Computer Security Symposium 2018
22 - 25 October 2018

出現頻度の高い可読文字列を用いた未知のマルウェアの検知手法

伊藤 良[1,a)]　三村 守[1,b)]　田中 秀磨[1,c)]

概要：マルウェアの解析において，実行ファイルから取得可能な可読文字列は補助的に用いられている．近年の自然言語処理技術の発展に伴い，可読文字列のみを用いたマルウェアの検知手法が提案された．その手法では，すべての可読文字列を単語に分割し，良性と悪性の実行ファイルで出現する単語の違いを利用している．そのため，出現頻度の低い余計な単語を除外することで，検知精度が向上すると考えられる．本研究では，良性および悪性の実行ファイルから出現頻度の高い単語を均等に抽出し，精度を向上させる手法を提案する．検証実験では年代別の FFRI Dataset を用いて精度を評価し，F 値は 0.982 以上に改善することを確認した．

キーワード：実行ファイル，マルウェア，検知，機械学習，自然言語処理，Bag-of-Words，Doc2Vec，LSI

Extracting Frequent ASCII Strings to Detect Unseen Malware

Ryo Ito[1,a)]　Mamoru Mimura[1,b)]　Hidema Tanaka[1,c)]

Abstract: The ASCII strings extracted from executable files are helpful for analyzing malware. With the recent development of natural language processing technology, a detection method with these strings was proposed. The method divides these strings into words, and distinguishes the difference of the words between benign and malignant executable files. These strings, however, contain uncommon words. Therefore, we consider that reducing the uncommon words improves the detection rate. In this paper, we propose a method that extracts frequent words to improve the detection rate. In our method, the frequent words are selected with the same number of both words. Our experiment shows that the F-measure achieves more than 0.982 with the FFRI Dataset.

Keywords: executable file, malware, detection, machine learning, SVM, natural language processing

1. はじめに

今日，特定の個人や組織が所有する機密情報等を盗み出すための手段の一つとして標的型攻撃が行われている．標的型攻撃には，実行ファイル形式のマルウェアを添付した電子メールが用いられることが多い [1], [2]．標的型攻撃を行う攻撃者は，標的とした受信者に対してマルウェアを添付した電子メールを送信する．受信者が攻撃者から送信された添付ファイルを実行した場合，受信者の端末はマルウェアに感染する．この結果，攻撃者は感染した端末の制御を奪い，標的が保有している組織の情報を盗み出すことが可能となる．

ウィルス対策ソフトにおいて，標的型攻撃に使用されるマルウェアを検知できる可能性は低い．その理由は，攻撃者が最新のパターンファイルを適用したウィルス対策ソフトで検知されないことをあらかじめ確認し，その未知のマルウェアを用いることが多いためであると考えられている．ウィルス対策ソフトがマルウェアを検知できるようにするためには，そのマルウェアに対応したパターンファイルを検体から作成する必要がある．しかしながら，新しい

[1] 防衛大学校情報工学科
National Defense Academy in Japan
[a)] em57031@nda.ac.jp
[b)] mim@nda.ac.jp
[c)] hidema@nda.ac.jp

マルウェアに対応したパターンファイルを作成するまでには時間を要する．パターンファイルで検知できない未知のマルウェアに対しては，サンドボックスでファイルを動作させ，その挙動を確認することでマルウェアを検知する手法がある．この検知手法は解析に時間がかかるため，大量のファイルを処理する用途には適しているとは言いがたい．また，環境の構築や維持にかかる費用が高額であるという欠点もある．そのため，高速かつ簡易な未知のマルウェアの検知手法が必要とされる．

近年の自然言語処理技術の発展に伴い，機械翻訳や検索エンジン等において使用者の意図した出力ができるようになってきている．たとえば，機械翻訳では，翻訳する単語に対してその前後の単語関係から最適な意味を表現する翻訳語を選択することで，翻訳された文章を自然な文章に変換できるようにしている．自然言語処理技術をマルウェアの検知に応用した先行研究には，実行ファイルから取得した可読文字列を用いてマルウェアを検知する手法がある [9]．表層解析で取得可能な可読文字列は，マルウェアを検知するための補助的な手段として用いられてきた．この手法では，正常な実行ファイルおよび悪性の実行ファイルから取得した可読文字列を，分かち書きをして単語に分割する．これら全ての単語をコーパスとし，良性および悪性に出現する単語の出現頻度の違いをマルウェアの検知に用いている．そのコーパスには，出現頻度が低く，分類にはあまり貢献しない単語も含まれている．したがって，出現頻度の高い単語を均等に抽出してコーパスを作成すれば，良性あるいは悪性の区別が容易となり，マルウェアの検知率が向上すると考えられる．本研究では，良性および悪性の実行ファイルからそれぞれユニークな単語を，出現頻度の高い順に均等に抽出してコーパスを作成し，検知率を向上させる手法を提案する．良性および悪性の実行ファイルから取得した可読文字列を自然言語処理技術である言語モデルにより特徴ベクトルに変換する．Support Vector Machine(SVM) を用いた分類器を訓練データから生成した特徴ベクトルで訓練し，テストデータから未知のマルウェアを検知して精度を評価する．

本論文の構成を示す．第 2 節では，本研究で用いる自然言語処理技術について説明する．第 3 節では，関連研究について説明し，提案手法と先行研究の差異を明確にする．第 4 節では，提案手法の詳細について述べる．第 5 節では，検証実験とその結果について述べる．第 6 節では，考察を述べる．最後の第 7 節では，本研究のまとめと今後の課題を述べる．

2. 自然言語処理技術

この節では，本研究で使用する自然言語処理技術について説明する．自然言語処理技術では，自然言語をコンピュータで処理するために数値に変換する．本研究の提案手法では単語の出現頻度に着目するため，Bag-of-Words および LSI を使用する．また，先行研究と検知率を比較するために Doc2Vec を用いる．

2.1 Bag-of-Words

Bag-of-Words(BoW) は，ある文書を単語の出現頻度を元に表現することでベクトルに変換すモデルである．d を文書，w を単語，n を w に対応した出現頻度として表した場合，文書 d は (1) 式で表すことができる．

$$d = [(w_1, n_{w_1}), (w_2, n_{w_2}), (w_3, n_{w_3}), ..., (w_j, n_{w_j})] \quad (1)$$

この (1) 式を元に n の位置を固定することで単語 w を省略すると，d を数値で表現することができる．これにより各文書 $\hat{d}_i$ における単語の出現数を表した文書と単語の関係性をベクトル（文書-単語マトリクス）で表現できる．

$$\hat{\mathbf{d}}_\mathbf{i} = (n_{w_1}, n_{w_2}, n_{w_3}, ..., n_{w_j}) \quad (2)$$

BoW を用いることで，各文書はユニークな単語数で固定された次元数のベクトルに変換される．

2.2 Latent Semantic Indexing

Latent Semantic Indexing(LSI) は，文書群と文書に含まれる単語群の関連性を分析する自然言語処理技術である．LSI では，BoW により求めた文書-単語マトリクスの各成分に重み付けをしたベクトルに対して特異値分解を行うことで，文書の特徴を計算し関連性を分析できる．最後に導き出された関連性は，潜在的意味を示すとされている．

文書-単語マトリクスの各成分の重み付けをする技術には，term frequency - inverse document frequency（tf-idf）を用いる．tfi-df は，$n_{i,j}$ を文書 i における単語 j の出現頻度，$|D|$ は総文書数，$\{d : d \ni t_j\}$ を単語 i を含む総文書数を用いた場合，(3) 式で表すことができる．

$$tf_{i,j} * idf_i = \frac{n_{i,j}}{\sum_k d_{k,j}} * \log \frac{|D|}{\{d : d \ni t_j\}} \quad (3)$$

次に，tfi-df による重み付けされたベクトルに対して特異値分解を行う．文書群である行列 X の成分 (i, j) は，単語 i の文書 j における tf-idf 値を表すとする．この行列の行は 1 つの単語に対応したベクトルを示し，各成分が各文書との関係を示している．同様に，この行列の列は 1 つの文書に対応したベクトルを示し，各成分が各単語との関係を示している．X を分解して直交行列 U と V，対角行列 Σ が存在すると仮定する．このような分解が特異値分解である．単語の相関と文書の相関を与える行列積は，下記の行列式で表すことができる．この行列式の U は，列直交行列であり列ベクトルに関して 1 次独立である．そのため，U が文書ベクトル空間の基底となっている．この各列ベクトルが潜在的意味を示す．

$$X = \begin{bmatrix} x_{1,1} \ldots x_{1,j} \\ \vdots \ \ddots \ \vdots \\ x_{i,1} \ldots x_{i,j} \end{bmatrix} = U \Sigma V^T$$

$$= \begin{bmatrix} u_{1,1} \ldots u_{1,r} \\ \vdots \ \ddots \ \vdots \\ u_{i,1} \ldots u_{i,r} \end{bmatrix} * \begin{bmatrix} \sigma_{1,1} \ldots \ 0 \\ \vdots \ \ddots \ \vdots \\ 0 \ \ \ldots \sigma_{r,r} \end{bmatrix} * \begin{bmatrix} v_{1,1} \ldots v_{1,r} \\ \vdots \ \ddots \ \vdots \\ v_{j,1} \ldots v_{j,r} \end{bmatrix}$$

2.3 Doc2Vec

Doc2Vec は，Paragraph Vector[3] の実装の 1 つである．Paragraph Vector は，Mikolov らによって提案された単語の特徴ベクトルを生成するモデルである word2vec[4] を拡張し，単語ではなく文書の特徴ベクトルを生成するモデルである．word2vec は，文書中の単語の前後関係をニューラルネットワークを用いて学習させ，その単語の特徴を表すベクトルに変換する．$qyeen = king - man + woman$ は，word2vec で生成した各単語ベクトルを用いた演算の例である．Mikolov らは，word2vec に用いるコーパスの単語が多い場合，単語ベクトルの次元数も大きくなり，単語ベクトルによる演算結果の精度が高くなることを確認している．高精度な分類を行うためには，コーパスに用いる単語数に応じて単語ベクトルの次元数を調整する必要がある．word2vec を拡張した doc2vec にも同様のことが言える．word2vec を実装するためのニューラルネットワークは，CBOW モデルと Skip-gram モデルが提案されている．Paragraph Vector には，CBOW モデルを拡張した Paragraph Vector with Distributed Memory(PV-DM) モデルと Skip-gram モデルを拡張した Paragraph Vector with Distributed Bag of Words(PV-DBOW) モデルが提案されている．

3. 関連研究

本研究では，表層解析を用いて実行ファイルから取得可能な可読文字列を用い，自然言語処理技術により単語の出現頻度から特徴ベクトルを作成し，SVM を用いて未知のマルウェアを検知する．以下，本研究に関連する先行研究について述べる．

文献 [6], [7] は，実行ファイルから取得した可読文字列からマルウェアに含まれる可読文字列との共通部分を抽出し，その類似性を用いてマルウェアファミリを分類する手法を提案している．本研究では，実行ファイルに含まれる可読文字列の出現頻度の違いを用いており，マルウェアを検知することを目的としている．

文献 [5] は，実行ファイルから取得した可読文字列を用い，SVM によりマルウェアを検知およびファミリを分類する手法を提案している．この文献では，複数の SVM 分類器を用いた多数決によるマルウェアの検知およびファミリの分類をしている．本研究では，取得した文字列の特徴ベクトルに変換する手法として，自然言語処理技術である LSI および Doc2Vec を用いており，単独の SVM を用いてマルウェアの検知を行う．

文献 [8] は，実行ファイルから英単語辞書に含まれている可読文字列を取得し，tf-idf により特徴ベクトルに変換して SVM によりマルウェアを検知する手法を提案している．この文献では，特徴ベクトルに用いる可読文字列を英単語辞書に含まれる単語に限定している．本研究では，実行ファイルから取得する可読文字列を制限していない．一見，意味のないように思われる単語にも焦点を置くことにより，単語の出現頻度の違いを特徴とした検知精度の向上を目指している．

文献 [9] は，実行ファイルに含まれる全ての可読文字列を Doc2Vec により特徴ベクトルに変換し，ＳＶＭを用いてマルウェアを検知する手法を提案している．この文献では，全ての単語をコーパスとし，可読文字列を Doc2Vec により特徴ベクトルに変換している．そのコーパスには，出現頻度の低い余計な単語が多く含まれている．そのため，余計な単語を除外することで検知精度が向上すると考えられる．本研究では，良性と悪性の実行ファイルからそれぞれユニークな単語を，出現頻度の多い順に均等に抽出してコーパスを作成し，検知率を向上させることを目指す．

4. 提案手法

本節では，本研究で提案する未知のマルウェアの検知手法について詳細を述べる．図 1 に，提案手法の概要を示す．既知の実行ファイルおよび未知の実行ファイルには，それぞれ正常な実行ファイルと悪性の実行ファイルが含まれている．以下，提案手法の概要を図 1 の番号順に示す．

1. 既知の実行ファイルから可読文字列を抽出する．
2. 抽出した可読文字列を単語に分割し，高い出現頻度の単語を選択してコーパスを作成する．
3. 自然言語処理技術を用いて言語モデルを構築し，可読文字列から特徴ベクトルに変換する．
4. 可読文字列から生成した特徴ベクトルを用い，SVM の分類器を訓練する．
5. 未知の実行ファイルから可読文字列を抽出する．
6. 可読文字列から 2. で選択した単語を抽出し，単語を削減する．
7. 3. で構築した言語モデルを用い，可読文字列を特徴ベクトルに変換する．
8. 4. で訓練した分類器を用い，変換した特徴ベクトルを分類する．

以下，各手順の詳細を述べる．

4.1 可読文字列の抽出

図 1 中の「1　可読文字列の抽出」では，実行ファイルに含まれる可読文字列の抽出を行う．可読文字列は，Strings

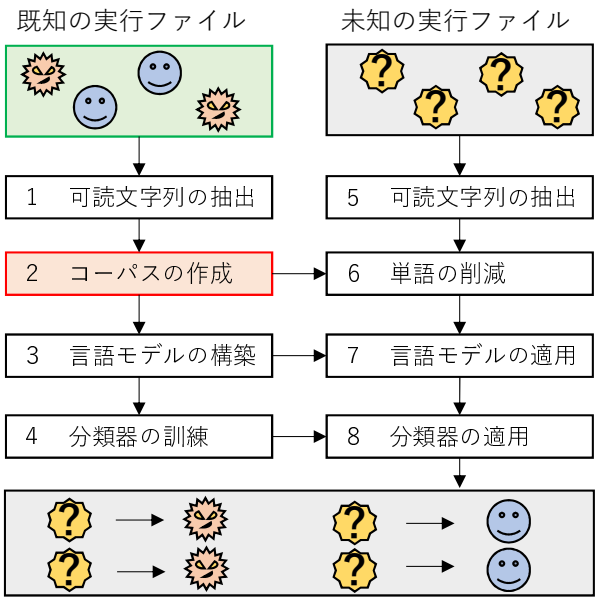

図 1　提案手法の概要

コマンド [15] を用いて実行ファイルから抽出可能であり，アルファベットと記号で構成されている．可読文字列は文章のように連なっているが，英語と同様に空白で分かち書きすることにより単語に分割する．可読文字列に含まれている単語には，英単語辞書にない単語も多く含まれている．

4.2　コーパスの作成

図 1 中の「2　コーパスの作成」では，実行ファイルから抽出した単語からコーパスを作成する．全てのユニークな単語をコーパスとして検知に用いる従来の手法に対し，本研究では単語を選択してコーパスを作成する．

良性および悪性の実行ファイルからそれぞれユニークな単語を，出現頻度の高い順に均等に抽出してコーパスを作成する．コーパスで使用する最適なユニークな単語の数は，予備実験により決定する．予備実験の方法は，次節の検証実験で説明する．

4.3　言語モデルの構築

図 1 中の「3　言語モデルの構築」では，既知の実行ファイルから取得した可読文字列を用いて言語モデルを構築する．提案手法で使用する自然言語処理技術は，LSI および Doc2Vec の 2 種類である．LSI では，BoW および tf-idf を用いた特異値分解により可読文字列から特徴ベクトルに変換する言語モデルを構築する．Doc2Vec では，PV-DM モデルを用いた分散表現により，可読文字列から特徴ベクトルに変換する言語モデルを構築する．

4.4　分類器の訓練

図 1 中の「4　分類器の訓練」では，訓練データにより教師あり学習モデルである SVM の分類器の訓練を行う．分類器の訓練に用いる訓練データは，既知の実行ファイルから作成する．訓練データは，Doc2Vec または LSI を用い変換した各特徴ベクトルに，良性または悪性のラベルを付与したものである．

4.5　未知の実行ファイルからマルウェアの検知

図 1 中の未知の実行ファイルを，実際に良性または悪性の実行ファイルに分類する手順を示す．まず，未知の実行ファイルから Strings コマンドを用いて可読文字列を抽出する (5)．抽出した可読文字列から，既知の実行ファイルから生成したコーパスに含まれる単語を選択し，可読文字列に含まれる単語を削減する (6)．次に，LSI または Doc2Vec の言語モデルを用い，可読文字列を特徴ベクトルに変換する (7)．最後に，既知の実行ファイルで訓練された分類器を用いて，特徴ベクトルを良性または悪性に分類してマルウェアを検知する (8)．

4.6　実装

提案手法の実装には，python2.7 を用いた．python には機械学習モデルと自然言語処理技術を実装したライブラリが存在しており，提案手法の実装が容易である．

機械学習ライブラリには，SVM，ランダムフォレスト等を含む scikit-learn[10] を使用した．本研究で使用する SVM は，パターン識別用の教師あり機械学習モデルの 1 つである．SVM のセットアップに必要な制御パラメータ C，カーネル関数 γ 等は，従来手法 [9] と比較するため初期値を使用する．

自然言語処理技術のライブラリは，Bag-of-Words，LSI，Doc2Vec 等を含む gensim[11] を使用した．特徴ベクトルの次元数である LSI の *num_topics* および Doc2Vec の *size* は，従来手法と同条件とするために 100 と設定した．Doc2Vec の他のパラメータについては，従来手法と比較するため，$epochs = 30$ の他は初期値とした．

5.　検証実験

検証実験では，正常な実行ファイルおよび悪性の実行ファイルから可読文字列を抽出した良性データおよび悪性データを準備し，未知のマルウェアを作為した時系列分析を行う．未知のマルウェアを作為するため，訓練データとして 2013 年の悪性データを使用し，テストデータには 2014 年〜2017 年の悪性データを使用した．以下，検証実験の詳細を述べる．

5.1　データセット

検証実験で使用するデータセットは，正常な実行ファイルから作成した良性データと株式会社 FFRI が提供する FFRI Dataset（悪性データ）に区分される．以下，良性データと悪性データについて説明する．

5.1.1 良性データ

良性データは，Windows10 における Windows フォルダ配下にあるすべての実行ファイル (windows file と呼ぶ．) および，窓の杜 [13] とフリーソフト１００ [14] から収集した実行ファイル（third party file と呼ぶ.）を正常な実行ファイルとみなし，可読文字列を抽出したものである．third party file は，フリーソフトを無料で提供しているサイト [13], [14] のトップページからリンクをたどり，実行形式のファイル (*.exe) のみをダウンロードした．実行ファイルのダウンロードには，３秒間の待機時間を設けることでアクセスおよびダウンロードによるサーバへの負担を軽減している．クローリングの期間は３日間である．収集した実行ファイルから可読文字列を抽出するために，Windows Sysinternals が提供している strings64.exe[15] を使用した．**表 1** は，良性データのファイル数およびユニークな単語数の一覧である．windows file は，ファイル名のアルファベット順に並び替え，奇数番目のファイル (windows file A と呼ぶ．) および偶数番目のファイル (windows file B と呼ぶ．) に分割した．windows file A および B でファイル数が異なる理由は，可読文字列の取得中に発生したエラーにより取得できなかったファイルを除外したためである．ユニークな単語数は，全ての実行ファイルから抽出した可読文字列を分割して得られた単語から重複を除外した単語の数である．

表 1 良性データのファイル数とユニークな単語数

データ名	ファイル数	ユニークな単語数
windows file A	1,206	2,858,679
windows file B	1,147	4,089,830
third party file	1,105	37,109,008

5.1.2 悪性データ

悪性データは，株式会社 FFRI が提供する FFRI Dataset から可読文字列のみを抽出したものである．FFRI Dataset は，MWS Datasets2018[12] の一部として提供されており，FFRI が保有している検体をサンドボックスで動作させた動的解析の結果が JSON 形式で保存されている．**表 2** は，悪性データのファイル数およびユニークな単語数の一覧である．本研究では，2013 年から 2017 年の FFRI Dataset を使用し，データセットに含まれる strings 項目のみを抽出して使用する．strings 項目には，マルウェアに含まれる可読文字列が記録されている．ユニークな単語数は，すべてのデータから抽出した可読文字列を分割して得られた単語から重複を除外した単語の数である．

表 2 悪性データのファイル数とユニークな単語数

データ名	ファイル数	ユニークな単語数
FFRI 2013	2,641	1,093,494
FFRI 2014	3,001	3,919,973
FFRI 2015	3,000	2,544,502
FFRI 2016	3,019	3,290,247
FFRI 2017	3,003	757,187

5.2 評価指標

本研究で用いる評価指標について説明する．評価値を算出するために必要な予測値と真の値の関係を**表 3** に示す．真の値は，悪性データ (Positive) と良性データ (Negative) である．予測値は，真の値に対して正解 (True) か不正解 (False) で値を決める．たとえば，悪性データを誤って良性データだと予測した場合，予測値は False Negative となる．本実験では，評価指標として Accuracy，Precision，Recall，F-measure を用いる．各評価指標の定義は，(4)〜(7) 式に示すとおりである．

表 3 予測値と真の値の関係

		真の値	
		悪性	良性
予測結果	悪性	True Positive (TP)	False Positive (FP)
	良性	False Negative (FN)	True Negative (TN)

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$F - measure = \frac{2Recall * Precision}{Recall + Precision} \tag{7}$$

5.3 予備実験

予備実験では，コーパスで使用する単語数の最適値を求める．コーパスは，良性データおよび悪性データからそれぞれ，出現頻度の高い単語を均等に抽出して作成する．訓練データは，良性データを windows file A，悪性データを FFRI 2013 とする．テストデータは，良性データを windows file B，悪性データを FFRI 2014 とする．コーパスの総数を 1,000 から 1,000 ずつ 10,000 まで増加させて 10 分割交差検証を実施した．その結果を図 2 に示す．図 2 の縦軸は評価指標の値であり，横軸はコーパスに用いたユニークな単語の数である．図 2 から，F-measure はコーパスの総数を 4,000 としたとき 0.982 となり，最適値となった．そのため，検証実験では，良性データと悪性データから出現頻度の高い順に各々 2,000 のユニークな単語を用いた，合計 4,000 のコーパスを使用する．

5.4 実験内容

表 4 に検証実験で使用する訓練データとテストデータの組み合わせを示す．訓練データは，windows file A を良性データ，FFRI 2013 を悪性データとして使用する．テス

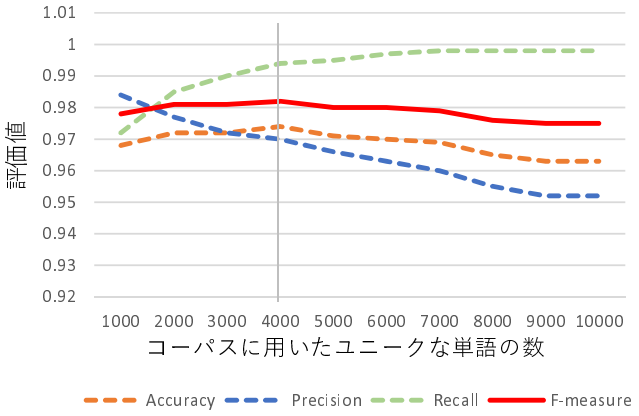

図 2　予備実験の結果

トデータは，windows file B または third party file を良性データ，FFRI 2014 から FFRI 2017 を悪性データとして使用する．検証実験で用いたテストデータは，2 種類の良性データと 4 種類の悪性データを用いるため，計 8 とおりの組み合わせとなる．訓練データの良性および悪性の実行ファイルから 4,000 語を抽出してコーパスを作成する．Doc2Vec および LSI の 2 種類の自然言語処理技術を用いて，可読文字列を特徴ベクトルに変換する．また，コーパスに全てのユニークな単語を用いる従来手法でも実験を行い，提案手法と結果を比較する．Doc2Vec では，特徴ベクトルの値が実験の度に変化するため，5 回の平均値を算出する．

表 4　検証実験で使用する訓練データとテストデータの組み合わせ

訓練データ		テストデータ	
良性データ	悪性データ	良性データ	悪性データ
windows file A	FFRI 2013	windows file B	FFRI 2014
			FFRI 2015
			FFRI 2016
			FFRI 2017
		third party file	FFRI 2014
			FFRI 2015
			FFRI 2016
			FFRI 2017

検証実験の環境は，**表 5** に示すとおりである．

表 5　実験環境

CPU	Core i7-5820K 3.30GHz
Memory	32.0GB
OS	Windows 8.1 Pro
使用言語	python2.7

5.5　実験結果

時系列分析の実験結果を**表 6** に示す．表 6 中の値は従来手法と，Doc2Vec または LSI を用いた提案手法の F-measure を示す．テストデータの良性データを windows file B とした場合，LSI を用いた提案手法の平均 F-measure は 0.988 であり，Doc2Vec は 0.820 であった．従来手法の平均 F-measure は 0.958 であった．テストデータの良性データを third party file とした場合，LSI を用いた提案手法の平均 F-measure は 0.999 であり，Doc2Vec は 0.811 であった．従来手法の平均 F-measure は 0.988 であった．

表 7 は，従来手法と提案手法の処理時間の平均を示している．言語モデルの構築時間は，自然言語処理技術を用いて特徴ベクトルを生成するための言語モデルの構築に要する時間である．特徴ベクトルの生成時間は，テストデータを言語モデルに入力して特徴ベクトルを生成するために要する時間である．分類器の訓練は，訓練データの特徴ベクトルを用いた分類器の訓練に要する時間である．検証実験の結果，各処理時間が最短となったモデルは LSI であった．

表 6　従来手法と提案手法の検証実験結果

テストデータ		F-measure		
良性データ	悪性データ	従来手法	Doc2Vec	LSI
windows file B	FFRI 2014	**0.934**	0.930	**0.982**
	FFRI 2015	0.960	0.898	0.985
	FFRI 2016	0.970	0.808	0.993
	FFRI 2017	0.970	0.646	0.994
平　均		**0.958**	0.820	**0.988**
third party file	FFRI 2014	0.992	0.891	0.997
	FFRI 2015	0.993	0.956	1.0
	FFRI 2016	0.992	0.752	1.0
	FFRI 2017	0.992	0.645	1.0
平　均		**0.992**	0.811	**0.999**

表 7　従来手法と提案手法の処理時間の比較 (s)

良性データ	時間区分	従来手法	Doc2Vec	LSI
windows file B	言語モデルの構築	104	61	18
	特徴ベクトルの生成	**89**	53	**13**
	分類器の訓練	0.5	0.5	0.5
	分類器による分類	0.5	0.6	0.6
third party file	言語モデルの構築	105	62	19
	特徴ベクトルの生成	**4,584**	36	**6**
	分類器の訓練	0.5	0.5	0.5
	分類器による分類	0.5	0.6	0.6

6.　考察

6.1　分類精度

LSI を用いた提案手法では，検知精度が従来手法より向上した．提案手法と従来手法の違いは，コーパスの作成手法と自然言語処理技術に LSI を用いたことである．このことから，単語を削減したコーパスと LSI による特徴ベクトルへの変換は，精度を向上させる上で有効であると考えられる．提案手法のコーパスは，低い出現頻度の単語を除外し，良性データと悪性データの区別を明確にすることがで

きたものと考えられる．また，可読文字列から特徴ベクトルに変換する過程で，LSI の処理の 1 つである tf-idf を適用することにより，出現頻度の高い単語をより正確に抽出できたものと考えられる．

悪性データの内，訓練データを FFRI 2013 とし，テストデータを FFRI 2014 から FFRI 2017 とした場合には，F-measure の値はほとんど変化しなかった．このことから，検知に貢献する単語は，年代に係らず一貫しているものと考えられる．

次に，良性のテストデータの内，windows file B を third party file に変更とした場合，F-measure の値にはほとんど変化はなかった．このことから，提案手法は windows file だけでなく，ほとんどの実行ファイルにおいて有効であると考えられる．

6.2 処理速度

検証実験の結果，LSI を用いた提案手法では従来手法よりも各処理時間を短縮することができた．

マルウェアの検知に要する時間は，未知の実行ファイルから抽出した可読文字列を特徴ベクトルに変換する時間および訓練された分類器により特徴ベクトルを分類する時間の合計となる．マルウェアの検知に要する時間は，従来手法では 90 秒を要したが，LSI を用いた提案手法では 14 秒で終えており，約 1/4 に処理時間を短縮することができた．提案手法では，約 4,000 件の実行ファイルを 14 秒で分類している．そのため，メールサーバ等において高速かつ簡易にマルウェアを検知する手段として，提案手法は実用的な時間内で動作可能であると考えられる．また，一般的な端末におけるマルウェアの検知手段としても，提案手法は未知のマルウェアを補助的に分類する手段として有益であると考えられる．

提案手法により処理時間が短縮された理由は，２つあると考えられる．１つ目は，従来手法に対してコーパスの総数が 1/100 になり，特徴ベクトルの生成にかかる計算コストが大幅に軽減されたことである．２つ目は，Doc2Vec を LSI に変更したことにより，計算に必要なコストが軽減できたことである．

6.3 誤検知の原因

提案手法を用いた LSI の言語モデルにおいて，分類結果が誤っていたデータを分析する．誤検知したファイルは，windows file B で 90 ファイル，FFRI 2014 で 17 ファイルであった．windows file B の 90 ファイルは，良性の実行ファイルを，悪性の実行ファイルとして過検知したものである．FFRI 2014 の 17 ファイルは，悪性の実行ファイルを良性の実行ファイルとして見逃したものである．**表 8** に，誤検知したファイルと各コーパスに共通する単語の数を示す．表 8 中の良性または悪性のコーパスは，提案手法により抽出した 4,000 語から作成したコーパスである．表中の単語数は，誤検知した 90 あるいは 17 のファイルに含まれている単語の内，良性または悪性のコーパスにも共通して含まれている単語の数である．過検知した windows file B の 90 ファイルに注目すると，悪性のコーパスにも含まれる単語が 1,199 も含まれていることが確認できる．つまり，これらのファイルには良性および悪性のコーパスに共通する単語が多く含まれていたため，過検知したものと考えられる．見逃した FFRI 2014 の 17 ファイルに注目すると，悪性の 869 に対して 1,563 もの良性のコーパスと共通する単語が含まれていることが確認できる．つまり，これらのファイルには悪性よりも良性のコーパスに含まれる単語が多く含まれていたため，良性として見逃したものと考えられる．したがって，良性データおよび悪性データで重複する単語を除外してコーパスを作成することで，誤検知の発生を抑えられるものと考えられる．windows file B の過検知したファイルでは，良性コーパスに含まれる単語は「w@」，「Yf」，「))」，「NN」等の 2 文字の英数字または記号が多く含まれており，悪性コーパスに含まれる単語は「:」，「’」，「)」，「-」等の 1 文字の英数字または記号が多く含まれていた．FFRI 2014 の見逃したファイルでは，良性コーパスに含まれる単語は「rrr」，「yY」，「x)」，「$」等が含まれており，悪性コーパスに含まれる単語は「xxx」，「Version」，「MMMMMM」，「)))))」等が含まれていた．

表 8 誤検知したファイルと各コーパスに共通する単語の数

	誤検知したファイル	
	windows file B 90 ファイル	FFRI 2014 17 ファイル
良性のコーパス	1,970	1,563
悪性のコーパス	1,199	869

6.4 研究倫理

本研究で使用したデータの内，悪性データは株式会社 FFRI が提供するデータセットを使用した．良性データは，フリーソフトを提供しているサイトのサーバに負荷がかからないように，待機時間を設けることで連続したアクセスを回避して収集した．そのため，本研究ではサービスを提供しているサーバおよびその利用者にほとんど影響を与えていない．

研究で使用した環境は，一般家庭でも使用されている端末であり，特別な機器は必要としない．本研究で使用したデータは，インターネットで公開されているフリーソフトウェアおよびオープンなデータセットから作成した．また，提案手法はオープンソースプログラミング言語および機械学習のライブブラリを用いて容易に実装可能である．本研究では，入手が困難な特別な機器や公開されていない独自のデータは一切使用していない．したがって，本研究

の再現性は極めて高い言える．

公益性に関しては，研究成果を公表したとしても，攻撃者がこの検知手法を回避することは困難であると判断した．攻撃者が提案手法を回避するマルウェアを作成するためには，大量の正常ファイルから可読文字列を抽出し，高い出現頻度の単語を選び，マルウェアに可読文字列を追加する必要がある．もし，攻撃者がこのようなマルウェアを作成したとしても，そのマルウェアから誤検知の原因になる単語をコーパスの生成過程で除外することで，検知することができるものと考えられる．また，そのマルウェアから言語モデルの構築と分類器の訓練を実施することで，提案手法を修正することなく，そのマルウェアを検知できるようになるものと考えられる．

7. おわりに

本研究では，良性と悪性の実行ファイルからそれぞれユニークな可読文字列を，出現頻度の高い順に均等に抽出してコーパスを作成することで，検知率を向上させる手法を提案した．検証実験の結果，LSI を用いた提案手法の平均 F-measure は 0.988 となり，従来手法は 0.958 となった．このことから，提案手法のコーパスの削減は有効であると考えられる．実行時間を比較すると，提案手法を用いることにより，処理時間を大幅に短縮することができた．

今後の課題は３つ考えられる．

１つ目は，各パラメータの値を調整することにより，分類精度を更に向上させることである．本研究で行った検証実験では多くのパラメータに初期値を設定しているため，各値を調整することにより分類精度を更に向上させることができるものと考えられる．

２つ目は，本研究で用いた SVM の分類器を異常検知の分類器に変更することである．本検証実験では悪性データに FFRI Dataset を用いたが，一般的に悪性データを入手することは困難であると考えられる．異常検知の分類器では，入手が容易な正常ファイルのみを用いてモデルを構築することができる．そのため，その運用性は向上するものと考えられる．

３つ目は，難読化されている正常な実行ファイルの評価である．難読化手法は，従来は商用ソフトウェアを保護する用途で用いられてきたが，マルウェアが解析を妨害する用途にも使用されている．マルウェアに類似する難読化手法が施された正常な実行ファイルについては，マルウェアとして誤検知する可能性が考えられる．また，正常な実行ファイルにおいて出現頻度の高い単語をあえてマルウェアに含めることで，マルウェアが検知を回避できる可能性についても検証する必要があるものと考える．

参考文献

[1] 独立行政法人情報処理推進機構技術本部セキュリティセンター :標的型攻撃／新しいタイプの攻撃の実態と対策, available from <https://www.ipa.go.jp/files/000024542.pdf/> (accessed 2018-7-20).

[2] 株式会社ラック：標的型攻撃 対策指南書 (第 1 版), available from <https://www.lac.co.jp/library/pdf/anti-apt_guidebook_ver1.pdf > (accessed 2018-7-20).

[3] Q. Le, T. Mikolov, : Distributed representations of sentences and documents, Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML'14) Vol32, pp.1088-1196 (2014).

[4] T. Mikolov, W. Yih, G. Zweig, : Linguistic Regularities in Continuous Space Word Representations, Proceedings of NAACL-HLT, pp.746-751 (2013).

[5] Y. Ye, L. Chen, D. Wang, T. Li, Q. Jiang, M. Zhao, : SBMDS: an interpretable string based malware detection system using SVM ensemble with bagging, Journal in Computer Virology, Vol5, No.4, pp.283-293 (2009).

[6] J. Lee, C. Im, H. Jeong, : A study of malware detection and classification by comparing extracted strings, Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, (2011).

[7] F. Mastjik, C. Varol, A. Varol, :Comparison of Pattern Matching Techniques on Identification of Same Family Malware, International Journal of Information Security Science, Vol.4, No.3, pp.104-111 (2015).

[8] 戸部和洋，森達哉，千葉大紀，下田晃弘，後藤滋樹，: 実行ファイルに含まれる文字列の学習に基づくマルウェア検出方法，マルウェア対策研究人材育成ワークショップ (2010).

[9] 菅野　賢輝, 三浦紘弥，三村　守, 田中　秀磨, : Doc2Vec を用いた静的解析によるマルウェア検知，情報処理学会第 80 回全国大会, 7W-05（2018）.

[10] Funding provided by INRIA and others. : scikit-learn, available from <http://scikit-learn.org/stable/> (accessed 2018-7-20).

[11] Copyright Radim Rehurek : gensim, available from <https://radimrehurek.com/gensim/> (accessed 2018-7-20).

[12] 高田　雄太，寺田　真敏，松木　隆宏，笠間　貴弘，荒木　粧子，畑田　充弘：マルウェア対策のための研究用データセット 〜MWS Datasets2018〜, IPSJ SIG Technical Report（2018）.

[13] Impress Corporation : 窓の杜, available from <https://forest.watch.impress.co.jp/> (accessed 2018-7-20).

[14] Copyright フリーソフト 100 : freesoft 100, available from <https://freesoft-100.com/> (accessed 2018-7-20).

[15] Windows Sysinternals : Stringsv2.53, available from <https://technet.microsoft.com/> (accessed 2018-7-20).