Macros Finder: Do You Remember LOVELETTER?

Hiroya Miura(✉), Mamoru Mimura, and Hidema Tanaka

National Defense Academy, Yokosuka, Japan
{em56030,mim,hidema}@nda.ac.jp

Abstract. In recent years, the number of targeted email attacks which use Microsoft (MS) document files has been increasing. In particular, damage by malicious macros has spread in many organizations. Relevant work has proposed a method of malicious MS document files detection. To the best of our knowledge, however, no method of detecting malicious macros exists. Hence, we proposed a method which detects malicious macros themselves using machine learning. First, the proposed method creates corpuses from macros. Our method removes trivial words in the corpus. It becomes easy for the corpuses to classify malicious macros exactly. Second, Doc2Vec represents feature vectors from the corpuses. Malicious macros contain the context. Therefore, the feature vectors of Doc2Vec are classified with high accuracy. Machine learning models (Support Vector Machine, Random Forest and Multi Layer Perceptron) are trained, inputting the feature vectors and the labels. Finally, the trained models predict test feature vectors as malicious macros or benign macros. Evaluations show that the proposed method can obtain a high F-measure (0.93).

Keywords: Macro · Machine learning
Natural language processing technique · Bag-of-Words · Doc2Vec

1 Introduction

In recent years, email has become one of the most popular communication tools. In this situation, targeted email attacks have become a big threat to society. A targeted email attack is a specific attack in which the attacker attempts to persuade a victim to run specific action. Depending on the specific action, there are two types of targeted email attacks. One is to open malicious links and to download a malware, and the other is to open malicious attachments. Attackers attempt to earn credibility with their victims through an eloquent mail text. Moreover, the attackers convince victims to unknowingly download a malicious file attachment or click-through to a malicious site. According to a report published by Sophos [1], most targeted email attacks are the attachment type. Moreover, the report shows that 85% of the attached files are Microsoft Office documents (MS documents) files. The report shows that most malicious

C. Su and H. Kikuchi (Eds.): ISPEC 2018, LNCS 11125, pp. 3–18, 2018.
https://doi.org/10.1007/978-3-319-99807-7_1

MS document files have malicious macros. Malicious macros have a long history. For example, the LOVELETTER worm of malicious macro infected more than 45 million computers, and some organizations suffered serious damage in 2000. After that, the occurrence of malicious macros gradually slacked off. However, they were enlivened again from 2014 onwards.

Next, we show the importance of detecting malicious macros themselves. Relevant work [5] analyzes the structure of docx files, and detects malicious docx files. The work does not, however, discriminate between malicious macros and benign macros. If the dataset contains benign macros and malicious macros, the work probably cannot detect malicious docx files. If malicious docx files are camouflaged with a structure of benign docx files, attackers can probably evade the detection model. Detecting malicious macros themselves can overcome these weaknesses. Hence, detecting malicious macros is an effective and important method.

We will introduce an outline of the proposed method. The proposed method detects malicious macros themselves using machine learning. First, the proposed method creates corpuses from macros. Our method reduces trivial words in the corpus. It becomes easy for the corpuses to classify malicious macros exactly. We use Term Frequency (TF) or Term Frequency-Inverse Document Frequency (TFIDF) in the reducing words process. TF is a method which weights value corresponding to frequency of words in a corpus. TFIDF is a method which weights a representative word in a corpus. Second, Doc2Vec (D2V) or Bag-of-Words represents feature vectors from the corpuses. D2V is a model that represents vectors from the context of the documents. Bow is a method that represents vectors corresponding to the frequency of the words.

Next, Support Vector Machine (SVM), Random Forest (RF) and Multi Layer Perceptron (MLP) are trained, inputting the feature vectors and the labels. Finally, the trained models predict test feature vectors as malicious macros or benign macros.

Next, we will show three viewpoints of verification experiments in this paper.

(1) Which is more effective, TF or TFIDF?
(2) Which is more effective, D2V or Bow?
(3) What is the best combination of these methods and classifiers?

In order to answer these questions, we conducted verification experiments. Based on the results of these verification experiments, this paper makes the following contributions:

(1) we confirmed that D2V was effective in classifying malicious macros.
(2) we confirmed that reducing words using TF was effective in classifying malicious macros.
(3) we confirmed that classifiers in which the strong point was to solve the problem of linear separability, were effective in classifying malicious macros.

We will introduce the structure of this paper. Section 2 introduces relevant work and reveals the differences between this paper other and relevant study.

Section 3 presents an overview of the background. Section 4 presents the proposed method. Section 5 describes experiments. Section 6 discusses the results of the experiments. Finally, we conclude this paper.

2 Related Work

In targeted email attacks, attackers use document files in which are embedded malicious source codes, or executable files. Most malicious executable files are camouflaged with a change of document icon and the file extension. Methods of detecting these malicious files can be categorized into static analysis and dynamic analysis. Dynamic analysis is the testing and evaluation of a program by executing data in real-time. Static analysis is a method of program debugging that is done by examining the code without executing the program.

Malicious document files are roughly categorized into MS document files and PDF files [1]. This section presents the relevant work in two parts (MS document files detection and malicious PDF files detection). Since our method does not execute specimens for detecting malicious macros, our method is the static analysis. Therefore, we present the relevant work within the static analysis range, in this section.

We will show representative work for malicious executable files detection using the dynamic analysis, as a reference. Rieck et al. [2] proposed a framework for the dynamic analysis of malicious executable binaries behavior using machine learning. Bayer et al. [3] proposed a tool which monitors the behavior of Windows API, to classify malicious executable files. Next, we will show representative work for malicious executable files detection using the static analysis. Perdisci et al. [4] proposed a framework which detects malicious executable files using the static analysis. The framework classifies malicious executable codes using n-gram analysis. Even if executable files are packed, the framework is able to classify.

MS Document File. This section presents relevant work on the detection of malicious MS document files. Nissim et al. [5] proposed a framework (ALDOCX) that classifies malicious docx files using various machine learning classifiers. ALDOCX created feature vectors from the path structure of docx files. This is a practical method, because ALDOCX framework has updatability and incorporates new unseen malicious docx files created daily. Naser et al. [6] proposed a method to detect malicious docx files. The method parses structure of docx files, and analyzes suspicious keywords. These works do not support the classification of Excel files and Power Point files. Our method, however, can be applied to malicious MS document files which are Word files, Excel files and Power Point files.

Otsubo et al. [7] proposed a tool (O-checker) to detect malicious document files (e.g. rtf, doc, xls, pps, jtd, pdf). O-checker detects malicious document files which contain executable files, using deviation of file format specifications. O-checker focuses on embedded executable files however, and cannot classify

macros themselves. Even if the malicious documents do not contain executable files, our proposed method can detect malicious macros.

Boldewin implemented a tool (OfficeMalScanner) [8] to detect MS document files which contain malicious shellcodes or executable files. The tool scans entirely malicious files, and detects features of strings of Windows API, shellcode patterns and embedded OLE data [9]. The tool scores each document corresponding to each of the features. If the scores are more than a certain threshold, the tool judges the file as a malicious file.

Mimura et al. [10] de-obfuscate embedded executable files in a malicious document file (e.g. doc, rtf, xls, pdf) and detect them. The detection rate was verified in the work, and it was confirmed that the detection rate was higher than the detection rate of OfficeMalScanner. [7,8,10] focused on embedded malicious executable files or shellcodes, but not, however, on detecting malicious macros themselves.

PDF File. Next, this section presents related work which deals with the detection of malicious PDF files. Igino Corona et al. [11] proposed a method that refers to the frequency of suspicious reference APIs, to classify malicious PDF files. Liu et al. [12] proposed a method that analyzes obfuscated scripts to classify malicious PDF files. This method uses the characteristics of obfuscation, which is common to our method. However, these methods classify only malicious PDF files, and are fundamentally different from our method, which classifies malicious macros.

3 Relevant Techniques

3.1 Malicious Macros

This section describes the behavior of malicious macros, and reveals their features. There are two types of malicious macros, Downloader and Dropper.

Downloader is a malicious macro which enforces download malware upon a victim. An attacker uses a slick text of the type that the victim expects, and induces the victim to open an attachment. When the victim opens the attachment, the computer is forced into connecting to a malicious server. When Downloader connects to the server, it tends to uses external applications (Internet Explorer, etc.). Finally, the computer downloads and installs a malware from the server.

With Dropper, malicious codes (a binary of EXE files, etc.) are embedded in Dropper itself. When a victim opens the attachment of a phishing email, Dropper executes the codes contained in it as an executable file. The difference between Dropper and Downloader is that Dropper itself is able to fraudulently operate upon victim computers. Unlike Downloader, Dropper can infect victims without communicating an external resource (server or database, etc.).

Table 1. Typical obfuscation methods

#	summary
1	Obfuscation of replacing statement name, etc.
2	Obfuscation of encoding and decoding ASCII code
3	Obfuscation of character string by encoding conversion using exclusive OR
4	Splitting characters
5	Using reflection function

3.2 Obfuscation of Malicious Macros

The source codes of most malicious macro tend to be obfuscated. Therefore, capturing the characteristics of obfuscation can effectively predict malicious macros. We will show some obfuscation methods of the source codes.

Table 1 shows typical obfuscation methods in the source codes. Method 1 replaces class names, function names, etc with random strings. The random strings tend to be more than 20 characters. Method 2 encodes and decodes strings to ASCII codes. Macros provide AscB function and ChrB function. AscB function encodes character strings to ASCII codes. ChrB function encodes ASCII codes to character strings. Using AscB functions, attackers can conceal strings to encode the hexadecimal of ASCII codes. Moreover, ChrB function can convert ASCII codes to readable character strings. Method 3 performs using exclusive-OR any strings with a key. Many of the keys are intricately calculated and perform logic operations. Method 4 subdivides strings. Subdivided character strings are assigned to variables. By adding together those variables, the original strings are restored. Method 5 uses reflection functions which execute strings as instructions. For example, the strings are function names, class names and method names. CallByName function is a reflection function in the source codes. Using the CallByName function, attackers can hide the executing function.

3.3 Bag-of-Words

Words and documents need to be represented by feature vectors so that computers can interpret a natural language. Bow is the most basic natural language processing technique. Bow is a method of representing the frequency of a token in a sentence to an element of a vector corresponding to the token. Bow does not consider word order or meaning of tokens. In Bow, the number of unique tokens and the number of elements are the same. When the number of unique tokens diverges, the number of elements likewise diverges. Therefore, when Bow represents feature vectors, it may be necessary to adjust the number of dimensions.

3.4 Doc2Vec

D2V is a natural language processing technique. D2V is a model that is improved Word2Vec (W2V). First of all, we will introduce W2V. W2V is a model that is

used to represent word embeddings. W2V is a two-layer neural network that is trained to reconstruct the linguistic context of words. W2V has a hidden layer and an output layer. The input of W2V is a large corpus of documents, and W2V represents the input in feature vectors. The number of dimensions of the feature vector is typically several hundred. Each unique token in the corpus is assigned a corresponding element of the feature vector. Word vectors are positioned in the vector space such that common contexts in the corpus are positioned in close proximity to one another in the space. This is based on the probability of words co-occurrence around a word. W2V has two algorithms, which are Continuous Bag-of-Words (CBow) and Skip-Gram. CBow is an algorithm which predicts a centric word from surrounding words. Skip-Gram is an algorithm which predicts surrounding words from a centric word. Using these algorithms, W2V can obtain similarity of words, and also predict equivalent words.

D2V has two algorithms which are Distributed Memory (DM) and Distributed Bag-of-Words (DBow). DM is an algorithm that is improved CBow. In addition to a large corpus of documents, those document-IDs input into DM. DBow is an algorithm which improves Skip-Gram. The input of DBow is not the words of documents but document-IDs. Using these algorithms, D2V can obtain similarity of documents, and also vectorize the documents.

3.5 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency ($TFIDF$) is a numerical value that determines the importance of the words in the corpus. We will introduce how $TFIDF$ value is calculated.

$$TFIDF = frequency_{i,j} \times \log_2 \frac{D}{document\_freq_i}$$

The $frequency_{i,j}$ is the frequency of a token i in a document j. The $document\_freq_i$ is the frequency of documents in which the token i appears. The TF is the $frequency_{i,j}$. The IDF is the logarithm of a value in which D (the number of total documents) is divided by the $document\_freq_i$. $TFIDF$ value is a value which is the multiplication of TF and IDF. Finally, $TFIDF$ value is normalized.

When a word appears rarely in an entire corpus and appears frequently in a document, the $TFIDF$ value increases the priority of the word.

4 Proposed Method

This section proposes our method. Figure 1 shows an outline of the proposed method. The purpose of the proposed method is to detect unseen malicious macros with high classification accuracy. Step 1 extracts macros from MS document files. Step 2 separates words in the macros to create corpuses. Step 3 replaces words with characteristics of the same types of obfuscation (such as

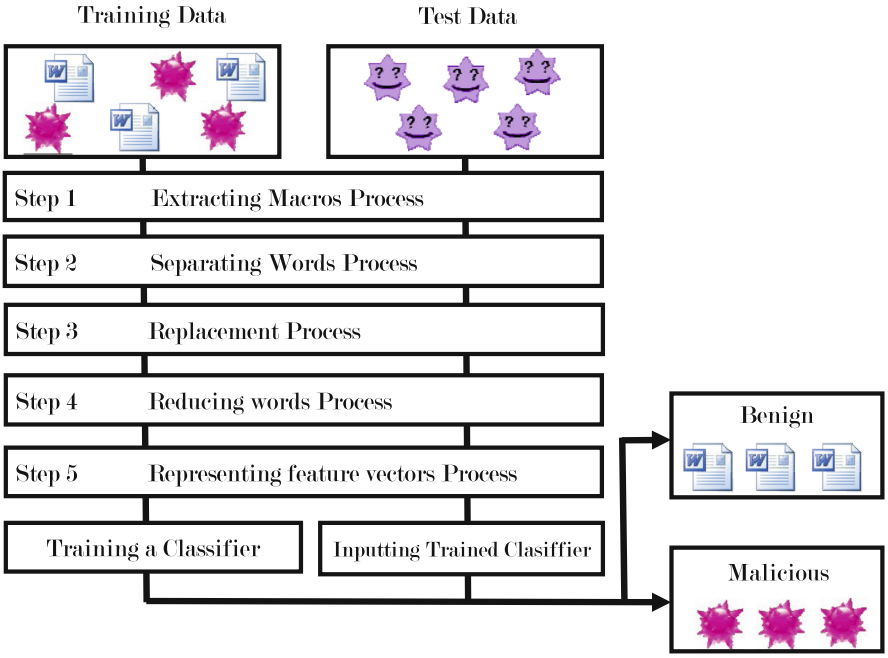

Fig. 1. Process procedure of unseen malicious macros detection

hexadecimal ASCII codes), with a word. Step 4 reduces trivial words in the corpuses using TF or TFIDF. Step 5 represents feature vectors from the corpus using Bow or D2V. The proposed method inputs the training feature vectors and the labels into classifiers (SVM, RF and MLP). Finally, we input the test feature vectors into trained classifiers, and obtain the labels.

4.1 Extract Source Code Process

The proposed method extracts macros from MS document files using Olevba [13]. Olevba is open source software that can extract macros from MS document files. Olevba can extract regardless of the platform.

4.2 Separating Words Process

The purpose of the separating words process is to create corpuses of the macros. The process replaces special characters that are shown in Table 2 with a blank. The source codes of malicious macros are intricately written, with many special characters. Therefore, the separating words process creates simple corpuses which are only alphabets and numbers.

Table 2. Replaced special characters

Special character	Name	Special character	Name
”	double quote	+	plus
’	single quote	/	slash
{	quare bracket	&	and
(	round bracket	%	percentage
,	comma		yen sign
.	period	$	dollar sign
*	asterisk	#	sharp
-	haihun	@	at mark

Table 3. Replacing specific strings

Methods	Characters pattern	Replaced strings
1	Hexadecimal (Type of 0xXX)	0xhex
2	Hexadecimal (Type of &HXX)	andhex
3	Asc, AscB, AscW	asc
4	A string of 20 or more characters	longchr
5	A number of 20 digits or more	longnum
6	Element of array	Elementofarray

4.3 Replement Process

This section discusses the replacement process. The purpose of the process is to collect words into each obfuscation. The process replaces words with characteristics of same type of obfuscation, with one token. Generally, malicious macros are obfuscated. The process can convert the corpuses to improve classification accuracy.

We will show an example of the process. The process replaces the hexadecimal values with one token as follows.

Before the process: 0xFF 0x14 0xA2

After the process: 0xhex 0xhex 0xhex
In this example, each hexadecimal value is treated as a different token before the process replacing the tokens. However, after the process of replacing characteristics, they are treated as the same token. The process replaces words regarded as different features with one word. Therefore, the replacement process improves the classification accuracy. Table 3 shows string patterns and replaced strings. These string patterns frequently appear in malicious macros.

4.4 Reducing Words Process

The purpose of the reducing words process is to reduce trivial words for improving classification accuracy. The frequency of words in macros is biased. A feature vectors of D2V and Bow are affected by the frequency of words. Thus, each word in a corpus has the some worth for the classification of malicious macros.

The process prioritizes words in the corpuses with TF or TFIDF. First, the process calculates the TF or $TFIDF$ of each word in all the corpuses. Next, we define a threshold. Finally, the process replaces words in which the TF or TFIDF value is less than the threshold, with "NONE". Through the process, the words which are bigger than the threshold remain in the corpuses.

4.5 Representing Feature Vectors Process and Classification of Macros

In the representing feature vectors process, D2V or Bow represents feature vectors by processed corpuses. Next, we input training feature vectors and the labels into classifiers (SVM, RF and MLP) in order to train the classifier. Test feature vectors are input into the trained classifier, and our method detects malicious macros.

5 Experiment

This section describes the verification experiments. The objective is to verify the next four factors.

1 Investigating the most effective corpus for improving F-measure
2 Comparing D2V and Bow
3 Comparing TF and TFIDF
4 Investigating the best combination of the above factors and classifiers (SVM, RF and MLP).

Verification Experiment 1 investigates effective corpuses which classify malicious macros. Bow represents feature vectors from malicious corpuses, benign corpuses and corpuses which are both. Next, SVM classifies each the feature vector, and obtains each classification accuracy.

The classification accuracy of D2V and Bow is compared in Verification Experiment 2. Each method represents feature vectors from corpuses. The best corpuses in Verification Experiment 1 are used in Verification Experiment 2. The feature vectors are classified using SVM, RF and MLP.

The classification accuracy of TF and TFIDF is compared in Verification Experiment 3. The best corpuses in Verification Experiment 1 are used in Verification Experiment 3. Feature vectors are represented using the best method (D2V or Bow) in Verification Experiment 2.

5.1 Experiment Environment

We implemented our proposed method with Python2.7 in the environment as shown in Table 4. We used gensim-2.0.0 [14] to implement Bow and D2V. Gensim has many functions related to natural language processing techniques. We used scikit-learn-0.18.1 [15] to implement SVM, RF and MLP. Scikit learn is a machine learning library and has many classification algorithms.

Table 4. Experiment environment

CPU	IntelCorei7 (3.30 GHz)
Memory	32 GB
OS	Windows8.1Pro

Table 5. Breakdown of The dataset

Specimens of 2015		Specimens of 2016	
Benign files	Malicious files	Benign files	Malicious files
622	515	1200	641

5.2 DataSet

This section presents the dataset of the experiments. Table 5 shows the breakdown of the dataset. This dataset was collected and provided by Virus Total [16]. We selected specimens which had been uploaded to Virus Total for the first time between 2015 and 2016. We collected macros whose file extensions have doc, docx, xls, xlsx, ppt and pptx. We treat the specimens which more than 29 out of 58 anti-virus vendors judged as malicious, as malicious specimens. We treated the specimens which all anti-virus vendors judged as benign, as benign specimens. There was no overlap in these specimens.

5.3 Evaluation Measure

This section presents the evaluation measures in the experiments. Malicious macros is treated as true label and benign macros is treated as false labels in the experiments. Table 6 shows the confusion matrix. We used Precision (P), Recall (R) and F-measure (F) as evaluation metrics. We will indicate the definition of each evaluation metric.

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

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

$$F - measure = \frac{2Recall \times Precision}{Recall + Precision}$$

Table 6. Confusion matrix

		Actual value	
		True	False
Predicted result	Positive	*TP*	*FP*
	False	*FN*	*TN*

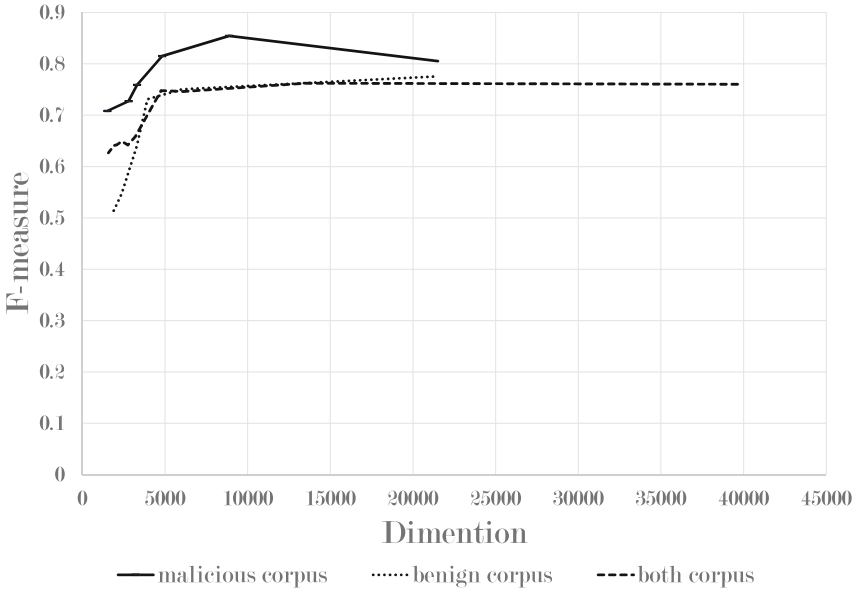

Fig. 2. The classification accuracy of each corpus in each dimension using TF and Bow

5.4 Verification Experiment 1

Experimental Approach. The objective is to investigate the most effective corpus for improving F-measure in Verification Experiment 1. The experiment selects from malicious corpuses, benign corpuses and corpuses which are both. The procedure of the experiment is shown next. The corpuses are created using Step 1 to Step 3 in Fig. 1. In the reducing process, we reduce the words in the corpus using the TF threshold. When the TF of a word is less than the TF threshold, the word is replaced with one word. Next, Bow represents three feature vectors from malicious corpuses, benign corpuses and corpuses which are both. Training feature vectors and the labels are input into an SVM classifier for training. Test feature vectors are input into the trained classifier to obtain predicted labels. The parameter of SVM is the default. Training data are specimens from 2015. Test data are specimens from 2016.

Result of Verification Experiment 1. Figure 2 shows the classification accuracy of each feature vector. The horizontal axis is the dimensions, and the vertical axis is the F-measure. When we represent feature vectors from malicious corpuses, the classification accuracy is higher than the classification accuracy of benign corpuses and corpuses which are both. Therefore, we conclude that the malicious feature vectors are effective in classifying malicious macros.

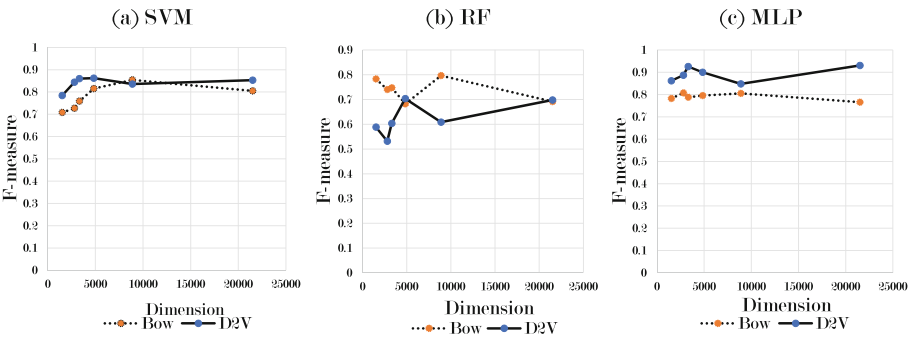

Fig. 3. The classification accuracy of each classifier using TF

5.5 Verification Experiment 2

Experimental Approach. The objective is to compare the classification accuracy of D2V and Bow in Verification Experiment 2. The procedure of the experiment is shown next. The corpuses are created using Step 1 to Step 3 in Fig. 1. In the reducing words process, we reduce the words in the corpus using TF. The feature vectors of two patterns are represented from malicious macros using D2V and Bow. D2V is set up such that the number of dimensions is 100, the number of epochs is 30 and the algorithm is DBow. Each of the feature vectors and the labels are input into three classifiers (SVM, RF and MLP) for training. The parameters of SVM and RF are default values. Verification Experiment 2 sets up MLP such that the input layer size is the number of unique tokens, the hidden layer size is 500, and the activation function is ReLU (Rectified Linear Unit). The classifier is input into the test feature vectors to predict malicious macros and benign macros. Finally, we obtain the F-measure of each of the feature vectors. Training data and test data are the same as for Verification Experiment 1.

Result of Verification Experiment 2. (a), (b) and (c) in Fig. 3 show the result of Verification Experiment 2 in each classifier. The horizontal axis is the dimensions, and the vertical axis is the F-measure. The F-measure of D2V is higher than Bow in (a) and (c). In contrast, the F-measure of Bow is higher than D2V in (b). In (c), when the number of dimensions is 21506 using D2V, the F-measure is the best (0.93). Moreover, the F-measure of a combination of MLP and D2V is stable.

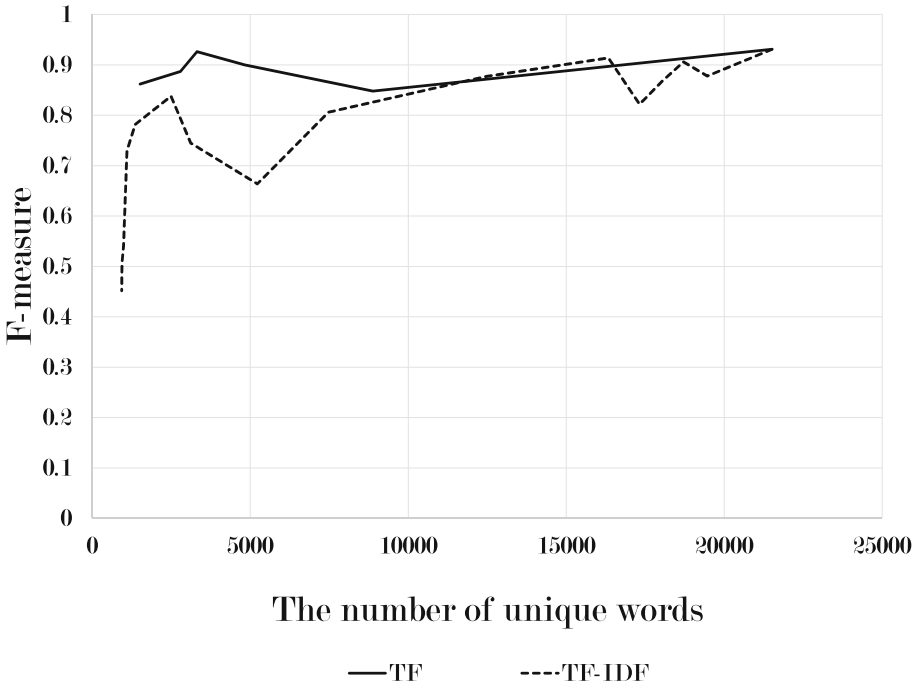

Fig. 4. The classification accuracy of each dimension using TF and TFIDF

5.6 Verification Experiment 3

Experimental Approach. The objective is to compare the F-measure of the TF and the TFIDF in Verification Experiment 3. The corpuses are created using Step 1 to Step 3 in Fig. 1. In the reducing process, we reduce the words in the corpus using two methods, which are TF and TFIDF. The feature vectors are represented using D2V. The settings of D2V are the same as in Verification Experiment 2. Each of the training feature vectors and the labels are input into MLP for training. MLP is input into the test feature vectors to predict malicious macros and benign macros. Finally, we obtain the F-measure of each of the feature vectors. The training data and the test data are the same as for Verification Experiment 1.

Result of Verification Experiment 3. Figure 4 shows the result of Verification Experiment 3. The horizontal axis is the number of unique tokens, and the vertical axis is the F-measure. Generally, the classification accuracy of the $TFIDF$ threshold is decreased. However, the classification accuracy of the TF threshold is stable and high. The highest F-measure is 0.93.

6 Discussion

6.1 Efficient Corpus

In Verification Experiment 1, we confirmed high classification accuracy using malicious corpuses. Benign macros are used for various purposes. Therefore, benign macros contain various tokens. In contrast, the purpose of malicious macros is simple. The purpose of malicious macros is to infect the victim's computer with malware. The source codes of the malicious macros contain many tokens which communicate to external servers, and are obfuscated. The source codes of the malicious macros frequently contain these characteristic tokens. Therefore, the replacement process can capture the characteristic, and the proposed method obtains high classification accuracy. In Bow, an element of the feature vectors is the frequency of a token. Therefore, malicious corpuses do better than other corpuses in Verification Experiment 1.

Table 7. Characteristic tokens

Token	Appearance ratio of malicious macro	Token	Appearance ratio of benign macro
Elementofarray	99%	Elementofarray	43%
Andchr	93.9%	Andchr	28%
Next	90.9%	Next	27.9%
Function	85.1%	Function	18.3%
String	83.3%	String	25.7%
Len	79%	Len	14.7%
Public	77.5%	Public	17.7%
Longchr	73.7%	Longchr	19.7%
Createobject	73%	Createobject	6.6%
Error	73%	Error	20.7%
Byte	56.1%	Byte	1.5%
Callbyname	51.3%	Callbyname	0.1%

6.2 Effectiveness of Bow and D2V

In Verification Experiment 2, we verified an efficient method of representing feature vectors. As a result, we concluded that D2V is better than Bow. As the TF threshold is high, low frequently-tokens are reduced and high frequently-tokens remain. In (c) of Fig. 3 using D2V, even if the dimension is the smallest (concretely, the dimension is 1515), the classification accuracy remains high. D2V represents word embedding. Therefore, we consider that the high-frequency token contains the context.

6.3 Efficient Classifier

In Verification Experiment 2, we confirmed that the classification accuracy of MLP and SVM was high. However, the classification accuracy of the RF classifier was low. Generally, the strong point of MLP and SVM is in solving the problem of linear separability. However, the strong point of RF is in solving the problem of linear inseparability. Therefore, we consider that the feature vectors of D2V can be separated linearly. In (a) and (c) of Fig. 3, the classification accuracy of D2V is higher than Bow. As a reason for this, we conclude that D2V tends to suit SVM and MLP in the classification of malicious macros.

6.4 Effectiveness of TFIDF

When the number of tokens is small, the classification accuracy decreased in Fig. 4 using TFIDF. Many words which are replaced in the replacement process, exist in the malicious corpus. Therefore, the TFIDF values of replaced words are small. While the $TFIDF$ threshold is high, replaced words are reduced. We indicate the representative tokens in Table 7. We define the appearance ratio as "Number of the files which contain the token / Number of files". Therefore, we consider that the classification accuracy decreased using the $TFIDF$ threshold.

In Fig. 4 using a TF threshold, the classification accuracy is more stable and we higher than TFIDF. As the TF threshold is high, the characteristic words of malicious macros remained and low frequently-word are reduced. Therefore, TF is more effective than TFIDF.

7 Conclusion

In this paper, we discussed effective methods of detecting unseen malicious macros. The proposed method reduces trivial words in corpuses. Next, the corpuses are converted to feature vectors using a linguistic approach. The training feature vectors and labels are input into a classifier. Finally, the test feature vectors are input into the trained classifier, and we obtain predicted labels. This paper investigated effective methods of reducing trivial words (TF and TFIDF), vectorizing methods (D2V and Bow) and classifiers (SVM, RF and MLP). As a result, it was seen that the combination of TF, D2V and MLP is effective for the detection of unseen malicious macros in our method. The highest F-measure is 0.93. We discussed effectiveness of the proposed method in this paper. We concluded that the feature vectors of D2V are effective in classifying unseen malicious macros. Our future work is to implement a tool which can detect malicious macros in real-time.

References

1. Wolf in sheep's clothing: a SophosLabs investigation into delivering malware via VBA. https://nakedsecurity.sophos.com/2017/05/31/wolf-in-sheeps-clothing-a-sophoslabs-investigation-into-delivering-malware-via-vba/
2. Rieck, K., Trinius, P., Willems, C., Holz, T.: Automatic analysis of malware behavior using machine learning. J. Comput. Secur. **19**(4), 639–668 (2011)
3. Bayer, U., Moser, A., Kruegel, C., Kirda, E.: Dynamic analysis of malicious code. J. Comput. Virol. **2**, 67–77 (2006). https://doi.org/10.1007/s11416-006-0012-2
4. Perdisci, R., Lanzi, A., Lee, W.: McBoost: boosting scalability in malware collection and analysis using statical classification of executables. In: Computer Security Applications Conference (2008). https://doi.org/10.1109/ACSAC.2006.53
5. Nissim, N., Cohen, A., Elovici, Y.: ALDOCX: detection of unknown malicious microsoft office documents using designated active learning methods based on new structural feature extraction methodology. IEEE Trans. Inf. Forensics Secur. **12**(3), 631–646 (2017)
6. Naser, A., Hadi, A.: Analyzing and detecting malicious content: DOCX files. Int. J. Comput. Sci. Inf. Secur. (IJCSIS) **14**(8), 404–412 (2016)
7. Otsubo, Y., Mimura, M., Tanaka, H.: O-checker: detection of malicious documents through deviation from file format specification. In: Black Hat USA (2016)
8. Boldewin, F.: Analyzing MSOffice malware with OfficeMalScanner. https://ja.scribd.com/document/21143233/Analyzing-MSOffice-Malware-With-OfficeMalScanner
9. OLE Background. https://msdn.microsoft.com/en-us/library/19z074ky.aspx
10. Mimura, M., Otsubo, Y., Tanaka, H.: Evaluation of a brute forcing tool that extracts the RAT from a malicious document file. In: 2016 11th Asia Joint Conference on Information Security (Asia JCIS) (2016). https://doi.org/10.1109/AsiaJCIS.2016.10
11. Corona, I., Maiorca, D., Giacinto, G.: Lux0R: detection of malicious PDF-embedded JavaScript code through discriminant analysis of API references (2014)
12. Liu, D., Wang, H., Stavrou, A.: Detecting malicious Javascript in PDF through document instrumentation. In: 2014 44th Annual IEEE/IFIP International Conference Dependable Systems and Networks (DSN), pp. 100–111, ISBN 978-1-4799-2233-8 (2014)
13. olevba. https://github.com/decalage2/oletools/wiki/olevba
14. python package index gensim 0.10.1. https://pypi.python.org/pypi/gensim/0.10.1
15. python package index scikit learn 0.19.0. https://pypi.python.org/pypi/scikit-learn/0.19.0
16. Virus Toral. https://www.virustotal.com/