Spring 2021

Malware Classification with BERT

Joel Lawrence Alvares
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Artificial Intelligence and Robotics Commons](#), and the [Information Security Commons](#)

Recommended Citation

Alvares, Joel Lawrence, "Malware Classification with BERT" (2021). *Master's Projects*. 998.
DOI: <https://doi.org/10.31979/etd.7n35-garb>
https://scholarworks.sjsu.edu/etd_projects/998

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Malware Classification with BERT

Presented to

Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

By

Joel Alvares

May 2021

The Designated Project Committee Approves the Project Titled Malware Classification

with BERT

by

Joel Lawrence Alvares

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

San Jose State University

May 2021

Prof. Fabio Di Troia

Department of Computer Science

Prof. William Andreopoulos

Department of Computer Science

Prof. Katerina Potika

Department of Computer Science

ABSTRACT

Malware Classification is used to distinguish unique types of malware from each other. This project aims to carry out malware classification using word embeddings which are used in Natural Language Processing (NLP) to identify and evaluate the relationship between words of a sentence. Word embeddings generated by BERT and Word2Vec for malware samples to carry out multi-class classification. BERT is a transformer based pre-trained natural language processing (NLP) model which can be used for a wide range of tasks such as question answering, paraphrase generation and next sentence prediction. However, the attention mechanism of a pre-trained BERT model can also be used in malware classification by capturing information about relation between each opcode and every other opcode belonging to a malware family. Word2Vec generates word embeddings where words with similar context will be closer. The word embeddings generated by Word2Vec would help classify malware samples belonging to a certain family based on similarity. Classification will be carried out using classifiers such as Support Vector Machines (SVM), Logistic Regression, Random Forests and Multi-Layer Perceptron (MLP). The classification accuracy of classification carried out by word embeddings generated by BERT can be compared with the accuracy of Word2Vec that would establish a baseline for results.

***Index Terms* – Natural Language Processing (NLP), Bidirectional Encoder Representations from Transformers (BERT), Word2Vec, Support Vector Machines (SVM), Logistic Regression, Multi-Layer Perceptron (MLP), Random Forests.**

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Fabio Di Troia for his constant support and guidance. The review committee members Prof. Katerina Potika and Prof. William Andreopoulos.

I have learned a lot at SJSU from all my professors. Thank you for all that you have taught me.

Finally, I would like to thank my family. All that I am and all that I ever will be is because of my family.

TABLE OF CONTENTS

I.	INTRODUCTION	3
II.	RELATED WORK	5
III.	BACKGROUND	7
A.	NLP MODELS	8
1)	<i>Word2Vec</i>	8
2)	<i>BERT</i>	12
B.	CLASSIFIERS	15
1)	<i>Logistic Regression</i>	15
2)	<i>SVM</i>	16
3)	<i>Random Forests</i>	17
4)	<i>MLP</i>	19
IV.	EXPERIMENTS AND RESULTS	21
A.	DATASET	22
B.	WORD EMBEDDINGS	23
C.	CLASSIFIER PARAMETERS	26
D.	LOGISTIC REGRESSION RESULTS	27
E.	SVM RESULTS	29
F.	RANDOM FOREST RESULTS	30
G.	MLP RESULTS	31
H.	FURTHER ANALYSIS	33
I.	SUMMARY	34

V. CONCLUSION AND FUTURE WORK.....	36
---	-----------

I. INTRODUCTION

Malware is a piece of code created with the intention to cause harm and damage to useful information or gain unauthorized access to a user's system. A malware could masquerade as a legitimate program but contain malicious content as seen in trojans or encrypt critical information with a ransomware program or modify legitimate code to execute malicious files [1].

Identification and classification of malware is very critical to information security. The world saw a massive shift in the workforce from in-person to remote workspaces when the COVID-19 lockdown began in March 2020. According to the Sophos 2021 threat report [13], malware contributed to 34% of all the breaches in a survey consisting of 3500 IT professionals who worked on remote infrastructure and cloud-based infrastructure.

Each malicious piece of code shares common characteristics within a certain family and tends to differ from malware samples belonging to a different family. It is necessary to identify these unique characteristics which would help classify malware codes belonging to numerous families [4].

Word embeddings can be used to quantify these unique characteristics of a malware sample. The word embeddings can be generated by state-of-the-art machine learning models such as BERT [2] and Word2Vec [5]. The embeddings would capture useful information that would serve as training features for classification models. In this research, the focus is on the effectiveness of the word embeddings generated in the context of malware classification.

As seen in Fig. 1, the input dataset of malware samples will be processed and transformed into inputs for BERT and Word2Vec that will generate the word embeddings.

Fig. 1. Illustration of the organization of the project

The word embeddings generated are classified with the help of multi-class classifiers such as SVM, Random Forests, MLP and Logistic Regression to their respective malware families. The overall accuracy would depend on the classification of the word embeddings which capture the essential characteristics of the malware samples.

The remainder of the report is organized in the following sections. It starts with a survey of relevant work in Section 2 followed in by the topics and building blocks of the research such as background of the word embedding models and classification models in Section 3. Next, experiments carried out and the results in Section 4 are discussed followed by the conclusion and future work Section.

II. RELATED WORK

Malware are written with the intention of causing harm by carrying out unauthorized access to personal information, causing harm to an innocent user's computer application by inserting malicious code, by providing false information to the user in order to extract money and a number of other ways. Malware writers are constantly on the lookout for a security faults that can be breached and malware are constantly updated. Malware developers try to mask a malicious code as a benign one so that it cannot be detected by malware recognition software [10]. This is the reason malware recognition has become a challenging task.

A lot of malware recognition techniques rely on signature-based detection. The anti-virus program that relies on signature-based detection generally computes the hash of the files and compares it with the hash of known malware signatures [46]. However, morphing or modifying the code by inserting benign code or dead code within malicious code is one easy way to avoid detection. It is also quite inefficient because all the files of a given user are scanned and compared with known available malicious signatures which takes a lot of time.

According to [45], a number of metamorphic malwares such as Evol, Zmist, Zperm, Regswap, MetaPHOR morph after each new infraction. Detecting these malware samples is challenging and it can beat signature-based malware recognition. Metamorphic malware morphs the code by using a combination of substitution, insertion, deletion, and transposition. However, the metamorphic malware can be identified by machine learning techniques because they are able to notice the subtle differences between malware and

benign samples despite the use of morphing. The different strategies that can be used to detect metamorphic malware using HMM is explained in [49].

The effectiveness of the different machine learning techniques depends on the input features extracted from the dataset. The different features that can be used are signatures [46], API calls [11], opcodes [7], opcode graphs [14] and many other features.

Natural language processing (NLP) techniques extract rich information from sentences of a language known as word embeddings and are able to identify the meaning of the sentence, generate sentences with similar meaning or fill the blanks within a sentence. The NLP models extract information of the relation of a word with every other word of a sentence. The model groups together words with similar meaning in the input dataset it is provided and maps it on to higher dimensional space where words of similar meaning are grouped closer together. This information helps NLP models carry out several classification and prediction tasks.

The NLP models can be used in the field of malware recognition to generate embeddings for malware samples. The malware samples that belong to the same family would have features that are closely related. This information could be used by classifiers to group together malware samples that belong to the same family.

BERT is one such NLP model that can be used to generate word embeddings that captures information of every component of the input with respect to every other component. [2] talks about the architecture of transformers and the attention mechanism of BERT while [3] focuses on analyzing the attention heads of the BERT model and not the model's output. The attention heads of BERT capture various patterns and linguistic notions.

Word2Vec was used in previous research to generate word embeddings for malware samples [20]. These malware samples were classified using classifiers such as MLP, k-nearest neighbors, random forests and SVM as explained in [8]. It performed very well when compared to traditional machine learning techniques such as HMM and PCA. The opcode sequences within malware samples are treated as a language in [6] and context is captured using Word2Vec. The classification is carried out using k-nearest neighbors (k-NN). Word2Vec is used to generate features for opcode sequences and classification is carried out using deep neural networks in [34]. Alternative strategies to generate embedding vectors is discussed in [32].

The results derived by utilizing word embeddings generated by Word2Vec to carry out malware classification proves that NLP based models can extract rich features that assist with classification accuracy. It calls for testing out newer NLP based models such as BERT that is a transformer-based model that consists of encoders and decoders along with an attention mechanism [2]. The BERT model will be explained in further detail in the next section. The experiments carried out in this research primarily focus on generating embeddings using BERT and comparing the classification accuracy with Word2Vec using the same dataset. The embeddings would be classified using a variety of classifiers. The accuracy would be assessed on challenging and recent malware samples.

III. BACKGROUND

This section provides more details on the key components i.e., the NLP models and the classifiers used in this research. It provides information about the background and architecture of the topics. The NLP Models will describe BERT and Word2Vec while the classifiers will describe SVM, Random Forests, Logistic Regression and MLP. The results

and the experiments carried out using these building blocks will be described in the next section.

A. NLP Models

Natural Language Processing (NLP) is the subfield of Artificial Intelligence (AI) that enables machines to understand the language spoken by humans. The models that help achieve it are known as NLP models. Training an NLP model from scratch is a tedious task and it requires a massive dataset and massive computation resources. For this reason, a pre-trained NLP model is used to carry out tasks related to NLP. Transfer learning is the technique used to transfer the knowledge gained by the model during training to carry out other tasks on a new dataset it has never been exposed to before. NLP tasks such as sentiment analysis, next sentence prediction, word embedding generation and so on. The following sections will cover two NLP models namely: BERT and Word2Vec. These two NLP models are used to generate word embeddings for malware samples. The word embeddings generated by the NLP models is used by classifiers to carry out multi-class malware classification.

1) Word2Vec

The original work that provides details about Word2Vec can be obtained from [30] while a paper provides enhancements that enable the use of Word2Vec with large datasets as explained in [31].

Word2Vec is used to convert the input sequence of words to vectors and map them to a higher dimension. [21] explains how Word2Vec uses neural networks to group together words with a similar meaning closer together. For example, consider the following set of words.

$w_0 = \text{"queen"}, w_1 = \text{"man"}, w_2 = \text{"woman"}, w_3 = \text{"king"}$

Fig. 2. Using Word2Vec to generate embeddings [25]

The words above are mapped to a higher-dimensional space by Word2Vec.

Cosine similarity works with any number of dimensions and words with the greatest

Cosine similarity are synonymous in nature.

An example word embedding for the word “king” is as follows [25]:

```
[ 0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 ,
-0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961
, -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 ,
-0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 ,
-1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042 ]
```

Let us color code the values based on the numbers such that red represents a value close to 2, blue represents a value close to -2 and white represents a value close to 0.

Fig. 3. Word embeddings represented as a color map [25]

Based on the Fig. 3 it can be observed that:

- The words woman and girl are quite similar to each other in a lot of positions.
- The words “boy” and “girl” are similar in certain positions to each other, but these positions are different from “woman” or “man”. It could be capturing something similar between the words “boy” and “girl” i.e., youth.
- The embeddings can be added and subtracted in order to form relations between words. For instance, in the following case where the word embedding for the word “queen” is subtracted with the word embedding for the word “woman” and the word embedding for “man” is added then the resultant word embedding is very close to the word embedding for the word king. It can be represented as follows:

$$\text{queen} - \text{woman} + \text{man} \approx \text{king}$$

Associating negative weights with frequently used words is another technique to improve the rate of training. The positive weights associated with the model are updated and only a sample set of the negative weights are updated while generating the output vectors. This reduces the impact of frequently used words while training the Word2Vec model.

Fig. 4. Network that generates Word2Vec embeddings [17]

The Word2Vec model is used to generate word embedding for malware samples in this research by using a window of size 6 and output size of 2 dimensions. We use the output generated by the Word2Vec model to generate unit vectors and plot a circular heat map which will be discussed in further detail in the next section.

2) BERT

BERT is a transformer-based NLP model that is used to carry out language-based tasks such as masked word prediction, sentiment classification and other classification tasks. The architecture consists of nothing but a stack of trained Transformer Encoders. BERT is able to generate the word embedding for a particular word by also taking into account the context in which it was used known as contextualized word embeddings.

Fig. 5. Trained BERT Components [15]

The encoder uses attention to map the input to a set of vectors which store information of a given word with respect to every other word in the sentence. Let us consider the following input sentence:

The boy drank water because he was thirsty.

The word 'he' in the above sentence is associated with the boy and the BERT model can identify this relation using attention. Attention helps BERT understand other relevant words in the sentence compared to the one that is currently being processed.

As seen in Fig. 5 the BERT model can accept at most 512 words as input. In general, a sentence in natural language does not exceed 512 words but the opcodes in a malware sample can exceed 512. For this reason, only the first 400 opcodes from each malware sample gave us promising results.

The BERT model used as a part of the experiments is DistilBERT which is a smaller version of BERT that was open sourced by the HuggingFace team. DistilBERT performs as good as BERT but it is lightweight and faster. The DistilBERT model used is pre-trained in the English language. However, the model is neither trained nor is it fine-tuned to carry out malware classification. Classification can be carried out using the word embeddings generated by DistilBERT. The [CLS] token from BERTs output captures the information about the entire sentence. In case of a malware sample, the [CLS] token would capture information about the entire malware sample. BERT uses this information to carry out NLP tasks such as next sentence prediction. This information can be used in malware classification as the [CLS] token from the embedding generated for a malware sample would capture information that would help with classification.

Fig. 6. Slicing BERT Word Embedding [15]

For instance, if there are 2000 malware samples that BERT is trained on and let 66 be the length of the tokens in the longest malware opcode sequence as seen in the Fig. 6 above. Out of the 768 hidden unit output of BERT only the first one representing the [CLS] token will be extracted. A label will be assigned to each of the 2000 sentences depending on the class of the malware sample as seen in the Fig. 7.

Fig. 7. Assigned labels to BERT embeddings [15]

The sliced embeddings of BERT along with the class labels will be classified by the classifiers. A total of 5 malware families are considered which will be described in the dataset section of report. The results of the classifiers and the parameters that attain the best results will be explored as part of the experiments.

B. Classifiers

Classification is the process of predicting the class or label of the input dataset. The input data set is mapped to the desired output class depending on the features of the input data. The machine learning models which enable the user to map the input data to its corresponding class is known as a classifier. This section will cover the background of all the classifiers used in the experiments.

1) Logistic Regression

Logistic regression is used to describe the input data and to find a correlation between them. A detailed explanation and various strategies guidelines for logistic regression can be found in [40] and [44]. The different applications of logistic regression and how it differs from linear regression can be found in [39] and [42]. A brief and excellent explanation of Logistic regression can be found in [43].

The result of logistic regression is dichotomous in nature. A logistic regression model used to fit more than two classes is referred to as multinomial logistic regression. The model carries out classification using multinomial probability distribution [9].

The hypothesis of logistic regression is a sigmoid function can be defined as follows:

$$f(x) = \frac{1}{1 + e^{-(x)}}$$

The pros and cons of logistic regression are similar to linear regression such as being prone to outliers, assumption of linearity amongst dependent variables and independent variables. However, logistic regression model provides

probabilities, and it is not just a classification model. It enables the user to identify the percentage with which a certain instance was assigned a class.

2) SVM

The main objective of SVM is to carry out classification between the dataset by maximizing the distance between the separating hyperplane and the dataset. As seen in the Fig. 8 the hyperplane with the maximum separation is chosen. The support vectors are the data points closest to the hyperplane. The support vectors are used by SVM to maximize the separation between the data points and the hyperplane. SVMs can be used to identify the subtle changes in malware samples belonging to a certain family as discussed in [47]. The classification process of SVMs and the mathematical proof can be found in [35].

Fig. 8. SVM for binary classification [27]

SVM identifies that the dataset may not be linearly separable by itself. The dataset can be mapped to a higher dimensional space where a separating hyperplane can classify the dataset. As seen in the Fig. 9 the data on the left is not linearly separable. However, the data can be easily separated by a

separating hyperplane if the data is mapped as seen on the right. One of the ways this can be achieved is by using a polynomial kernel. Identifying the right kernel can be a challenging task but it can significantly improve the classification accuracy without causing a major computation overhead.

Fig. 9. Mapping input data to a higher dimension [27]

3) *Random Forests*

Random Forests carries out the classification of the dataset using an ensemble of decision trees. Every tree in the random forest classifies the data and the class with the highest number of votes is selected as the class of the data.

As mentioned in [16], a large number of these trees carry out the classification together as a committee and the overall accuracy of such a committee outperforms the accuracy of an individual tree. An individual decision tree tends to overfit the input dataset. However, a group of trees tends to protect each other from their individual errors. The groups tend to move together in the right direction as seen in the Fig. 10.

The decision trees of random forests may be too correlated with each other. This is a problem that arises with Random Forests. Bagging which stands for Bootstrap Aggregation is to overcome the problem. It takes advantage of the fact that the decision trees are sensitive to the data they are trained on. The decision trees are formed using random samples of the training data which may or may not overlap. Bagging prevents the random forest from overfitting the data and reduces the correlation between the decision trees. A great explanation of random forests can be found in [36]

Fig. 10. Merging decisions from Decision Trees [33]

4) MLP

A neuron is the building block for a Multi Layered Perceptron (MLP). Multiple neurons known as McCulloch-Pitts Artificial Neuron [22] are placed in different layers and the inputs of the neurons in the hidden or intermediate layers are outputs of neurons in the previous layer. A neuron with 3 inputs and a single output is depicted in Fig. 11. The inputs are X_0 , X_1 and X_2 and w_0 , w_1 , w_2 are the weights associated with these inputs. The neuron generates an output Y $\{0, 1\}$ where 1 implies that the neuron was activated while 0 implies that the neuron remained inactive. The inspiration for this neuron comes from the neurons which form a complex network in the human brain as explained in [18] and [37]. The weights together with the input determine if the neuron should fire or not. If the $\sum w_i X_i$ is greater than the threshold T , then the neuron will activate.

Fig. 11. Neuron of a Neural Network [17]

The following equation as given in [48] represents the function that a neuron of an MLP utilizes. The bias b is introduced that is independent of the input and

the weights associated with the neuron. However, even the bias is updated during the training of the MLP.

$$f(X) = \sum_{i=0}^{n-1} w_i X_i + b$$

In case of binary classification if the above function generates a positive value, then we classify the input as class 1 and if the function generates a negative value, then the input is classified as class 2. The decision boundary of the binary classifier can be represented by the following equation. The decision boundary separates the inputs into the two classes in the output dimension space.

$$f(x, y) = w_0x + w_1y + b$$

An MLP consists of multiple layers of these perceptron's as shown in Fig. 12 which consists of two hidden layers. Each edge of the MLP has a weight associated with it and the weights of the MLP are finalized after training. The MLP can provide additional weightage to certain input features and generate a decision boundary that is quadratic, cubic and many other shapes depending on the one that best classifies the given input features.

SVMs can be compared to MLPs as a single layer perceptron will classify the data similar to an SVM with a linear kernel. The goal of the MLP is to fit a decision boundary that best classifies the data same as SVM. Additionally, the SVM tries to maximize the distance between the inputs and the separating hyperplane.

Fig. 12. Multi Layered Perceptron [17]

A neural network deals with a dataset that is not linearly separable by adding additional layers of neurons and this increases the dimensionality of the decision boundary. Such networks with multiple layers are known as deep neural networks. In order to achieve similar results with SVMs, a nonlinear kernel function needs to be used. However, the n -layers of an MLP can achieve much better results for a challenging input dataset where SVMs might fall short.

IV. EXPERIMENTS AND RESULTS

The dataset is described, followed by the word embeddings and the results based on classification carried out on the word embeddings generated by BERT and Word2Vec will be discussed in this section.

A. Dataset

All the experiments carried out as a part of this research were based on the malware families described in Table 1 along with the number of malware samples for each family.

Table 1: Malware dataset information

Malware Family	Malware Type	# of Samples
CeelInject	VirTool	899
FakeRean	Rogue	899
OnlineGames	Password stealer	900
Winwebsec	Rogue	897
Renos	Trojan Downloader	900

A brief description of each of the malware families listed above is as follows:

1. **CeelInject** is malware that is generally used along with other malware families as it is used to conceal the malware. The malware that CeelInject is used along with would be installed in a user's machine without requesting any permissions [24].
2. **FakeRean** alerts the user for issues or viruses that do not exist on the system and asks for a sum of money in order to assist the user [29].
3. **OnlineGames** is used to track the login information of online games and keeps track of information of online gamers without consent [26].
4. **Winwebsec** belongs to trojan family. It pretends to be a legitimate antivirus software and informs the user that the system is corrupt and needs to be fixed.

It tries to scare the user with the intention of extracting money from the user [28].

5. **Renos** is malware that informs the user of fake security warnings once it is downloaded and requests for payments to resolve any issues [23].

Samples from these malware families are classified. The results of the classification are discussed in the following sections.

B. Word Embeddings

Word Embeddings are used in natural language processing as a representation of the words of a sentence in vector values such that words of similar meaning are grouped closer together in the vector space. It helps group together words of similar meaning and identifying the meaning of the sentence. This information can be used by classifiers to identify key features and efficiently carry out classification.

The goal of this project is to carry out malware classification which cannot be done by feeding the malware samples to the classifiers. As the malware samples are a sequence of opcodes that cannot be used by the classifier to carry out classification. Features need to be extracted from the malware samples, which can be done by generating word embeddings from the malware samples. These word embeddings would capture information and group together features that are unique to a malware family.

The word embeddings are generated using NLP based models such as Word2Vec and BERT. These word embeddings generated for every opcode in a malware sample can be represented as unit vectors and plotted in a circular heat map as seen in the figures below.

Fig. 13. Vectors for CeelInject

Fig. 14. Vectors for FakeRean

Fig. 15. Vectors for OnlineGames

Fig. 16. Vectors for Renos

Fig. 17. Vectors for Winwebsec

As seen above the circular heat map representation of the opcodes seem to differ for every malware family and the opcodes with higher frequencies across malware families seem to be the opcodes push, mov and add.

C. Classifier Parameters

The parameters that were selected for the classification are shown in Table 2. The classifiers used to carry out the classification of the malware samples were imported from the libraries of scikit-learn [38]. After trying out multiple parameters, conducting numerous experiments and using GridSearchCV the ideal parameters obtained were these values [41].

Table 2. Parameters used by classifiers

Classifier Model	Parameter	Word2Vec	BERT
Logistic Regression	C solver multi_class	42.1 Lbfgs auto	42.1 newton-cg multinomial
SVM	C kernel gamma	1000 Rbf 1	1000 rbf 1
Random Forests	max_depth n_estimators	20 100	20 100
MLP	hidden layer size activation function solver numer of iterations learning rate	(150,150,100) relu adam 3000 constant	(100,100,100) relu adam 10000 invscaling

The classifier parameters obtained are almost identical for the features generated by BERT and Word2Vec as seen above.

D. Logistic Regression Results

The logistic regression model was used to classify a total of approximately 4500 malware samples. Optimal results were obtained by the logistic regression model using the regularization parameter value $C = 42.1$ with an overall test accuracy of 83.54% using the word embeddings generated by BERT. The parameters specified above were used by the logistic regression classifier giving a test accuracy of 81.2% using the word embeddings generated by Word2Vec. The confusion matrix of the results obtained is shown in Fig. 18 and Fig. 19.

Fig. 18. Confusion matrix of Logistic Regression for Word2Vec features

Fig. 19. Confusion matrix of Logistic Regression for BERT features

The different values for C were obtained using numpy's linspace function by dividing the range 0.0001 to 100 into 20 parts. The classifier is used to classify the word embeddings generated from both Word2Vec and BERT. The overall accuracy is poor when compared with the other classifiers. One of the reasons is that the model is overfitting the decision boundary to the training dataset. This causes the model to perform poorly when exposed to new data.

E. SVM Results

Experiments were carried out on the SVM model and the ideal set of parameters that produced the maximum accuracy were selected. The different types of kernels tried out were rbf, linear and polynomial along with the regularization parameter C in the range 10 to 1000 and gamma value in the range 0.001 to 0.1.

Fig. 20. Confusion matrix of SVM for Word2Vec features

Fig. 21. Confusion matrix of SVM for BERT features

SVM maps the input features to a higher dimension space in order to form a decision boundary that separates the features into different classes. For this reason, SVM is able to successfully leverage the features in the word embeddings and group together malware samples with similar features and give a high classification accuracy of around 91.01% using the embedding generated by BERT. The embeddings generated by Word2Vec gave a classification accuracy of 86.8%.

F. Random Forest Results

Random forest is a neighborhood-based algorithm that classifies input features by grouping features that are closer to each other and making decisions at different stages which segregate the inputs into different classes [19].

The results of the experiments conducted show that the random forest classifier performs better when the number of trees and the depth of the random forest is increased. The optimal parameters lead to a classification accuracy of 91.81% with embeddings generated by BERT while the embeddings generated by Word2Vec gave a classification accuracy of 89.6%.

Fig. 22. Confusion matrix of SVM for Word2Vec features

Fig. 23. Confusion matrix of SVM for BERT features

G. MLP Results

The multi-layered perceptron performs quite closely as the SVM by mapping the input features to a higher dimensional space and carrying out classification by forming a decision boundary and grouping features that are closer to each other. A constant learning rate with a 30,30,30 hidden layer size and relu activation function provided the best results. The classifier converged and gave optimal results at around 10000 iterations. The final accuracy obtained using the word embeddings generated by BERT was 86.83% which is quite close to the accuracy obtained by SVM. The word embeddings generated by Word2Vec gave a final accuracy of around 86.6% which is very close to the one obtained by BERT.

Fig. 24. Confusion matrix of SVM for Word2Vec features

Fig. 25. Confusion matrix of SVM for BERT features

H. Further Analysis

Random forest is a neighborhood-based classification model. It seems that the model performs poorly when the depth of the binary tree of the decision is shallow. It tends to overfit to the training data as the classification accuracy on training data was high, but the model performed poorly when tested on the test data. The Fig. 26 below shows that embeddings generated from both BERT and Word2Vec show improvement in classification accuracy when the depth of the Random Forest is increased. The accuracy plateaus at a depth of 10 and gradually increases beyond this point.

Fig. 26. Depth vs accuracy for RF using BERT and Word2Vec

On carrying out further analysis it was observed that behavior is similar when both the depth and number of trees of the random forest were increased. It is also observed that a larger number of trees in the random forest classification model compensate for shallow depths. As seen in the Fig. 27 the accuracy of the random forest model is high

even when the depth of the decision trees is around 2.5. Beyond a depth of 2.5 there is a gradual increase in classification accuracy as the number of trees of a random forest classifier is increased. As described in Section III of the report, a larger number of decision trees can generalize to the training data. A class is chosen for the input data only when a majority of the decision trees generate the same classification which protects the classification result from errors caused by individual decision trees. This is in line with the results obtained as a part of the experiments conducted.

Fig. 27. Accuracy for depth vs number of trees in random forest

I. Summary

Classification of malware samples carried out using BERT performs better overall in comparison to Word2Vec as seen in the Fig. 28 summarizing the results. Word embeddings were generated by BERT and Word2Vec and the word embeddings were classified using classifiers such as Logistic Regression, SVM, MLP and Random Forests. SVM, MLP and Random Forests perform better overall in comparison to

Logistic Regression which is an expected outcome. MLP and SVM perform similarly as they try to find the decision boundary that best fits the data without overfitting it. Random forests use an ensemble of decision trees to carry out the classification of the dataset and obtain a high classification accuracy.

The results of this experiment prove that word embeddings generated by BERT can be used to carry out multi-class malware classification of the dataset. The classification accuracy obtained using BERT embeddings is quite comparable to the classification accuracy obtained using Word2Vec embeddings for classification. The number of opcodes selected per malware sample was 400 since BERT required a maximum of 512 words per sentence. Some of the malware samples had over 10^5 opcodes. However, the classification accuracy was not impacted even though the number of opcodes used was significantly reduced for some of the malware samples. This proves that word embeddings can capture rich features with even a small subset of the opcodes for each malware sample.

Fig. 28. Accuracies obtained after classification

V. CONCLUSION AND FUTURE WORK

As a part of the experiments conducted in this research, it was observed that when the malware samples were mapped to word embeddings by capturing, grouping, and enriching the key components of input features, it led to an improvement in classification accuracy while carrying out malware classification. Word embeddings can be generated in several ways. However, there was no prior research conducted on using embeddings generated by BERT to carry out malware classification and comparing it with Word2Vec.

The results documented as a part of this research show that BERT performs very well by capturing information that helps the classifier improve the classification accuracy. The results are better than using Word2Vec using the same set of input parameters and the same set of classifiers. It proves that a transformer-based model such as BERT has applications beyond NLP.

In the future, more research can be conducted in this area by using different versions of BERT. Distilbert-base-uncased was used in these experiments but further research can be carried out using the other available versions of BERT. The BERT model is trained on natural language input, but the model will be able to generate more rich features if it is trained on malware samples. Research can be carried out using more malware families with more complex sets of data and observing how BERT captures the key information across multiple malware families. BERT can be compared with other word embedding techniques and scenarios, where BERT does not perform well, can be identified.

REFERENCES

- [1] P. Baldi and Y. Chavin. Smooth on-line learning algorithms for hidden Markov models. *Neural Computation*, 6:307–318, 1994.
<https://core.ac.uk/download/pdf/4881023.pdf>.

- [2] A. Vaswani et al. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017.
- [3] K. Clark, U. Khandelwal, O. Levy. and C.D. Manning. What does bert look at? An analysis of bert's attention. arXiv preprint arXiv:1906.04341, 2019.
- [4] S. Basole, F. Di Troia, and M. Stamp. Multifamily malware models. Journal of Computer Virology and Hacking Techniques, 2020.
- [5] S. Banerjee. Word2Vec — A baby step in deep learning but a giant leap towards natural language processing. <https://medium.com/explore-artificial-intelligence/word2vec-a-baby-step-in-deeplearning-but-a-giant-leap-towards-natural-language-processing-40fe4e8602ba>, 2018.
- [6] Y. Awad, M. Nassar, and H. Safa. Modeling malware as a language. In 2018 IEEE International Conference on Communications, ICC, pages 1–6, 2018.
- [7] D. Bilar. Opcodes as predictor for malware. International Journal of Electronic Security and Digital Forensics, 1(2):156–168, 2007.
- [8] A. Chandak et al. A Comparison of Word2Vec, HMM2Vec, and PCA2Vec for Malware Classification. In Malware Analysis Using Artificial Intelligence and Deep Learning (pp. 287-320). Springer, Cham, 2021.
- [9] C. Molnar, “Interpretable Models,” Interpretable Machine Learning, <https://christophm.github.io/interpretable-ml-book/>, 2021, ch. 4.
- [10] A. Dhammi and M. Singh. Behavior analysis of malware using machine learning. In Eighth International Conference on Contemporary Computing, IC3 2015, pages 481–486, 2015.
- [11] O. Hachinyan. Detection of malicious software on based on multiple equations of API-calls sequences. In 2017 IEEE Conference of Russian Rountg Researchers in Electrical and Electronic Engineering, EIConRus, pages 415– 418, 2017.
- [12] L. Hardesty. Explained: Neural networks. <http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>, 2017.
- [13] SophosLabs, Sophos Managed Threat Response, Sophos Rapid Response, Sophos AI, Cloud Security. <https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophos-2021-threat-report.pdf>, 2021.
- [14] H. Hashemi, A. Azmoodeh, A. Hamzeh, and S. Hashemi. Graph embedding as a new approach for unknown malware detection. Journal of Computer Virology and Hacking Techniques, 13:153–166, 2016.
- [15] A Visual Guide to Using BERT for the First Time <https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time>, 2019

- [16] M. Armanious, "Classification of Wins," Men's U-Sports Basketball Analysis, https://bookdown.org/michael_arman7/Basketball-Analysis/, 2019, ch. 4.
- [17] M. Stamp et al., "Alphabet soup of Deep Learning topics", <https://www.cs.sjsu.edu/~stamp/RUA/alpha.pdf>, 2019.
- [18] A. Liaw and M. Wiener. Classification and regression by randomForest. R news, 2/3:18–22, 2002.
- [19] Y. Lin and Y. Jeon. Random forests and adaptive nearest neighbors. Journal of the American Statistical Association, 101(474):578–590, 2006.
- [20] Y. Liu and Y. Wang. A robust malware detection system using deep learning on API calls. In 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC, pages 1456–1460, 2019.
- [21] C. McCormick. Word2vec tutorial - The skip-gram model. <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model>, 2016.
- [22] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 1943. <https://pdfs.semanticscholar.org/5272/8a99829792c3272043842455f3a110e841b1.pdf>.
- [23] Microsoft Security Intelligence. Renos. <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=TrojanDownloader:Win32/Renos&threatId=16054>, 2006.
- [24] Microsoft Security Intelligence. CeelInject. <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=VirTool%3AWin32%2FCeelInject>, 2007.
- [25] The Illustrated Word2vec <https://jalammar.github.io/illustrated-word2vec/>, 2019
- [26] Microsoft Security Intelligence. OnLineGames. <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=PWS%3AWin32%2FOnLineGames>, 2008.
- [27] M. Stamp. *Introduction to Machine Learning with Applications in Information Security*. Chapman and Hall/CRC, Boca Raton, 2017.
- [28] Microsoft Security Intelligence. Winwebsec. <https://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32%2fWinwebsec>, 2010.

- [29] Microsoft Security Intelligence. FakeRan. <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Win32/FakeRan>, 2011.
- [30] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. <https://arxiv.org/abs/1301.3781>, 2013.
- [31] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. <https://papers.nips.cc/paper/5021-distributed-representationsof-words-and-phrases-and-their-compositionality.pdf>, 2013.
- [32] C. Moody. Stop using word2vec. <https://multithreaded.stitchfix.com/blog/2017/10/18/stop-using-word2vec/>.
- [33] T. You, Understanding Random Forest <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
- [34] I. Popov. Malware detection using machine learning based on Word2Vec embeddings of machine code instructions. In 2017 Siberian Symposium on Data Science and Engineering, SSDSE, pages 1–4, 2017.
- [35] M. Stamp et al., “A Reassuring Introduction to Support Vector Machines,” *Introduction to Machine Learning with Applications in Information Security*, R. Herbrich, T. Graepel, Boca Raton, FL, U.S., CRC Pr. 2017, ch. 11, pp. 95-124.
- [36] M. Stamp et al., “Many Mini Topics,” *Introduction to Machine Learning with Applications in Information Security*, R. Herbrich, T. Graepel, Boca Raton, FL, U.S., CRC Pr. 2017, ch. 11, pp. 186-192.
- [37] F. Rosenblatt. Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. <http://www.dtic.mil/dtic/tr/fulltext/u2/256582.pdf>, 1961.
- [38] scikit-learn: Machine learning in Python. <https://scikit-learn.org/stable/>.
- [39] Me. Scott, *Applied logistic regression analysis* Vol. 106. Sage, 2002.
- [40] D. G. Kleinbaum, et al., *Logistic regression*, New York: Springer-Verlag, 2002.
- [41] sklearn.model selection.GridSearchCV. https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.
- [42] D. W. Hosmer Jr, S. Lemeshow, R.X. Sturdivant, *Applied logistic regression*, Vol. 398. John Wiley & Sons, 2013.
- [43] F.C. Pampel, *Logistic regression: A primer*, Vol. 132. Sage publications, 2020.

- [44] D.G. Kleinbaum et al. Logistic regression. New York: Springer-Verlag, 2002.
- [45] Symantec. Internet security threat report: Malware.
<https://interactive.symantec.com/istr24-web>, 2019.
- [46] P. Vinod, R. Jaipur, V. Laxmi, and M. Gaur. Survey on malware detection methods. In Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security, IITKHACK'09, pages 74–79, 2009.
- [47] M. Wadkar, F. Di Troia, and M. Stamp. Detecting malware evolution using support vector machines. Expert Systems with Applications, 143, 2020.
- [48] C. Wallis. History of the perceptron.
<https://web.csulb.edu/~cwallis/artificialn/History.htm>, 2017.
- [49] W. Wong and M. Stamp. Hunting for metamorphic engines. Journal in Computer Virology, 2(3):211–229, 2006