Spear-Phishing Detection Method Based on Few-Shot Learning

Qi Li(✉) and Mingyu Cheng

Beijing University of Posts and Telecommunications, Beijing 100876, China
{liqi2001,chengmingyu}@bupt.edu.cn

Abstract. With the further development of Internet technology, various online activities are becoming more frequent, especially online office and online transactions. This trend leads that the network security issues are increasingly prominent, the network security situation is more complex, and the methods and means of attacks are emerging in endlessly. Due to the characteristics of spear-phishing such as target accuracy, attack durability, camouflage concealment and damage severity, it has become the most commonly used initial means for attackers and APT organizations to invade targets. Thus, automated spear-phishing detection based machine learning and deep learning have become the focus of researchers in recent years. However, because of a smaller range and less attack frequency, the number of spear-phishing emails is very limited. How to detect spear-phishing based on machine learning and deep learning with small samples has become a key issue. Meanwhile, in machine learning and deep learning, few-shot learning aims to study a better classification model trained with only a few samples. Therefore, we propose a spear-phishing detection method based on few-shot learning that combines the basic features and the message body of emails. We propose a simple word-embedding model to analyzes the message body, which can process the message body of different lengths into text feature vectors with the same dimension, thus retaining the semantic information to the greatest extent. Then the text feature vectors are combined with the basic features of emails and input into commonly used machine learning classifiers for detection. Our proposed simple word-embedding method does not require the complex training of the model to learn a large number of parameters, thereby reducing the dependence of the model on a large number of training data. The experimental results show that the method proposed in this paper achieves better performance than the existing spear-phishing detection method. Especially, Especially, the advantages of our detection method are more obvious with small samples.

Keywords: few-shot learning · spear-phishing email · machine learning

1 Introduction

With the development of Internet technology, email has become not only an indispensable communication means in people's daily work, but also an important channel for obtaining work information or documents. Therefore, more and more attackers tend to use email as the carrier of malicious links or files, they always utilize social engineering to induce the recipients to click malicious links or download malicious attachments. Compared with the attack through malware directly, phishing emails have their special characteristics. For example, phishing emails have less technical requirements for attackers and it does not require the attackers to master the zero-day vulnerability or develop specific attack tools. Instead, it is usually centered on social engineering, that is, the attacker only needs to construct a seemingly innocuous email of interest to the recipients. In conclusion, phishing emails have become a popular way of network attack because of its lower cost and higher validity.

Spear-phishing is a special form of the phishing email. Compared with the general phishing email, it has the characteristics of target accuracy, attack durability, camouflage concealment and damage severity. At the beginning of the attack, the attacker needs to continuously investigate the target receiver or organization. After collecting enough personal information of the receiver, the attacker will carefully construct an exclusive email and send it at an appropriate time. Since the content of the email is closely related to the receiver, the success rate of the spear-phishing is usually high. According to the framework of "adversarial tactics, techniques, and common knowledge (ATT&CK)" proposed by MITRE, spear-phishing is often used as a starting point for advanced persistent threat (APT) attacks [1]. Many active APT organizations, such as OceanLotus and SideWinder, usually tend to use emails to deliver malware to their targets [2, 3]. There is evidence that the members of important organizations such as governments, companies, enterprises and authorities are the groups that are most vulnerable to phishing attackers [4]. Once the attack is successful, it will result in a leak of confidential data and bring serious damage for businesses, nonprofit organizations, governments and even national security.

Due to the huge damage caused by spear-phishing in recent years, relevant institutions and enterprises have begun to study automated detection methods to filter harmful emails. In the early days, researchers tried to analyze emails from multiple dimensions, including the sender's IP, domain, and subject keywords. They set up a blacklist or whitelist to filter out malicious emails [5–10]. However, the attackers can easily bypass the detection by forging identities or addresses. With the development of dynamic debugging technology, the sandboxes are widely used in phishing mail detection and have achieved good results [11, 12]. However, in recent years, attackers have begun to use anti-sandbox or cloud attachments technology to evade detection from anti-virus engines [13], even many attackers don't carry malicious attachments directly in the emails. At the same time, the statistics of Lawrence Berkeley National Lab shows that none of the successful spear-phishing attacks involved a malicious attachment [14]. In order to make up for the shortage of current methods, researchers introduced artificial intelligence to the detection of malicious emails. They first extracted a variety of email characteristics (mainly including sender's IP, sender's domain, sending time, embedded links, etc.), and then combined sandbox results to detect the potential malicious emails. Only few researchers take the semantic information into account into the detection process [15, 16]. It is well known that the content of the email shows the attacker's inducing skill and diction habits, which are very helpful for describing the characteristics of attacking organizations. In our previous work, we analyzed the message body of the email and use long-short term memory (LSTM) to judge whether an email is valid or not through the content of the email. However, this method cannot be used in spear-phishing detection directly. In spear-phishing, the attacker is more cautious, with a smaller range and less attack frequency, so the number of training samples cannot meet the demands of deep learning. Therefore, it is necessary to introduce the few-shot learning method to detect the most harmful attack.

In this paper, we proposed a few-shot learning method to detect spear-phishing emails, which extracted features from the mail header and message body respectively. When analyzing the message body of an email, we proposed a simple word-embedding method that used concatenated pooling to optimize features of the content. This method can reduce the dependence on model complexity, therefore perform better results without a large amount of training data. Moreover, in order to solve the problem caused by different lengths of training data, we used concatenated pooling to obtain vectors with consistent dimensions, which avoid information loss caused by padding and truncation during data processing. After processing the message body, we combined the processed result with other features as an input vector of machine learning methods.

The rest of the paper is organized as follows: In Sect. 2, we introduce the related works. In Sect. 3, we describe the framework of spear-phishing detection proposed in detail. The methods are presented in Sect. 4, including feature extraction of mails and processing of the body of mails. Section 5 details the experiment and evaluates the results. In Sect. 6, we conclude our work.

2 Related Work

2.1 Spear-Phishing Detection Technology

The traditional detection methods of phishing emails are mainly divided into three categories: detection methods based on the black-and-white list, methods based on sandbox and methods based on using machine learning and deep learning model.

The detection method based on blacklist and whitelist compare the URL connection and IP with lists in database. Currently, the commonly used blacklist of phishing websites includes Phishitank and Openphish. Jain et al. designed a method to automatically update the whitelist through historical data, so as to realize the reliability analysis of email domain name [8]. This method can complete the detection of phishing emails through simple and convenient query operations. However, since it relies on the comprehensive blacklist and whitelist, the attacker can bypass the detection by changing the IP address and URL link of email, which makes this method invalid for the unknown attack detection of the phishing email.

The detection method based on sandbox uses sandbox technology to analyze the static characteristics and dynamic behavior of phishing email attachments. Han et al. proposed a new sandbox tool to detect attachments of email [12], but this method cannot be applied to encrypted attachments. Besides, with the emergence of anti-sandbox technology, it is difficult to catch malicious behaviors of attachments simply using sandbox technology. At the same time, sandbox-based detection method cannot be applied to detecting phishing emails with cloud attachment.

Recently, the detection method based on machine learning and deep learning is the focus of attention in research [17–26]. This method trains machine learning or deep learning model by extracting the features to detect phishing emails. Du et al. proposed an anti-phishing technology to detect phishing emails by extracting mail features and calculating information entropy [27]. Peng et al. use natural language processing technology to detect phishing emails by extracting keyword information of message body [28]. In the detection of spear-phishing emails, it is very important to analyze the header characteristics and the message body of emails. Han et al. extracted the origin feature, text feature, attachment feature and recipient feature of email, and used KNN graph to realize the campaign attribution and early detection of spear-phishing emails [15]. Wang et al. raise a detection method for spear-phishing emails based on authentication considering the attacker's stylometric feature, gender feature and personality feature [16]. In addition, Grant et al. propose a new method which uses features derived from an analysis of fundamental characteristics of spear-phishing attacks and combines with a new non-parametric anomaly scoring technique for ranking alerts [14]. However, this method can only detect the spear-phishing emails, but cannot determine the organization, which is an important problem in the analysis of spear-phishing attack.

2.2 Few-Shot Learning Methods

With the development of artificial intelligence technology, few-shot learning has become the focus of current researchers. As we all know, machine learning and deep learning model need to complete the target task through a large number of data training, while few-shot learning aims to solve the model's dependence on large-scale data, and only using a small number of samples can get better results by prior knowledge. The methods of few-shot learning are mainly divided into three categories, including methods based on data, model and algorithm [29].

The data-based method is often called data augmentation through the operation of samples to get a richer and more sufficient sample set, which can support the training of the model. In the field of computer vision, data augmentation methods for images usually include image flipping, rotation, scaling, clipping, translation and adding noise [30, 31]. In the field of natural language processing, the easy data augmentation for text classification tasks mainly include synonym replace, random insert, random swap and random delete [32]. At the same time, the back translation method is also an important method, through the use of machine translation and back translation to complete data augmentation. In the field of speech recognition, methods such as adding noise, time shift, volume adjustment, speed adjustment, stretching signal and random same type extraction and splicing are often used [33]. In addition, the generation model can also be used to generate data similar to real data to complete data augmentation. Although there are many ways of data augmentation, however this method can only alleviate the problem of insufficient samples to some extent, and its ability is very limited.

The model-based methods include multi-task learning, metric learning and external memory-based methods. Multi-task learning method uses common information shared across tasks and specific information of each task to spontaneously learn multiple learning tasks. Embedding learning embeds all the samples in the training set into a low-dimensional separable space through a function F, then embeds the samples in the testing set into this low-dimensional space and calculates the similarity between the test samples and all the training samples to selects the tags of the samples with the highest similarity as the label of the test samples. Koch et al. explore a method for one-shot image recognition based on Siamese networks which employ a unique structure to naturally rank similarity between inputs [34]. Snell et al. propose typical networks for the problem of few-shot classification, which introduce the idea of mixed density estimation [35]. Sung et al. put forward that the difference between the relation network whose difference with other models is that the relation network autonomously learns an optimal measure function instead of a certain measure function given by humans, and realizes zero-shot learning to a certain extent [36]. Geng et al. propose a new induction network to learn generalized class wise representations, innovatively combining the dynamic routing algorithm with the typical meta-learning framework [37].

The algorithm-based method is to find the optimal parameters from the perspective of algorithm theory, corresponding to the optimal hypothesis space. Meta-learning is one of the most important ways, learning good initialization parameters, and then in the new task, as long as the initialization parameter is updated iteratively, it can adapt to the new task. The most classic model of this method is Model-Agnostic Meta-Learning, which can quickly adapt to new tasks with only a small amount of data by adjusting one or more gradients based on the initial parameters [38].

3 Simple Word-Embedding Model

The appropriate analysis of the message body will significantly improve the detection effect, but the current spear phishing detection methods did not make full use of the message body. In the process of semantic information utilization, there are some key problems that need to be solved.

First, due to the accurate target and low frequency of the spear-phishing attack, the number of email samples is quite small, which is not enough to support the deep learning model. However, when using machine learning to detect, we need to transform the message body information into a feature vector. How to choose an appropriate text vectorization method is the second key problem to be solved. Thirdly, since the different content of the message body, the length of the constructed message vector will be inconsistent. But the machine learning algorithm needs feature vectors with the same dimension. The existing approach usually addresses this problem by truncating or padding, which will result in missing or redundant information.

To solve the problems mentioned above, we proposed a simple word-embedding model. As shown in Fig. 1, the proposed model includes three important parts, that are vectorization of the message body, hierarchical-max pooling and hierarchical-min pooling.

- In vectorization of the message body, we segment the message body and use word2vec method to transform the message body into vector sequences. In word2vec, we use 128-dimensional word vectors, which fixes the final feature vectors of the message body to a length of 2 * 128 dimensions, and it is concatenated with the basic features of emails.
- In hierarchical-max pooling, we first use local max pooling to select the significant elements of the word vector, which can extract the feature with high representation ability. Then, the global average pooling is used to retain spatial information and

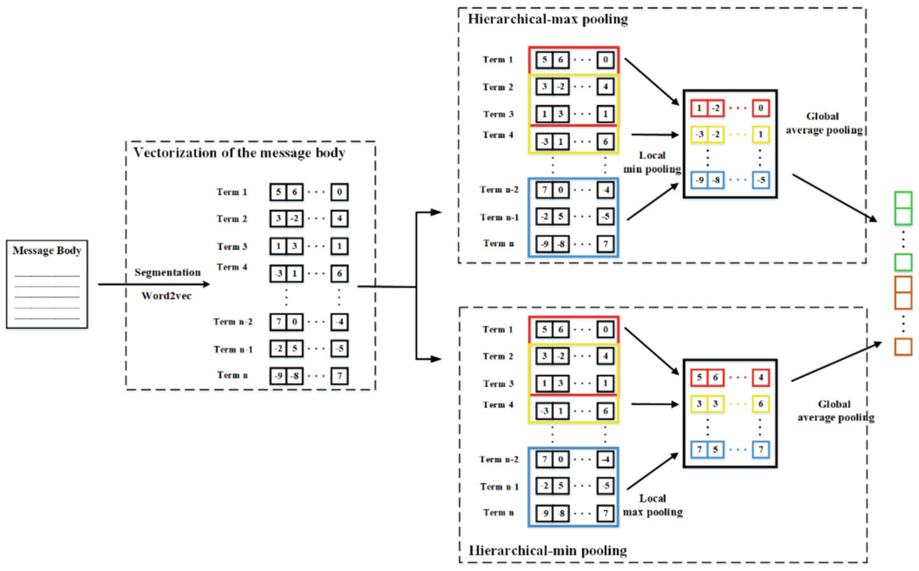


Fig. 1. The structure of our simple word-embedding model

word order. Meanwhile, the global average pooling can solve the problem of different lengths of the message body by calculating the average of all vector sequences.

- In hierarchical-min pooling, we use local min pooling to select the most easily ignored information in the vectors, which can reflect the attacker's characteristics in a subtle way. Similarly, the global average pooling is used to aggregate the intermediate features extracted from all local windows.

3.1 Hierarchical-Max Pooling

When dealing with the text classification or prediction task, the max pooling layer is usually used to extract the significant part of the word vector, that is, the keywords of the email. However, max pooling doesn't consider the spatial information and word order, which are very important in NLP. Therefore, we introduced the average pooling, and design a hierarchical pooling structure based on max pooling and average pooling.

Max Pooling: Since some keywords in a sentence or an article are important for prediction task or classification task, max pooling is used to extract these features from each dimension of word vectors, which is very similar to the max pooling in convolutional neural network. The formula of max pooling is shown as follows:

$$Z_j^{max} = \max_{i=1...L} v_{ij} \quad (1)$$

where v_{ij} is the j-th dimension of the first vector in the text, L is the length of the text, Z_j^{max} is the max value of all vectors in the j-th dimension, Z^{max} is the result of max pooling. According to the max pooling, we can ignore the words which are less relevant to the target task.

Average Pooling: Average pooling is the simplest strategy to transform a word vector sequence into a feature vector. This method calculates the average value of elements on a given word vector sequence by the following formula:

$$Z_j^{average} = \frac{1}{L}\sum\nolimits_{i=1}^{L} v_i \tag{2}$$

where v_i denotes the i-th vector in the message body, and L is the length of the text. The average pooling operation takes the average value of all word vectors in each dimension, and then obtains the feature vector which has the same dimension with the original word vector. Obviously, the average pooling strategy takes the elements of each sequence into account through a simple average operation, and gives each word vector the same weight. Unlike max pooling, the contribution of each word to the final feature vector is the same in average pooling.

Hierarchical-Max Pooling: Both average pooling and max pooling do not consider word order and spatial information, which are of great significance in dealing with text problems. Therefore, in this paper, we use the hierarchical pooling model to analyze the message body. For a given text vector $\mathbf{v} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_{\boldsymbol{L}}\}$, the local window denoted as $\boldsymbol{v}_{i:i+n-1}$, and there are n consecutive words in the local window, $\boldsymbol{v}_i, \boldsymbol{v}_{i+1}, \cdots, \boldsymbol{v}_{i+n-1}$. In the process of hierarchical-max pooling, we first complete the local max pooling by the $\boldsymbol{v}_{1:n}, \boldsymbol{v}_{1+s:n+s}, \boldsymbol{v}_{1+2s:n+2s}, \cdots, \boldsymbol{v}_{L-n+1:L}$ stride-sliding local window, where n denotes the size of the sliding window, and s denotes the stride. The process of sliding is similar to that of the convolution neural network. After the local max pooling, we aggregate the intermediate features extracted from all windows through the global average pooling operation, and finally form the text feature vector. On the one hand, the hierarchical-max pooling strategy retains the construction information from a single word window (i.e. n-gram) to the text sequence, that is, this method can obtain the local spatial information of the text sequence, instead of learning fixed-length representations for the n-grams that appears in the corpus, which is similar to the bag of n-gram method.

3.2 Hierarchical-Min Pooling

In the traditional method of extracting text features, max pooling is used to select the part with high representation ability of the vector, but this method cannot retain the most easily ignored information in the vector, which can reflect the attacker's characteristics in a subtle way. Therefore, we combined features extracted by hierarchical-min pooling with features extracted by hierarchical-max pooling, so as to retain the information of email as much as possible.

Min Pooling: In the field of computer vision, since the pixel of images is distributed in the range of 0 to 255, only max pooling is usually used which can extract the key features. Instead, if we use min pooling, we will often get all zero feature maps and can't extract the key texture features of the image. However, in the word embedding method, each dimension of word vector after word2vec is not limited in the range from 0 to 255. At this time, the min pooling will extract features different from the max pooling, which is easy to be ignored, but they can reflect the attacker's characteristics in a subtle way. The formula for the min pooling is as follows:

$$Z_j^{min} = \min_{i=1...L} v_{ij} \tag{3}$$

Hierarchical-Min Pooling: After local min pooling, we use global average pooling to process feature vector, which is similar to the process of local hierarchical-max pooling.

Concatenated Pooling: Due to the difference between the max pooling and the min pooling in the processing of text information, it is unreasonable to use a pooling method alone to extract information. Therefore, in order to retain the original semantics of the statement to the greatest extent, we use the concatenated pooling method. For a word vector, the results of hierarchical-max pooling and hierarchical-min pooling are combined together to form the concatenated pooling feature vector.

4 Spear-Phishing Detection Method Based on Few-Shot Learning

In our spear-phishing detection method, we first preprocess the email and extract key discrete features. For the message body, we first use word2vec to vectorize the message body, and then input the feature vector to simple word-embedding model to implement dimension reduction and further refine the semantic information, which can obtain a lightweight word vector that reflects the information of semantics and word order at the same time. Finally, we combine the discrete features with the feature vector of the message and input them to the machine learning model for detection (Fig. 2).

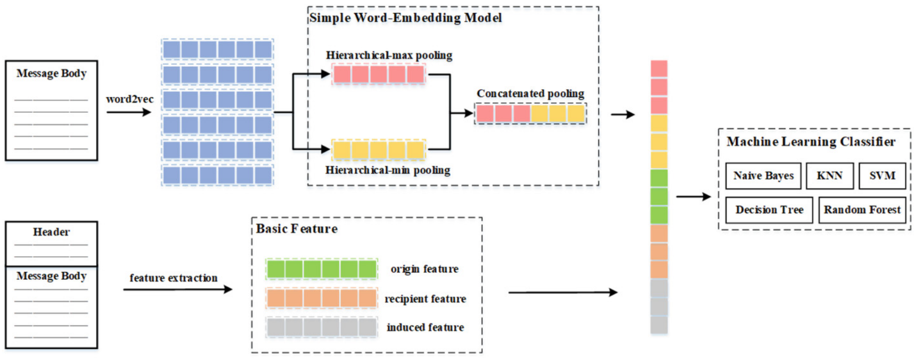


Fig. 2. The framework of spear-phishing detection method based on discrete feature and message body of the mails

4.1 Basic Feature Extraction

In this paper, we select some wildly used features commonly used in spear-phishing detection and the details of features are shown in Table1.

Table 1. The basic feature of email

Type of the feature	Abbreviation	Feature
Origin feature	ADO	attacker's domain
	SIP	source IP
	OCT	origin country
	SDT	sent date
Recipient feature	RDO	recipient's domain
	OIN	organization information
Induced feature	CLK	containing link or not
	CSL	containing short-links or not
	CPT	containing picture or not
	CKW	containing key words or not(such as "click", "download")

These features are divided into three categories: origin feature, recipient feature, and induced feature. The origin feature reflects the sender's attributes of the email, such as domain, IP and origin country which can uniquely identify the attackers and reflect the organizational characteristics. Besides, we also extract the day of week as an additional categorical feature. In the recipient feature, we extract the recipient's IP, domain name, and recipient's organization and industry information to reflect the recipient characteristics of the email, that is, the target attribute in spear-phishing. However, these header features of emails, especially the origin feature, can be easily forged by attackers to improve the concealment of identity. To this end, in our method, we also extract induced features. Because the contents of emails can reflect the attacker's potential attack habits, which are often carefully constructed by the attacker for the target, such as using phishing links or short links, embedding pictures and obvious induced word.

4.2 Visualization and Explanation of Message Body Feature Vector

In Sect. 3, we have described the simple word-embedding model in detail. In order to demonstrate the effectiveness of the model more intuitively, we visualize the organizational characteristics of the feature vector. We use the t-distributed stochastic neighbor embedding (t-SNE) algorithm to map the high-dimensional features of all samples to a two-dimensional space according to Ref. [41]. In Fig. 3, we demonstrate the feature distribution of 7 APT organizations (Krast, Layork, Elderwood, Nitro, Bisrala, Darkmoon, Samskams). We can find that samples of the same organization are clustered, and there are also obvious divisions between different organizations. Especially, due to the limitation of the number of samples and the two-dimensional space, the distribution of Bisrala's email sample is more scattered. However, the overall effect of visualization shows that the simple word-embedding model proposed in this paper can retain the characteristics of mail samples from different organizations.

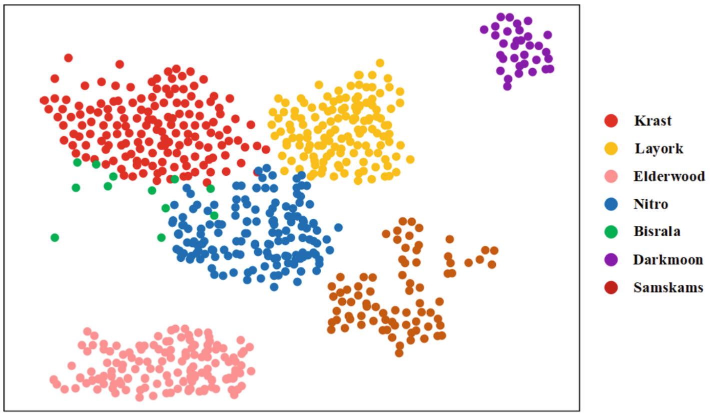


Fig. 3. The feature visualization of our simple word-embedding method representation

4.3 Feature Fusion

In feature fusion, we use the simple word-embedding model to process the message body to obtain a text feature vector, $V_{message}$. Subsequently, $V_{message}$ is concatenated with the discrete feature vector of the mail, $V_{existing}$, to obtain the final email representation vector, $V_{email} = (V_{message}, V_{existing})$, which will be input into the machine learning model.

4.4 Classifier

To choose the appropriate classifier, we try five different machine learning algorithms including KNN, Naive Bayes, Decision Tree, Random Forest, and Support Vector Machine. KNN algorithm classifies the samples by selecting the labels of data adjacent to the samples in the feature space. Naive Bayes algorithm is based on the assumption that the characteristics of the samples are independent of each other, and uses Bayes principle to calculate the probability of samples belonging to each category. Decision Tree selects the key feature of samples through entropy or Gini index, and summarizes a set of classification rules. Random Forest classifies samples by integrating the results of multiple Decision Tree classifiers. Support Vector Machine calculates the distance from the sample points to the hyperplane to finds the maximum margin hyperplane to realize the classification task.

5 Experiments and Result Discussion

5.1 Datasets

The dataset of phishing emails in this experiment contains 1342 mails from 7 organizations [15]. The origin organization of each email is determined manually. In order to ensure the balance of data samples, we will select a small number of balanced samples from each class as training data, which will improve the effectiveness of the model.

In few-shot learning, the number of training samples are very limited, and researchers usually use n-way k-shot to describe the setting of training data. Usually, n-way k-shot refers to randomly selecting n classes with k samples in each class. Such a process is called a task. In each task, the training set is called support set, and the test set is called query set. For example, 5-way 10-shot 5-query means to select 5 class in complete dataset, 10 samples from each class as the support set and 5 samples as the query set. Such a process is called an episode and one task includes many episodes. Following the standard experimental setup in few-shot learning, we carried out 7-way classification tasks for 1-shot, 5-shot and 10-shot respectively, and set the query set size to 15.

5.2 Experimental Setup and Evaluation Criteria

All experiments carried out in the Ubuntu 14.04. LTS environment, using python 3.5.4 and Keras 2.1.2 neural network library to build networks. We use Tensorflow 1.4.1 as a backend computing framework. CPU server is Inter(R) Xeon(R) CPU E5–2637 v4 @ 3.50 GHz, GPU is TITAN(X) (Pascall).

In the experiment, we choose accuracy, precision, recall Macro F1 and Micro F1 as the evaluation criteria. In each task, the evaluation criteria will be calculated separately, and the final evaluation criteria of the model is calculated by averaging results of all randomly generated episodes on the testing dataset. The formulas of the metrics are as follows:

$$\mathrm{Acc} = \frac{1}{n}\sum\nolimits_{i=1}^{n} \mathrm{Acc}_i = \frac{1}{n}\sum\nolimits_{i=1}^{n}\left(1 - \frac{errors_{num}}{sum}\right) \times 100\% \tag{4}$$

$$\mathrm{P} = \frac{1}{n}\sum\nolimits_{i=1}^{n} \mathrm{P}_i = \frac{1}{n}\sum\nolimits_{i=1}^{n} \frac{TP}{TP + FP} \times 100\% \tag{5}$$

$$\mathrm{R} = \frac{1}{n}\sum\nolimits_{i=1}^{n} \mathrm{R}_i = \frac{1}{n}\sum\nolimits_{i=1}^{n} \frac{TP}{TP + FN} \times 100\% \tag{6}$$

$$\mathrm{Macro\,F1} = \frac{1}{n}\sum\nolimits_{i=1}^{n} \mathrm{F1}_i = \frac{1}{n}\sum\nolimits_{i=1}^{n} \frac{2 \times P_i \times R_i}{P_i + R_i} \tag{7}$$

$$\mathrm{Micro\,F1} = \frac{2 \times P \times R}{P + R} \tag{8}$$

where TP, FP, FN represent True Positive, False Positive, False Negative respectively. A represents the accuracy, which refers to the proportion of correct results predicted by the model. P, R and F1 are calculated by dividing the multi-classification evaluation into the multi binary-classification evaluation, and calculating the average value. P represents the precision, which refers to the proportion of the samples identified as a certain class correctly. R represents the recall, which refers to the proportion of all samples of a certain class that are correctly identified as that class. Macro F1 is calculated by averaging the F1 score of each classification. Micro F1 is calculated by the sum of TP, FP and FN in each binary-classification respectively.

5.3 Experimental Results

1) The influence of different method of word2vec

In the processing of the message body, we transform the message body into vector sequences. Different schema of word2vec will affect the performance of our simple word embedding model. Therefore, we design some experiments to choose the optimal one.

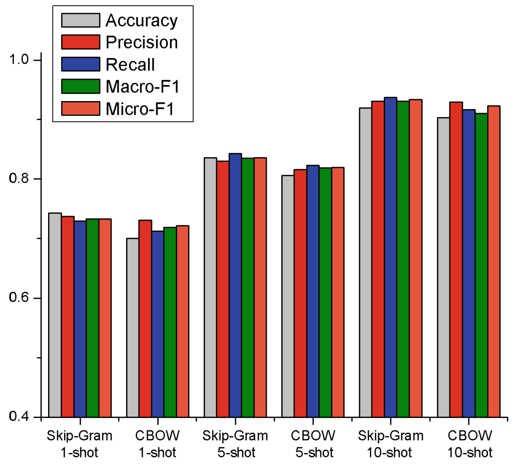


Fig. 4. The result of using different feature vectors

- CBOW: This method uses surrounding words to predict the center word. The input of the model is the word vector corresponding to the context of the target word, and the output is the word vector of the target word.
- Skip-gram: This method uses the center word to predict surrounding words. The input of the model is the word vector of the center word, and the output is the word vector corresponding to the context of the target word.

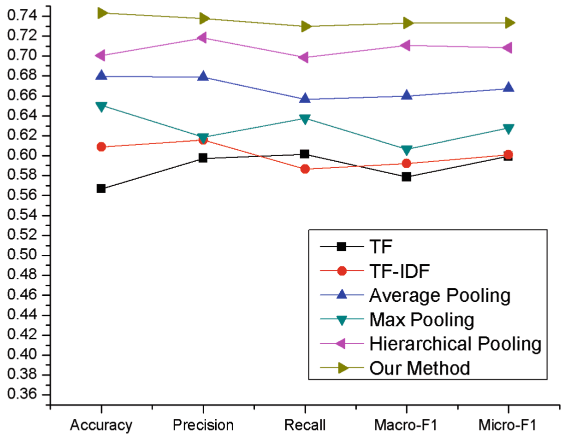
As shown in Fig. 4, the accuracy of Skip-gram method is higher than that of CBOW method in the message body analysis to be solved in this paper, so we use the feature vector based on Skip-gram.

2) Comparison with other simple word-embedding method

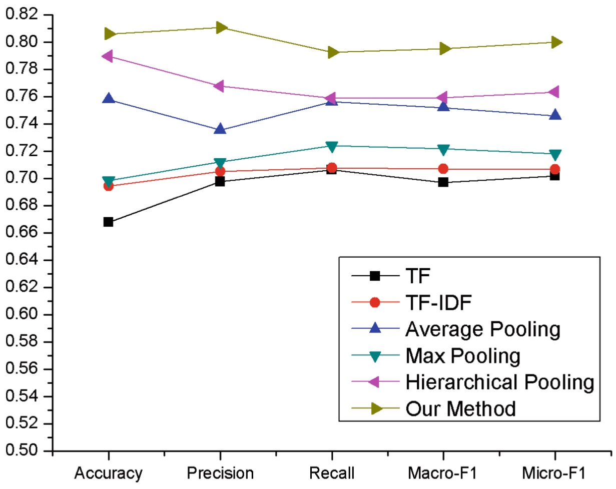
Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF) are two commonly used and simplest word-embedding methods, which use the characteristics of the bag-of-word model. We take these two methods as the baseline in this paper. At the same time, we also compare the performance of the proposed method with other commonly used simple word-embedding methods, including average pooling, max pooling and hierarchical pooling (max pooling and average pooling). The experimental results are shown in Fig. 5.

From the results in Fig. 5, the proposed simple word-embedding model is significantly better than TF, TF-IDF and commonly used simple word-embedding methods.

The hierarchical pooling is the closest to the results of the proposed method, because the structure of hierarchical pooling is the most similar to our proposed method, retaining the spatial information. However, the existing simple word-embedding methods fail to combine local features with global features, or lose the details that are easy to be ignored. Our proposed method not only integrates the max pooling and the min pooling, but also combines the local features and the global features through the global average pooling to retain the original semantics, so as to extract the characteristics of the attacker in the message body to the greatest extent.

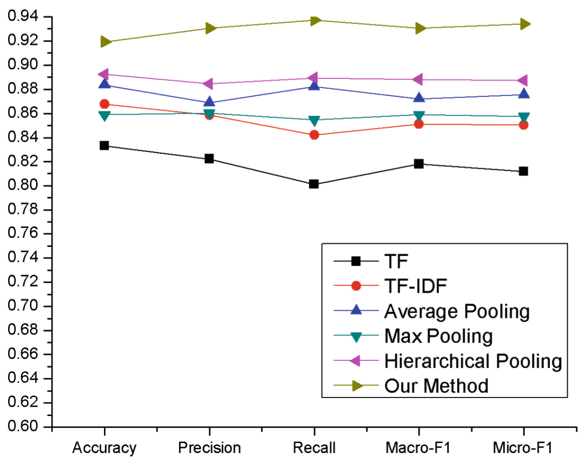


(a) 1-shot



(b) 5-shot

Fig. 5. The result of using different feature vectors



(c) 10-shot

Fig. 5. (*continued*)

3) The effect of sliding window parameters

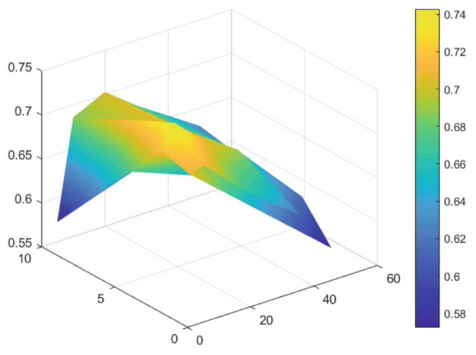
In the proposed simple word-embedding model, the size and stride of the sliding window are the super parameters of the model, which need to be given in advance. In order to explore the influence of these parameters in our method, we conduct experiments with window size selected from {5, 10, 20, 30, 40, 50} and stride selected from {1, 3, 5, 10}.

As shown in Fig. 6, we can find that when the window size is 10, the accuracy of the model reaches the peak. With the increase of the window size, the accuracy has a obvious decline, which is because the window size is too large to extract local features with fine-grained. In addition, the results show that the stride has a small influence on the experiments, but with the increase of stride, the accuracy of the model has a significant decline. Moreover, when the window size is smaller than the stride, the word vectors cannot be fully utilized, which leads to the poor effect of the model.

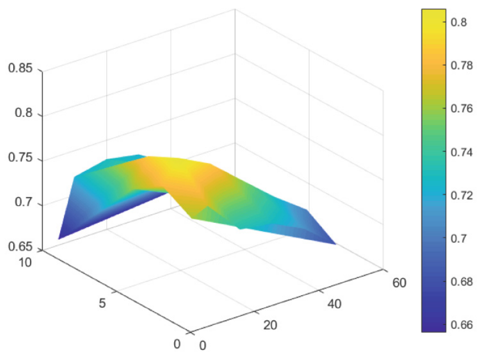
4) The influence of different classifier

To choose the appropriate classifier, we try five different machine learning algorithms including KNN, Naive Bayes, Decision Tree, Random Forest, and Support Vector Machine. In KNN, we use cross-validation to select the best K value. For Naive Bayes, we use Gaussian Naive Bayes model. In Decision Tree and Random Forest model, we use Gini coefficient index as criterion. For Support Vector Machine, the linear kernel function is selected according to the number of samples and the dimensions of feature vector. The experimental results of different classifiers are shown in Fig. 7.

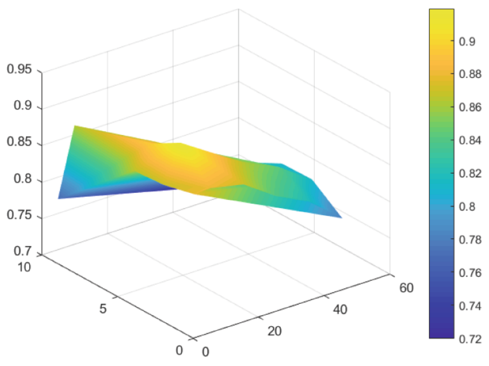
The results of KNN, Naive Bayes and Decision Tree are similar, and Random Forest is slightly better than KNN, Naive Bayes and Decision Tree. However, SVM has the best detection effect, because when the dimension of feature vector is large enough but the number of samples is small, the linear kernel can still achieve a good classification effect, which makes SVM model show its natural advantages even when facing the problem of small samples.



(a) 1-shot



(b) 5-shot



(c) 10-shot

Fig. 6. The effect of window size and stride

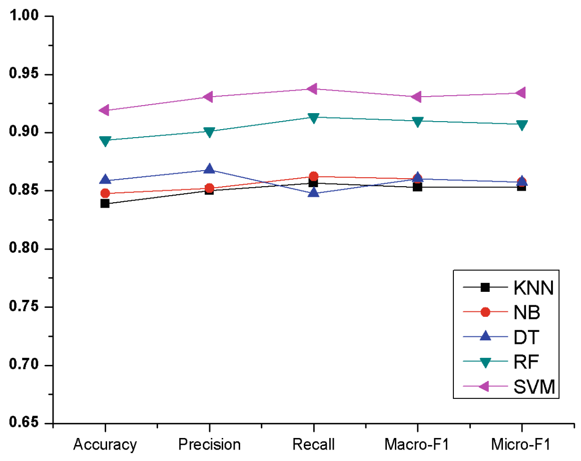


Fig. 7. The influence of different classifiers

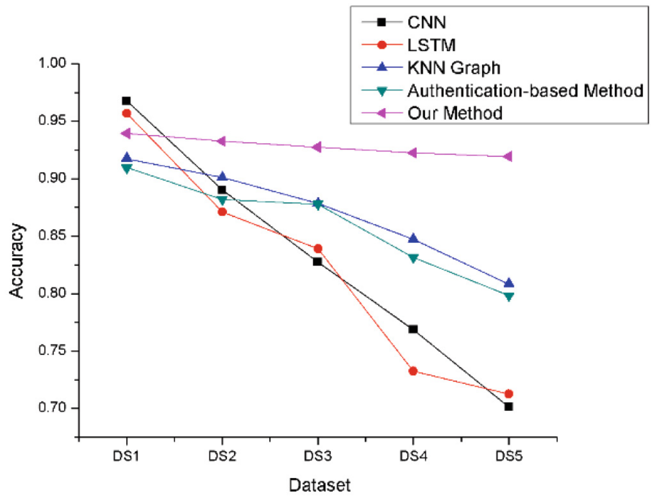
5) Comparison with other spear-phishing detection method

Table 2. The number of samples in dataset

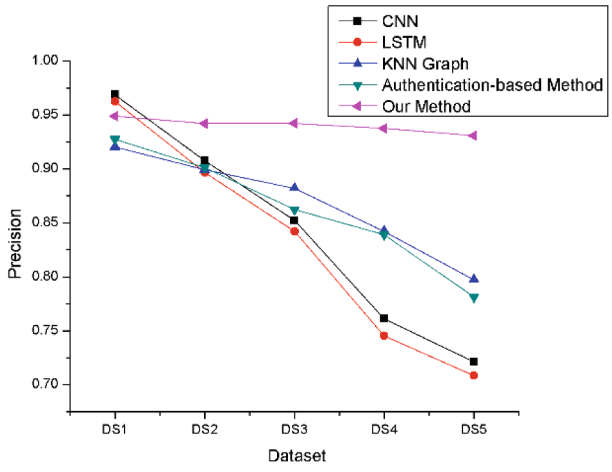
Organization\Datasets	DS_1	DS_2	DS_3	DS_4	DS_5
Krast	157	100	50	20	10
Layork	139	100	50	20	10
Elderwood	770	100	50	20	10
Nitro	153	100	50	20	10
Bisrala	12	12	12	12	10
Darkmoon	33	33	33	20	10
Samskams	78	78	50	20	10
Total	1342	523	295	132	70

In traditional phishing and spear-phishing detection methods, CNN, LSTM and other deep learning models are usually used to analyze the message body, and machine learning models are used to learn the extracted features of emails. In order to verify the effect of the method proposed in this paper on the spear-phishing detection with small samples, we designed enough experiment to compare with the commonly used detection methods. We set up 5 datasets with different quantities of samples, as shown in Table 2. In the experiment, we compared our method with the detection methods using only deep learning models (CNN and LSTM). In addition, we compare our method with the methods proposed by Han et al. and Wang et al. Han et al. extracted the origin feature, text feature, attachment feature and recipient feature of the email and used KNN graph. Wang et al. proposed an authentication-based spear-phishing detection method, paying more attention to the attacker characteristics reflected in the email, including stylometric feature, gender feature and personality feature.

It can be seen from the results in Fig. 8 that when the number of samples is sufficient, the detection effect of the CNN and LSTM models is slightly better than our proposed method, but as the number of samples decreases, the detection effect of the deep learning

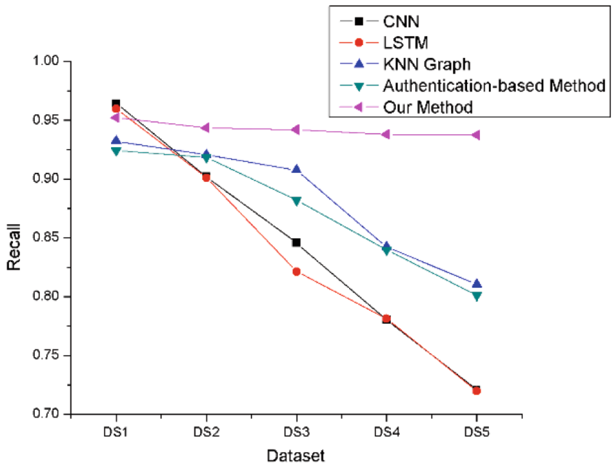


(a) accuracy

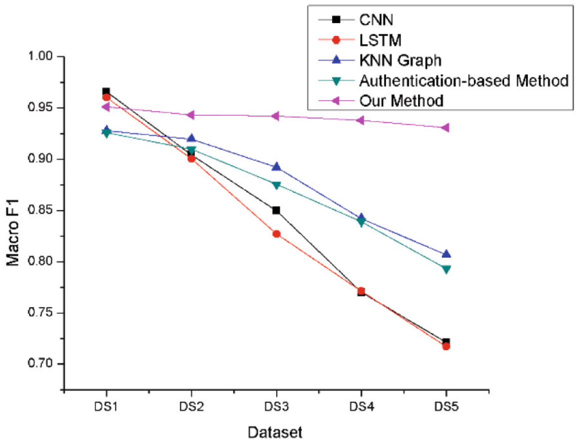


(b) precision

Fig. 8. The result of comparing with other spear-phishing detection method



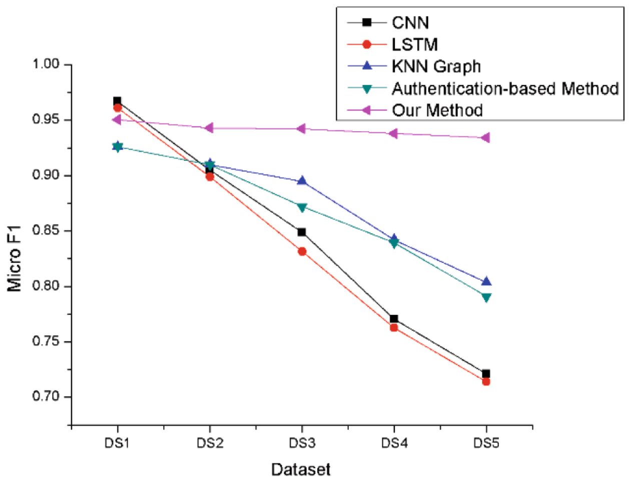
(c) recall



(d) Macro F1

Fig. 8. (*continued*)

model deteriorates significantly, due to the fact that the deep learning model contains too many parameters to be trained. However, the spear-phishing attack event has the characteristics of fewer range and less attack frequency, resulting in insufficient samples to support the training of deep learning models. The detection effect of our proposed method is significantly better than the other two methods (KNN graph and authentication-based method). Because our method not only extracts wildly used features, but also uses the entire message body of emails to the detection, instead of only the statistical



(e) Micro F1

Fig. 8. (*continued*)

characteristics of the message body. In that way, the original semantic information of the message body can be retained to the greatest extent.

6 Conclusion

In this paper, we analyzed the existing spear-phishing attack detection methods. We found that, the detection method based on the artificial definition of features may cause incomplete feature extraction due to the lack of prior knowledge while the detection method based on deep learning has high requirements on training samples. In order to slove this problem, we proposed a spear-phishing detection method based on few-shot learning, which can work well with limited labeling samples. On the one hand, we extracted the wildly used features such as attacker IP, target domain, induction keywords and so on. On the other hand, the simple word embedding-model is used to output the word vector of message body information with small samples. Then, we combine these two kinds of characteristics together and input them into the machine learning model for classification, including KNN, NB, DT, RF, SVM. Moreover, we also determined the optimal values of window size and stride of the sliding window in simple word-embedding model through sufficient experiments. The experimental results show that the spear-phishing detection method based on few-shot learning proposed in this paper has a better effect than the existing spear-phishing detection methods. Especially, our method can achieve better performance with small samples.

References

1. The MITRE Corporation: Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK), Tactics, initial access. https://attack.mitre.org/tactics/TA0001/
2. FreeBuf: Analysis on the attack samples of vulnerability exploitation of (2017). https://www.freebuf.com/articles/web/155747.html
3. FreeBuf: Attack event report of APT organization SideWinder (2019). https://www.freebuf.com/articles/paper/213799.html
4. Fireye: Best defense against spear-phishing attacks (2018). https://www.fireeye.com/current-threats/best-defense-againstspearphishing-attacks.html
5. Jansson, K., von Solms, R.: Phishing for phishing awareness. Behav. Inf. Technol. **32**, 584–593 (2013)
6. Nikolaos, T., Nikos, V., Alexios, M.: Browser blacklists: the utopia of phishing protection. E-Bus. Telecommun. **554**, 278–293 (2014)
7. Wang, Y., Agrawal, R., Choi, B.: Light weight anti-phishing with user whitelisting in a web browser. In: 2008 IEEE Region 5 Conference, pp. 39–42 (2008)
8. Jain, A., Gupta, B.: A novel approach to protect against phishing attacks at client side using auto-updated white-list. EURASIP J. Inf. Secur. **1**, 2016 (2016)
9. Marchal, S., François, J., State, R.: Proactive discovery of phishing related domain names. In: Balzarotti, D., Stolfo, S.J., Cova, M. (eds.) RAID 2012. LNCS, vol. 7462, pp. 190–209. Springer, London (2012). https://doi.org/10.1007/978-3-642-33338-5_10
10. Cao, Y., Han, W., Le, Y.: Anti-phishing based on automated individual white-list. In: Proceedings of the 4th ACM Workshop on Digital Identity Management, pp. 278–293 (2008)
11. Nissim, N., Cohen, A., Glezer, C., Elovici, Y.: Detection of malicious PDF files and directions for enhancements: a state-of-the art survey. Comput. Secur. **48**, 246–266 (2015)
12. Han, X., Kheir, N., Balzarotti, D.: PhishEye: live monitoring of sandboxed phishing kits. In: ACM SIGSAC Conference on Computer & Communications Security, pp. 1402–1413 (2017)
13. FreeBuf: APT-C-12, Nuclear Crisis Action Revealing (2018). https://www.freebuf.com/column/176675.html
14. Ho, G., Sharma, A., Javed, M., Paxson, V., Wagner, D.: Detecting credential spearphishing in enterprise settings. In: 26th USENIX Security Symposium (2017)
15. Han, Y., Shen, Y.: Accurate spear phishing campaign attribution and early detection. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp. 2079–2086 (2016)
16. Wang, X., Zhang, C., Zheng, K., Tang, H., Tao, Y.: Detecting spear-phishing emails based on authentication. In: IEEE International Conference on Computer and Communication Systems, pp. 450–456 (2019)
17. Tewari, P., Singh, R.: Machine learning based phishing website detection system. Int. J. Eng. Res. Technol. **4**, 172–174 (2015)
18. Jain, A., Gupta, B.: A machine learning based approach for phishing detection using hyperlinks information. J. Ambient Intell. Humaniz. Comput. 2015–2028 (2018)
19. Jain, A., Gupta, B.: Comparative analysis of features based machine learning approaches for phishing detection. In: International Conference on Computing for Sustainable Global Development, pp. 2125–2130 (2016)
20. Abdelhamid, N., Thabtah, F., Abdel-jaber, H.: Phishing detection: a recent intelligent machine learning comparison based on models content and features. In: IEEE International Conference on Intelligence & Security Informatics, pp. 72–77 (2017)
21. Chiew, K., Tan, C., Wong, K., Yong, K., Tiong, W.: A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. Inf. Sci. **484**, 153–166 (2019)

22. Sahingoz, O., Buber, E., Demir, O., Diri, B.: Machine learning based phishing detection from URLs. Expert Syst. Appl. **117**, 345–357 (2019)
23. Yadollahi, M., Shoeleh, F., Serkani, E., Madani, A., Gharaee, H.: An adaptive machine learning based approach for phishing detection using hybrid features. In: International Conference on Web Research, pp. 281–286 (2019)
24. Zhu, E., Chen, Y., Ye, C., Li, X., Liu, F.: OFS-NN: an effective phishing websites detection model based on optimal feature selection and neural network. IEEE Access **7**, 73271–73284 (2019)
25. Phoka, T., Suthaphan, P.: Image based phishing detection using transfer learning. In: Annual International Conference on Knowledge and Smart Technology, pp. 232–237 (2019)
26. Smadi, S., Aslam, N., Zhang, L.: Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. Decis. Support Syst. **107**, 88–102 (2018)
27. Du, Y., Xue, F.: Research of the anti-phishing technology based on e-mail extraction and analysis. In: International Conference on Information Science & Cloud Computing Companion, pp. 60–65 (2014)
28. Peng, T., Harris, I., Sawa, Y.: Detecting phishing attacks using natural language processing and machine learning. In: IEEE International Conference on Semantic Computing, pp. 300–301 (2018)
29. Wang, Y., Yao, Q., Kwok, J., Ni, L.: Generalizing from a few examples: a survey on few-shot learning. ACM Comput. Surv. **1**(1) (2020)
30. Huynh-The, T., Hua, C., Kim, D.: Encoding pose features to images with data augmentation for 3-D action recognition. IEEE Trans. Industr. Inf. **16**(5), 3100–3111 (2020)
31. Liu, Z., et al.: Automatic diagnosis of fungal keratitis using data augmentation and image fusion with deep convolutional neural network. Comput. Methods Program. Biomed. **187** (2020)
32. Wei, J., Zou, K.: EDA: easy data augmentation techniques for boosting performance on text classification tasks. In: Conference on Empirical Methods in Natural Language Processing & International Joint Conference on Natural Language Processing (2019)
33. Park, D., et al.: SpecAugment: a simple data augmentation method for automatic speech recognition. In: Conference of the International Speech Communication Association (2019)
34. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: International Conference on Machine Learning (2015)
35. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Annual Conference on Neural Information Processing Systems, vol. 30 (2017)
36. Sung, F., Yang, Y., Zhang, L., Xiao, T., Torr, P., Hospedales, T.: Learning to compare: relation network for few-shot learning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1199–1208 (2018)
37. Geng, R., Li, B., Li, Y., Ye, Y., Jian, P., Sun, J.: Few-shot text classification with induction network. In: Conference on Empirical Methods in Natural Language Processing (2019)
38. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning (2017)
39. Shen, D., et al.: Baseline needs more love: on simple word-embedding-based models and associated pooling mechanisms. In: Annual Meeting of the Association-for-Computational-Linguistics, pp. 440–450 (2018)
40. Pan, C., Huang, J., Gong, J., Yuan, X.: Few-shot transfer learning for text classification with lightweight word embedding based models. IEEE Access **7**, 53296–53304 (2019)
41. Maaten, V., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2625 (2008)