Computer Security Symposium 2018
22 - 25 October 2018

fastTextを用いたマルウェア亜種判別

岩元 遼太[1]　大久保 潤[1]

概要：本研究は機械学習の手法の一つであるサポートベクトルマシン（SVM）を用いて，未知のマルウェアがある既知のマルウェアの亜種であるかどうかを少ないデータ数で判定することを試みる．SVM を利用するためにマルウェアを多次元のベクトルとして数値化する必要がある．そのための手法として自然言語処理の分野において用いられている word2vec が，動的解析によって得られた API コール列の特徴表現を作るために利用できることは知られている．今回，word2vec を拡張した fastText がマルウェア亜種判別に有用であることを示す．

キーワード：マルウェア, 亜種型, fastText, word2vec, SVM

Identification of malware variants using fastText

Ryota Iwamoto[1]　Jun Ohkubo[1]

Abstract: This study attempts to judge whether an unknown malware is a variant of known ones or not with a small number of data by using support vector machine (SVM) which is one of machine learning methods. To perform the judgement, we need to characterize a malware with a multidimensional vector. In natural language processing research fields, word2vec is a famous method to characterize words and to make multidimensional vectors, and it has been shown that the word2vec is available to characterize API call sequence obtained by dynamic analysis of malwares. Here, we will show that fastText, which is an extension of word2vec, is useful to identify malware variants.

Keywords: malware, variant, fastText, word2vec, SVM

1. はじめに

McAfee 社のレポート [1] によると，2017 年第 4 四半期には平均で 1 秒あたり 8 つの新しいマルウェアが発見され，過去最高の 6340 万件が確認されている．この増加する要因の一つに亜種の存在がある．亜種とはあるマルウェアに対して，大枠の機能や動作は変更せず一部に変更を加えたものであり，ツールを用いることで誰もが容易に作成することができる．亜種型は元のマルウェアとハッシュ値が異なるため，従来のシグネチャベースによる検知は困難である．そこで現在では機械学習を用いて，マルウェアの特徴を機械が自動で学習することで，新種や亜種型の検知が可能であると期待されている．機械学習を用いる際，マルウェアを多次元のベクトルで数値化する必要がある．この特徴ベクトルは人手によって生成されるものであるため，先行研究では様々な手法が提案されている．

本研究では，未知のマルウェアがある既知のマルウェアの亜種型であるかどうかを判別することを目的として，機械学習の手法の一つであるサポートベクトルマシン （SVM）を用いることを前提とし，その SVM の入力として用いる特徴ベクトルの生成方法を提案する．亜種型であると判定できれば，既存の知識や解析手法を用いることができたり，マルウェアのグループ分けができたりと効率的に解析することが可能となる．本研究における中心となるアイデアは，自然言語処理の分野で単語ベクトルを生成する手法である

[1] 埼玉大学理工学研究科
Graduate School of Science and Engineering, Saitama University
[2] 埼玉大学理工学研究科
Graduate School of Science and Engineering, Saitama University

word2vec[2], [3] を拡張した fastText[4], [5] を用いて，マルウェアの特徴ベクトルを生成することである．具体的には，本研究で使用するデータセットを収集するために用いられたオープンソフトウェアである Cuckoo Sandbox[6] による動的解析から得た API コール列をコーパスとし，fastText によって各 API コールのベクトルを生成する．そして得られたベクトルを用いることで，人手が介在せず自動でマルウェアの特徴ベクトルを生成できる．本研究の重要な提案は，API コール列からコーパスを作成する方法と，API コールのベクトルからマルウェアの特徴ベクトルを生成する方法の 2 点である．

2. 関連研究

先行研究 [7] では，動的解析によって得た API コール列をコーパスとして word2vec で学習した後，学習した各 API コールのベクトルからマルウェアの特徴ベクトルを生成し，亜種を判別している．word2vec はニューラルネットワークを用いたモデルで，Skip-gram と CBoW(Continuous Bag of Words) の 2 つの学習モデルがある．Skip-gram は**図 1** に示すように現在の単語から周辺の単語を推測するモデルで，CBoW は**図 2** に示すように Skip-gram のモデルを逆にしたもので，周辺の単語から現在の単語を推測するモデルである．ここで w_t は t 番目の単語，V は語彙数，N は単語ベクトルの次元数，C は推定したい単語数を表す．

word2vec は単語のベクトルの次元数を自分で決めることができ，単語間の関係もベクトルによって表せることから n-gram よりも低次元でかつ精度が高いという利点がある．しかし word2vec は学習した単語以外はベクトルで表すことができないため，未知のデータに未知の単語が含まれている際は対応できない．そのため本研究では未知の単語にも対応できるようにするために fastText を用いる．

3. 提案手法

本章ではマルウェアの特徴ベクトルの生成方法について述べる．

3.1 fastText

fastText は単語のベクトル表現を作ることと，テキスト分類することの 2 つを高速にしたモデルである [4], [5]．このモデルは word2vec を拡張したもので，基本となる学習モデルは word2vec と同様に Skip-gram と CBoW の 2 つである．word2vec では周辺の単語からのみ現在の単語を推定していたが，fastText ではそれに加えて単語の内部の情報も利用し，活用形も考慮できることでより精度の高い単語ベクトルを得ることができる．例えば “goes” は，“go”，“oe”，“es” というサブワードに分解され，それらが持つベクトル大きさの和の平均で表される．そうすることで，“goes” は “go” の情報も持つことができる．また，未知の単語についても word2vec では対応できない一方で，fastText では学習の際に用いたサブワードの情報を用いることで生成できる利点がある．本研究では fastText を使用し，動的解析によって得られた API コール列をコーパスとして各 API 関数を単語として学習する．学習モデルである CBoW と Skip-gram では Skip-gram の方が精度が高いとされていることから，本研究では Skip-gram を用いる．fastText で学習するためのコーパス作成にあたって各マルウェアの API コール列を一つのデータにまとめる必要があり，それによって文脈がおかしくなる場所が出てしまい，単語の精度が落ちる可能性がある．そこで文脈がおかしくなることを緩和するために，**図 3** に示すように各データの先頭の文字の前には始点の記号 “S” を，末端の文字の後には終点の記号 “E” を用いたコーパスの生成方法を提案する．

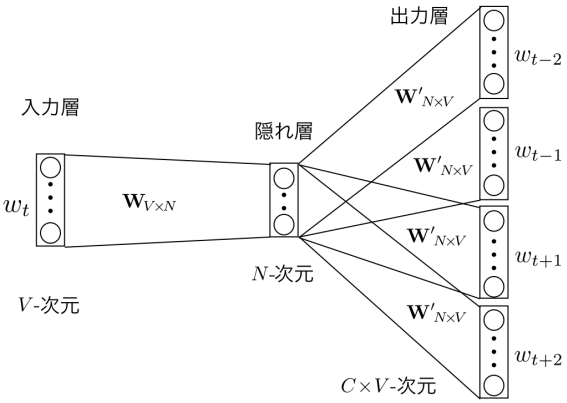

図 1 Skip-gram のモデル
Fig. 1 The model of Skip-gram

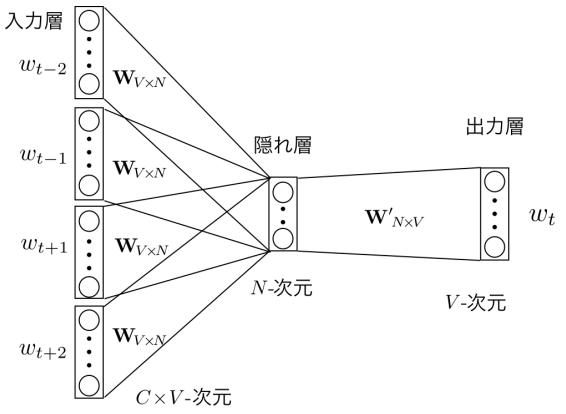

図 2 CBoW のモデル
Fig. 2 The model of CBoW

3.2 特徴ベクトル生成

fastText から得られた API 関数を用いてマルウェアの特徴ベクトルを生成するにあたって，2 つの生成手法を提案する．1 つ目は BoW(Bag of Words) のように，マルウェア内で使用された API 関数に対応するベクトルについて，重

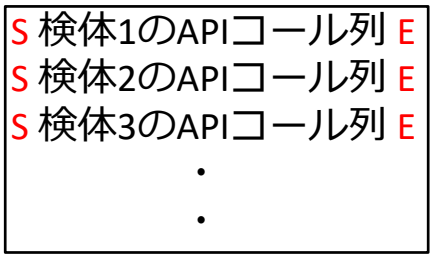

図 3　コーパスの生成方法
Fig. 3　How to create a corpus

複を許してすべて足し合わせて平均を取ったものを特徴ベクトルとする手法，2 つ目は同じ API 関数が複数回使用された場合でも足し合わせるのは 1 回だけとして平均を取ったものを特徴ベクトルとする手法である．例えば，あるマルウェアの API コール列を “A B C A”(A，B，C は任意の API 関数) とし，$v(\cdot)$ を fastText で得られた API 関数のベクトルをすると，1 つ目の手法は，

$$\frac{v(\mathrm{A}) + v(\mathrm{B}) + v(\mathrm{C}) + v(\mathrm{A})}{4}$$

で表し，2 つ目の手法は，

$$\frac{v(\mathrm{A}) + v(\mathrm{B}) + v(\mathrm{C})}{3}$$

で表す．

以降，1 つ目の手法を “BoW” とし，2 つ目の手法を “Bit BoW” とする．また，本研究では “BoW” や “Bit BoW” で特徴ベクトルを生成する際に，始点の記号 “S” と終点の記号 “E” のベクトルも加算に含めた場合も考慮する．

3.3　サポートベクターマシン (SVM)

3.2 節の手法によって得られた特徴ベクトルを SVM の入力として与えることで，推定したい亜種マルウェアごとに分類器を生成する．ラベル付けの方法については 4.2 節で述べる．SVM は本来線形識別器であるが，カーネル関数を用いることで非線形の識別器にも対応できる．本研究ではカーネル関数にガウシアンカーネルを採用し，グリッドサーチにより SVM に必要な 2 つのハイパーパラメータであるコストパラメータ C とガウシアンカーネルのパラメータ γ を最適化したものを用いる．また，SVM の実装は Python のオープンソース機械学習ライブラリである scikit-learn[8] を用いた．

4.　実験環境

本研究の学習の流れを図 **4** に示す．

4.1　データセット

本研究では MWS2018 の研究用データセット [9] の一部として FFRI 社が提供している FFRI Dataset 2018（含む FFRI Dataset 2013 – 2017）を用いる．このデータセットは，株式会社 FFRI が独自に収集した PE(Portable Executable) 形式かつ Windows プラットフォーム上で実行可能な検体を Cuckoo Sandbox を用いて動的解析したログである．

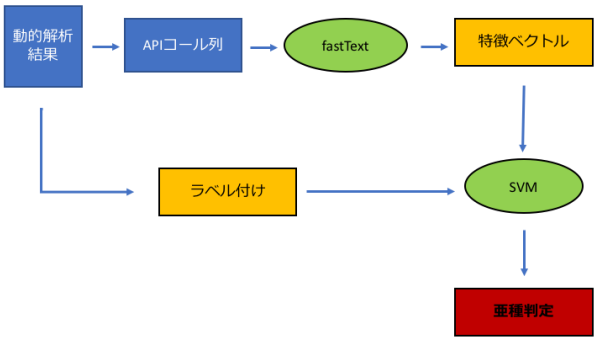

図 4　学習の流れ
Fig. 4　Flow of learning

図 5　Kaspersky によるマルウェア名
Fig. 5　Malware name by Kaspersky

4.2　ラベル付け

各マルウェアの名前付けをするために，マルウェアの検出精度が高く，先行研究においても利用されることが多い Kaspersky[10] の検知結果 (図 **5**) を用いる．Prefix と Variant は，それぞれ検体を検知したサブシステムと亜種の名前で，検体によっては存在しない場合がある．Behaviour はトロイの木馬やランサムウェア，バックドアといった検体の挙動，Platform は検体の実行環境（OS），Name は公式の検体のファミリー名を示す [11]．本研究では Behaviour.Platfom.Name をマルウェアのファミリー名とし，同じファミリー名で Variant が異なるマルウェアを亜種型と定義する．

本研究では SVM が少ないデータで学習できることから，FFRI Dataset 2013 – 2015 のみを用い，その中で亜種が多く存在する上位 11 種類のファミリーを実験に用いる．ただし Name が Generic となっている検体は多く存在するが「その他」として扱われるため，本研究の実験対象から除く．**表 1** は本研究に用いる 11 種類のマルウェアファミリー名を示す．

データセットは推定したい亜種マルウェア（マルウェアのファミリー名）をランダムに 100 検体，それ以外のファミリーのマルウェアを 100 検体（残りの 10 種類のファミリーからランダムに 10 検体ずつ) 集めたものを 11 通り（マルウェアのファミリー数分) 用意する．そして，これらの

表 1　マルウェアファミリー名
Table 1　Names of malware family

マルウェアファミリー名
Trojan-Spy.Win32.Zbot
Hoax.Win32.ArchSMS
Worm.Win32.WBNA
Trojan-Win32.Yakes
Trojan-Win32.Jorik
Trojan-Win32.Inject
Trojan-Win32.Agent
Worm.Win32.Vobfus
Trojan-Ransom.Win32.Foreign
Backdoor.Win32.Androm
Trojan-PSW.Win32.Tepfer

表 2　特徴ベクトル作成方法ごとの F 値
Table 2　F-measure for each method in feature vector case.

次元数 50	word2vec (BoW)	word2vec (Bit BoW)	fastText (BoW)	fastText (Bit BoW)
提案 1	0.9175	0.9029	0.9212	0.9148
提案 2	0.9172	0.9266	0.9237	0.9267
提案 3	0.9116	0.9265	0.9270	0.9373

表 3　提案 3 を用いた次元数ごとの特徴ベクトルの F 値
Table 3　F-measure for each dimensional feature vector case using method3.

	fastText(BoW)	fastText(Bit BoW)
次元数 (10)	0.9187	0.9305
次元数 (50)	0.9270	0.9373
次元数 (100)	0.9180	0.9319

データセットを用いて，10 分割交差検証法によって各ファミリーごとの性能評価値を算出する．すなわち，各ファミリーの合計 200 検体を亜種型とその他のマルウェアの比率が同じになるようにランダムに 10 分割し，そのうち 180 検体で学習をし，20 検体で検証する．最後にすべての実験結果の平均値を取ることで性能評価を行う．

5. 実験結果

本研究では性能評価値として F 値を用いた．また性能比較のため，図 4 の fastText を word2vec に置き換えた実験も実施した．

表 2 は，word2vec や fastText で各 API コールのベクトルを次元数 50 で学習した後，“Bow” や “Bit Bow” で各マルウェアの特徴ベクトルを生成したときの F 値を表す．提案 1 は word2vec や fastText の用いるコーパスを，各マルウェアの API コール列を連結して一つのデータにまとめたものを用いたもので，提案 2 は各 API コール列を連結する際に，各データの先頭の文字の前には始点の記号 “S” を，末端の文字の後には終点の記号 “E” 入れたコーパスで学習したものである．また，提案 3 はマルウェアの特徴ベクトルを生成する際に，そのマルウェアに含まれる API コールのベクトルに加えて始点の記号 “S” と終点の記号 “E” のベクトルも加算に含めたものである．表 2 より，word2vec よりも fastText を用いた手法の方が F 値が高く，特に提案 3 の特徴ベクトルの生成方法が高い値を示している．**表 3** は次元数を変えた際の提案 3 における F 値を表しており，10 次元でマルウェアの特徴ベクトルを生成しても F 値が高く，また “Bow” と “Bit Bow” の生成手法では “Bit Bow” を用いたほうが F 値が高かった．

以上の結果から，fastText を用いると未知のマルウェアにおいて学習には使用されなかった API コールが使用されている場合にも対応できるだけでなく，word2vec よりも精度も向上することが示された．また，マルウェアの特徴ベクトルの生成方法では，提案 3 と “Bit Bow” を用いたものが最も精度が高かった．

6. まとめ

本研究では API コール列を自然言語処理の手法である fastText に適用して，マルウェアの特徴ベクトルを生成する手法がマルウェア亜種を判定することに有用であることを示した．加えて，fastText で学習する際のコーパスの作成方法や，API コールを用いたマルウェアの特徴ベクトルの生成方法にも考慮することで精度が向上することを示した．fastText は低次元のベクトルでも精度が高く，n-gram や word2vec とは異なり未知の単語についても対応できることから，API コール列以外にも文脈があるものであれば fastText を用いると精度が上がる可能性がある．

本研究ではマルウェアの API コール列のみを用いていたので，今後の方針として動的解析によって得たその他の関数の戻り値や引数，マルウェア内に含まれる文字列等も特徴として用いることでさらなる精度向上を目指す．また提案したマルウェアの特徴ベクトルがクラスタリングにおいても有用かどうかも試みる．

参考文献

[1] “McAfee 脅威レポート 2017 年第 2 四半期”, https://www.mcafee.com/enterprise/ja-jp/assets/reports/rp-quarterly-threats-jun-2018.pdf, 2018.

[2] T.Mikolov, K.Chen, G.Corrado and J.Dean, “Efficient Estimation of Word Representations in Vector Space.”, In Proceedings of Workshop at ICLR, arXiv:1301.3781, 2013.

[3] T.Mikolov, K.Chen, G.Corrado and J.Dean, “Distributed Representations of Words and Phrases and their Compositionality.”, Neural Information Processing Systems(NIPS), pp.3111–3119, 2013.

[4] A.Joulin, E.Grave, P.Bojanowski and T.Mikolov, “Bag of Trick for Efficient Text Classification.”, arXiv preprint arXiv:1607.01759, 2016.

[5] P.Bojanowski, E.Grave, A.Joulin and T.Mikolov, “Enriching Word Vectors with Subword Information.”, arXiv preprint arXiv:1607.04606, 2016.

[6] “Cuckoo Sandbox”, http://www.cuckoosandbox.org.
[7] 佐藤 拓未, 武部 嵩礼, 後藤 滋樹, “Word Vector を用いたマルウェアの亜種判別法.”, 早稲田大学修士論文, 5-35(2017-01-30).
[8] “SVM scikit-learn”, http://scikit-learn.org/stable/modules/svm.html.
[9] 高田雄太, 他: マルウェア対策のための研究用データセット – MWS 2018 Datasets –, 情報処理学会, Vol.2018-CSEC-82, No.40, 2018 年 7 月.
[10] “Kaspersky”, http://www.kaspersky.com.
[11] “Rules for naming”, http://securelist.com/threats/rules-for-naming.

付　　録

A.1　3 次元プロット

付録では fastText によるマルウェアの特徴ベクトルの生成方法が低次元においても精度が高かったことから，可視化できる 3 次元を用いた場合について紹介する．

表 A·1 から 3 次元を用いた場合，F 値はわずかに下がったものの，fastText では 0.9 以上と高い値を示していることから，3 次元を用いた場合でも，fastText によって作られた API に対応する特徴ベクトルが，マルウェアの解析に有用であることがわかる．実際に F 値が最も高かった提案 3 と “Bit BoW” を用いたマルウェアの特徴ベクトルをプロットしてみた結果が**図 A·1** と**図 A·2** であり，赤色が判別したいマルウェアの亜種型で青色がその他のマルウェアを表す．図 A·1 は判別に用いたマルウェアの中で F 値が 0.9693 と最も高かったときのマルウェアをプロットしたもので，下部に亜種型，上部にその他のマルウェアが多く集まっていることから SVM による分類が比較的容易であると考えられる．一方で図 A·2 は F 値が 0.8474 と最も悪かったときのマルウェアをプロットしたもので，亜種型の集まりがいくつかあるものの，SVM による分類がしづらいと考えられる．本研究では特徴ベクトルをマルウェアごとに平均化してしまったが，API コールに対応する特徴ベクトルを個別に調べることで，さらに詳細にマルウェアの振る舞いを分析できる可能性がある．word2vec や fastText は言葉の意味のようなものを含めた抽出し，特徴ベクトルを作成する技術である．可視化しやすい 3 次元でも十分な分類性能を有することが判明したため，この知見が今後，API コール同士のベクトルの関係性を調べるなど詳細な検討を実施する際に役立つことが期待される．

表 A·1　3 次元の特徴ベクトル作成方法ごとの F 値
Table A·1　F-measure for each method in three-dimensional feature vector case.

次元数 3	word2vec (BoW)	word2vec (Bit BoW)	fastText (BoW)	fastText (Bit BoW)
提案 2	0.8673	0.8823	0.9060	0.9123
提案 3	0.8672	0.8793	0.9006	0.9138

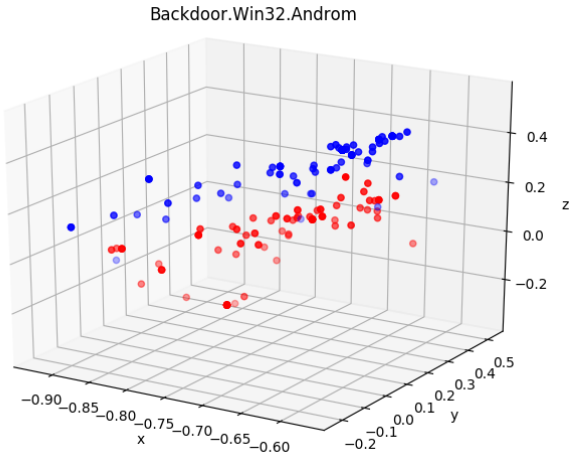

図 A·1　F 値が高いマルウェアの特徴ベクトル
Fig. A·1　Feature vector of malware with high F-measure

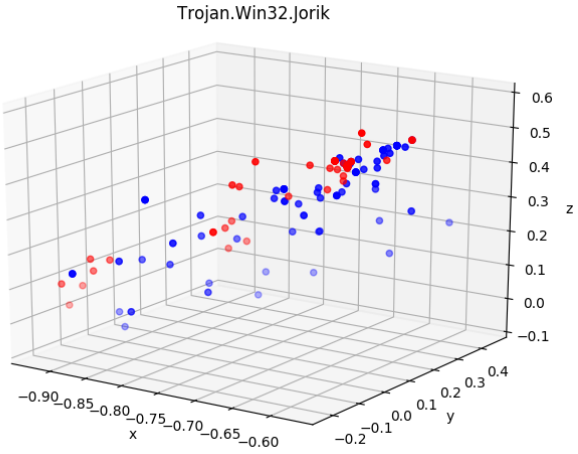

図 A·2　F 値が低いマルウェアの特徴ベクトル
Fig. A·2　Feature vector of malware with low F-measure