Semantics-aware Representation Learning for Malware Information Retrieval

Anonymous Author(s)

ABSTRACT

멀웨어에서 쉽게 추출한 Structural Feature로부터 semantic similarity 를 계산하기는 쉽지 않다. 어떻게 하면 멀웨어의 의미를 이해하는 Malware Information Retrieval System을 만들 수 있을까? 우리는 이를 가능하게 하기 위해 딥러닝을 활용하여 Semantic-aware representation vector 를 학습할 수 있는 목적함수인 Multilabel Center Loss(MCL)을 제안한다. malware 샘플의 의미는 그 멀웨어가 갖고있는 악성 행위들의 조합을 통해 표현될 수 있다. MCL 로스를 통해, 이런 속성을 갖고있는 Malware Semantic space에 malware representation vector들을 임베딩하였다. 그리고 이를 통해 학습된 벡터들로 구축된 Malware Information Retreival System 이 멀웨어의 의미를 잘 이해하고 있다는 것을 정량, 정성 평가 결과를 통해 관측했다.

CCS CONCEPTS

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

KEYWORDS

Malware, Information Retrieval, Semantic Space, Deep Learning

1 INTRODUCTION

하루에 30만개 이상의 멀웨어가 생성되고 있고 이 숫자는 계속 늘어나고 있다. 같은 기능을 하지만 해시는 다른 멀웨어들을 자동으로 제너레이션 하는 멀웨어 obfuscation 툴들의 발전 및 배포가 그 이유이다.

반면 보안 전문가의 시간은 늘어나지 않고 있다. 보안 전문가의 업무 중 큰 부분을 차지하는 두 가지 일이 있다. 하나는 해시로 걸러지지 않은 악성코드의 카테고리를 분류하는 일이다. 두 번째는 주요 악성코드에 deep dive 해서 obfuscation 방법이나 악성 행위 방법을 공부하고 후에 나올 악성코드에 잘 대응할 수 있게 하는 것이다.

이 두 가지 주요 업무 중에서 머신러닝이 쉽게 도움을 줄 수 있는 부분은 두 번째 태스크이다. 사실상 수많은 new 악성코드들을 보안 전문가들이 하나하나 정적 혹은 동적 분석을 통해 분류하는 일은 불가능하다. 따라서 자동으로 멀웨어를 분류하는 시스템은 예로부터 많이 만들었다. PE 스트럭쳐나 콜그래프 등의 정적 피쳐를 받아서 SVM, RF, DNN 등으로 멀웨어를 분류하는 모델이 연구되어왔다.

하지만 이렇게 만들어진 분류기의 정확도가 만족스럽지 못하거나 새로운 샘플에 대해 대응을 잘 하지 못한다면 보안 전문가는 멀웨어 분류를 위해 학습된 머신러닝 모델을 사용하고나서 그 결과를 확인해야하는 수고를 해야한다. 그리고 그 확인 과정은 비슷한 멀웨어들을 검색해주는 시스템이 구축되어있지 않다면 수고스럽고 오래 걸린다.

따라서 classification model 을 검증하는데에 사용할 수 있으면서도 동시에 위에서 언급한 보안 전문가의 두번째 태스크인 deep dive 에도 도움을 줄 수 있는 멀웨어 IR System 에 대한 니즈가 있다. 멀웨어 IR System 은 자연어 혹은 비젼의 IR System 과 마찬가지로 의미적으로 비슷한 멀웨어 샘플들을 랭킹하고, retrieve 해 줄 수 있는 시스템을 말한다. 따라서 우리는 멀웨어의 의미를 담을 수 있도록 멀웨어 IR 시스템을 구축해야한다.

시스템이 Semantics - aware 라는 말은, 사람이 생각하는 멀웨어 샘플들간의 의미론적 관계가 시스템에 담겨있다는 뜻이다. 구조적으로 다르다고 해도 멀웨어의 행위가 일치하면 같은 의미를 갖는 샘플이라고 볼 수 있다. 마찬가지로 구조적으로 거의 같다고 해도 멀웨어의 행위가 전혀 다르다면 두 샘플은 다른 의미를 가졌다고 할 수 있다. 또한 샘플들 간 의미 차이 까지 고려될 수 있다면 그 시스템은 Semantics-aware한 시스템이라고 할 수 있다.

Traditionally Semantic-aware Feature 를 사용해서 Semantics 를 담는 시스템을 구축하고자하는 시도가 있었다. 가장 멀웨어의 행위를 잘 나타낼 수 있는 동적 피쳐는 모든 경우의 수를 탐색하기 어렵기에 얻기가 어렵다는 단점이 있다. 정적 분석으로 얻을 수 있는 Syntatic 피쳐로 멀웨어 샘플의 의미를 표현하기 위해 보안 전문가가 피쳐 엔지니어링을 해서 특정 악성 행위의 공통적인 패턴을 조금 더 잘 검출하는 hand-crafted 피쳐를 획득할 수 있었지만, 복잡한 의미 관계까지 표현하는 머신러닝 모델을 만들기 위해서는 다양한 도메인에서 증명된 딥러닝같은 특징 표현 학습 기법을 멀웨어 도메인에 적용하는것을 고려해야한다.

딥러닝은 보안 전문가에 의해 피쳐 엔지니어링 된 피쳐들 뿐만 아니라, malware 바이너리, 소스코드, 메타데이터, 리소스, 동적 행위로그, 스크린샷 등 엔지니어링 되지 않은 인풋들에 대해서도 같은 의미를 갖는 멀웨어 샘플들을 표현하기 위한 공통 레이턴트 피쳐 표현을 하이러키컬하게 배울 수 있기에 Generalization 관점에서 이점이 있다. 또한 풍부한 Capacity 로 복잡한 의미관계를 학습할 수 있다는 장점도 있다.

그렇다면 좋은 인풋 feature 와 딥러닝의 레이턴트 피쳐 표현 학습 능력이 있다면 멀웨어의 의미 관계까지 배울 수 있는가? 자연어 도메인에서는 의미 관계를 학습하기 위해서 문장에서의 앞 뒤 관계를 활용하거나[word2vec] 컨텍스트에 기반하여 co-occurence 를 확률적으로 표현한 메트릭스를 활용하는[GloVe?] 등, Self- supervised 방법을 활용한다. 하지만 멀웨어는 self-supervised learning 을 할 만한 요소가 없기 때문에, supervised learning 을 통해 의미 관계를 학습시키는 방법을 사용해볼 수 있다. 지금까지 알려진 대부분의 Deep Learning 기반 Malware Classifier는 어떤 파일이 어떤 malware family로 명명되어 있는 지를 분류하도록 가르친다. 하지만 이를 학습시켜서 얻은 representation latent feature 들은 거리가 가까우면 거의 동일한 멀웨어이고 거리가 멀면 동일하지 않은 멀웨어라는 것을 알 수 있을 뿐, 거리의 차이가 의미의 차이를 표현해주지는 않는다.

그러면 어떻게 해야 멀웨어의 시멘틱을 딥러닝으로 학습할 수 있는가? 하나의 멀웨어가 가질 수 있는 악성 행위 혹은 속성은 여러개가 될 수 있다. 멀웨어의 시멘틱을 악성 행위 혹은 속성들이

의미하는 시멘틱 컴포넌트들의 조합이라고 생각해볼 수 있다. 따라서 우리는 멀웨어의 태그들을 정제하여 딥러닝 모델에게 제공하였고 이를 통해 딥러닝 모델이 학습할 멀웨어의 Semantics를 지도해주었다. 학습 과정에서는 멀웨어의 표현 벡터를 시멘틱 컴퍼넌트 벡터들의 리니어 컴비내이션이 되도록 학습하여 특정 의미를 갖게 하였다. 또한 우리가 제안한 메트릭 러닝 방법을 통해, 벡터 스페이스에서의 멀웨어 샘플들 간 거리가 의미의 차이와 같아지도록 학습하였다. 그리하여 멀웨어 샘플의 의미(Semantics)를 벡터로 표현할 수 있는 멀웨어 시멘틱 스페이스를 얻을 수 있었다.

우리가 제안한 메트릭 러닝 방법은 센터로스[cite]를 어그멘트해서 설계한 Multilabel Centerloss Obejctive function을 크로스엔트로피 로스와 함께 사용하는 것이다. 이 로스 펑션의 역할은 멀웨어의 Representation vector 가 그것의 시멘틱스 벡터와 가깝게 하는 것이다. 이 것의 장점은 어디에든 쉽게 붙일 수 있다는 것이며 구현이 어렵지 않다는 것이다. 더불어 Information Retrieval Task에서 사용하던 Scoring 기법과 동이랗게, 특정 시멘틱 컴포넌트에 Weight를 부여하여 다른 컴포넌트들보다 랭킹에 큰 역할을 할 수 있도록 제약을 줄 수 있다.

이렇게 학습된 Semantic Space에서 의미적으로 유사한 멀웨어 또는 어떤 semantic의 차이를 갖는 malware, 혹은 특정 의미를 갖는 멀웨어들을 검색할 수 있는 유연한 IR system을 소개한다. 더불어 딥러닝을 활용하여 그리고 정량 평가와 정성 평가를 통해 시멘틱 스페이스가 얼마나 잘 만들어졌는지 확인한다.

우리의 메인 컨트리부션은 이렇게 요약할 수 있다.

- 우리는 악성코드의 Semantics를 벡터로 모델링하고 어떤 악성코드의 latent feature representation을 semantic components의 linear combination으로 표현할 수 있는 Semantics Space 위에 임베딩했다. 그리고 이를 학습하기 위해 Deep Learning 기반 Malware Classifier를 사용한다.
- 우리는 Malware Classifier가 학습하는 Semantic Space에서 metric function이 semantics의 차이를 의미하도록 학습하기 위해 multilabel centerloss에 기반한 Metric Learning 기법을 제안한다.
- 학습된 Semantic Space에 존재하는 멀웨어 샘플 벡터들에 대해 멀웨어, 시멘틱, 멀웨어와 시멘틱의 조합 세 가지 방법으로 malware를 검색할 수 있는 유연한 IR system을 소개하고 그 성능을 평가한다.

페이퍼 구성은 다음과 같다. 딥러닝, IR, Metric learning, Semantic Space 등 주요 배경 지식에 대한 설명하는 Background 가 섹션 2에 위치한다. 섹션 3는 멀웨어 IR 시스템에 대한 구조와 desired properties 를 정의한다. 섹션 4에서는 우리가 학습할 멀웨어 시멘틱 스페이스를 설명하고 이를 학습하기 위한 우리의 제안을 보여준다. 섹션 5에서는 실험에 쓰인 데이터셋과 뉴럴넷 하이퍼파라미터 설정들, 그리고 정량 평가와 정성 평가 실험 결과에 대해서 설명한다. 섹션 6에서는 related works 를, 섹션 7에서는 Future works 그리고 세션 8 에서는 conclusion 이 위치한다.

2 BACKGROUND

2.1 Deep learning

neural networks는 훈련 데이터의 통계적 특성을 기반으로 특징을 추출할 수 있는 machine learning 방법이다. 또한 DNN은 network 를 수 층의 hidden layers로 구성함으로써 hierarchical feature representation을 얻을 수 있다. 일반적으로 DNN은 input layer와 수 개의 hidden layer, output layer로 구성된다. Classification task의 경우, input layer는 task의 목적이 되는 sample의 feature vector representation을 입력으로 받는다. Output layer는 주어진 input vector에 관계된 class의 probability를 vector로 출력한다. 이를 통해 DNN은 input vector에 따른 class를 predict 할 수 있다.

특히 Image classification task의 경우 Convolutional network를 사용함으로써 모델의 성능을 향상시킬 수 있다.[ref?] Convolutional network는 convolutional layer와 pooling layer, 일반적인 neural network layer로 구성되며, convolutional layer의 경우 filter를 input에 대해 convoluting함으로써 feature를 추출한다. Pooling layer의 경우 영향을 적게 주는 feature를 down sampling하는 효과를 얻는다.

2.2 Information Retrieval

Information retrieval 은 본래 유저의 string 쿼리로부터 관련 문서를 찾는 태스크로 자연어 처리 분야에서 발전하였다. 하지만 이를 다른 도메인으로 확장시켜 이미지[1, 8], 음악[7], 의료[2, 4], 멀웨어[6] 등 다양한 분야에서 사용되고 있다.

Boolean Model. Information Retrieval System 을 만들기위한 모델로, Boolean model 이 제안되었다. boolean model 은 boolean 로직과 집합 이론을 배경으로 하여 문서들과 유저의 쿼리들을 집합으로 대하고 검색을 시행하는 모델이다. 이는 직관적이고 구현하기 쉽다는 장점이 있었지만, 유사도가 Discrete 하게 계산되어 리트리벌 결과가 아주 많아지거나 정확히 일치하지 않으면 검색되지 않는 현상이 일어나고, 모든 term 들이 똑같은 비중을 가지는 한계들이 있었다.[3?]

Vector space model. 또 다른 방법으로 Vector space model 이 있다. 이는 문서들과 유저의 쿼리를 모두 벡터스페이스에서 표현함으로써 벡터 간 거리가 곧 유사도가 되도록 하는 모델이다. 선형 대수에 기반한 간단한 모델로, 텀에 대한 가중치를 다르게 줌으로써 리트리벌 결과를 더 좋게 만들 수 있고, 쿼리의 유사도를 continuous Value 로 나타낼 수 있다는 장점이 있다.[5] 문서와 유저의 쿼리를 벡터로 만드는 방법[]과 벡터 간 거리를 측정하는 방법[]에 따라 여러가지 모델들이 존재한다.

Learn vector representation using deep learning. 앞서 설명했듯, 딥러닝을 사용하여 머신러닝 태스크를 수행하면 hierarchical representation 을 배울 수 있기 때문에 Generalization 이 더 잘 되는 장점이 있다. 따라서 Vector space model 을 이용하는 여러 태스크들과 딥러닝은 조합이 좋았고, 이로인해 많은 태스크에서 성능 상 breakthrough 를 이루어냈다.[refs : imagenet, word2vec, speech recognition, ...]. 더불어 Vector space model 을 사용하는 IR 분야 또한 딥러닝과 만나면서 한번 더 Breakthrough를 이루어냈다.[refs : document retrieval, Retrieval based QA, Image retrieval].

벡터 representation 을 학습시키는 방법으로는 supervised 와 unsupervised 가 있다. Unsupervised 로 가르치는 대표적인 방법은 Autoencoder를 이용하는 것으로, 이를 활용해서 멀웨어를 인코딩하려는 시도[ref]가 있었다. Supervised 로 가르치는 방법은 인간이 레이블을 주고 그 레이블로의 Classification 혹은 Regression 태스크를 수행하는 딥러닝 모델을 만들어, 정보를 가장 압축적으로 담고있는 bottleneck layer 의 아웃풋을 해당 샘플의 representation vector 로 보는 것이다. 이 방식은 vector representation 의 위치를 가르치는 사람이 어느정도 컨트롤 할 수 있게 된다. 예컨데 메트릭 러닝을 위한 objective function 을 사용해서 어떤 샘플들을 가깝게 보내고 어떤 샘플들을 멀게 보낼지를 결정할 수 있고[siamese, triplet, centerloss ...], 이는 one shot or few-shot learning 의 태스크를 수행할 수 있기 때문에, 벡터 간 distance 를 기준으로 랭킹을 결정하고 적은 샘플에도 대응할 수 있어야 하는 information retrieval 의 성능을 향상시킬 수 있다.

2.3 Metric Learning

Metric Learning 은 거리 함수를 학습하는 기법이다. 좀 더 자세히는, 메트릭 러닝을 통해서 어떤 샘플들을 서로 가깝게 볼 것이고 어떤 샘플들을 멀다고 볼 것인지, 혹은 어느정도 다르면 거리가 어느정도 될지를 결정짓는 거리 함수를 학습시킬 수 있다.

딥러닝을 활용한 메트릭 러닝의 대표적인 예는 Siamese Network 이다. 샤미즈 네트워크는 입력 데이터로 동일하다고 볼 샘플의 쌍과 동일하지 않다고 볼 샘플의 쌍을 주고, 동일하다고 볼 샘플의 쌍에 대해서는 특징 표현 벡터의 거리 차가 0으로, 그렇지 않은 샘플의 쌍에 대해서는 거리 차가 1 이상 으로 나도록 가르친다.

또 다른 예시로 Centerloss 가 있다. 이는 각 Single label classification Task 에서, class 별 Template vector 를 만들고 같은 클래스에 속하는 샘플들의 Representation Vector 들이 해당 클래스의 template vector 와 같아지도록 하는 목적 함수를 기존 Cross-entropy 로스에 추가하는 방식으로 학습된다. 이를 통해 intra-class variance 를 낮추어 representation vector 들이 Retrieval와 같은 태스크에 사용될 수 있도록 한다.

이렇게 메트릭 러닝으로 학습된 거리 함수는 유사한 데이터를 찾는 여러 실생활 태스크에 사용되는데, 대표적으로 Face recognition 이나 image retrieval, Text retrieval 등이 있다.

2.4 Semantic Spaces for Natural Language

Semantic spaces 를 만드는 것은 본래 자연어 도메인에서, 의미를 capture 할 수 있는 자연어의 표현을 만드려는 목표를 갖는다. Semantic Spaces의 오리지널 모티베이션은 자연어의 두 가지 챌린지에 있는데 하나는 다른 의미이지만 음이 같은 동음이의어(ambiguity)를 구분 가능하게 표현하는 것이고 다른 하나는 생김새(Syntactic)는 다르지만 의미가 같은 용어들을 동일하게 표현하는(Vocabulary mismatch)것이다.

최근에 뉴럴넷이 다른 새 접근법들과 조합되면서 주된 발전을 이뤘는데, 이를테면 word2vec, glove, fastText 등이 있다. 이렇게 자연어들에 대한 표현을 Semantic spaces 에서 잘 만들어 놓으면, 이를 문서 분류, 문서 검색, Question and Answering, 음성인식, 번역 등 여러 태스크에 사용할 수 있다.

3 MALWARE INFORMATION RETRIEVAL SYSTEM

이번 세션에서는 제안된 malware information retrieval system 의 동작과 구성 그리고 가져야하는 속성들을 설명한다. 우리의 시스템은 크게 학습과 검색(retrieval), 두 페이즈로 나뉜다. 학습 페이즈에서는 원할한 검색을 위해 시멘틱스를 반영하는 학습 데이터의 표현 벡터를 학습한다. 검색 페이즈에서는 유저로부터 입력된 멀웨어 혹은 태그 쿼리와 미리 저장해놓은, 학습된 표현 벡터들로부터 랭킹 결과를 출력한다. 이어지는 서브 섹션에서는 두 페이즈에서 사용될 노테이션들을 정의하고, 이들을 이용해서 각 페이즈에서 어떤 태스크를 행해야하는지를 자세히 설명한다.

3.1 Notations

우리는 먼저 학습 멀웨어 셋으로 $X = \{x_1, x_2, ..., x_N\}$ 를 갖고있다. 그리고 각 멀웨어에 해당하는 멀티레이블 셋 $Y = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$ 이 있다. 그리고 각 멀웨어로부터 손으로 추출된 피쳐들의 셋을 $V = \{v_1, v_2, ..., v_n\}$ 로 정의한다.

멀웨어 IR 시스템은 $[u, h, E, d, R]$ 5개의 원소를 갖는 tuple 로 정의된다. u malware 로부터 hand-crafted feature 를 추출하는 Feature extractor 모듈이다. 는 h 는 멀웨어 피쳐로부터 벡터 representation 을 얻을 수 있는 임베더 함수다. E 는 검색이 될 멀웨어의 임베딩 벡터 셋이다. d 는 입력된 쿼리와 임베딩 벡터 간 거리를 계산하는 함수이다. R 은 랭킹 모듈이다. 학습 페이즈에서는 적절한 h 를 학습하고 이를 통해 V로부터 E를 얻는다. 리트리벌 페이즈에서는 멀웨어 샘플 쿼리 q 를 입력받고 가장 가까운 k 개의 neighbor 를 랭킹 모듈 R 을 통해 반환한다.

Table 1: List of Symbols

Symbol	Meaning
X	Set of malwares
x_i	ith malware sample
v_i	Extracted features of ith malware
V	Set of extracted features
Y	Set of multilabels
$\mathbf{y}_i$	Multilabels of ith malware
u	Feature extractor
h	Neural embedder
θ	Parameters of neural embedder
g	Classifier
W, 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
$\mathbf{e}_q$	Query vector

3.2 Tasks

학습페이즈에서는 IR system 에서 사용하기 위한 벡터 리프레젠테이션을 얻기 위한 Auxiliary Task를 수행한다. Auxiliary Task 는 멀웨어 분류 태스크를 학습하는 것으로 싱글 레이블 클래시피케이션이나 멀티레이블 클래시피케이션이 될 수 있다. $f : V \rightarrow Y$ 인 f 는 멀웨어 피쳐로부터 레이블을 추정하는 함수이다. 함수 f 는 다시 g 와 h 의 합성 함수로 정의할 수 있다. h 는 위에서 설명한, 멀웨어 피쳐로부터 vector representation 을 얻을 수 있는 임베딩 함수이다. g는 임베딩으로부터 label 을 추정할 수 있는 클래시파이어이다. 시멘틱을 내포하는 임베딩 벡터를 학습시키는 방법에 대해서는 섹션4에서 자세히 이야기한다.

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

where

$$g(e_i) = \sigma(W * e_i + b)$$

리트리벌 페이즈에서는 멀웨어 샘플 쿼리 q를 입력받아 학습 페이즈에서 구했던 h 함수를 이용하여 임베딩한다. 임베딩된 쿼리 eq 와 E 의 원소들 간 거리를 d 함수를 통해 측정하고, 가까운 k 개의 neighbor 를 랭킹 모듈 R 을 통해 반환한다.

$$Results = \{x_j | j = argmin_i(-d(e_q, e_i))\}$$

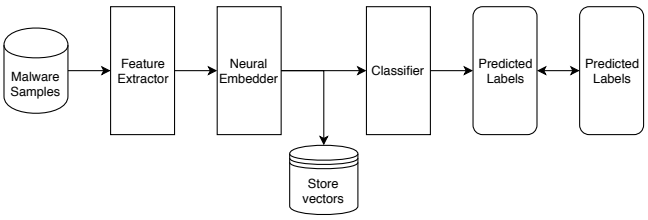

Figure 1: train phase

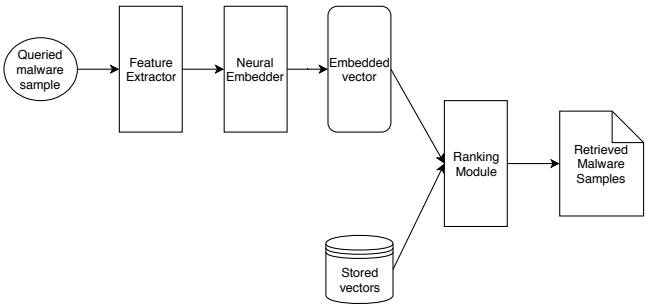

Figure 2: retrieval phase

3.3 Modules

두 페이즈의 구조도는 Figure 과 같다. 두 페이즈에서 공통으로 존재하는 모듈은 Feature Extractor와 Neural Embedder 가 있다. 벡터 학습 페이즈에는 Classifier 가, 리트리벌 페이즈에는 랭킹 모듈이 포함되어있다.

Feature Extractor. Feature extractor u 는 Malware Rawdata 로부터 Handcrafted feature 를 추출하는 모듈이다. Handcrafted feature로 PE 같은경우에는 Size 와 Entropy, Histogram of API Calls, … 등을[*] 추출할 수 있다. APK 같은 경우에는 추가적으로 Permission 등의 피쳐를 추출할 수 있다.

Neural Embedder. Neural embedder 는 Feature extractor 모듈에서 추출된 멀웨어의 피쳐들로부터 Representation vector 를 뽑는 모듈이다. theta 로 Parameterized 되어있는 뉴럴 네트워크이며, 파라미터는 벡터 학습 페이즈에서 Auxiliary Task 를 수행하면서 Optimize 된다. 그리고 리트리벌 페이즈에서 파라미터는 업데이트하지 않는다. Neural Embeder 의 네트워크 구조는 CNN 을 사용하였다. Thumbnail 을 받아서 CNN 5 층과 fully 2 층을 쌓은 구조를 사용한다. feature 의 성격에 따라 다른 네트워크 구조로 임베딩 피쳐벡터를 추출함으로써 더 나은 generalization 성능을 얻을 수 있다. thumbnail 피쳐는 [malimg 논문 인용] 악성 코드영역 혹은 악성코드를 특징짓는 영역의 로컬리 시프트 인베리언트한 특성을 담기 위해 CNN 을 사용해서 임베딩 피쳐벡터를 추출한다.

Classifier. Classifier 모듈은 Neural embedder 를 통과해서 나온 embedding vector 를 받아서 각 레이블 별 확률을 출력해준다. 1층의 fully connected 로 파라미터라이즈드 되어있으며 태스크에 따라 softmax 혹은 sigmoid Activation을 사용하였다. Neural Embeder 모듈과 마찬가지로 벡터 학습 페이즈에서 back propagation 에 의해 파라미터들이 학습되고, 리트리벌 페이즈에서는 업데이트되지 않는다.

Ranking Module. 학습 페이즈에 저장해두었던 학습 셋의 임베딩 벡터들과 추출된 벡터의 거리를 거리함수 d 를 사용하여 측정하고 k 개의 nearest neighborhood 를 가까운 순서대로 정렬하고 출력한다.

3.4 Desired properties

멀웨어 IR 시스템은 한계들이 존재했다. 특히 Raw binary file 로부터 좋은 피쳐를 뽑기가 수월하지 않고, intraclass variance 는 크고 innerclass variance 는 작으며 계속해서 변종과 새로운 종이 생겨나는 멀웨어 도메인의 특징으로 말미암아 좋은 멀웨어 IR 시스템이 가져야 할 속성들은 다른 도메인의 IR 시스템과는 조금 다르다.

Semantic understanding. 쿼링 샘플에 대해 구조적으로 비슷한 샘플 뿐만 아니라 의미적으로도 비슷한 샘플들을 랭킹하고 retrieve 할 수 있어야 한다. 여기서 의미가 비슷하는 것은 멀웨어의 행동 혹은 사람이 생각하기에 중요한 멀웨어의 속성들이 비슷하다는 것을 의미한다. 구조적으로 다르다고 해도 멀웨어의 행위가 일치하면 같은 의미를 갖는 샘플이라고 볼 수 있다. 마찬가지로 구조적으로 거의 같다고 해도 멀웨어의 행위가 전혀 다르다면 두 샘플은 다른 의미를 가졌다고 할 수 있다. 더불어 우리는 Locky 와 Cerber 라는 두 랜섬웨어 패밀리에 해당하는 샘플 간 유사도가 Locky 와 Coinminer 에 해당하는 두 샘플의 유사도보다 더 작다고 생각한다. 이러한 샘플들 간 의미 차이 까지 고려될 수 있다면 그 시스템은 Symantics-aware한 시스템이라고 할 수 있다.

Robustness to novel samples. 멀웨어의 새로운 변종들은 계속해서 나타나기 때문에 이에 빠르게 대응할 수 있어야 한다.

Efficiency. 세상에 존재하는 멀웨어 샘플 개수는 매우 많고 기하급수적으로 늘어나고 있다. 따라서 수많은 샘플들에서 k 개의 상위 랭크된 결과를 적절한 시간 내에 retrieve 하려면 랭킹 모듈이 효율적이어야한다.

4 SEMANTICS AWARE REPRESENTATION LEARNING

이번 섹션에서는 Auxiliary task 를 통해 학습된 malware vector representation을 멀웨어 시멘틱 스페이스 위에서 해석하기 위한 방법들을 설명한다. 먼저 우리가 정의한 멀웨어 시멘틱 스페이스가 무엇인지 설명한다.

4.1 Semantic Spaces for Malwares

Definition. Let $X = \{x_1, \ldots, x_n, \ldots, x_N\}$ be a set of malwares. Let E and 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 $\mathbf{c} = \{c_1, ..., c_k\}$ We call a vector 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 $d(e_i, e_j)$ for a metric of a set E and it means the function that defines a distance between semantics of two malwares.

The meaning of a learning malware semantic space. Malware semantic space 를 배운다는 것은 무슨 뜻일까. 언어를 예를 들어보자. 우리는 머릿속으로 어떤 단어들의 의미을 생각해볼 수 있다. 그 의미들의 관계를 생각해보자. 도시 이름들과 머신 러닝에

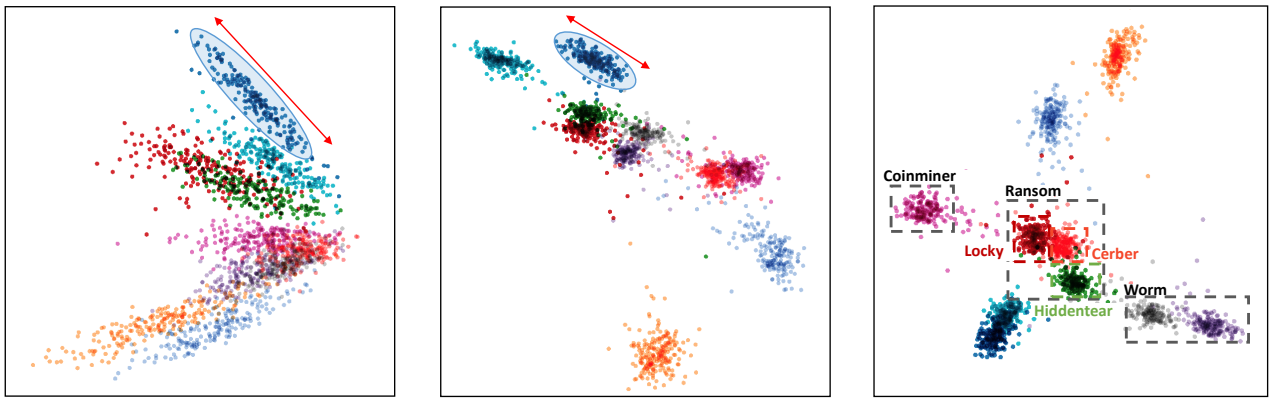

Figure 3: concept

관련된 단어들은 멀리 위치한다고 생각해볼 수 있다. 그리고 각각은 가까울 것이라고 생각할 수 있다. 단어들 간 관계의 유사성을 생각해볼 수 도 있다. King 과 Queen 의 관계는 Man 과 Woman 의 관계와 비슷하다는 생각을 해볼 수 있다. 이렇게 머릿 속에 존재하는 단어들의 의미를 최대한 approximate 해서 표현하려면, 벡터 스페이스 위에 개념들을 벡터로서 올려보는 방법이 있다. 의미가 가깝다면 벡터 간 거리가 가깝게, 의미가 멀다면 벡터 간 거리가 멀게 위치시킬 수 있다. king - queen = man - woman 와 같이, 벡터스페이스에서 정의된 연산을 이용해서 우리가 머릿속으로 생각하는 단어들 간의 관계를 모사(approximate)하여 표현해볼 수도 있다.

마찬가지로 우리는 머릿속에서 악성코드의 의미를 생각해볼 수 있다. 우리는 악성코드가 악성 컴포넌트들의 조합이라고 여기고, 이를 벡터 스페이스의 연산으로 모사(approximate)하여 표현하기 위해, basis 와 Linear combination 이라는 개념을 이용하였다. 즉, 멀웨어의 sematnic vector 가 semantic component vector의 선형 조합으로 표현되도록 가르치는 것은 우리의 머릿 속에 존재하는 Semantics 를 벡터 스페이스에서 최대한 따라하겠다는 것이고 이것의 의미가 바로 Semantic space 를 학습한다는 의미이다.

Evidences. 우리가 위에서 정의한대로 Semantic space를 잘 학습했다면 다음 쿼링 태스크들이 제대로 작동해야한다. 먼저 멀웨어 샘플로 쿼링해서 의미가 비슷한 샘플들을 검색할 수 있어야 한다. 둘째로, semantic components의 조합 만으로 검색할 수 있어야 한다. 이를테면, ransom, downloader, agent 세 가지 속성을 모두 갖는 멀웨어를 검색할 수 있어야한다. 마지막으로, 샘플과 semantic components 의 조합으로 검색할 수 있어야 한다. 이를테면, ransom, downloader 의 속성을 갖는 샘플과 agent 라는 semantic component 를 함께 검색하여 세 속성 모두 갖는 샘플들을 검색할 수 있어야 한다. 이 쿼리 방법들에 대한 그림 설명은 Figure.4 에서 확인할 수 있다.

4.2 Solution Overview

Multilabel Classification. 멀웨어 샘플의 의미를 반영하는 Vector representation E 를 학습시키기 위해 멀티레이블 분류 학습을 Auxiliary Task 로 선택하였다. 이 분류 태스크를 학습하기 위해 뉴럴 임베더 h는 표현 벡터들을 선형 분류기가 분류할 수 있도록 위치시킨다. 기존 대부분의 멀웨어 분류기를 학습하는 연구들은 하나의 레이블로 분류하도록 분류기를 학습한다.

하지만 멀웨어 도메인에서 하나의 레이블을 특정하고 이를 기준으로 가르치는 Auxiliary Task로 Malware IR System 을 만든다면, 학습하는 임베딩 벡터가 그 샘플의 의미를 담기에 부족한 점이 있다. 첫 째, 멀웨어에 레이블링을 하는 프로세스가 멀웨어의 의미를 표현하는데에 합리적이지 못하다. 멀웨어는 여러 악성 행위들을 동시에 하는 경우가 많다. 하지만 멀웨어를 대분류, 중분류, 소분류 등 하이러키컬한 분류 체계를 갖도록 레이블링 하는 경우가 대부분이다. 또한 네이밍 규칙이 글로벌하게 일관되지 못하고 심지어 로컬하게도 분석가들 마다 다른 기준으로 네이밍을 하는 경우가 많다. 유행하는 멀웨어에 대해서는 그 멀웨의 레이블을 붙일 때 그 멀웨어의 행위와 상관 없는 특징을 이름으로 붙이기도 한다. 이러한 레이블링 프로세스들은 머신러닝 모델이 멀웨어를 입력으로 받아서 레이블을 예측하는 Supervised Learning Task 를 수행할 때 멀웨어의 의미를 배우는 것을 방해한다.

둘 째는 하나의 레이블이 만약 Trojan이나 Adware 같은 대분류라면 같은 분류 안의 샘플들을 좀 더 세세하게 구분하여 information retrieval task 를 수행할 때에 정확하게 같은 의미의 샘플을 검색하기가 쉽지 않다. 샘플들의 벡터의 차이가 정확하게 어떤 의미인지를 알 수 없기 때문이다. 레이블이 만약 소분류라면 거의 동일한 샘플들만 같은 분류 안에 속해있을 가능성이 높고, 분류의 개수가 너무 많기 딥러닝 모델의 캐퍼시티가 충분하다면 샘플들을 외우고 하이러키컬 representation 을 학습하지 않아 Generalization 효과가 떨어지게 될 것이다. 따라서 정확하게 같은 샘플들 혹은 아주 약한 변종들이 검색될 수 있지만 의미가 비슷한, 동일하지 않은 샘플들에 대해서는 검색되지 않을 가능성이 높다.

우리는 같은 의미를 가진 스트럭쳐가 다른 샘플들 뿐만 아니라 비슷한 의미를 가진 멀웨어 샘플들에 대해서도 Retrieval이 가능하도록 하기 위해, 멀티 레이블을 분류하도록 Auxiliary Task 를 학습시키고, 거기에서 얻은 Vector representation 을 검색 시스템에서 사용한다. 이는 여러 의미 계층에 대한 공통 피쳐를 딥러닝 모델이 모두 학습할 수 있도록 도와준다.

Centerloss. 위의 방식대로 Multilabel Classification 을 학습시키면서 임베딩한 멀웨어의 Representation vector 들이 우리가 정의한 Semantic Space 위에 존재하도록 하려면 SingleLabel 센터로스를 발전시켜 만든 Multilabel Centerloss(MCL) 를 사용해야 한다. The first reason is about the variance of samples in the same class. 일단 기존 센터로스의 효과는 인트라 클래스의 베리언스를 낮춰주는 것이라고 백그라운드에서 설명했다. 약간 더 부연 설명을 하자면, 센터로스를 사용하면 inner class variance 가 너무 커서

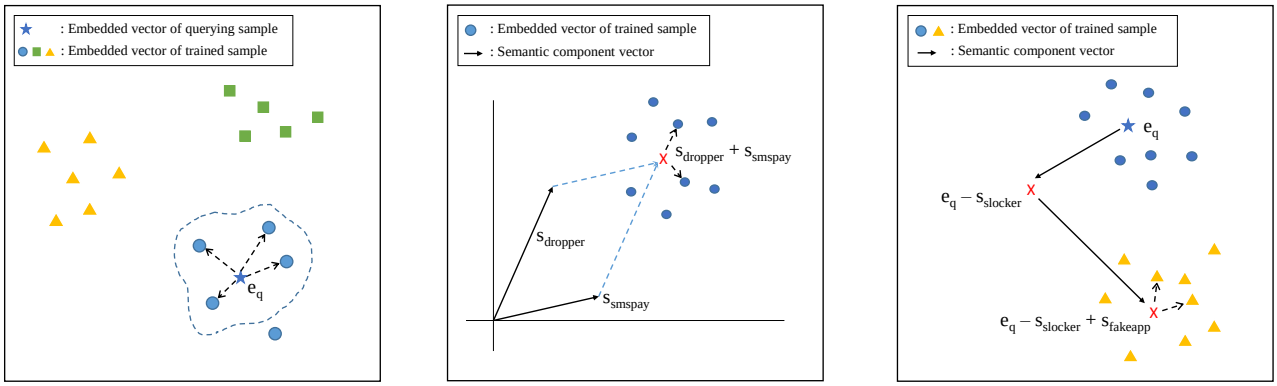

Figure 4: Queryings

ranking module 에서 같은 클래스 내 두 샘플의 거리가 서로 다른 클래스 내의 두 샘플 간 거리보다 더 커지는 현상을 막아주어 결과적으로 IR 결과를 향상시킨다.

두 번째 이유는 우리가 정의한 시멘틱 스페이스의 성질을 만족시키도록 멀웨어 representation vector 를 임베딩 시킬 수 있다는 것이다. 싱글레이블 센터로스를 제안했던 논문에서는 각 레이블별로 템플릿 벡터를 익스터널 메모리에 저장해놓고 그 템플릿 벡터와 보틀넥의 거리가 작아지도록 MSE 에러를 설정한다. 이는 곧 템플릿 벡터의 속성을 정의하지는 않았지만, 특정 클래스에 해당하는 샘플의 보틀넥은 그 클래스에 해당하는 템플릿 벡터가 되도록 가르칠 수 있다는 것을 의미하고 이 부분이 우리가 센터로스를 사용하여 우리의 문제를 해결해야겠다고 영감받은 부분이다. 마찬가지로 각 시멘틱 컴포넌트 별로 벡터를 익스터널 메모리에 저장해놓고, 보틀넥이 시멘틱 컴포넌트 벡터들의 linear combination 이 되도록 MSE 에러를 로스펑션으로 설정했다. 이 로스에 의해서 우리는 멀웨어의 시멘틱스 벡터가 시멘틱 컴포넌트의 리니어 컴비내이션으로 만들어지는 것을 학습시킬 수 있다.

Learn to Rank. 멀웨어의 레이블 중에는 우리가 중요하게 여기는 레이블들과 아닌 레이블들이 있다. 멀웨어 retrieval 시스템을 만들 때 이 중요도를 고려하는 것은 semantic understanding property 를 만족시키기 위한 중요한 요소 중 하나이다. 예를 들어 PE 포멧의 레이블들 중, agent 나 downloader보다 ransom, coinminer 등의 레이블에 더 가중치를 주어서 랭킹을 한다면 더 semantics-aware 한 검색 시스템이라고 여길 수 있을 것이다. 이렇게 어떤 시멘틱 컴포넌트에 대해서 더 잘 검색되게 할 지를 IR 시스템을 만들 때 반영하기 위해, 우리는 태그들의 중요도를 constraint 로 추가한 Centerloss 를 사용한다. 여기에서 Constraint 는 시멘틱 컴포넌트들의 Norm을 결정짓는다. 우리가 리트리벌에서 사용하는 Metric function 인 유클리디언 디스턴스는 거리를 계산할 때 두 벡터의 Norm 에 영향을 받는다. Norm 이 큰 semantic component vector 와 다른 컴포넌트 벡터와의 거리는 Norm 크지 않은 컴포넌트 벡터들간의 거리보다 더 크게 될 가능성이 높다. 따라서 euclidean distance 로 랭킹을 하는 IR 시스템에서는 중요도가 큰 semantic component 를 갖고있는 멀웨어를 검색했을 때, 해당 semantic component 를 포함하는 멀웨어 샘플들이 검색될 가능성이 더 높아지게 된다. 이는 기존 IR 시스템에서 특정 속성을 가진 문서들이 더 잘 검색되게 만드는 Scoring 기법 (ex. tf-idf) 과 같은 역할을 하는 것이라고 볼 수 있다.

4.3 Proposed Objective Functions

Overview 에서 설명했던 방법들로 말미암아 시멘틱 스페이스를 배우기 위한 새로운 objective function 을 제안한다.

Multilabel Center Loss(MCL). 우리가 풀고자 하는 constained optimization problem 의 식은 Eq.1와 같다. 이는 cross-entropy loss 의 근간이 되는 Negative Log Likelihood function 최적화 문제에 target semantic vector 와 representation vector 의 거리가 ϵ 미만이어야 한다는 constraint 를 추가한 것이다.

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

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.

위의 식을 만족시키는 parameters 를 찾기 위해 cross-entropy loss 와 multilabel centerloss 를 1: lambda 의 비율로 더해서 최종 loss function 을 만들었다[]. 우리가 제안하는 loss function 을 수식으로는 Eq.2과 같이 formulate할 수 있다.

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

where

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

Weighted Multilabel Center Loss(WMCL). 앞서 설명한 것처럼 우리의 Malware Information Retrieval System 에 Learn to rank 기법을 적용시키기 위해 Importance coefficients 를 각 시멘틱 컴포넌트 별로 부여한다. 기존 Multilabel Center Loss 에 semantic component vector 의 norm 과 importance coefficient 값이 같도록 하는 constraint 를 하나 더 추가하여 중요도를 반영하였다.

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

cross-entropy 로스는 multilabel classification 의 accuracy 를 높이기 위해 $\theta, W, \mathbf{b}$의 파라미터들을 업데이트시킨다. multilabel centerloss 을 통해 업데이트하는 대상은 크게 두 가지가 있다. 하나는 embedding vector 가 $\mathbf{s}_{\text{target}}$ 와 가까워지도록 $\theta, W, \mathbf{b}$ 의 parameter 들을 back propagation 을 통해 업데이트시킨다. 두 번째는 $\mathbf{s}_{\text{target}}$

가 embedding vector 와 가까워지도록 semantic compoenents 를 업데이트시킨다. 업데이트 할 때는 다음 step 에서의 $\mathbf{s}_{\text{target}}$ 가 지금의 $\mathbf{s}_{\text{target}}$ 와 임베딩 벡터의 α: $1-\alpha$ 내분점이 되도록, 두 벡터의 difference vector를 각 semantic compoenent vector 에게 균등하게 나눠 더해주어 업데이트 한다. semantic component vectors 를 더해서 target semantic vector 를 만드는 모델(summation model) 과 평균으로 만드는 모델(average model) 이 있다. 각 모델의 성능은 Section 5. 에서 확인할 수 있다.

파라미터와 semantic component update 의 과정은 Figure.5 와 Algorithm.1 그리고 Algorithm.2에서 확인할 수 있다.

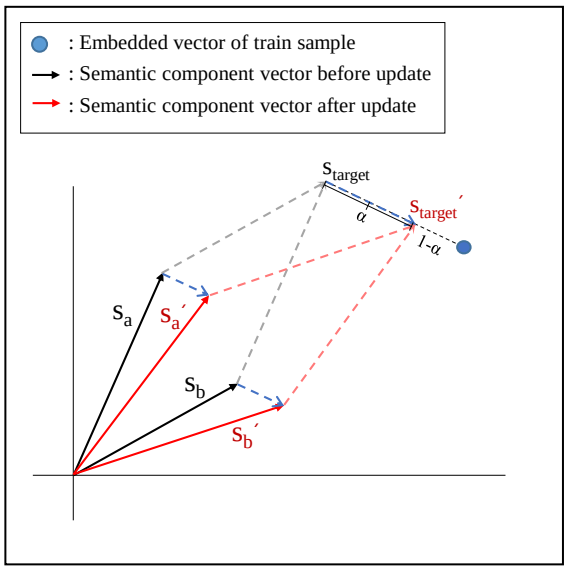

Figure 5: update

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 $\mathbf{s}$.Importance coefficients $\mathbf{c}$. Hyperparameter λ, α, learning rate μ.

Output: The parameters θ, W, b

repeat

s_{target} = `getTargetSemanticVectors`(s_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

5 EVALUATION

이번 섹션에서는 우리가 제안하는 멀웨어 IR 시스템의 성능을 여러 방법으로 평가하여, 앞에서 우리의 시스템의 강점이라고 주장했던 semantic-aware malware retrieval 태스크를 수행하였는지 확인한다.

5.1 Implementation and Setup

Datasets. 모델의 학습과 평가에 사용된 dataset은 VirusTotal로부터 [언제부터 언제까지] crawling된 샘플들 중 security expert가 검수하여 확진을 내린 샘플들로 구성되어있다. 또한 VirusTotal에서 제공하는 수십개 이상의 AntiVirus 엔진의 탐지결과로부터 Labeling을 자동화 하였다. VirusTotal로부터 각 AntiVirus 엔진들이 탐지해낸 결과를 의미있는 word 단위로 파싱하고, 샘플들로부터 얻어진 word token들을 대상으로 security expert가 의미있는 토큰과 그렇지 않은 토큰을 선별하는 작업을 수행하였다. 추가로, 의미 있는 토큰에 대해 중요도에 따라 우선순위를 매기도록 하였다. 이를 통해 학습 샘플에 대해 expert가 일일히 수동으로 label을 다는 시간을 줄이면서도 높은 accuracy의 label을 얻을 수 있다. 또한, 하나의 샘플에 대표 레이블과 멀티레이블 두 종류의 레이블을 부여했으며, 멀티 레이블에는 대표 레이블이 포함된다. train 셋과 테스트셋은 7:3으로 랜덤샘플링해서 나누었다. 그리고 대표 레이블에 대한 샘플의 분포는 유니폼하게 샘플링하였다.

- **Dataset 1. PE1300.** VirusTotal로부터 crawling된 샘플들 PE 5만여개 중 security expert가 검수하여 확진을 내린 PE 샘플 1300여개 11종의 labels. 이 중 8개는 대표 레이블이며 대표 레이블에 대한 샘플들의 수는 거의 같다.

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

- **Dataset 2. APK19000.** VirusTotal로부터 crawling된 샘플들 APK 8만여개 중 security expert가 검수하여 확진을 내린 APK 샘플 2만여개 83종의 labels. 이 중 19개는 대표 레이블이며 대표 레이블에 대한 샘플의 수는 1000개로 같다.

Hyperparameters. 실험에 사용되었던 하이퍼파라미터들은 다음과 같다. 옵티마이져로는 Adam optimizer 를 사용했고 learning rate 는 0.001 을 사용했다. Centerloss 에서의 α 는 0.1을 사용하였다. PE1300 데이터셋을 학습시킬때는 cross-entropy loss 와 centerloss 의 비율을 의미하는 λ 값으로 0.1을, APK19000을 학습할때는 0.001 을 사용하였다. 뉴럴네트워크의 구조로는 batch normalization을 사용한 CNN 5층 과 Fully Connected 2층이다. representation vector 를 output 으로 갖는 마지막에서 하나 전 층은 leaky relu(0.2) 를 사용하였고 나머지 activation function 은 모두 relu 를 사용하였다. representation vector 의 dimension 은 100을 사용하였다.

Models. 실험에 사용된 모델은 총 6가지이다. Single label 모델은 Loss function 으로 softmax-cross-entropy 를 사용하여 대표 레이블을 분류하는 Auxiliary Task 를 학습한 모델로, 우리 실험의 baseline 이다. Multi label 은 Loss function 으로 sigmoid cross-entropy 를 사용하여 학습시킨 모델로 여러 레이블을 multi task 로 분류하도록 설계된 딥러닝 모델로, 우리 실험의 두 번째 baseline 이다. 다음으로 Centerloss 를 적용시킨 효과를 알아보기 위해서 멀티레이블 모델에 centerloss 를 추가하였는데, 이는 semantic components 를 평균으로 조합하는 모델과 더하여 조합하는 모델이 있다. 그리고 scoring의 효과를 알아보기 위해 여기에 importance coefficient 를 constraint 로 추가한 weighted center loss 모델이 있다. 마찬가지로 average 과 summation 두 버젼이 있다.

Feature extraction 우리는 실험에서 PE, APK 공통으로 MalImage 논문에서 제안하는 Image feature를 사용하였다. 다만 APK 실험에서는, 멀웨어 파일의 압축을 풀고 classes.dex파일로부터 Image feature을 추출해서 사용하였다.

5.2 Querying Quantitative Test

이번 파트에서는 각 모델 별로 멀웨어 샘플로 Querying 하여 얻은 Precision at K 값, 그리고 representation vector 의 class variance 의 비교 통해 로스의 효과를 분석해본다.

Metrics. 정량 평가에 사용된 메트릭들을 소개한다. Malware IR System 의 성능을 평가하기 위한 Precision과, Scoring 을 평가하기 위한 Weighted Precision 그리고 intra, inter class variance 를 사용하였다.

- **Precision**

  Querying sample q의 Labels 를 $y_{q1}, y_{q2}, ...y_{q_m}$ 라고 하자. Ranking module R 을 통해 얻은 k개의 검색 결과 샘플 $r_1, r_2, ..., r_k$의 Labels 를 각각 $y_{r1}, y_{r2}, ..., y_{r_l}$이라고 하자. 이럴 때 Top K 개의 리트리벌 결과에 대한 Precision 는 다음과 같이 구해진다.

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

  where $|Y|$ is the number of elements of set Y.

- **Weighted Precision**

  Weighted Preicision 는 Precision 과 같은 방식으로 구하되, 각 label의 importance coefficient를 가중하여 더해준다. 이 메트릭은 중요하다고 여기는 레이블에 대해서 잘

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

검색해줄 수록 값이 높게 나온다.

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

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

- **inner class variance**

  Inner class variance 는 같은 클래스 내 샘플들의 representaion vector 의 Variance 를 나타낸다. 즉, 같은 클래스 끼리 얼마나 모여있는지를 측정하는 메트릭이다. 우리 샘플들은 Multi label 을 갖고 있기때문에, 같은 label 의 조합을 가진 샘플들에 대해서 Variance 를 측정하고 이를 평균 내었다.
- **inter class variance**

  Inter class variance 는 다른 클래스끼리 얼마나 떨어져있는지를 나타내는 메트릭이다. 역시 Multi label 이기 때문에, label의 조합이 다른 샘플들과 자신의 차이 벡터들의 Variance 를 측정하고 이를 평균내었다.

Sample Querying Results. Table.2 와 Table.3의 결과로부터, 우리는 다음의 관측을 할 수 있었다. 첫 째, Precision at K 값이 Centerloss 를 사용했을 때 Multilabel 베이스라인 모델보다 더 좋다. 특히 K 가 커질수록 multilabel 베이스라인 모델의 Precision 값은 급격히 떨어지지만 centerloss 모델들은 그렇지 않다. 둘 째, Weighted AP 의 값이 centerloss 에 importance coefficient 를 추가한 모델들이 아닌 모델들보다 더 좋다. 셋 째, semantic components 의 combination 방법으로 average 사용하는 것이 summation 를 사용하는것 보다 살짝 더 성능이 좋다는 것을 관측했다.

위의 실험으로부터 우리는 Multilabel Centerloss 를 사용하여 MIR System 을 구축했을 때 위의 정량 수치가 그렇지 않을 때보다 대체적으로 더 높다는 것을 확인할 수 있었고, 이는 MIR System 의 Desired property 1 번인 Semantic Understanding 속성을 다른 방법들에 비해 더 만족시킨다는 것을 알 수 있다. 또한 weighted centerloss 를 사용하여, 딥러닝을 이용하여 만든 IR 시스템에서도 scoring 이 가능하다는 것을 보였다.

Class variances. 우리는 위의 결과를 얻을 수 있었던 원인을 파악하기 위해 class variance 를 측정하였고 결과는 Table.4에서 볼 수 있다. centerloss 를 사용하면 inner class variance 가 줄어든다는 것을 확인했다. 더불어 모델들의 상대적인 class variance 를 측정하기 위해, 각 모델의 임베딩 벡터들의 intra class variance 와 inter class variance 를 측정하고 둘을 나누어서 비교하였다. interclass variance / intraclass variance 의 값이 클수록 representation vector 들이 discriminative 하다는 의미이므로 리트리벌 결과가 더 좋아질 수 있다는 것을 추론해볼 수 있다.

Auxiliary Task Results. Table.5 은 Auxiliary task 의 Validation set 에 대한 AUC 를 나타낸다. Auxiliary task 는 Malware IR System 에서 사용할 representation vector 를 학습시키기 위한 서브태스크이지만 centerloss 를 사용할 경우 약간의 성능 향상을 확인할 수 있었다. 이는 centerloss 를 사용함으로써 Discriminative feature learninig[**?**]을 수행하였고, 이를 통해 Generalization 효과를 얻은 것으로 해석해 볼 수 있다.

5.3 Querying Qualitative Test

이번 파트에서는 섹션4.1에서 소개한 세 가지 방법으로 우리가 구축한 MIR 시스템에 쿼링을 하여 얻은 top k 개의 결과를 리스팅한다. 그리하여 우리의 proposal 을 통해 Malware Information Retrieval System 의 성능이 좋아졌다는 것을 보여준다. 각 쿼링 테스스트는 Figure.4 의 왼쪽, 중앙 그리고 오른쪽 그림에서 쿼링 방법을 확인할 수 있다.

Querying by malware sample. agnet, slocker, jisut, ransom 이렇게 4개의 semantic component 를 갖는 APK 멀웨어 샘플을 model A, model B, 그리고 centerloss(summation) 모델로 학습한 MIR 시스템에 각각 쿼링하였다. 리트리벌 결과는 Table.6 에서 관측할 수 있다. 다른 두 모델보다 centerloss(summation) 모델로 학습한 시스템이 쿼링 결과가 좋다는 것을 확인할 수 있다.

$$query = e_q$$

Querying by semantic components. 다음 테스트는 semantic components 를 쿼링하여 그런 의미를 갖는 샘플들을 리트리브하는 것이다. centerloss(summation) 모델로 학습한 APK Malware Information Retrieval 시스템을 대표로 테스트했으며, 쿼리로는 agent, ewind, downloader 에 해당하는 semantic component vector 의 합을 사용하였다.

$$query = s_{\text{agent}} + s_{\text{ewind}} + s_{\text{downloader}}$$

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

결과는 Table.7 에서 확인할 수 있다. 우리 학습 데이터셋 안에 agent, ewind, downloader 세 semantic component 만을 갖고 있는 샘플은 총 89개이다. top 90 까지의 샘플 안에 모두 검색된 것을 확인할 수 있었다.

Querying by the combination of semantics and sample. 마지막으로, 멀웨어 샘플에 어떤 시멘틱 컴포넌트들을 더하거나 빼서 쿼링하는 테스트를 하였다. 위의 두 실험과 마찬가지로, APK에 대해 실험했다. 실험에 사용된 APK 샘플은 smsreg, agent, smspay, smssend 네 개의 시멘틱 컴포넌트를 갖고 있다. 이 샘플에서 smssend component 를 빼고 dropper component 를 더하여 쿼리문을 작성하였다.

$$query = e_q - s_{\text{smssend}} + s_{\text{dropper}}$$

우리 학습 데이터셋 안에 smsreg, agent, smspay, dropper 네 semantic components 를 가진 데이터는 총 56개이고, top 69 안에 모두 검색된 것을 Table.8에서 확인할 수 있었다.

6 FUTURE WORKS

현재 방법은 멀티 레이블이 다 정확해야 한다. 아니라면 모델이 noise 로부터 오해를 쉽게 한다. 하지만 이 도메인에서 정확한 레이블을 얻기란 쉽지 않다. 따라서 label noise-robust MLC 를 만들어 멀웨어의 feature 로부터 멀웨어 샘플의 리얼 레이블을 denoising 을 통하여 추정하고, 이를 통해서 IR 시스템을 만든다면 더 semantic-aware 한 Malware Information System 을 만들 수 있을 것이라 추정된다. 더불어 새로운 샘플에 대한 빠른 대응도 IR 시스템에서 중요한 부분인데, 딥러닝 모델이다보니 학습해야할 시간이 필요하다. continual learning 혹은 online update 를 사용해서 이를 해결하는 것도 우리의 Future work이다.

7 RELATED WORKS

Nataraj et al. 은 large scale malware search and retrieval system을 제안하였다. 이 논문에서는 malware image로 부터 fingerprints를 얻고, 이를 nearest neighbor search를 통해 비슷한 샘플을 retrieval 하는 방법을 제안하였다.

Upchurch et al. 은 similarity testing을 통해 variant malware를 탐지하는 framework를 소개했다. 이 framework는 BitShred, TLSH, sdhash, ssdeep 등의 방법으로 정적 feature를 추출하고 이를 비교함으로써 유사한 malware인지의 여부를 판단하였다.

Palahan et al. 은 system call dependency graph로부터 significant malicious behaviors를 추출하고 이를 비교함으로써 malware간의 similarity를 비교하는 방법을 제시했다.

Neural IR document retrieval domain에서 많은 neural IR 모델들이 제안되었고 그 중 많은 모델들이 text의 좋은 representation을 얻기 위해 연구되었다[1),2),3)].

Multilabel embedding Chih-Kuan et al. 은 multilabel embedding을 얻는 방법으로 label-correlation sensitive loss function을 사용하는 Canonical Correlated AutoEncoder 모델을 학습하였다. 논문에서는 dependency가 있는 label간의 관계를 End-to-End로 학습하고 이를 통해 Multi-label classification task를 해결하였고, 나아가 multilabel classification의 missing label 문제에 효과적임을 입증했다.

8 CONCLUSIONS

우리는 이 페이퍼에서, deep learning 을 활용하여 malware information retrieval system 을 만드는 방법을 소개했다. 그리고 시스템이 semantic understanding 속성을 갖게 하기 위하여, 제안된 Loss function 인 multilabel centerloss 를 사용하여 semantic space 를 approximate 하였다. 우리가 제안한 방법으로 만든 MIR 시스템에 멀웨어 샘플, 시멘틱 컴포넌트들, 멀웨어 샘플과 시멘틱 컴포넌트의 조합 이렇게 세 가지 방법으로 쿼리를 하였다. 그리고 정량, 정성 평가 결과를 통해 제안된 방법이 시스템을 semantic-aware 하게 만들었음을 보였다.

A LOCATION

Note that in the new ACM style, the Appendices come before the References.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (1975), 613–620.

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

[6] 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.

[7] 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.

[8] 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.