Telecommunication Systems (2022) 79:33–45
https://doi.org/10.1007/s11235-021-00850-6

Application of word embedding and machine learning in detecting phishing websites

Routhu Srinivasa Rao[1] · **Amey Umarekar**[2] · **Alwyn Roshan Pais**[2]

Accepted: 19 October 2021 / Published online: 19 November 2021

Abstract
Phishing is an attack whose aim is to gain personal information such as passwords, credit card details etc. from online users by deceiving them through fake websites, emails or any legitimate internet service. There exists many techniques to detect phishing sites such as third-party based techniques, source code based methods and URL based methods but still users are getting trapped into revealing their sensitive information. In this paper, we propose a new technique which detects phishing sites with word embeddings using plain text and domain specific text extracted from the source code. We applied various word embedding for the evaluation of our model using ensemble and multimodal approaches. From the experimental evaluation, we observed that multimodal with domain specific text achieved a significant accuracy of 99.34% with TPR of 99.59%, FPR of 0.93%, and MCC of 98.68%

Keywords URL · Phishing · Anti-phishing · TF-IDF · Hostname · Random forest

1 Introduction

Phishing attacks are on the rise periodically resulting in loss of billions of dollars. The possibility of attacks generating fear of online service usage is increasing year after year. From the statistics of APWG (2017–2018),[1] 785,920 number of attacks were observed in 2018 compared to 612,075 in 2017. Use of HTTPS in phishing websites is also increased from 46 to 58% in 2019. As per the proofpoint report attacks increased from 76% in 2017 to 83% in 2018.[2] The organizations most affected by phishing are banking sectors with suffered 25.78% attacks followed by internet portals 19.82% and payment systems with 17.33% attacks.[3]

The attackers design phishing sites that look visually similar to target legitimate sites. The visual resemblance may include logos, favicons, textual content etc. The brand of the website may be present in textual content at headers, copyright, title etc. Hence, we explored textual content for identifying the defense mechanism for the phishing websites in this paper.

Word Embeddings are vector representations of the words or phrases. These vectors helps to identify a word provided in the given corpus. When using these embeddings it makes Natural Language Processing tasks easier such as syntactic analysis. This numerical representation helps to understand the semantic and contextual information of the words present in the corpus and is useful in machine learning algorithm to work with better performance. Word Embeddings can be helpful in various tasks where our systems need to understand language and context which is being used. To the best of our knowledge, there exists no works that use word embeddings ensemble or multimodal in detecting phishing site in the literature. Hence, we have attempted to use various embedding algorithms for exploring the textual content based

[1] https://www.antiphishing.org/resources/apwg-reports/.
[2] https://www.proofpoint.com/us/security-awareness/post/2019-state-phish-attack-rates-rise-account-compromise-soars.

[3] https://securelist.com/spam-and-phishing-in-q1-2019/90795/.

✉ Routhu Srinivasa Rao
routh.srinivas@gmail.com

Amey Umarekar
amey08umarekar@gmail.com

Alwyn Roshan Pais
alwyn.pais@gmail.com

[1] Department of Computer Science and Engineering, GMR Institute of Technology, Rajam, Andhra Pradesh 532127, India

[2] Information Security Research Lab, Department of Computer Science and Engineering, National Institute of Technology, Surathkal, Karnataka 575025, India

defense mechanism. There exists various phishing detection techniques including list based techniques, heuristic based methods where features are extracted from the source code of the website and from the third party services such as page rank and search engine results. Limitation of these methods is that they might misclassify in case of newly registered legitimate website. Phishing websites which are hosted on compromised servers may get bypassed by the above techniques.

Following are the contributions of our proposed work:

- We propose an ensemble model from the dictionaries of various embedding algorithms for the classification.
- We analysed multimodal of dictionaries of various word embedding algorithms with several machine learning algorithms.
- The proposed technique is deployed as a chrome extension to provide real time protection at the users system

Following are the advantages of the proposed method:

- *Independent from third party services* the technique does not rely on third party features like page ranking, age and search engine indexing.
- *Semantic extraction* word embeddings in the proposed technique capture textual semantic features which helps in identifying phishing websites with similar textual contents.
- *Client-side adaptability* the proposed technique extracts features from textual content unlike visual content hence, it is adaptable at client-side.

The remaining sections of the paper consist of literature survey of existing anti-phishing techniques in Sect. 2. Proposed work is discussed in Sect. 3. Experimentation and results are given in Sect. 4. We conclude the paper in Sect. 5.

2 Literature survey

There are various machine learning and heuristic based phishing detection techniques with features extracted from the websites URL, source code and from the third party based services. These techniques are elaborated below:

1. *List based techniques* these techniques involve use of whitelist or blacklist for detection of phishing websites. In whitelist based techniques, a list of permitted URLs are maintained which are allowed to be accessed and loaded by web browsers. It is decided based on this whitelist if particular URL is legitimate or phish. Following works of Wang et al. [32], Belabed et al. [5], Cao et al. [6], Azeez et al. [3] involves use of white list for classification of URLs.
   Browsers like Google Chrome and Firefox maintains a list of the URLs which are suspicious to be loaded for the user by the browser. Google safe browsing API[4] maintains a list of blacklisted URLs which are used to compare with the URL to which the user names the https request. Similarly, Firefox web browser[5] also maintains a similar black list which is to check the suspicious website URLs. PhishNet Prakash et al. [17] which uses different heuristics such as IP Address, replacing TLD's, brand name equivalence, query string substitution,finding similar directory to calculate a similarity score which helps in predicting status of given url. Rao and Pais [18] proposed a technique which maintain a blacklist of features instead of urls whose signature is generated using SimHash. These are compared with that of the suspicious url to classify as legitimate or phishing.
2. *URL based techniques*: in URL based techniques, classification is done based on the features extracted from the url of the suspicious website. These features include count based, presence of special characters and phish hinted words. Existing works such as Gowtham and Krishnamurthi [11], He et al. [12], Sahingoz et al. [24], Zhang et al. [40], Mohammad et al. [15], Shirazi et al. [25], Zamir et al. [38], Rao et al. [22], Mourtaji et al. [16] and Afzal et al. [1] extracts count based heuristics such as length of URL, number of dots, @, hyphens in the URL as well as the redirection links present in the website source.
   Some of the works such as Gastellier-Prevost et al. [10], Li et al. [14], Mohammad et al. [15], Shirazi et al. [25], Rao and Pais [20], Tharani and Arachchilage [29] consists of heuristics such as presence of black listed words, IP address, special characters (*, /, ?, −), protocols, ports used and the brand name in the URL. There exists techniques such as Xiao et al. [35], Rao et al. [21], Wei et al. [33], Xiao et al. [34] that use deep learning algorithms for the extraction of features from the given URL to further classify the phishing URLs.
   Some techniques such as Xu et al. [36], Zhang et al. [39] and Cheng et al. [7] also make use of third party features such as Alexa page rankings, WHOIS, search engine results and age of domain.
3. *Content based techniques*: in content based techniques, features are extracted from the source code of the website itself. Recent works like Rao and Pais [19], Shirazi et al. [25], Tharani and Arachchilage [29], Wang et al.

[4] https://developers.google.com/safe-browsing.

[5] https://support.mozilla.org/en-US/kb/how-does-phishing-and-malware-protection-work.

[31] and Rao et al. [23] used features extracted from content of the website including hyperlinks, textual content (headers, titles, copyright) and image based features.

As our work fall under the source code based technique, we discuss some of the work in source code based phishing detection in detail. These techniques use hyper link based, text based features for classification of websites to be phish or legitimate.

- Zhang et al. [42] proposed CANTINA in which textual content is extracted from the source code of the website and text with high TF-IDF scores is given input to search engine. This input is a signature of ve highest TF-IDF score terms. If the domain of suspicious URL is in top N results of search engine results page, it is considered as legitimate else phishing. CANTINA correctly classied 95% of phishing sites.

- Fang et al. [9] proposed a technique which uses word representation for the email phishing detection technique. It uses character level for email body and word level representations for email header and body. Intuition behind character level is that it helps to capture uniqueness of content in email body as each individual has their own habits. All the representations are generated using Word2Vec model which are then fed to a recurrent convolutional neural network. THEMIS model reached accuracy of 99.848% for phishing email detection.

- Rao and Pais [19] extracts hyper link based features from website source code as well as third party based features. These features are then fed to various machine learning algorithms for the classification. With oblique random forest, the model achieved an accuracy of 99.55%.

- Zhang et al. [41] proposed a technique which uses textual semantic features for website phishing detection. They extracted features such as sensitive words in the source text, licenced copyright information from the website, age of domain and hyper-links present in the website source code. Textual semantic features are captured using word embedding which is generated using Word2Vec is used with CBOW model. For classification AdaBoost, Bagging, Random Forest and SMO were used. AdaBoost with word embedding and statistical features performed better with F-measure of 0.998%. For more details on the phishing techniques in the existing literature, it can be found in Basit et al. [4], Vijayalakshmi et al. [30], Aleroud and Zhou [2], Chiew et al. [8], da Silva et al. [27], Jain and Gupta [13]

3 Proposed work

The architecture of the proposed work is given in Fig. 1. The proposed work is carried with two types of input. Firstly, PlainText (PT) from the source code is extracted and fed to the word embedding algorithms to generate the dictionaries. The feature vector is created with the generated dictionaries and the vector is further fed to machine learning algorithms for the classification. Secondly, domain specific text (DST) is extracted from the source code which are located at various locations such as copyright, title, headers etc. Similar to PT, DST is also fed to various embedding algorithms to generate dictionaries which are further used for generating feature vectors. The generated vectors are given as input to various classifiers to identify the optimal classifier for our dataset.

Following are the details of the components of our architecture:

- *Parsing and lemmatization*: source code of each website is present in text file. Content is loaded and then parsed using the library Beautifulsoup. With the help of it, extraction of the required text from the html source code becomes convenient. While parsing, html tags and css code present in the source code is ignored. Domain specific text such as title, headings, copyright information, description and paragraphs is extracted. With this process, we get the text present in the complete html source as well as we obtain domain specific text.
  After parsing html source and getting text from each text file of different websites, we tokenize the text using the tokenizer provided by nltk library. After tokenizing lemmatization is performed. Lemmatization tries to remove the inflectional endings from the word and provide the dictionary form of the word. This is done using the vocabulary and morphological analysis i.e studying the structure and formation of the word. So after performing lemmatization, we would get results for car, cars, car's as car, geese to goose, running to run, has to have and are, is, being as be etc.. Lemmatization helps as it takes into consideration the context of the word which is being used.

- *Word embeddings*: word embeddings gives us the numerical representations of the words in the corpus in the vector form. So, it is just vector of numerical values. This helps in reducing the time and space complexity while performing machine learning on the data. Following algorithms are used in our proposed work:

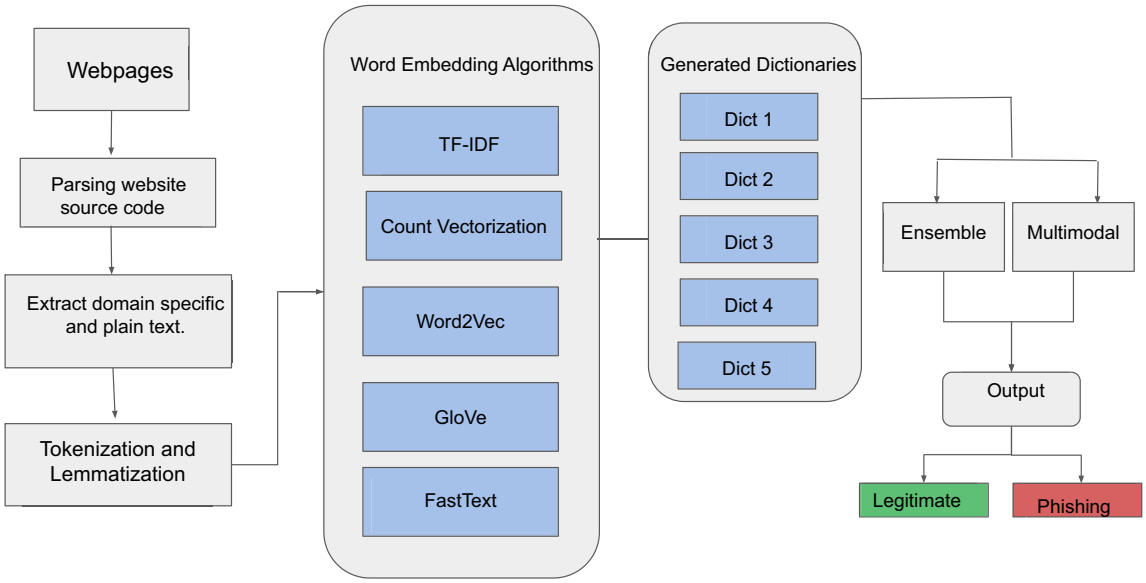


Fig. 1 Architecture of proposed work

– *Frequency based*
  - *TF-IDF*: Term frequency and inverse document frequency vectorization identifies the unique and most frequent words that describes the document. It assigns weights to each word based on their frequency in the document and its existence in number of documents in the corpus.
    TF The number of times a word has appeared in a document. Further we can divide it with total number of words in document. Therefore,

$$TF(t,d) = \frac{t}{T}$$

    where t = number of times word occur in a document, T = number of words in the document.
    IDF It measures uniqueness of the word across the corpus.

$$IDF = log(\frac{N}{n})$$

    where n number of document word is present in, N total number of documents

$$TFIDF = TF * IDF$$

  - *Count vectorization*: In count vectorization a matrix of size $d \times n$ will be created where d is size of the corpus i.e the number of documents and n is the number of unique tokens in the documents. This matrix holds the count of each words appearing in a document. The similarity between each of the vector generated is calculated using the cosine similarity i.e the angle between the two vectors.

– *Prediction based*
  - *Word2Vec*: Word2Vec is a predictive model which tries to learn embeddings from the given text. It has 3 layer architecture with a small hidden layer which does the task of generating embeddings from given text. The size of input and output given is generally same. Word2Vec has two algorithms to work with. They are:
    CBOW: continuous Bag of Words tries to predict the word with the help of the context. This context can be a single word or a group of words. In this network, we have two types of weight matrices, one is between input layer and hidden layer (say W) and othe other is between hidden layer and output layer (say W'). At each time step t, we consider n words before and n words behind as input. These input words are multiplied with W and thereby averaged to result single vector v'. The hidden output-weights i.e. W' is multiplied with v' to get the output. The predicted output and actual output is compared to calculate the error. The error is reduced through backpropagation by adjusting the W and W'. Finally, the weight between the hidden layer and output layer is considered as word representation. Note that CBOW also doesnot consider word order but the vector sum is meaningful enough to deduce target word.

Skip gram model: skip gram model tries to predict the context of the given word. The model follows a similar method as the CBOW model but it has architecture opposite to that of the CBOW model. It starts with a single word embedding and try to predict the surrounding words. The input words are one hot encoded based on the dictionary of unique words where position of the word is set as 1 and remaining locations as 0. Note that, while training the model, the input is a one hot vector and output is also an one hot vector representing both input word and output word. But, while evaluating model, the output is a probability distribution vector and not a one hot vector. The network consists of hidden layer with no activation function but contains softmax at output layer. The softmax converts the each output neuron into a value between 0 and 1 where sum of all the values of output neurons results to 1.

- *GloVe*: Global Vectors are generated using the co-occurrence matrix statistic from a corpus which also incorporates global statistics from the corpus with local statistics. It combines advantages of both global matrix factorization and local context window methods. It leverages the statistical information by training only on the nonzero elements in the word to word co-occurrence matrix rather than on the sparse matrix or individual context windows in large corpus. The element $X_{i,j}$ of co-occurrence matrix X represent the number of times word j appears in context of word i. $X_i = \sum_k X_{ik}$ be the number of times any word appears in the context of word i. $P_{i,j} = X_{i,j}/X_i$ gives the probability of the word j occurs in context of word i. These probabilities can provide some potential to encode some form of underlying meaning of the context meaning. GloVe learns and optimizes word vector of a word such that dot product of two word vectors equals to the logarithm of their probability of them occurring together.
- FastText: FastText is a library developed by Facebook Research. It uses two algorithms which are Continuous bag of words and Skip Gram model similar to Word2Vec. Instead of learning from individual work, this technique represents words as n gram of characters due to which it generates better embeddings for rare words. Also, this approach allows to generate vector representations of previously unseen words in the text.

– In this paper, we work on plain text and domain specific text. Plain text has been extracted from the source code of the websites using the library BeautifulSoup. This plain text is then stored in a csv file. Domain specific text is extracted from the source code using same library but from some specific locations in source code such as title, headers mainly h1 and h2, copyright and description. Also, the URL maps are used from which it's components have been extracted and are concatenated to domain specific text. Assumption behind this is that there could be presence of important information such as brand names etc. Plain text and domain specific text are later fed to generate word embedding.

– *Dictionary Generation*: After getting the word embeddings our saved model has dictionary of words with their vector representations using different word embedding algorithms mentioned above. This helps when our testing data comes and we have to generate their vectors representations.

– *Ensemble*: In ensemble model, we generate word embeddings of our corpus using the algorithms mentioned and these are fed to the best performing machine learning algorithm. After the results obtained max voting is applied and evaluation is done on the defined metrics.

– *Multimodal*: In multimodal model, word embeddings are generated for the data using the word embeddings algorithms. Now these word embeddings are concatenated together to generate feature vector which is further fed to the machine learning algorithm for classification of websites into phishing or legitimate.

– *Machine learning algorithms*: To evaluate our ensemble and multimodal model of our word embeddings created various machine learning algorithms are used. The machine learning algorithms used in our model are random forest, SVM, logistic regression, decision tree and XGBoost. The features from various embedding algorithms are integrated to generate feature vector which is further used to train the model.

4 Experimentation and result

The input to our model is the suspicious URL where plaintext and domain specific text is extracted from the source code of the URL and outputs the status of the website as either legitimate or phishing. From the extracted text we obtain the vector representations of the words using the word embeddings which are then fed to the machine learning algorithms.

4.1 Dataset

Dataset consists of text file consisting of HTML source code of the websites. This is a labeled dataset with 5438 instances of phishing websites and 5076 instances of legitimate websites with their source codes as text files and their urls. From these text files, source code of the websites are parsed and word embedding algorithms are applied on the text. While parsing, we obtain both plain text as well as domain specific text from the text files for our further experiments.The dataset and source code of our work is located at https://tinyurl.com/phishdata. We have split the dataset into training and testing sets in which 80% of data contributes to training and remaining 20% of data is used for testing. Based on the experimental analysis, the tuned parameters that are used for the final model are as follows: For TF-IDF and word2vec, maximum features is set as $150 * 150$; For word2vec, min_count =5, workers = 10, iterations = 50, vector size = 150, window = 10; For Glove, $no\_components$=150, $learning\_rate$=0.05, epochs=30, window = 10 ; For FastText, $vector\_size$ = 100, window = 10, min_count = 5, workers = 4. For the machine learning algorithms, the tuned parameters are as follows: RandomForest ($n\_estimators$ = 100), Decision Tree (criterion = 'gini', splitter='best'), SVM(gamma = 'auto', C = 1.0, kernel = 'rbf', degree = 3), XGBClassifier: batches=5, Logistic Regression(solver='lbfgs', max_iter = 100)

4.2 Evaluation

For evaluation of our model, we use following metrics such as true positive rate (TPR) or Sensitivity, true negative rate (TNR) or specificity, Matthew MCC, F Score, Precision. We consider positive as phishing and negative as legitimate. Correctly predicted phishing websites is termed as true positive (TP), correctly predicting legitimate websites as true negative (TN), incorrectly predicting legitimate website as false positive (FP) and incorrectly predicting phishing websites as false negative (FN).

1. True positive rate (TPR): % of phishing websites predicted correctly out of total number of phishing websites. TPR is also known as Recall or Sensitivity.

$$TPR = \frac{TP}{TP + FP} \times 100 \tag{1}$$

2. True negative rate (TNR): % of correctly predicted legitimate websites out of total number of legitimate websites. TNR is also known as specificity.

$$TNR = \frac{TN}{TN + FN} \times 100 \tag{2}$$

3. Precision: % of correctly predicted phishing websites out of total number of predicted phishing websites.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

4. Accuracy: % of phishing and legitimate websites predicted correctly to total number of websites.

$$Accuracy = \frac{TP + TN}{F + N} \tag{4}$$

5. F-Score: F Score is the measure of test's accuracy and is the weighted harmonic mean of test's precision and recall.

$$F - Score = 2 * \frac{Precision.Recall}{Precision + Recall} \tag{5}$$

6. MCC: Mathews Correlation Coefficient takes into account all the values of the confusion matrix to evaluate quality of binary classifier.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{6}$$

To evaluate our proposed model, we conducted several experiments using word embedding and various machine learning algorithms. While generating word embeddings with different algorithms, the embedding length of 150 is used throughout the experimentation. Hence, in the following experiments, we generate word embedding with specific length of 150 for both the frequency based as well as prediction based embedding.

4.3 Experiment 1: evaluation of model with plain text

In this experiment, plain text is extracted from the source code using BeautifulSoup library. The plain text is used to generate word embeddings using the above mentioned word embedding algorithms. The experimental results are mentioned in Table 1. From the results, it is observed that Random Forest outperformed other classifiers in frequency based word embedding algorithms where as logistic regression outperformed other classifiers for prediction based embedding algorithms.

4.4 Experiment 2: evaluation of model with domain specific text

In this experiment, we extract the domain specific text from specific sections like copyright, description of website, title

Table 1 Results of various word embedding algorithms with plain text

Word embedding	Classifiers	Precision	TPR	TNR	MCC	F Score	Accuracy
TF-IDF	RF	97.9	98.27	97.92	96.2	98.08	98.09
	SVM	90.33	95.68	90.78	86.34	92.93	93.1
	LR	95.07	96.25	95.1	91.35	95.66	95.67
	DT	95.54	97.11	95.58	92.69	96.32	96.33
	XGBoost	95.54	98.15	95.63	93.75	96.82	96.86
Count vectorization	RF	98.09	97.9	98.1	96	97.99	98
	SVM	90.52	95.21	90.9	86.03	92.8	92.96
	LR	94.97	96.25	95.01	91.25	95.61	95.62
	DT	93.93	96.7	94.07	91.01	95.42	95.48
	XGBoost	95.64	97.86	95.7	93.55	96.74	96.76
Word2Vec (CBOW)	RF	95.18	96.02	95.34	91.34	95.6	95.67
	SVM	96.1	96.67	96.3	92.95	96.39	96.48
	LR	98.25	96.83	98.3	95.15	97.53	97.57
	DT	96.34	93.63	96.32	89.95	94.97	94.95
	XGBoost	94.84	96.06	95.12	91.15	95.44	95.57
Word2Vec (skip gram)	RF	96.97	95.94	97.09	93.06	96.45	96.52
	SVM	96.1	96.67	96.3	92.96	96.39	96.48
	LR	98.24	97.95	98.32	96.28	98.1	98.14
	DT	97.17	93.25	97.19	90.47	95.17	95.19
	XGBoost	94.84	96.06	95.12	91.15	95.44	95.57
GloVe	RF	94.65	93.55	94.82	88.4	94.1	94.12
	SVM	93.48	89.06	93.45	82.51	94.21	91.2
	LR	97.47	97	97.57	94.57	97.57	97.29
	DT	94.55	89.83	94.51	84.33	92.13	92.1
	XGBoost	97.37	92.68	97.36	90	94.97	94.95
FastText (CBOW)	RF	96.58	95.92	96.73	92.67	96.25	96.33
	SVM	96.57	96.48	96.38	92.86	96.52	96.43
	LR	94.06	97.13	93.95	91.11	95.57	95.53
	DT	90	95.1	90.01	85.11	92.47	92.48
	XGBoost	95.64	96.09	95.43	91.53	95.86	95.76
FastText (skip gram)	RF	96.31	96.95	96.29	93.25	96.63	96.62
	SVM	90.29	96.82	90.4	87.19	93.44	93.48
	LR	97.87	98.78	97.77	96.57	98.32	98.28
	DT	94.16	95.53	94.31	89.83	94.84	94.91
	XGBoost	96.3	96.75	96.1	92.86	96.52	96.43

and headings of the websites. The rationale behind choosing this text is due to more probability of containing domain information about the designed website and for phishing sites this text can also provide target website information. Using the word embedding algorithms, we generate embedding for these domains which are further fed to different classifiers like Random Forest, SVM, Logistic Regression, Decision Tree and XGBoost. Experimental results are mentioned in Table 2. With domain specific text also, we observed that Random forest algorithm performed better than other classifiers in case of frequency based embedding approaches. Whereas Logistic Regression performed better than other classifiers with predictive based embedding approaches.

4.5 Experiment 3: evaluation of model with ensemble of word embedding in plain text and domain specific data

In this experiment, we evaluate our model with ensemble of various word embedding algorithms with RF and LR classifiers for the classification. We used both plain text and domain-specific text as input for the ensemble model. For the given corpus, dictionaries are generated with word embed-

Table 2 Results of various word embedding algorithms with domain specific text

Word embedding	Classifiers	Precision	TPR	TNR	MCC	F Score	Accuracy
TF-IDF	RF	97.23	95.06	97.18	92.22	96.13	96.10
	SVM	89.94	92.30	90.14	82.43	91.11	91.20
	LR	90.51	93.43	90.75	84.16	91.95	92.05
	DT	93.26	96.75	93.46	90.17	94.97	95.05
	XGBoost	88.33	95.88	89.13	84.76	91.95	92.24
Count vectorization	RF	97.23	94.88	97.17	92.04	96.04	96.00
	SVM	89.65	91.12	89.77	80.9	90.38	90.44
	LR	94.97	96.25	95.01	91.25	95.61	95.62
	DT	91.93	94.32	91.58	85.87	92.82	92.91
	XGBoost	88.70	96.20	89.47	85.42	92.30	92.58
Word2Vec (CBOW)	RF	96.27	97.64	96.71	94.27	96.95	97.14
	SVM	96.77	93.93	97.04	91.08	95.33	95.53
	LR	97.78	97.29	98.01	95.32	97.53	97.67
	DT	96.27	93.81	96.58	90.50	95.02	95.24
	XGBoost	95.16	96.32	95.72	92.00	95.74	96.00
Word2Vec (skip gram)	RF	96.83	98.44	97.27	95.61	97.63	97.81
	SVM	97.24	93.80	97.51	91.47	95.49	95.72
	LR	98.47	97.18	98.64	95.90	97.82	97.95
	DT	96.02	92.62	96.40	89.18	94.29	94.58
	XGBoost	96.12	95.56	96.59	92.26	95.88	96.14
GloVe	RF	97.10	94.20	97.10	91.29	95.62	95.62
	SVM	91.21	81.39	90.33	71.35	86.02	85.40
	LR	97.49	92.15	97.41	89.50	94.74	94.67
	DT	96.62	88.50	96.40	84.67	92.38	92.15
	XGBoost	96.04	90.45	95.91	86.28	93.16	93.05
FastText (CBOW)	RF	98.12	98.31	98.26	96.57	98.21	98.28
	SVM	95.51	97.02	95.23	92.30	96.26	96.14
	LR	93.40	98.36	93.24	91.66	95.81	95.76
	DT	94.41	98.00	94.20	92.26	96.17	96.10
	XGBoost	96.61	97.41	96.37	93.81	91.01	96.90
FastText (skip gram)	RF	99.48	98.86	99.51	98.40	99.17	99.20
	SVM	97.28	92.71	97.34	90.10	94.94	94.99
	LR	99.42	98.19	99.46	97.68	98.80	98.84
	DT	99.42	97.94	99.45	97.44	98.68	98.71
	XGBoost	95.98	96.64	96.27	92.89	96.31	96.45

ding algorithms. These dictionaries are used to generate word vectors which are further used to train the model.

In this ensemble model with various embedding algorithms, we used RF and LR on all the dictionaries (frequency and prediction). Also, we used RF for frequency based word embedding data and LR for Prediction based word embedding data and vice versa. The reason to use only RF and LR for the generation of ensemble is that RF outperformed other classifiers with frequency based word embedding data whereas LR outperformed others with prediction based word embedding data as shown in Experiments 1 and 2. In case RF-LR model, the word vectors from frequency-based are given to random forest and word vectors from prediction based are given to logistic regression. Now with the predictions made from the algorithm for each word embedding are used to generate ensemble model with majority voting strategy. Results of ensemble model are mentioned in Table 3. From the table, it is observed that LR-RF model with input DST outperformed other models with an accuracy of 97.95% and MCC of 95.9% whereas RF-LR with PT outperformed other models with a significant accuracy of 98.53% and MCC of 97.06%.

Table 3 Result of Ensemble model with text from website source code

Metrics (%)	Ensemble with DST				Ensemble with PT			
	RF	LR	RF + LR	LR + RF	RF	LR	RF+LR	LR+RF
Precision	98.39	97.31	97.56	97.83	96.7	98.15	98.12	96.22
TPR	96.65	98.05	98.45	98.27	96.79	98.98	98.96	96.83
TNR	98.34	97.19	97.32	97.59	96.74	98.06	98.09	95.78
MCC	94.97	95.72	95.81	95.9	93.52	97.05	97.06	92.65
F Score	97.51	97.9	98	98.05	96.88	98.56	98.54	96.529
Accuracy	97.47	97.86	97.91	97.95	96.76	98.53	98.53	96.33

Table 4 Result of multimodal model with plaintext and domain specific text

Metrics (%)	Multimodal with DST					Multimodal with PT				
	RF	SVM	LR	Decision tree	XGBoost	RF	SVM	LR	Decision tree	XGBoost
Precision	99.13	94.72	98.79	98.46	96.41	96.72	97.11	97.71	95.22	95.42
TPR	99.59	98.15	99.65	99.7	96.86	93.91	94.39	97.13	92.29	93.47
TNR	99.07	94.55	98.71	98.37	96.18	96.91	97.29	97.89	95.5	95.73
MCC	98.68	92.76	98.39	98.1	93.05	90.9	91.75	95.05	87.86	89.26
F Score	99.36	96.41	99.22	99.07	96.64	95.29	95.73	97.42	93.73	94.44
Accuracy	99.34	96.35	99.19	99.05	96.53	95.44	95.86	97.53	93.91	94.63

4.6 Experiment 4: multimodal of word embedding in domain specific data

In the multimodal model, we generate word embedding using all the frequency based as well as prediction based algorithms. Initially, dictionaries are generated using the Embedding algorithms and thereby they are concatenated to result comprehensive dictionary which are further used for the classification task. For identifying the suitable classifier we have applied various classifiers on the concatenated data and the results are given in Table 4. From the results it is observed that, with DST as input to the model, RF outperformed others with a significant accuracy of 99.34% and MCC of 98.68%. With PT as input to the model, LR outperformed others with an accuracy of 97.53% and MCC of 95.05%.

4.7 Experiment 5: comparison with existing techniques

In this experiment, we compare our technique with existing works that use features extracted from URL and content from the source code. We have implemented Shirazi et al. [26], Su et al. [28], Yang et al. [37] and Li et al. [14] for the comparison. For simplicity, we follow the convention as W1 for Shirazi et al. work, W3 for Su et al. W4 for Yang et al work, W2 for our work. The results are given in Table 5. From the results it is observed that the proposed multimodal outperformed existing works with a significant accuracy of 99.34% and MCC of 98.68%

Table 5 Comparison of our work with existing works

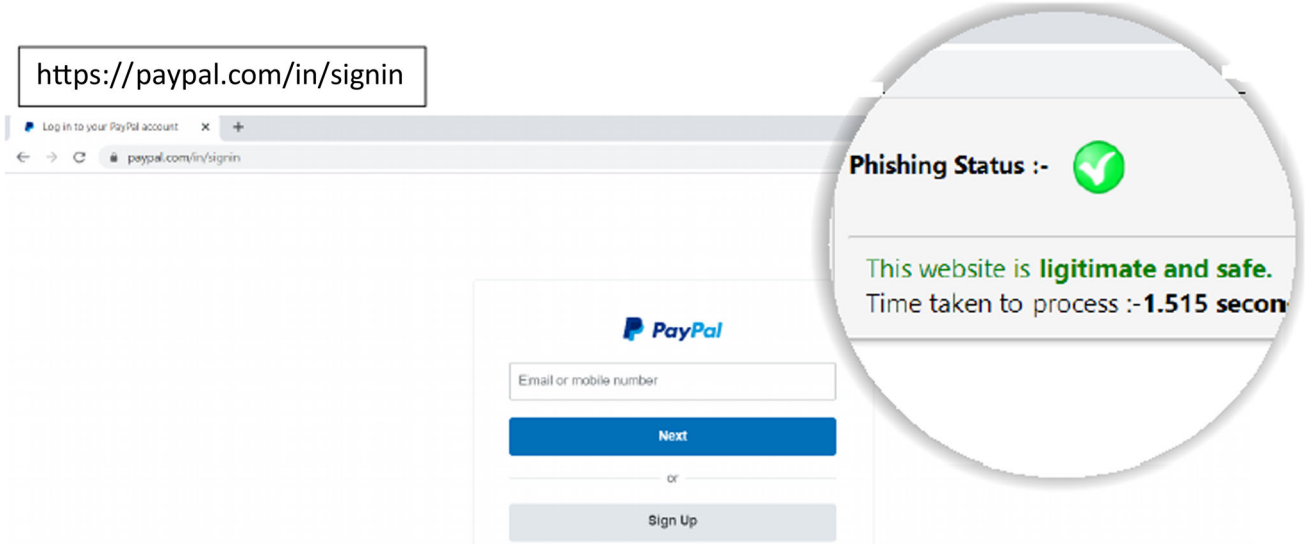
Metrics (%)	W1	W2	W3	W4	W5
TPR	93.04	99.59	75.01	97.06	96.61
FPR	4.1	0.93	15.17	1.96	2.34
Accuracy	94.67	**99.34**	80.61	97.55	97.13
Precision	94.47	99.13	78.84	98.06	97.66
F Score	93.75	99.36	76.87	97.56	97.14
MCC	89.12	**98.68**	60.25	95.1	94.26

Bold values indicate that the final parameters of our work used to compare with existing works

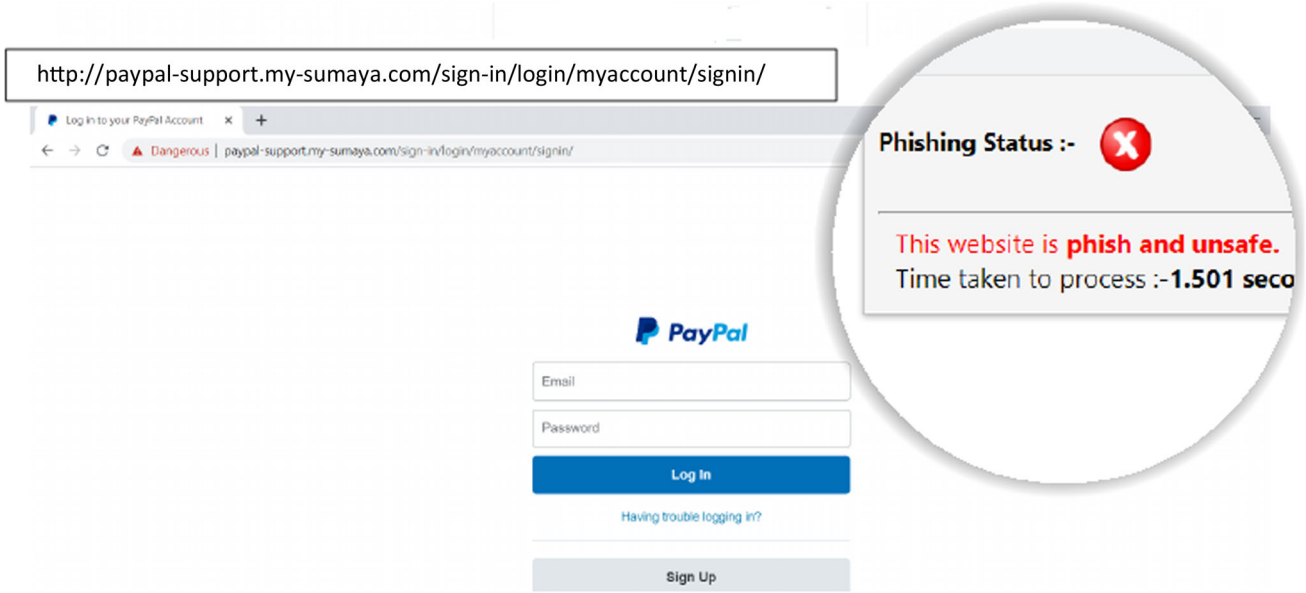
5 Deployment of the model and limitations

5.1 Design of chrome extension

The main intention of the proposed work is to design a real time application for the detection of phishing sites. Hence, we designed a chrome extension which takes input as URL and classifies it as either legitimate or phishing URL. Javascript is used to design the extension which extracts the document object model (DOM) from the given input URL. The DOM is sent to the remote server through REST API service. Note that the remote server is the place where the actual execu-



(a) Legitimate site detection



(b) Phishing site detection

Fig. 2 Output of chrome extension

tion of the proposed work takes place and also it is the same place where the REST API also keeps running for the transfer of data from extension to remote server. The REST API is hosted on an Intel Xeon 16 core Ubuntu server with 2.67 GHz processor and 16 GB RAM. Note that, the REST API is implemented with Spring framework and POST method is used for transferring the DOM, whereas GET method is used for transferring the URL and Title to the server.

As our main goal is to design real time detection mechanism, hence we need a tool which detects the phishing URL with a very low response time. To achieve the same, we have preloaded the dictionaries of various classifiers and moreover, we have parallelized the process of generating word embeddings for both DST and PT with the help of multiple cores in the remote system.

The designed extension is tested with various phishing and legitimate sites for the identification of detection time which is observed to be an average time 1.56 s. The demo of the extension is shown in Fig. 2. From the figure, it is observed that the proposed model detected a legitimate paypal as legitimate with a response time of 1.515 s whereas the model correctly detected phishing site targetting PayPal website in 1.501 s. The extension is designed such that no extra clicks or key press are required i.e. on single click of

extension in chrome browser, the extension displays the status of the website as legitimate or phishing. The status of the website is shown in pop up window with green color text for legitimate site and red color text for phishing website.

5.2 Limitations

- As the proposed model completely depends on plain text and domain specific text, it fails when the text is replaced with an image.
- The Model is language dependent as it is trained on English websites.
- Due to large memory requirement for the word embedding, we had to average the size of each document which would in turn affect our model's efficiency.

6 Conclusion

In this paper, we employed different word embedding with various classifiers to identify the best word embedding algorithms suitable for phishing detection. Also the experiment results demonstrate that FastText with Skip Gram performs better for domain specific text than plain text as well as other word embedding algorithms. Another noteworthy observation is that logistic regression works better with prediction based word embedding algorithms whereas random forest outperforms other classifiers better with frequency based algorithms. We have evaluated the model with both ensemble and multimodal approach and the experimental results demonstrated that multimodal approach achieved significant performance with MCC 98.68%, accuracy 99.34%, TPR 99.59% and TNR 99.07%.

The proposed model is implemented as a chrome extension using which client gives input as URL and obtain the legitimacy of the URL as output. In the future, we would like to apply deep learning algorithms on the ensemble of embeddings extracted from the websites. We also intend to include additional lexical features along with content based features to improve the performance of the model.

Acknowledgements The authors would like to thank Ministry of Electronics and Information Technology (Meity), Government of India for their support in part of the research.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Afzal, S., Asim, M., Javed, A. R., Beg, M. O., & Baker, T. (2021). Urldeepdetect: A deep learning approach for detecting malicious urls using semantic vector models. *Journal of Network and Systems Management, 29*(3), 1–27.
2. Aleroud, A., & Zhou, L. (2017). Phishing environments, techniques, and countermeasures: A survey. *Computers & Security, 68*, 160–196.
3. Azeez, N., Misra, S., Margaret, I. A., & Fernandez-Sanz, L. (2021). Adopting automated whitelist approach for detecting phishing attacks. *Computers & Security, 108*, 102328
4. Basit, A., Zafar, M., Liu, X., Javed, A. R., Jalil, Z., & Kifayat, K. (2021). A comprehensive survey of AI-enabled phishing attacks detection techniques. *Telecommunication Systems, 76*(1), 139–154.
5. Belabed, A., Aïmeur, E., & Chikh, A. (2012). A personalized whitelist approach for phishing webpage detection. In *2012 7th international conference on availability* (pp. 249–254). IEEE: Reliability and Security.
6. Cao, Y., Han, W., & Le, Y. (2008). Anti-phishing based on automated individual white-list. In *Proceedings of the 4th ACM workshop on Digital identity management (DIM '08)* (pp. 51–60). Association for Computing Machinery, New York, NY. https://doi.org/10.1145/1456424.1456434
7. Cheng, Y., Chai, T., Zhang, Z., Lu, K., & Du, Y. (2021). Detecting malicious domain names with abnormal whois records using feature-based rules. *The Computer Journal*.
8. Chiew, K. L., Yong, K. S. C., & Tan, C. L. (2018). A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Systems with Applications, 106*, 1–20.
9. Fang, Y., Zhang, C., Huang, C., Liu, L., & Yang, Y. (2019). Phishing email detection using improved rcnn model with multilevel vectors and attention mechanism. *IEEE Access, 7*, 56,329–56,340.
10. Gastellier-Prevost, S., Granadillo, GG., & Laurent, M. (2011). Decisive heuristics to differentiate legitimate from phishing sites. In *2011 conference on network and information systems security* (pp 1–9). https://doi.org/10.1109/SAR-SSI.2011.5931389
11. Gowtham, R., & Krishnamurthi, I. (2014). A comprehensive and efficacious architecture for detecting phishing webpages. *Computers & Security, 40*, 23–37.
12. He, M., Horng, S. J., Fan, P., Khan, M. K., Run, R. S., Lai, J. L., Chen, R. J., & Sutanto, A. (2011). An efficient phishing webpage detector. *Expert Systems with Applications, 38*(10), 12,018–12,027.
13. Jain, A. K., & Gupta, B. (2021). A survey of phishing attack techniques, defence mechanisms and open research challenges. *Enterprise Information Systems,* 1–39.
14. Li, Y., Yang, Z., Chen, X., Yuan, H., & Liu, W. (2019). A stacking model using URL and HTML features for phishing webpage detection. *Future Generation Computer Systems, 94*, 27–39.
15. Mohammad, R. M., Thabtah, F., & McCluskey, L. (2012). An assessment of features related to phishing websites using an automated technique. In *2012 international conference for internet technology and secured transactions* (pp. 492–497). IEEE.
16. Mourtaji, Y., Bouhorma, M., Alghazzawi, D., Aldabbagh, G., & Alghamdi, A. (2021). Hybrid rule-based solution for phishing URL detection using convolutional neural network. *Wireless Communications and Mobile Computing, 2021,* 8241104. https://doi.org/10.1155/2021/8241104.
17. Prakash, P., Kumar, M., Kompella, R. R., & Gupta, M. (2010). Phishnet: Predictive blacklisting to detect phishing attacks. In *2010 proceedings IEEE INFOCOM* (pp. 1–5). https://doi.org/10.1109/INFCOM.2010.5462216
18. Rao, R. S., & Pais, A. R. (2017). An enhanced blacklist method to detect phishing websites. In *International conference on information systems security* (pp. 323–333). Springer.
19. Rao, R. S., & Pais, A. R. (2019). Detection of phishing websites using an efficient feature-based machine learning framework. *Neural Computing and Applications, 31*(8), 3851–3873.

20. Rao, R. S., & Pais, A. R. (2019). Two level filtering mechanism to detect phishing sites using lightweight visual similarity approach. *Journal of Ambient Intelligence and Humanized Computing, 11,* 1–20.
21. Rao, R. S., Vaishnavi, T., & Pais, A. R. (2019). Phishdump: A multi-model ensemble based technique for the detection of phishing sites in mobile devices. *Pervasive and Mobile Computing, 60*(101), 084.
22. Rao, R. S., Vaishnavi, T., & Pais, A. R. (2020). Catchphish: Detection of phishing websites by inspecting URLs. *Journal of Ambient Intelligence and Humanized Computing, 11*(2), 813–825.
23. Rao, R. S., Pais, A. R., & Anand, P. (2021). A heuristic technique to detect phishing websites using TWSVM classifier. *Neural Computing and Applications, 33*(11), 5733–5752.
24. Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications, 117*, 345–357.
25. Shirazi, H., Bezawada, B., & Ray, I. (2018). "kn0w thy doma1n name" unbiased phishing detection using domain name based features. In *Proceedings of the 23nd ACM on symposium on access control models and technologies* (pp 69–75).
26. Shirazi, H., Bezawada, B., & Ray, I. (2018). "kn0w thy doma1n name" unbiased phishing detection using domain name based features. In *Proceedings of the 23nd ACM on symposium on access control models and technologies* (pp. 69–75).
27. da Silva, C. M. R., Feitosa, E. L., & Garcia, V. C. (2020). Heuristic-based strategy for phishing prediction: A survey of URL-based approach. *Computers & Security, 88*(101), 613.
28. Su, K. W., Wu, K. P., Lee, H. M., & Wei, T. E. (2013). Suspicious URL filtering based on logistic regression with multi-view analysis. In *2013 8th Asia joint conference on information security* (pp. 77–84). IEEE.
29. Tharani, J. S., & Arachchilage, N. A. (2020). Understanding phishers' strategies of mimicking uniform resource locators to leverage phishing attacks: A machine learning approach. *Security and Privacy, 3*(5), e120.
30. Vijayalakshmi, M., Shalinie, S. M., Yang, M. H., et al. (2020). Web phishing detection techniques: A survey on the state-of-the-art, taxonomy and future directions. *IET Networks, 9*(5), 235–246.
31. Wang, S., Khan, S., Xu, C., Nazir, S., & Hafeez, A. (2020). Deep learning-based efficient model development for phishing detection using random forest and BLSTM classifiers. *Complexity, 2020,* 8694796. https://doi.org/10.1155/2020/8694796.
32. Wang, Y., Agrawal, R., & Choi, B. Y. (2008). Light weight anti-phishing with user whitelisting in a web browser. In *2008 IEEE region 5 conference* (pp. 1–4). IEEE.
33. Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., & Woźniak, M. (2020). Accurate and fast URL phishing detector: A convolutional neural network approach. *Computer Networks, 178*(107), 275.
34. Xiao, X., Zhang, D., Hu, G., Jiang, Y., & Xia, S. (2020). CNN-MHSA: A convolutional neural network and multi-head self-attention combined approach for detecting phishing websites. *Neural Networks, 125*, 303–312.
35. Xiao, X., Xiao, W., Zhang, D., Zhang, B., Hu, G., Li, Q., & Xia, S. (2021). Phishing websites detection via CNN and multi-head self-attention on imbalanced datasets. *Computers & Security, 108,* 102372.
36. Xu, L., Zhan, Z., Xu, S., & Ye, K. (2013). Cross-layer detection of malicious websites. In *Proceedings of the 3rd ACM conference on data and application security and privacy, association for computing machinery, New York, CODASPY '13* (pp. 141–152). https://doi.org/10.1145/2435349.2435366
37. Yang, P., Zhao, G., & Zeng, P. (2019). Phishing website detection based on multidimensional features driven by deep learning. *IEEE Access, 7*, 15,196–15,209.
38. Zamir, A., Khan, H. U., Iqbal, T., Yousaf, N., Aslam, F., Anjum, A., & Hamdani, M. (2020). Phishing web site detection using diverse machine learning algorithms. *The Electronic Library, 38*(1), 65–80. https://doi.org/10.1108/EL-05-2019-0118
39. Zhang, D., Yan, Z., Jiang, H., & Kim, T. (2014). A domain-feature enhanced classification model for the detection of Chinese phishing e-business websites. *Information & Management, 51*(7), 845–853.
40. Zhang, W., Jiang, Q., Chen, L., & Li, C. (2017). Two-stage elm for phishing web pages detection using hybrid features. *World Wide Web, 20*(4), 797–813.
41. Zhang, X., Zeng, Y., Jin, X., Yan, Z., & Geng, G. (2017). Boosting the phishing detection performance by semantic analysis. In *2017 IEEE international conference on big data (big data)* (pp. 1063–1070).
42. Zhang, Y., Hong, J. I., Cranor, L. F. (2007). Cantina: A content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web* (pp. 639–648).

Routhu Srinivasa Rao is an Assistant Professor in Department of Computer Science and Engineering, GMR Institute of Technology Rajam, India. He completed his B.Tech. (Computer Science and Engg.) from SRKR Engineering College, Andhra University, India and M.Tech. (Computer Science and Engg.) from NIT Kurukshetra, Haryana, India. He had pursued his Ph.D from NITK Surathkal,India. He has His area of interest include Information Security, Cyber Security and Phishing.

Amey Umarekar is student of Master of Technology in Computer Science - Information Security in the Department of Computer Science and Engineering, NITK Surathkal, India. He completed his B.E (Information Technology) from Jabalpur Engineering College, India. His area of interest includes Information Security and Machine Learning.

Alwyn Roshan Pais is Assistant Professor in Department of Computer Science and Engineering, NITK Surathkal, India. He completed his B.Tech. (CSE) from Mangalore University, India, M.Tech. (CSE) from IIT Bombay, India and Ph.D. from NITK, India. His area of interest includes Information Security, Image Processing and Computer Vision.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com