function for obtaining vector representations from the malware features described above. g is a classifier that can estimate labels from representation vectors. See Section 4 for details on how to train representation vectors that contain semantics.

$$f(v_i) = (g \circ h)(v_i) = g(h(v_i)) = g(\mathbf{e_i}) = y_i$$

where

$$g(\mathbf{e_i}) = \sigma(W\mathbf{e_i} + \mathbf{b})$$

In the retrieval phase, the malware sample query, q, is embedded using the h function that already trained at training phase. The distance between the elements of the representation vector of the query, $\mathbf{e_q}$ and E, is measured through the d function, and the k neighbors are returned through the ranking module, R.

3.3 Modules

The structure of the two phases is shown in the Figure.1 and Figure.2. The common modules in both phases include the feature extractor and the neural embedder. The vector training phase includes the classifier, and the retrieval phase contains the ranking module.

Feature extractor. The feature extractor is a module that extracts hand-crafted features from raw malware sample. For example,

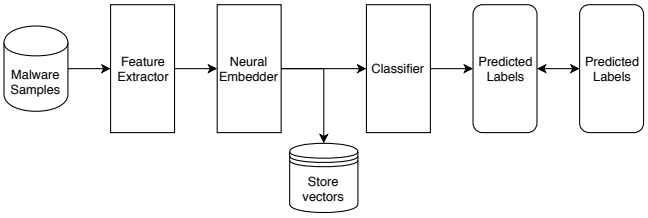

Figure 1: The structure to perform training phase.

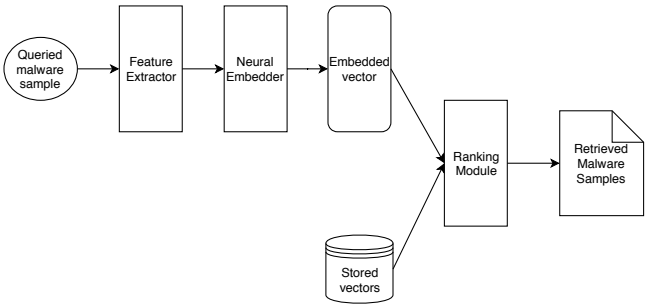

Figure 2: The structure to perform retrieval phase.

hand-crafted features, such as size, entropy, and histogram-of-API-calls can be extracted. In the case of APK, an additional permission feature can be extracted.

Neural embedder. The neural embedder creates a representation vector from the features of malware extracted from the feature extractor module. This is a neural network parameterized with theta, and parameters are optimized while performing the auxiliary task in the vector training phase. During the retrieval phase, the parameters are not updated.

The network structure of the neural embedder is CNN. The malware image[25] of the raw data is passed through five convolutional layers and two fully connected layers. The convolutional layers extract features to contain local shift invariant characteristics of the malignant region or the region that characterizes the malware.

Classifier. The classifier module represents the probability that the sample corresponds to each label based on the representation vectors from the neural embedder. It is parameterized in one layer of fully connected and it uses softmax or sigmoid activation depending on the task. Like the neural embedder module, the parameters are trained by back-propagation in the vector training phase and not updated in the retrieval phase.

Ranking module. We measure the distance between the embedding vectors obtained from the input sample, and the embedding vectors of the training data set using the distance function, d, and output k nearest neighborhoods.

3.4 Desired Properties

Extant MR systems have limitations. They could not retrieve samples that have similar meanings with different structure. It also fails to cope with the emerging variants and new families. To become a malware IR system that solves this problem, the following attributes must be satisfied.

Semantic understanding. The system should be able to rank and retrieve structurally similar samples and semantically similar samples for the queried samples. This similarity means that the malware attributes or behaviors are like what security experts expect. Even if the structure of two malwares is different, both have the same meaning if the behavior of the malwares are the same. Likewise, if the behavior is quite different, even if it is structurally nearly the same, the two samples will have different meanings. For example, we think that the similarity of samples between ransomware families, Locky and Cerber, is smaller than the similarity between the two samples of different family, Locky and Coinminer. If the semantic differences between these samples are considered, the system is a semantics-aware system.

Robustness to novel samples. New malware variants continue to appear and the system must respond quickly.

Efficiency. A large number of malware currently exist, and this number is increasing exponentially. Thus, to retrieve the k-ranked results of many samples in a reasonable time, the ranking module should be efficient.

4 SEMANTICS AWARE REPRESENTATION LEARNING

This section describes how the space in which malware is embedded approximates the semantic space. First, we describe malware semantic space and we propose a loss function for training semantic-aware malware representation vectors.

4.1 Semantic Spaces for Malwares

Definition. Let $X = \{x_1, \ldots, x_n, \ldots, x_N\}$ be a set of malwares. Let S be a subset of vector space V of dimension d and let semantic component set $S = \{s_1, ..., s_k, \ldots s_K\}$ be a linearly independent subset of V. Then for every $\mathbf{e} \in E$, there is an unique linear combination of the sementic component vectors that equals e.

$$\mathbf{e} = c_1\mathbf{s_1} + c_2\mathbf{s_2} + \ldots + c_k\mathbf{s_k}$$

where c_i is the i'th element of coefficient vector $\mathbf{c} = \{c_1, ..., c_k\}$ and we call it as importance coefficents. We call a vector $\mathbf{e}$ as a semantic vector of malware x and there is a nonlinear mapping from X to E.

Metric function. we use an Euclidean Distance $d(\mathbf{e_i}, \mathbf{e_j})$ for a metric of a set E and it means the function that defines a distance between semantics of two malwares.

The meaning of approximate semantic space of malwares We now take an example of language, thinking of the meaning of certain words. We consider the relationship of those meanings. We can assume that city names and words related to machine learning are located far away from each other. We can also imagine that the words will be near. The similarity of the relationships between words can thus be considered. The relationship between "King'' and "Queen'' is similar to the relationship between "Man'' and "Woman.'' To maximize the meanings of these words, we represent them as vectors on a vector space. If the meaning is similar, the distance between the vectors is near. If the meaning is disparate, the distance between the vectors is far away. We also approximate the semantic relationship between words by using operations defined in the vector space, such as "king - queen = man - woman.''

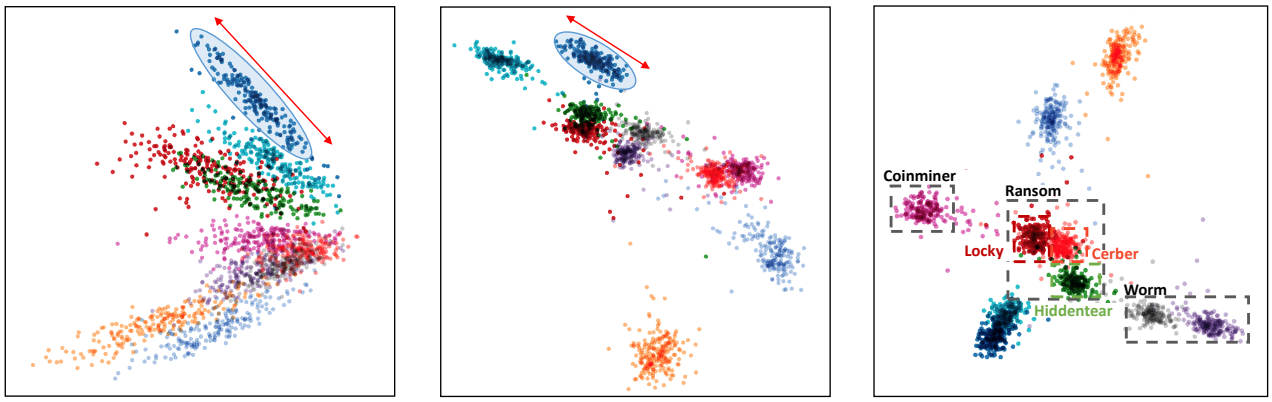

Figure 3: The concept of our proposed method. Left figure shows PCA plot of representation vectors which are trained by single label classificaion task. Center figure shows the vectors trained by training single label classification task with centerloss. We observe that of center figure's intra-class variance model is smaller than left figure's. Right figure shows the vectors which are trained by multi-label classification task with multi-label centerloss. It shows that representation vectors whose semantics are simliar are nearly located then others.

Similarly, we can imagine the semantic meaning of malicious code. We consider malware a combination of malicious components. We also use the concept of basis and linear combination to approximate it as a vector space operation. In other words, we can approximate the semantic space by training the semantic vector of malware to be represented as a linear combination of the semantic component vector.

Evidences. If we approximate the semantic space well, then the following querying tasks should work correctly. First, we should be able to retrieve semantically similar malwares by querying malware samples. Second, it should be possible to retrieve by only a combination of semantic components. For example, we should be able to retrieve malwares with these three attributes: "ransom'', "downloader'', and "agent''. Finally, we should be able to retrieve related samples by querying a combination of samples and semantic components such as a sample with attributes "ransom'' and "downloader'' combined with semantic component "agent''.

4.2 Solution Overview

Multilabel classification. To train the vector representation, E, that reflects the meaning of the malware sample, multi-label classification learning was selected as the auxiliary task. To train this classification task, the neural embedder, h, places the representation vectors so that they can be classified by the linear classifier. Most studies that train malware classifiers learn classifiers to classify malware under single label.

However, if you identify a label in a malware domain and create a malware IR system based on it, then the trained embedding vector is insufficient to include the meaning of the sample. Typically, the process of labeling malware is irrational in expressing the meaning of malware. Malware often displays multiple malicious behaviors at the same time. However, existing malware labeling systems do not reflect these attributes. Additionally, naming rules are often globally inconsistent, and even locally, analysts name via differing criteria. Regarding non-prevalent malware, when one labels malware, a name is provided unrelated to the malware's behavior. These labeling processes prevent the machine learning model from learning the appropriate meanings.

We use the auxiliary task to classify the labels into multiple labels, and we use vector representations from the auxiliary task to enable retrieval of malware samples of similar meanings.

Centerloss. To allow the representation vectors of malwares to exist on the semantic space while training multi-label classification in the above way, we use multi-label centerloss. Once in the background, the effect of centerloss is lowering the intra-class variance. This causes centerloss to prevent the intra-class variance from becoming too large and the distance between two samples in the same class to be greater than the distance between two samples of different classes, resulting in improved IR results.

Second, we can locate representation vectors to approximate the semantic space. In the paper proposing a single label centerloss, the template vector is stored in the external memory for each label, and the mean square error (MSE) is set so that the distance between the template vector and the representation vector of sample becomes small. As a result, the representation vector of the sample and the template vector of the class are approximated. Likewise, we store vectors for each semantic component in external memory, and we set the MSE error to a loss function so that representation vector is a linear combination of semantic component vectors.

Learn to rank. In malware labels, there are important labels and those that are not. Considering this importance in creating a MR system is one of the important factors to satisfy the semantic understanding property. For example, in PE format labels, ranking more than labels such as ransom and coinminer rather than agent or downloader would be considered as a semantics-aware search system. To solve this problem, we use weighted centerloss which reflects the importance of labels with constraint. The constraint determines the norm of the semantic components. Euclidean distance, the metric function that we use in retrieval, is affected by the norm of the two vectors when computing distances. The distance

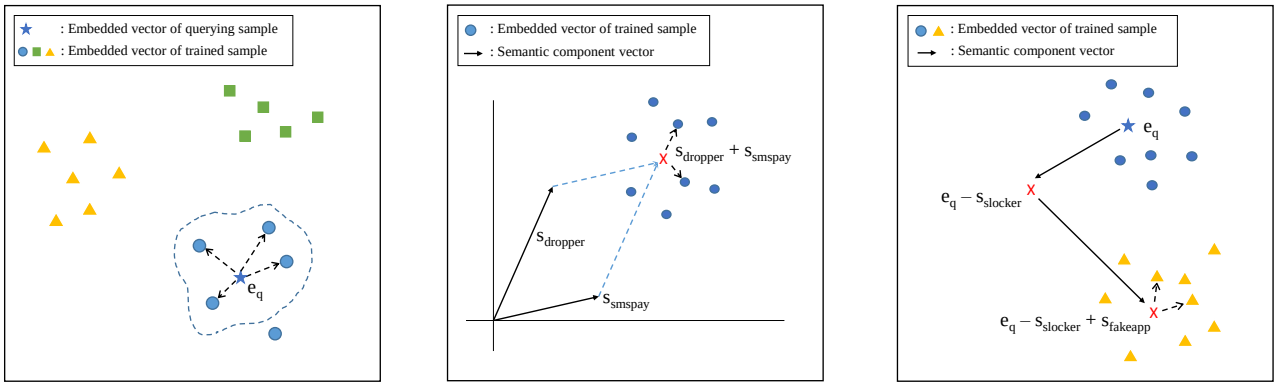

Figure 4: Our malware retrieval system was queried in three ways: malware samples(left), semantic components(center) and a combination of malware samples and semantic components(reight).

between semantic component vector, having a large norm and another component vector is likely to be larger than the distance between component vectors having a small norm. Therefore, in the MR system ranking with Euclidean distance, when malware having a semantic component with a high importance is queried, the malware samples, including the semantic component, are more likely to be searched. This can be regarded as a scoring technique (i.e., tf-idf[1]), making it possible to search documents with specific properties more easily in the existing information retrieval system.

4.3 Proposed Objective Functions

We propose a new objective function to learn the semantic space by the methods described in the Section 4.2.

Multi-label center loss(MCL). The expression of the constrained optimization problem we want to solve is equal to Eq.1. This adds a constraint that the distance between the target semantic vector and the representation vector should be less than ϵ on the negative log likelihood optimization problem, which is the basis of the cross-entropy loss.

$$\min_{\theta, W, \mathbf{b}} J(\theta, W, \mathbf{b}) = -\sum_i \sum_j y_{mij} \log \hat{y_{ij}} \tag{1}$$

s.t.

$$h(v_i; \theta)) - \mathbf{s}_{\text{target}} < \epsilon,$$

for $i \in \{1, 2, ..., N\}$ where $\epsilon >= 0$ and s_{target} is a target semantic vetcor.

To find the parameters satisfying the above equation, the final loss function was constructed by adding cross-entropy loss and multilabel centerloss at a ratio of 1:λ[40]. We can formulate the loss function we propose as Eq.2.

$$\begin{aligned} L &= L_s + \lambda L_c \\ &= -\frac{1}{N} \sum_i \sum_j y_{mij} \log \hat{y_{ij}} + \lambda \frac{1}{N} \sum_i (\mathbf{s}_{\text{target}} - h(v_i; \theta))^2 \end{aligned} \tag{2}$$

where

$$\hat{y_{ij}} = \frac{\exp(Wh(v_i; \theta) + \mathbf{b})}{1 + \exp(Wh(v_i; \theta) + \mathbf{b})}$$

Weighted multi-label center loss(WMCL). As described above, we apply importance coefficients for each semantic component to apply the learn to rank technique to our Malware Information Retrieval System. We added a constraint such that the importance coefficient value is equal to the norm of the semantic component vector.

$$||\mathbf{s_i}|| = c_i \text{ for all i } \in \{i = 1, 2, ..., M\}$$

The cross-entropy loss updates the parameters of $\theta, W, \mathbf{b}$ to improve the accuracy of multilabel classification. Multi-label center-loss updates parameters as follows. First, the parameters of $\theta, W, \mathbf{b}$ are updated so that the embedding vector is close to $\mathbf{s}_{\text{target}}$ via back propagation. Second, the semantic components are updated so that $\mathbf{s}_{\text{target}}$ is close to the embedding vector. When updating, we divide difference vector between $\mathbf{s}_{\text{target}}$ of current step and embedding vector equally and we add it to each semantic component vectors, such that $\mathbf{s}_{\text{target}}$ of next step to be locate at Internally dividing point of current $\mathbf{s}_{\text{target}}$ and embedding vector of sample.

There are several methods to combine semantic component vectors, we propose summation model and average model. The performance of each model can be found in Section 5.

The process of parameters and semantic component update can be found in Figure. Ref fig: update, Algorithm. Ref alg: centerloss and Algorithm. Ref func: functions.

5 EVALUATION

In this section, we evaluate the performance of our MR system in several ways, and verify that we have performed the semantic-aware malware retrieval task well.

5.1 Implementation and Setup

Datasets. The dataset used for learning and evaluation of the model is composed of samples crawling from VirusTotal[37] and certified by security expert. In addition, Labeling was automated from the detection results of dozens of AntiVirus engines provided by VirusTotal. The detection results of each AntiVirus engine in VirusTotal are parsed in a meaningful word tokens, and the security expert selected labels from the word tokens. In addition, labels were

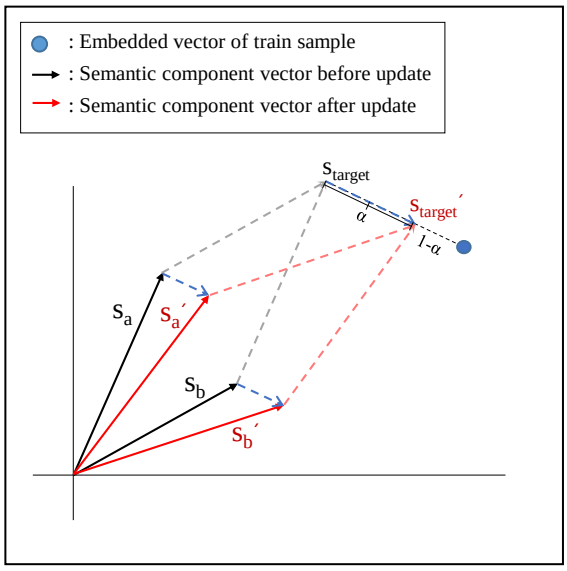

Figure 5: The semantics-aware representation vector learning algorithm. When updating, we divide difference vector between s_{target} of current step and embedding vector equally and we add it to each semantic component vectors, such that s_{target} of next step to be locate at Internally dividing point of current s_{target} and embedding vector of sample.

ALGORITHM 1: The semantics-aware representation vector learning algorithm.

Input: Extracted handcrafted features of training data v_i and their semantic components $\mathbf{y}_i$. Initialized parameters θ in neural embedder. parameters W, b in classifier. initialized semantic component vectors **s**.Importance coefficients **c**. Hyperparameter λ, α, learning rate μ.

Output: The parameters θ, W, b

repeat

s_{target} = `getTargetSemanticVectors`($s_{\mathbf{y}}$)

Compute total loss by $L = L_s + \lambda L_c$

Update parameters θ by $\theta \leftarrow \theta - \mu \frac{\partial L}{\partial \theta}$

Update parameters W by $W \leftarrow W - \mu \frac{\partial L}{\partial W}$

Update parameters b by $b \leftarrow b - \mu \frac{\partial L}{\partial b}$

Update semantic component vectors s by $s \leftarrow$ `getNewSemanticComponentVectors`()

until *converge*;

ALGORITHM 2: Definitions of functions.

Function `getTargetSemanticVectors`(**s**):

$M \leftarrow$ The number of semantic components

if *Summation Model* **then**

$s_{\text{target}} \leftarrow \sum_i^M s_i$

end

else if *Average Model* **then**

$s_{\text{target}} \leftarrow \frac{1}{M} \sum_i^M s_i$

end

return s_{target}

Function `getNewSemanticComponentVectors`(**s**, s_{target}, e, α, **c**):

$M \leftarrow$ The number of semantic components

foreach $i \in \{1, 2, ..., C\}$ **do**

if *Summation Model* **then**

$s_i \leftarrow s_i - \frac{\alpha}{M}(s_{\text{target}} - e)$

end

else if *Average Model* **then**

$s_i \leftarrow s_i - \alpha(s_{\text{target}} - e)$

end

if *Weighted Model* **then**

$s_i \leftarrow \frac{s_i}{||s_i||} c_i$

end

end

return s

prioritized according to their importance. This saves the time of the security expert in labeling the malware samples, while achieving accurate labels. Finally, for one sample, we have two types of labels: a representative label and a multi-label. The multi-label includes a representative label. Among the datasets, train set and test set were randomly sampled by 7: 3. And the distribution of samples over the representative label is even.

- **Dataset 1. PE1300.** Among the more than 50,000 PE samples crawled from VirusTotal, the security expert verified about 2,000 PE samples with 10 labels. Four of the labels are representative labels and the number of samples for the representative label is almost the same.
- **Dataset 2. APK19000.** In the case of APK, there are 19,000 samples from 190,000 samples crawled by VirusTotal, which were verified by expert, and used 83 kinds of labels. Of these, 19 labels are representative labels, and the number of samples for each representative label is 1000.

Hyperparameters. The hyper parameters used in the experiment are as follows. Adam optimizer[16] was used as the optimizer and the learning rate was 0.001. α in centerloss is 0.1. When learning the PE1300 dataset, we used 0.1 as λ, which is the ratio of cross-entropy loss and centerloss, and 0.001, when learning APK19000. The structure of the neural network is the five layers of CNN and two Fully Connected layers using batch normalization[12]. The previous layer of the last layer used leaky relu[41] as activation function and the rest of the activation function are relu[24].

Models. There are six models used in the experiment. First, the single-label model is a first baseline of our experiments, in which the auxiliary task classifies representative label using softmax-cross-entropy as a loss function. The multi-label model is a deep learning model designed to classify multiple labels into multi-task learning using a sigmoid cross-entropy as a loss function. This is a second baseline of our experiment. Next, to find out the effect of applying centerloss, we added centerloss to the multi-label model. There are

Table 2: APK19000 Querying Results

Dataset	Querying trainset						Querying validset					
Metric	precision			weighted precision			precision			weighted precision		
k	top1	top10	top100	top1	top10	top100	top1	top10	top100	top1	top10	top100
weighted center(sum)	0.9846	0.9743	0.9374	0.9869	**0.9802**	**0.955**	0.8717	0.862	0.8272	0.8808	0.8746	0.8534
weighted center(avg)	**0.9856**	0.9772	**0.9481**	0.9855	0.9779	0.9528	**0.8929**	**0.8837**	0.8559	**0.8893**	0.8815	0.8591
center(sum)	0.9843	0.9737	0.9358	0.984	0.9733	0.9388	0.8736	0.8676	0.8373	0.8752	0.872	0.8462
center(avg)	0.987	**0.9776**	0.9474	0.9872	**0.9776**	0.9472	0.8845	0.8779	**0.8562**	0.8892	**0.8829**	**0.8631**
multilabel(baseline2)	0.9764	0.9617	0.9115	0.9752	0.9593	0.9094	0.8871	0.8713	0.8184	0.8869	0.8704	0.8236
singlelabel(baseline1)	0.9035	0.8727	0.8061	0.9004	0.8667	0.7946	0.8489	0.8231	0.7446	0.8434	0.8151	0.733

Table 3: PE1300 Querying Results

Dataset	Queyring trainset						Queyring validset					
Metric	precision			weighted precision			precision			weighted precision		
k	top1	top10	top100	top1	top10	top100	top1	top10	top100	top1	top10	top100
weighted center(sum)	1	1	0.9487	1	1	0.9329	0.9082	0.9075	0.8875	0.8816	0.881	0.8547
weighted center(avg)	1	1	**0.9613**	1	1	**0.9671**	**0.9203**	**0.9186**	**0.9029**	**0.9142**	**0.9085**	**0.8916**
center(sum)	1	1	0.9483	1	1	0.9319	0.9058	0.9058	0.8839	0.8845	0.8845	0.8487
center(avg)	1	1	0.9544	1	1	0.965	0.9179	0.9179	0.8999	0.9048	0.9048	0.888
multilabel(baseline2)	0.9877	0.9665	0.7949	0.9877	0.9668	0.7793	0.8913	0.8894	0.8033	0.8666	0.8676	0.7692
singlelabel(baseline1)	0.9647	0.9004	0.7397	0.9608	0.8961	0.7235	0.8865	0.8597	0.7805	0.8784	0.8413	0.7501

two models that combine semantic components as average and summation. And to see the effect of scoring, here is the weighted center loss model with the importance coefficient added as a constraint. This method also has two versions of average and summation models.

5.2 Querying Quantitative Test

In this part, we will analyze the effect of centerloss by comparing precision at top K value obtained by querying with malware samples for each model and class variance of representation vectors.

Metrics. The metrics used in quantitative evaluation are as follows. We used precision to evaluate the performance of MR system, weighted precision to evaluate scoring, and intra and inter class variance.

- **Precision** Let the labels of the querying sample q be $y_{q1}, y_{q2}, ...y_{q_m}$. Let the labels of the k search result samples $r_1, r_2, ..., r_k$ obtained from Ranking module R be $y_{r1}, y_{r2}, ..., y_{r_l}$ respectively . In this case, precision at top K is obtained as follows.

$$Precision_K = \frac{1}{N} * \sum_{i=1}^{N} \frac{\sum_{k=1}^{K} |Y_{q_i} \cap Y_{r_k}|}{\sum_{k=1}^{K} |Y_{r_k}|}$$

  where $|Y|$ is the number of elements of set Y.
- **Weighted precision**

  Weighted precision is calculated in the same way as precision, adding weighted importance coefficients of each label. This metric provides better results when model hits important labels.

$$WeightedPrecision_K = \frac{1}{N} * \sum_{i=1}^{N} \frac{\sum_{k=1}^{K} |Y_{q_i} \cap Y_{r_k}|_w}{\sum_{k=1}^{K} |Y_{r_k}|_w}$$

  where $|Y|_w = \sum_i c_{y_i}$ and c_j is importance coefficient of j'th label.
- **Inner class variance**

  Inner class variance represents the variance of the representation vector of samples in the same class. That is, it is a metric that measures how much the same classes are gathered together. Because our samples have a multi-label, we measure the variance for a sample with the same label combination and average it.
- **Inter class variance**

  Inter class variance is a metric that indicates how far apart other classes are. Likewise, the distances between sample vectors with different label combinations were measured and average it.

Sample querying results. From the results of Table. 2 and Table. 3, we were able to observe the following. First, the precision is better than the multi-label baseline model when using centerloss. Especially, as the K increases, precision value of the multi-label baseline model falls sharply, but the centerloss models do not. Second, the models with the importance coefficient added to the centerloss have better weighted precision results than the other models. Finally, we observed that using average as a combination of semantic components was slightly better than using summation method.

From the above experiments, we can confirm that the above quantitative values are generally higher when the MR system is

Table 4: Class Variances

Dataset	APK19000			PE1300		
Variance	intraclass	interclass	inter/intra	intraclass	interclass	inter/intra
weighted center(sum)	0.0473	2.793	59.0486	0.129	2.169	16.8140
weighted center(avg)	**0.0101**	**1.1906**	**117.8812**	**0.0361**	**1.2698**	**35.1745**
center(sum)	0.2268	5.2967	23.3541	0.1748	1.8299	10.4685
center(mean)	0.0258	1.2785	49.5543	0.1332	1.6826	12.6321
multilabel(baseline2)	2.4956	21.8435	8.7528	25833.582	840.54	0.0325
singlelabel(baseline1)	1.2103	15.18	12.5423	2310.7649	267.683	0.1158

Table 5: Auxiliary Task Results

Dataset	APK19000	PE1300
weighted center(sum)	0.9618	0.9801
weighted center(avg)	0.9636	**0.9812**
center(sum)	**0.9728**	0.9753
center(avg)	0.9705	0.9574
multilabel(baseline2)	0.9606	0.9495
singlelabel(baseline1)	0.9508	0.946

constructed using multi-label centerloss. This shows that the semantic understanding attribute of desired property of the MR system is more satisfactory than the other methods. We also showed that scoring is possible in MR systems using weighted centerloss with deep learning.

Class variances. We measured the class variance to determine the cause of the above results. The results can be seen in the table. 4. From this result, we confirmed that using centerloss reduces the inner-class variance. In addition, to measure the relative class variance of the models, the intra-class variance and the inter-class variance of the embedding vectors were measured and compared. The larger the value of the inter-class variance / intra-class variance is, the more discriminative the representation vectors are, meaning that retrieval results may be better.

Auxiliary task results. Table.5 represents the AUC for the validation set of the auxiliary task.The purpose of the original Auxiliary task is literally a subtask to learn the representation vector to be used in the MR system, but with the centerloss, we can see a slight performance improvement.

5.3 Querying Qualitative Test

In this part, we list the top k results obtained by querying the MR system we built in the three ways introduced in Section 4.1. Thus, our proposal shows that the performance of the MR system has improved. For each querying test, we can see the querying method in the figure. 4.

Querying by malware sample. A sample of the APK malware with a semantic component consisting of "agent'', "slocker'', "jisut'', and "ransom'' was queried to the learned MR system with the models described above. The retrieval result can be observed in Table. 6. The system learned by the centerloss (add) model is better than the other two models.

$$query = \mathbf{e_q}$$

Querying by semantic components. The next test is to querying the semantic components and retrieve samples. We tested the APK MR system learned by the centerloss (add) model. We used the sum of the semantic component vectors corresponding to "agent'', "ewind'', and "downloader'' as a query.

$$query = \mathbf{s}_{\text{agent}} + \mathbf{s}_{\text{ewind}} + \mathbf{s}_{\text{downloader}}$$

Results can be found in Table. 7. In our learning data set, there are 89 samples consisting of "agent'', "ewind'', and "downloader''. We could see that these samples were all searched in the samples up to the top 90.

Querying by the combination of semantics and sample. Finally, we tested to add or subtract some semantic components to the malware samples for querying. As in the previous two experiments, we tested for APK. The APK samples used in the experiment have four semantic components: "smsreg'', "agent'', "smspay'', "smssend''. Among these samples, the "smssend'' component was omitted and the "dropper'' component was added to create the query statement.

$$query = \mathbf{e_q} - \mathbf{s}_{\text{smssend}} + \mathbf{s}_{\text{dropper}}$$

In our learning data set, there are 56 samples consisting of four semantic components of "smsreg'', "agent'', "smspay'', and "dropper'', and all the samples are found in top 69 results.

6 FUTURE WORKS

The current method requires that all labels be correct. If not, the model can be vulnerable to noise. But getting the correct label in malware domain is not easy. Therefore, label noise-robust model should be created to estimate the real label of malware samples from features of malware through denoising. If we create MR system through this, we could make more semantic-aware malware information system. In addition, quick response to new samples is an important part of the MR system, but in case of deep learning model, it takes time to train. It is also our future work to solve by using continual learning or online update.

7 RELATED WORKS

Nataraj et al. [26] proposed a large scale malware search and retrieval system. In this paper, he proposes a method to retrieve fingerprints from malware image and retrieves similar samples

Table 6: Queyring by sample

Semantics of queried sample	top	center(sum)	multi	single
agent, slocker, jisut, ransom	1	agent, slocker, jisut, ransom	agent, slocker, jisut, ransom	slocker, jisut, ransom
	2	agent, slocker, jisut, ransom	agent, slocker, jisut, ransom	slocker, jisut, ransom
	3	agent, slocker, jisut, ransom	agent, slocker, jisut, ransom	slocker, jisut, ransom
	4	agent, slocker, jisut, ransom	agent, slocker, jisut, ransom	dropper, agent, slocker, jisut, ransom
	5	agent, slocker, jisut, ransom	agent, slocker, jisut, ransom	slocker, jisut, ransom
	6	agent, slocker, jisut, ransom	agent, slocker, jisut, ransom	dropper, agent, slocker, jisut, ransom
	7	agent, slocker, jisut, ransom	slocker, jisut, ransom	slocker
	8	agent, slocker, jisut, ransom	agent, slocker, jisut, ransom	slocker, jisut, ransom
	9	agent, slocker, jisut, ransom	slocker, jisut, ransom	agent, slocker, jisut, ransom
	10	agent, slocker, jisut, ransom	slocker, jisut, ransom	slocker, jisut, ransom

Table 7: Quering by semantics

Queried semantic components	top	centerloss(sum)
agent, ewind, downloader	1~86	agent, ewind, downloader
	87	agent, ewind, downloader, ransom
	88~90	agent, ewind, downloader
	91	agent, ewind, downloader, ransom
	92	agent, ewind, downloader, ransom
	93	downloader, agent, ewind, fakeinst
	94	ewind, downloader

Table 8: Querying by the combination of semantics and sample

Query		top	centerloss(sum)
Sample	Semantic components		
smsreg, agent, smspay, smssend	- smssend + dropper	1~34	smsreg, dropper, agent, smspay
		35	smsreg, dropper, smspay
		36~45	smsreg, dropper, agent, smspay
		46	mobilepay, smsreg, dropper,agent, smspay
		47~49	smsreg, dropper, agent, smspay
		50	smsreg, dropper, agent, smspay, dowgin
		51	smsreg, dropper, agent, smspay

through nearest neighbor search. Upchurch et al. [38] introduced a framework for detecting variant malware through similarity testing. This framework extracted the static features using BitShred, TLSH, sdhash, ssdeep, and compared them to determine whether they were similar malware. Palahan et al. [27] proposed a method of comparing similarities between malware by extracting significant malicious behaviors from the system call dependency graph and comparing them. Several neural IR models have been proposed in the document retrieval domain, and many models have been studied to obtain a good representation of text[5, 11, 21].

Chih-Kuan et al. [43] trained Canonical Correlated AutoEncoder models using label-correlation sensitive loss function as a way to obtain multilabel embedding. In this paper, he showes that the relationship between labels with dependency is trained end-to-end and solves the multi-label classification task, and proved to be effective for missing label problem of multilabel classification.

8 CONCLUSIONS

We observe that single-label model(baseline1) couldn't cope with malware retrieval task well. To solve this problem, we supervised multi-label and use our metric learning method, MCL. When using MCL(average) the performance is improved about 2% for top 1, 4% for top10, and 11% for top 100 than single-label model for the same valid set. Additionally we verified that our model could learn the semantics of malwares by observing intra-class and inter-class variances and results of other qualitative tests.

REFERENCES

[1] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*. Vol. 463. ACM press New York.

[2] Jinrong Bai, Junfeng Wang, and Guozhong Zou. 2014. A malware detection scheme based on mining format information. *The Scientific World Journal* 2014 (2014).

[3] Zhongqiang Chen, Mema Roussopoulos, Zhanyan Liang, Yuan Zhang, Zhongrong Chen, and Alex Delis. 2012. Malware characteristics and threats on the internet ecosystem. *Journal of Systems and Software* 85, 7 (2012), 1650–1672.

[4] Mihai Christodorescu, Somesh Jha, Sanjit A Seshia, Dawn Song, and Randal E Bryant. 2005. Semantics-aware malware detection. In *Security and Privacy, 2005 IEEE Symposium on*. IEEE, 32–46.

[5] Daniel Cohen and W Bruce Croft. 2016. End to end long short term memory networks for non-factoid question answering. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*. ACM, 143–146.

[6] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. 2008. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (Csur)* 40, 2 (2008), 5.

[7] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891* (2016).

[8] Lorraine Goeuriot, Gareth JF Jones, Liadh Kelly, Henning Müller, and Justin Zobel. 2016. Medical information retrieval: introduction to the special issue. *Information Retrieval Journal* 19, 1-2 (2016), 1–5.

[9] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 55–64.

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

[11] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. ACM, 2333–2338.

[12] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[13] Jiyong Jang, David Brumley, and Shobha Venkataraman. 2011. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*. ACM, 309–320.

[14] Xuxian Jiang, Xinyuan Wang, and Dongyan Xu. 2007. Stealthy malware detection through vmm-based out-of-the-box semantic view reconstruction. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*. ACM, 128–138.

[15] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. FastText.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651* (2016).

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

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.

[18] Arash Habibi Lashkari, Fereshteh Mahdavi, and Vahid Ghomi. 2009. A boolean model in information retrieval for search engines. In *Information Management and Engineering, 2009. ICIME'09. International Conference on*. IEEE, 385–389.

[19] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

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

[21] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1291–1299.

[22] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137* (2016).

[23] André Mourão, Flávio Martins, and João Magalhães. 2015. Multimodal medical information retrieval with unsupervised rank fusion. *Computerized Medical Imaging and Graphics* 39 (2015), 35–45.

[24] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 807–814.

[25] Lakshmanan Nataraj, Sreejith Karthikeyan, Gregoire Jacob, and BS Manjunath. 2011. Malware images: visualization and automatic classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*. ACM, 4.

[26] Lakshmanan Nataraj, Dhilung Kirat, BS Manjunath, and Giovanni Vigna. 2013. Sarvam: Search and retrieval of malware. In *Proceedings of the Annual Computer Security Conference (ACSAC) Worshop on Next Generation Malware Attacks and Defense (NGMAD)*.

[27] Sirinda Palahan, Domagoj Babić, Swarat Chaudhuri, and Daniel Kifer. 2013. Extraction of statistically significant malware behaviors. In *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 69–78.

[28] Younghee Park, Douglas Reeves, Vikram Mulukutla, and Balaji Sundaravel. 2010. Fast malware classification by automated behavioral graph matching. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*. ACM, 45.

[29] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[30] Alessandro Reina, Aristide Fattori, and Lorenzo Cavallaro. 2013. A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors. *EuroSec, April* (2013).

[31] Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. 2016. Using word embeddings for automatic query expansion. *arXiv preprint arXiv:1606.07608* (2016).

[32] Igor Santos, Xabier Ugarte-Pedrero, Felix Brezo, Pablo Garcia Bringas, and José María Gómez-Hidalgo. 2013. Noa: An information retrieval based malware detection system. *Computing and Informatics* 32, 1 (2013), 145–174.

[33] Joshua Saxe and Konstantin Berlin. 2015. Deep neural network based malware detection using two dimensional binary program features. In *Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on*. IEEE, 11–20.

[34] Markus Schedl, Emilia Gómez, Julián Urbano, et al. 2014. Music information retrieval: Recent developments and applications. *Foundations and Trends® in Information Retrieval* 8, 2-3 (2014), 127–261.

[35] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 373–382.

[36] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. 2014. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems*. 1988–1996.

[37] Virus Total. 2012. VirusTotal-Free online virus, malware and URL scanner. *Online: https://www. virustotal. com/en* (2012).

[38] Jason Upchurch and Xiaobo Zhou. 2015. Variant: a malware similarity testing framework. In *Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on*. IEEE, 31–39.

[39] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. 2014. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM, 157–166.

[40] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. 2016. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*. Springer, 499–515.

[41] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015).

[42] Lok-Kwong Yan and Heng Yin. 2012. DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis.. In *USENIX Security Symposium*. 569–584.

[43] Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. 2017. Learning Deep Latent Space for Multi-Label Classification.. In *AAAI*. 2838–2844.

[44] Jun Yu, Dacheng Tao, Meng Wang, and Yong Rui. 2015. Learning to rank using user clicks and visual features for image retrieval. *IEEE transactions on cybernetics* 45, 4 (2015), 767–779.

[45] Zhenlong Yuan, Yongqiang Lu, Zhaoguo Wang, and Yibo Xue. 2014. Droid-sec: deep learning in android malware detection. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 371–372.

[46] Mu Zhang, Yue Duan, Heng Yin, and Zhiruo Zhao. 2014. Semantics-aware android malware classification using weighted contextual api dependency graphs. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1105–1116.