Classification of Phishing Email Using Word Embedding and Machine Learning Techniques

Somesha M.* and Alwyn R. Pais

Information Security Research Lab, Department of Computer Science and Engineering, National Institute of Technology karnataka, Surathkal, Karnataka, India, 575025
E-mail: somesha.187co004@nitk.edu.in; alwyn@nitk.edu.in
**Corresponding Author*

Received 23 August 2021; Accepted 10 March 2022;
Publication 26 April 2022

Abstract

Email phishing is a cyber-attack, bringing substantial financial damage to corporate and commercial organizations. A phishing email is a special type of spamming, used to trick the user to disclose personal information to access his digital assets. Phishing attack is generally triggered by emailing links to spoofed websites that collect sensitive information. The APWG survey suggests that the existing countermeasures remain ineffective and insufficient for detecting phishing attacks. Hence there is a need for an efficient mechanism to detect phishing emails to provide better security against such attacks to the common user. The existing open-source data sets are limited in diversity, hence they do not capture the real picture of the attack. Hence there is a need for real-time input data set to design accurate email anti-phishing solutions. In the current work, it has been created a real-time in-house corpus of phishing and legitimate emails and proposed efficient techniques to detect phishing emails using a word embedding and machine learning algorithms. The proposed system uses only four email header-based

Journal of Cyber Security and Mobility, Vol. 11_3, 279–320.
doi: 10.13052/jcsm2245-1439.1131

heuristics for the classification of emails. The proposed word embedding cum machine learning framework comprises six word embedding techniques with five machine learning classifiers to evaluate the best performing combination. Among all six combinations, Random Forest consistently performed the best with FastText (CBOW) by achieving an accuracy of 99.50% with a false positive rate of 0.053%, TF-IDF achieved an accuracy of 99.39% with a false positive rate of 0.4% and Count Vectorizer achieved an accuracy of 99.18% with a false positive rate of 0.98% respectively for three datasets used.

Keywords: Email phishing detection, Word embedding, Machine Learning, Word2ec, FastText, TF-IDF..

1 Introduction

The advent of the internet since 1990 has brought much convenience along with threats to the privacy of users. Since its inception, the users availing of the facilities provided by the internet are increasing in an unprecedented way, and its current users are 4 billion out of 7 billion population on the earth. Most internet users are unaware of the technicalities of the internet and probable to fall into a trap designed by some malicious users [1]. In India, there has been an increase in rural internet users over the past three years with the introduction of affordable data rates. Out of India's 1.3 billion population, about 600 million use the internet, those figures will only rise in the future. The increase in the number of users also invites phishers fraudulent practices to attack people's privacy by exploiting their analphabetism from internet functionality.

1.1 Phishing Emails:

Phishing is a fraudulent activity conducted by cybercriminals to get unauthorized access to the user's resources. Many studies [2–6] suggest that the young population is attached to digital devices and communication networks for their communication, education, entertainment, and financial transactions. Hence this population is most vulnerable to phishing email attacks. Phishing is most generally triggered by sending emails containing links to spoofed websites. Email phishing has become a real and serious threat to electronic commerce, because, it contains a message from credible looking sources requesting to disclose and gaining access to steal sensitive information from individuals and financial organizations. The number of phishing attacks

Table 1 APWG Email phishing statistics 2019

Quarter-1	Phishing Emails	Quarter-2	Phishing Emails	Quarter-3	Phishing Emails	Quarter-4	Phishing Emails
January	34630	April	37045	July	35530	October	45057
February	35364	May	40177	August	40457	November	42424
March	42399	June	34932	September	42273	December	45072
Total - Q1	**112393**	**Total – Q2**	**112154**	**Total – Q3**	**118260**	**Total – Q4**	**132553**

Table 2 APWG Email phishing statistics 2020

Quarter-1	Phishing Emails	Quarter-2	Phishing Emails	Quarter-3	Phishing Emails	Quarter-4	Phishing Emails
January	52407	April	43282	July	119181	October	143950
February	43270	May	39908	August	119180	November	119700
March	44008	June	44497	September	128926	December	133038
Total – Q1	**139685**	**Total – Q2**	**127687**	**Total – Q3**	**367287**	**Total – Q4**	**396688**

continued to rise in the fall of 2019 according to the Phishing Activity Trends Report ([7] and [8]) of the Anti-Phishing Working Group (APWG). APWG [7] identified a total of 266,387 phishing sites from July through September 2019. This was 46 percent higher than the 182,465 seen in the second quarter of 2019 and almost double the 138,328 seen in [9]. The email phishing activities reduced drastically in the year 2019, four-quarter statistics of APWG 2019 unique phishing emails are tabulated in Table 1.

Table 1 shows the amount of unique phishing email messages received from clients by APWG. According to the latest four quarter reports of 2020, 1031347 unique phishing e-mails (campaigns) were recorded from the consumers. The four quarter monthly results of 2020 are tabulated in Table 2. The cyber-criminals use COVID-19 related disaster content for phishing against health care warriors, hospitals, and healthcare facilities. According to the APWG – 2015 survey report ([10] and [11]), the highest number (1,413,978) of phishing emails was identified in the year 2015. The statistics of phishing emails recorded from 2010 to 2020 can be seen in Figure 1. The APWG survey clearly states that email phishing is one of the major threats to be considered for further research to identify efficient anti-phishing solutions. According to the Mimecast survey report [12], the majorly affected organizations are Finance 68%, Professional services 66%, and Manufacturing 66%. According to Kaspersky's third quarter report of 2019 [13], educational institutions and university sensitive documents are stolen and sold in the dark market.

As the phishing websites and phishing emails are often nearly identical to legitimate websites and emails, current filters have limited success in detecting these attacks, and leaving users vulnerable to a growing threat.

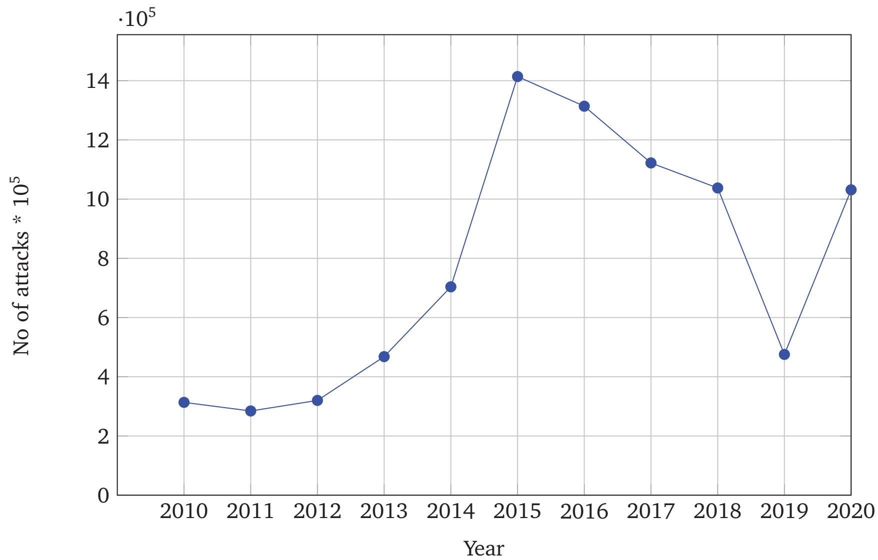

Figure 1 Email Phishing attacks from 2010 to 2020.

Users become a victim and end up revealing private (sensitive) information due to,

- Lack of computer system knowledge
- Inadequate knowledge of security and security indicators
- Replication of original sites with minor change mostly goes unnoticed by users
- Ignoring security warnings.

1.2 Word Embeddings:

Word embedding is a collective term for a group of language models and methods for selecting features often referred to as word representation. Its primary objective is to map textual terms or phrases into a continuous low dimensional space. Word embeddings convert human textual language meaningfully into a numeric representation. The converted text to numbers may be a different numeric representation of the same text.

- *Need of word embeddings:* Many machine learning algorithms and all deep learning algorithms or architectures are incapable of processing strings or plain text in their raw form. They require numbers as inputs to perform any sort of jobs (classification, regression, etc). A huge amount

of data present in the text format is imperative to extract knowledge out of it and build applications. The real world text based applications are sentiment analysis of reviews by business organizations, document or news classification or clustering by google, etc. A word embedding format generally tries to map a word using the dictionary to a vector.
Example: Sentence = "Word embeddings are word converted into numbers" Words in the sentence are 'embeddings' or 'numbers'. A dictionary may be the list of all unique words in the sentence, so a dictionary may look like ['Word','embeddings','are','converted','into','numbers'].

- *Word embedding types:* Word embedding is categorized into two types: (1). Frequency-based, and (2). Prediction-based embedding techniques.

1.2.1 Frequency-based word embedding

Frequency-based word embedding counts words in each document and is a very basic, fast, and easy method to create word vectors. There are two types of embeddings, TF-IDF and Count vectorizer.

TF-IDF: Term Frequency and Inverse Document Frequency vectorization works, by finding out the most unique words present not just in the document but in the entire corpus of the documents. The intuition behind this is that the more frequent words may not be the relevant words. Some words just appear in the documents more number times. IDF works by finding such words and giving better unique words present in the entire corpus of documents as the more frequent irrelevant words hold little to no new information. TF, the number of times a word has appeared in a document. Further, it can be divided by the total number of words in a document. Therefore,

$$TF(t,d) = \frac{x}{y} \tag{1}$$

where x is the count of t in document d, and y is the number of words in document d.
IDF measures the uniqueness of the word across the corpus.

$$IDF(t,d) = log(\frac{N}{n}) \tag{2}$$

where N is the total number of documents present in the corpus, and n is the number of documents where the term t appears.

$$TFIDF = TF * IDF \tag{3}$$

Count Vectorization (CV): In count vectorization, a matrix of size $d \times n$ will be created. Where d is the size of the corpus i.e number of documents and n is the number of unique tokens in the documents. This matrix holds the count of each word appearing in a document. The similarity between each of the vectors generated is calculated using the cosine similarity i.e the angle between the two vectors.

1.2.2 Prediction-based word embedding

Prediction-based word embeddings are more efficient and accurate language modeling techniques in modern research and are considered a byproduct of language models according to Almeida and Xexéo [14]. Some of the prediction-based word embedding techniques such as Word2Vec, FastText, and GloVe are discussed below.

Word2Vec (W2V): To infer any relationship between two words is difficult in their one-hot encoding representation. Sparsity is another issue with the one-hot encoding as there are many redundant ”0” in their vector representation. Word2Vec solves these problems by using surrounding words in representing the target words. Word2Vec is a predictive model which tries to learn embedding from the given text. It is a three layer architecture with a small hidden layer that does the task of generating embeddings from given text. The size of input and output given is generally the same. Word2Vec has two algorithms, those are Continuous Bag of Words (CBOW) and SkipGram (SG).

- **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. CBOW uses a neural network as continues distributed representation of the context of words and predicts a word as an output. The working and architecture of the CBOW and SkipGram models were presented by Mikolov et al [15]. This model predicts the probability of occurrence of a word given the context of words surrounding it.
- **SkipGram model:** SkipGram model tries to predict the context of the given word. The architecture is just opposite to that of the CBOW model. SkipGram takes the input as the target word and outputs the context words that surround the target word.
  For example: Given the sentence, ”It was an apple pie”, if the input is ”a”, the output would be ”It”, ”was”, ”apple”, and ”pie” for the window size of 5. The dimension of all the input and output data is the same and one-hot encoded. This model consists of one hidden layer with a dimension equal to the embedding size, which is lesser than the vector size of input/output. A softmax activation function is applied at the end

of the output layer which describes the likelihood of the appearance of a specific word in the context.

Two challenges that appear with Word2Vec are,

- **Out of Vocabulary (OOV) words:** Word2Vec can handle only words it has encountered during its training. For example, Word2Vec vocabulary containing words such as ”tensor” and ”flow” can not handle embedding for the word ”tensorflow”, i.e., a compound word. Thus it leads to an ”out of vocabulary” error.
- **Morphology:** Word2Vec does not do any parameter sharing for words such as ”eat” and ”eating” with the same radicals. Each word is uniquely learned based on the context in which the word appears. Thus the internal structure of the word can be utilized properly to make the embedding more efficient.

FastText (FT): FastText is the library developed by Facebook AI Research (FAIR) [16], and it is given as an open-source free library. FastText uses unsupervised or supervised learning algorithms to create word vectors. It uses two algorithms which are Continues Bag of Words and Skip Gram model similar to Word2Vec. FastText is used to develop word embedding for a word using the n-gram of each character. Thus, FastText allows the generation of vector representations of previously unseen words in the text. And also used for finding semantic similarities, text classification, and fast training of large datasets.

GloVe: Glove represents Global Vectors, and it is an unsupervised learning distributed word representation model to obtain vector representation of words. Global Vectors are generated using the co-occurrence matrix statistic from a corpus. The matrix denoted by X, $X_{i,j}$ represents the number of times word j appears in the context of the word i. $P_{i,j} = X_{i,j}/X_i$ which gives the probability that the word j occurs in the context of the word i. These probabilities can provide some potential to encode some form of the underlying meaning of the contextual meaning.

The ***research contributions*** of our work are listed below:

- Creation of In-house real-time phishing and legitimate email datasets.
- Proposed a novel phishing email detection technique using word embedding and machine learning.
- The proposed novel architecture uses only FOUR email header features and achieved competitive accuracy.

- Proposed work outperformed all other existing works on publicly available datasets.
- The proposed work justified that the newly created datasets are accurate and obtained nearly similar results as with publicly available datasets.

The rest of the paper is organized as follows: In section 2, the literature survey about phishing email detection and word embedding techniques used are provided. The major section of our paper is the proposed work and is discussed in section 3. This section clearly describes the architecture and its stages. Processing of emails, heuristics selection, feature extraction, vector generation, and classification of emails is the core part of this section. Section 4 describes the implementation of the proposed model. Results and discussions are given in section 5 to justify the performance of the proposed method in comparison with all existing works. The conclusion of our work and the future enhancement is given in section 6.

2 Literature survey

”Phishing Email” is a deceptive activity performed by fraudsters by spoofing emails ostensibly from some trusted companies or organizations to gain financial benefits from victims by camouflage emails. This is usually done by including a link that appears to take the victim to the fake website to fill victim’s personal information. The provided information goes directly to the crooks behind the scam.

Phishing attacks are categorized into deceptive and malware phishing. Deceptive phishing is the major concern of this literature survey. Deceptive phishing is related to social engineering schemes, which depend on forged email claims that appear as originated from a legitimate organization. And also, an embedded link redirects the user to fake websites to obtain personal information to defraud the user. The emails are classified as phishing or legitimate by using filtering (particularly keywords or learning-based filters which analyze a collection of labeled training data). Email messages majorly consist of two parts, header(s) and body.

Email header consists of a structured set of fields, such as From, To, Subject, Message-id, etc. The body is the content of the email message. The taxonomy (Figure 2) and structure (Figure 3) of the email clearly describe the format of the email message. These figures are obtained from Almomani et al [17]. The strategy of a phishing email is to attract victims and direct them to a particular phishing website. The received emails are embedded

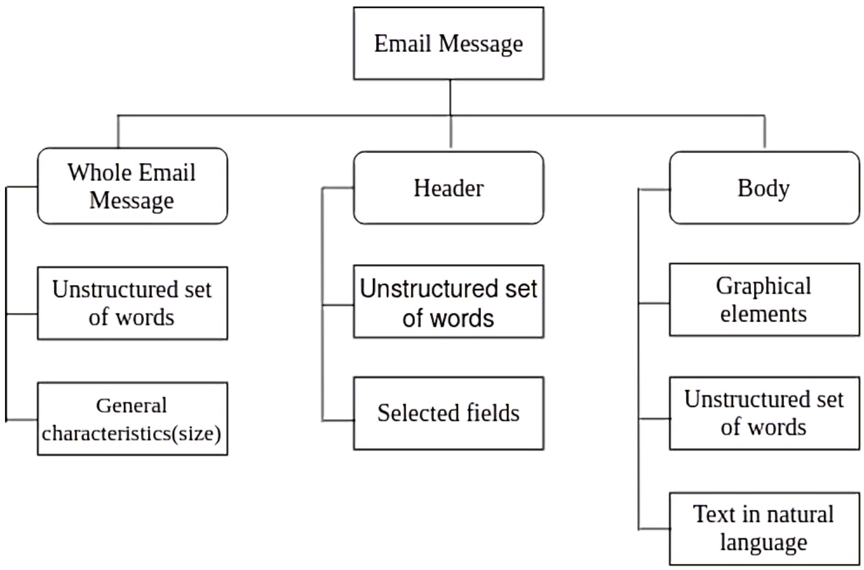

Figure 2 Email message Taxonomy – Courtesy [17].

with URLs and trick the user to mouse click on the embedded link to reveal confidential information. Several email phishing features are identified by many researchers and found many different mechanisms to achieve the identification of legitimate and fake emails. According to Almomani et al [17], there exist three types of feature sets, such as basic features, latent topic features, and dynamic Markov chain features. The author [17] identified different anti-phishing approaches contributed by researchers. A brief survey of techniques used in machine learning, deep learning, word embedding, and natural language processing is discussed in detail.

2.1 Machine Learning Based Techniques

Fette et al [18] proposed a live filtering solution, based on PILFER a machine learning based classification approach. They used 10 features, out of which nine features were extracted from the email itself and the tenth feature represents the age of the linked-to-domain names. Toolan and Carthy [19] proposed an extension to the work of Fette et al [18], using classifier ensembles for the classification of phishing and non-phishing emails. They used the C5.0 algorithm and achieved a very high precision. Toolan and Carthy [19]

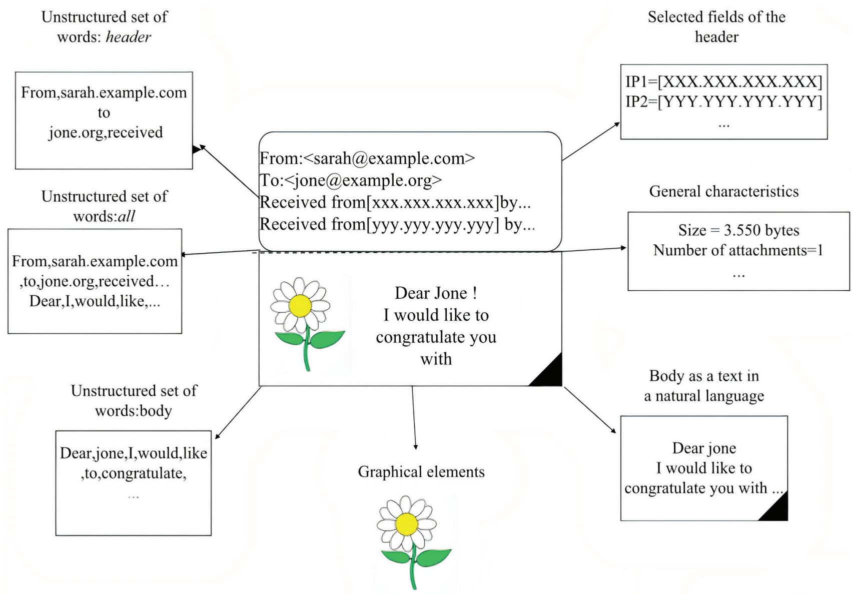

Figure 3 Email message structure – Courtesy [17].

used only FIVE features on approximately 8000 emails, half of which were phishing and remaining legitimate.

Bergholz et al [20] proposed new filtering approaches by selecting novel features suitable to identify phishing emails. The selected features suite of statistical models for low dimensional descriptions of email topics. The work was carried out by sequential analysis of email text, external links, and detection of embedded logos as well as indicators for hidden salting (inserting white text on white background). They used 27 basic features with two novel features (logo detection and hidden salting) and obtained an f-measure of 99.46%. Toolan and Carthy [21] identified 40 features extracted from the email body of over 10,000 emails which are divided among ham, spam, and phishing. The selected features are evaluated using an information gain algorithm and classified as Best-IG, Median-IG, and Worst-IG features. Best-IG features outperformed among all with an average accuracy of 97.1%. The freely available datasets from SpamAssassin and Phishing corpus were used (4202-ham, 1895-spam, and 4563-phish).

Khonji et al [22] proposed feature subset evaluation and feature subset searching methods. The primary focus of this is to enhance the classification accuracy of phishing emails by finding the effective feature subsets from the number of previously proposed features. There are a total of 21 features selected (email body, email header, URL, JavaScript, and external features) from Fette et al [18], Bergholz et al [20], Toolan and Carthy [21], and Gansterer and Pölz [23]. After evaluating with various feature selection methods, Wrapper with RF performed the best with 21 features and an f1-score of 99.396%. The authors used publicly available datasets of 4116 phishing emails from monkey.com[1] and 4150 ham emails from SpammAssasin.com[2]. Abu-Nimeh et al [24] proposed distributed phishing detection by applying variable selection using Bayesian Additive Regression Trees (BART). They presented a distributed client-server architecture to detect phishing e-mails by automatic variable selection. BART improves its predictive accuracy when compared to other classifiers. This architecture is also used to detect phishing attacks in a mobile environment. Abu-Nimeh et al [24] used 71 features for training and testing of 6 Machine learning algorithms (RF, LR, SVM, Nnet, CART, BART), and proved that there is no standard classifier for phishing email prediction.

Chandrasekaran et al [25] proposed a technique to classify phishing based on the structural properties of phishing e-mails. They used one-class SVM to classify phishing e-mails before it reaches the user's inbox, essentially reducing the human exposure based on selected features. The prototype sits between users Mail Transfer Agent (MTA) and Mail User Agent (MUA) and process each arriving email. Their results claim a detection rate of 95% of phishing e-mails with a low false positive rate. Cohen et al [26] proposed a novel set of general descriptive features for enhanced detection of malicious emails using machine learning methods. The proposed features are extracted directly from the email itself, therefore the features are independent, don't require internet or any other tools, and meet the needs of real-time systems. These features are from all components, i.e., header, body, and attachments. The authors used 33142 emails which contain 38.73% of malicious and 61.27% benign emails. Applied 30 most prominent features of the 100 features extracted by applying three main feature selection approach those are, Filter methods, wrapper methods, and embedded methods. Random Forest (RF) classifier achieved the highest detection accuracy of 92.9%, TPR 94.7%, FPT 0.03 among 9 commonly used machine learning classification algorithms (J48, RF, NB, Bayesian Networks, LR, LogitBoost, Sequential Minimal Optimization, Bagging, and Adaboost).

---

[1] https://monkey.org/ jose/phishing/

[2] https://spamassassin.apache.org/old/publiccorpus/

2.2 Deep Learning Based Techniques

Smadi et al [27] proposed a framework that is a combination of neural networks and reinforcement learning to detect phishing attacks in the online mode. The proposed model can adapt itself to identify newly arrived phishing emails for newly explored behaviors of emails dynamically. A novel algorithm adopts to explore new phishing behaviors in the new datasets. The dynamic system achieves an accuracy of 98.63%, TPR of 99.07%, and TNR of 98.19%. The drawback of this approach is learning dynamic updates of features and datasets for every email may slow down the system. Nguyen et al [28] presented a deep learning model with hierarchical-LSTM and a supervised attention mechanism. The hierarchical LSTM structure is implemented first for words at the lower level, whose results are then passed to the LSTM structure in the sentences at the upper level to generate vector representation for the email. An attention mechanism is used to combine these two levels and to assign the contribution weights to each of the words and sentences in the email. A deep learning model is used to automate the feature engineering process for phishing email detection. With the use of both the email headers and body, they achieved precision, recall, and F1 scores of 0.990, 0.992, and 0.991 respectively.

Li et al [29] proposed LSTM based phishing detection model for big email data. The proposed model includes the sample expansion stage and testing stage. The model combines KNN with K-means algorithms to expand the training data for deep learning. In this work, the author used private data set generated from their email servers, and mailboxes collected from some organizations. In this work, they used seven header and content-based features and achieved an accuracy of 95%. Alhogail et al [30] proposed Graph Convolutional Network (GCN) based phishing email detection using body-based text features and natural language processing techniques. The author achieved an accuracy of 98.2% and a false positive rate of 0.015, and used the CLAIR collection of fraud emails dataset.

2.3 Word ebedding Based Techniques

Fang et al [31] proposed a model called THEMIS, which is a combination of deep learning and Word2Vec techniques for phishing email detection. The model uses improved Recurrent Convolution Neural Networks (RCNN) with multilevel vectors and attention mechanisms. They used word level and character level vectorization for a rich set of vectors. The extracted vectors are tuned, trained, and tested using RCNN to obtain an efficient phishing accuracy of 99.848% and FPR of 0.043%. The obtained results are competitive, but the same would have been trained and tested with other word embedding and deep learning techniques. The author used multiple datasets to have a bulky dataset. Bagui et al [32] proposed approach uses deep semantic analysis, ML, DL techniques to classify phishing and legitimate emails. They used private datasets collected from various industries in the USA. The proposed approach uses hybrid features as text and achieved an accuracy of 98.89% with word phrasing and 96.34% without word phrasing using n-gram analysis and one-hot encoding techniques. In the proposed work, Bagui et al [32] claim that stop words are not removed and used both header and body features.

Castillo et al [33] proposed email threat detection using a distinctive neural network approach. The author described different approaches for detecting malicious content in emails. The proposed model is a combination of machine learning and natural language processing and used publicly available and private datasets. The model uses only email contents as input data set to classify emails as malicious or benign. In this work, the Gensim-Word2Vec model is used to generate numeric word vectors and achieved testing accuracy of 95.68% with 1025 emails. Ra et al [34] used the combination of word embedding, a neural bag of n-grams, and some of the deep learning models such as CNN, RNN, LSTM, MLP for the detection of phishing emails. Deep learning models are used to extract the optimal features and non-linear activation functions are used for classification. All the models are trained on an anti-phishing shared task corpus at IWSPA-AP 2018. The proposed model achieved a training accuracy of 99.1% with word embedding vector and LSTM network.

Hiransha et al [35] made use of IWSPA-AP 18 datasets to train the model consisting of Keras word embedding and CNN. The proposed model combining word embedding and CNN gives a vector representation for the words in the emails which are then used in the classification of legitimate and phishing emails. The proposed model without an email header has an accuracy of 96.8% and has an accuracy of 94.2% with the email header. Harikrishnan et al [36] made use of TF-IDF and some classical machine learning algorithms such as RF, AdaBoost, NB, DT, and SVM. The proposed method uses TF-IDF for vector representation of words and SVD, NMF for feature extraction, and dimensionality reduction. This model is trained on IWSPA-AP 18 datasets. The proposed model has a testing accuracy of 90.29% for emails with headers using TF-IDF and NMF representation.

Verma et al [37] proposed ”Detecting phishing emails the natural language way” in the year 2012, the first scheme used natural language processing techniques and contextual information in detecting phishing emails. The scheme uses all parts of the email including the header, text in the body, and the links present in an email. The proposed model named ”PhishNet-NLP” operates between a mail transfer agent and a mail user agent. The proposed method achieves an accuracy of 97%. The obtained accuracy is comparatively less than in other works. Gutierrez et al [38] proposed a model called SAF$_E$-PC for detecting a new form of phishing attacks, a semi-automated feature generation model for phishing classification. The model uses a huge corpus received from the Purdue university's central IT organizations with the help of a state-of-the-art email filtering tool called Sophos installed on a Microsoft exchange server. The author used three datasets as caught, uncaught, and benign of size 388,264 emails, 37,606 and 158,444 emails respectively, and used 806 features from email header, body, and links. The authors also tested their model with SpamAssassin open-source corpus and noticed the model performed better with collected real-time datasets. The authors claim that the proposed work is an extension of work carried out by Verma et al [37]. Used features in the proposed work are huge and may require more time to process in a real-time environment.

Valecha et al [39] used a new convention called Persuasion cues instead of features, keywords, or phishing techniques used by other researchers. The proposed technique uses Word2Vec with four machine learning classifiers and compared the candidate model for gain, loss, and gain_loss persuasion cues with the baseline model and achieved an improvement of approximately 5 to 20%. The model with Word2Vec and SVM achieved the highest gain accuracy of 96.52%, loss of 96.16%, and gain_loss accuracy of 95.97%.

A summary of major works based on machine learning to detect phishing emails is tabulated in Table 3. The table describes related works based on seven parameters, those are authors of the paper, model or algorithms used, number of features used, input dataset, the accuracy achieved, weakness, and techniques used. Some major related works based on word embedding with machine learning or deep learning to detect phishing emails are tabulated in Table 4. It may be observed that the related works are also evaluated based on the parameters used. According to the survey conducted, the majority of the research works carried out on both the header and body of an email, and some works on attachments along with the header and body of an email. None of the works focused exclusively on email header features or word vectors based on email headers. After analyzing the research gaps, this

Table 3 Summary of the related works based on Machine Learning

Author	Model / Algorithm	Features	Dataset(s)	Accuracy (%)	Weakness
Smadi et al [27]	Dynamic evolving Neural Networks based on Reinforced learning	50 Hybrid (email header, body, & attachment) feature	Phishing corpus + PhishTank: 4559 SpamAssasin:4559	Acc: 98.63 TPR:99.07 TNR:98.19	Run time for training and detection is high.
Islam and Abawajy [40]	Multi-tire classification Model (SVM, AdaBoost, and NaiveBayes)	21 Hybrid features	SpamAssasin, Phishing corpus	Acc: 97 FP: 2 FN: 9	Complexity of analysis is high. Misclassification of test data is high (3%)
Khonji et al [41]	Lexical URL Analysis (LUA) technique with RF	48 Hybrid features (including LUA as a feature)	Phishing corpus: 4116, SpamAssasin: 4150	F-score: 99.37 FP: 0.59	Fails to detect phishing emails having non textual body with higher accuracy
Gansterer and Pölz [23]	SVM & J48	30 Hybrid features	Phishing corpus: 5000, SpamAssasin: 5000	Acc: 97	Higher cost because of online features. Internet connection decides the classification speed.
Ramanathan and Wechsler [42]	phishGILLNET, probabilistic latent semantic analysis, AdaBoost	200 topic Email body features	400,000 emails from SpamAssasin, Phishing corpus, Enron, SPAM Archive, and PhishTank	Acc: 97.7	Take more memory and computation time because of the complex architecture
Ma et al [43]	Machine Learning algorithms (DT, RF, MLP, NB, SVM)	7 Email Header and Body features	Live emails received by West-Pac and their customers. Phishing-46,525 Legitimate-613,048	Acc: 99	Did not use a verified dataset of phishing and legitimate emails.
Toolan and Carthy [19]	Ensemble model with C5-Decission Tree, K-NN, LR, SVM, R-Boost algorithms used	5 Hybrid features (Header and body)	SpamAssasin: Ham-4202, Spam-1895 Nazario: Phishing corpus – 4563	F-Score: 99.31	The only motive of this work is to re-label false-negatives to boost the true positive rate.
Hamid and Abawajy [44]	Machine Learning	7 Hybrid features	Total: 4594 (Nazario and SpamAssasin)	Acc: 96	Accuracy reduces when dataset size increases.
Abu-Nimeh et al [24]	Multi classifiers Algorithms (LR, CART, SVM, NNET, BART, RF)	71 Hybrid features	Phishing corpus:1403, Legit (Financial): 178 Legit (Others): 4974	Acc: 97.09 (SVM)	Consumes more time and memory due to large set of features
Chandrasekaran et al [25]	Classifiers based model, Simulated annealing algorithm and SVM	25 structural features (language, layout, and structure of phishing e-mails before getting into Inbox)	In-House generation Phishing: 200 Legitimate: 200	Acc: 95 (SVM)	Used very small dataset size of 200. Time consuming
Fette et al [18]	PILFER – RF & SVM based classifier	10 Hybrid features	Phishing corpus: 860 SpamAssassin: 6950	Acc: 96	Resulted in 0.12% false positive and 7.35% false-negative rates. Achieved low performance with large datasets.
Alhogail et al [30]	Graph Convolutional Networks (GCN)	Body features	CLAIR collection of fraud emails Phishing: 3685 Legit: 4894	Acc: 98.2	Used only body based features, accuracy is less compared to some existing works. Unknown No. of features.

Table 4 Summary of the related works based on Word Embedding

Author	Model/ Algorithm	Features	Dateset (s)	Accuracy (%)	Observations
Bagui et al [32]	Deep Semantic Analysis, ML, DL, and Word Embedding	Hybrid	Non-public	98.89	Used only one-hot encoding for vector generation, and used both header and body text features.
Nguyen et al [28]	NLP, DL, and H-LSTM	Hybrid	IWSPA-AP 2018	99.0	Used only Hierarchical LSTM
Castillo et al [33]	DNN, CN, RNN, abd Word2Vec	Body	Enron, APWG, and Non-public	95.68	A general classifier used for email and other types of message services inclu-ding SMS.
Ra et al [34]	CNN, MLP, LSTM,and Word Embedding	Hybrid	IWSPA-AP 2018	99.1	Imbalance datasets are used, and not tested the model with balanced data-sets to justify the results
Hiransha et al [35]	Word Embedding, CNN	Body	IWSPA-AP 2018	96.8	The dataset is highly imbalanced and leads to decrease in accuracy.
Harikrishnan et al [36]	Classical ML techniques, TF-IDF	Hybrid	Combination of different publicly available datasets	90.29	Achieved accuracy is very low compared to all existing works. Over fitting due to unbalanced datasets
Valecha et al [39]	Word2Vec and Machine learning techniques.	Hybrid	Millersmile[a]	96.52	Used Persuasion cues, Didn't compare efficiency with other works, compared only with baseline model

[a] https://millersmiles.co.uk/

work proposes a model which uses only email header-based heuristics for efficient phishing email detection. The study of related works gave clarity to adopt word embedding for our research with machine learning classifiers. The architecture in the next section evaluates the best possible combination of the classification process with multiple word embedding and machine learning classifiers. Based on the research summary, it may be concluded that word embedding techniques may generate more suitable word vectors to classify given emails as phishing or legitimate using different machine learning classifiers. Hence, a new word embedding and machine learning based phishing email detection technique is proposed in this paper.

3 Proposed work

The architecture of the novel work proposed to detect phishing emails is shown in Figure 4. The architecture has multistage functionality to process and classify the email as phishing or legitimate. The steps involved in this process are:

- Input Emails
- Feature extraction
- Dictionary creation
- Vectorization
- Classification

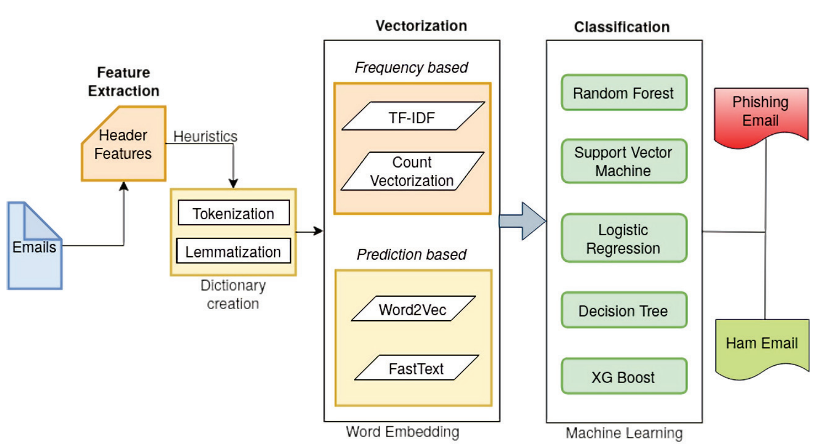

Figure 4 Architecture of Phishing Email Detection.

3.1 Input Emails

Emails are the actual messages communicating between two or more known peers in electronic messaging media. Initially, the raw emails are grouped as datasets repositories. These emails are processed to extract the required heuristics. The required heuristics are extracted from email headers of pre-processed datasets taken from public and in-house generated repositories.

3.2 Feature Extraction

Feature extraction is the first step in the proposed model. In this stage, the required heuristics are extracted from the emails.

Python scripts are written to extract only required header heuristic features from MBOX format files. After extracting the required heuristics, unwanted tags, text, garbage characters, and some special symbols are removed. The extracted data from individual emails are stored and saved as a CSV file. The generated CSV file is an input to the proposed architecture to classify phishing or legitimate emails. The selected heuristics are discussed below.

3.2.1 Description of selected heuristic features

As discussed in section 2, most of the existing works used hybrid features, and some works have used only content-based features. In this work, only four header labels are selected as heuristic features. The selected heuristic features are From, Return-Path, Subject, and Message-id from the email header.

- **From:** This is a label for the sender's email address and name. The address may be a person, a company, or an association that created an internet account from the email service provider. The genuine email account from address will be used by fraudsters to send web links, malware, and other means to defraud users.
- **Return-Path:** Email header generated by SMTP protocol to keep track of reverse path, and is used for collecting and processing bounced emails. These bounced email return paths are used by fraudsters to steal sensitive information by broadcasting vulnerable links to targeted users.
- **Subject:** This is a brief description of an email message to convey the information. The subject is a major vulnerable heuristic among all four fields. Subject contains catchy, urgency, bank, financial, and account-related information for conducting fraudulent activity.

- **Message-ID:** This is a globally unique identifier for every individual email and has a specific format. The generated Message-ID is specific to the email address and message, thus no two emails have the same Message-ID. The generated message-ID will be used by a fraudster to project as a legitimate user interacting to gain the personal information of the end-user.

3.3 Dictionary creation

The extracted heuristics are tokenized using the ***nltk*** library. Tokenization is the process of splitting input documents into smaller units such as words or terms. The input document may be a phrase, sentence, paragraph, or an entire document. Each of these smaller units is called a token. The output of tokenization is fed to the lemmatization process. Lemmatization tries to remove the inflectional endings from the word and provide the dictionary form of the words. This process is achieved using vocabulary and morphological analysis i.e studying the structure and formation of the word. Lemmatization converts and correctly identifies a word to its base form and helps in considering the context of the word which is being used.

Example: Let us consider the following sample feature,
Subject = ”Transaction alerts for your State Bank of India Debit Card”
When the subject is tokenized, the string looks like,

[’Transaction’, ’alerts’, ’for’, ’your’, ’State’, ’Bank’, ’of’, ’India’, ’Debit’, ’Card’]

The output from the tokenizer is fed to the lemmatizer, the obtained output from the lemmatizer as below,

[’Transaction’, ’alert’, ’for’, ’your’, ’State’, ’Bank’, ’of’, ’India’, ’Debit’, ’Card’]

3.4 Vectorization

The vectorization processes are generally unsupervised learning methods to convert words into a numeric format. The obtained numeric vectors are trained using classification models. Before generating vectors, pre-processing of extracted features from the emails should be cleaned up by removing special symbols, extra space, irrelevant numeric data, and garbage characters as a tokenization process. The extracted words are lemmatized to remove inflectional endings and lowered. In this work, two common vectorization methods, such as frequency (count of words/context co-occurrences) and prediction-based methods are used. The prediction-based methods are FastText & Word2Vec, and the frequency-based methods used are TF-IDF & Count vectorization. These word embedding techniques are used to represent words, allowing machine learning algorithms to understand words that have similar meanings. Word vectors are just numerical vectors that represent the meaning of a word. Word vectors are multidimensional floating-point values that represent semantically comparable words that are mapped to approximate positions in geographic space.

Word2Vec – SkipGram: In the proposed mechanism, Word2Vec SkipGram takes a dictionary of words generated in the earlier step as input and generates corresponding vectors. SkipGram works well with unknown words. Word2Vec has several parameters as input and cosine similarity techniques for the generation of vectors. The used parameters are vector size, window size, minimum count, number of workers, number of iterations, and input datasets. Fine-tuning of these parameters results in obtaining suitable vectors to achieve better efficiency.

Word2Vec – CBOW: CBOW works opposite of the SkipGram model. In the proposed mechanism, CBOW takes a dictionary of words generated and predicts the target word by taking context words as an input. CBOW is faster and works better with frequently occurring words. All the parameters used in Word2Vec – CBOW model is the same as with the SkipGram model to generate corresponding word vectors of real numbers. The vectors generated depends on the vector size and other parameters assigned to a Word2Vec function.

FastText – SkipGram: FastText is a library developed by FAIR based on two papers Bojanowski et al [16] & Joulin et al [45]. The proposed library function includes a set of parameters namely input corpus, vector size, window size, minimum count, and the number of workers to generate vectors. This algorithm uses hierarchical classifiers and n-gram techniques for unlabeled datasets to train the model. In the proposed mechanism, the FastText SkipGram model takes a dictionary of words generated in the earlier section 3.3 as input and generates corresponding real-valued vectors.

FastText – CBOW: In this method, the model captures all the words in a surrounding window and uses some of their vectors to predict the target.

TF-IDF: Term Frequency and Inverse Document Frequency works by finding out the most frequent unique words present in the entire corpus of the documents. TF identifies the most frequent words in a document. IDF provides words uniquely present in the corpus of documents which does provide some relevant information about the document. The vectors are generated by replacing true or false conditioned boolean values of vector size for the words of the dictionary. TF-IDF terms need to be normalized to reduce the bias in term frequency from terms in short or longer documents. The corresponding sparse matrix is generated to identify term frequency and inverse document frequency of corpus.

Count Vectorization: Count Vectorizer tokenizes the text along with performing very basic pre-processing. It removes the punctuation marks and converts all the words to lowercase. The vocabulary of known words is formed which is also used for encoding unseen text later. An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document. We need to normalize Count Vectorizer terms to reduce bias in term frequency from terms in short or longer documents. The normalized resultant vector generated of size 100 and the corresponding sparse cse_matrix is obtained from the count vectorizer. The generated vectors are further fed into machine learning classifiers to classify the given email as phishing or legitimate. The working of word embedding algorithms used in the proposed work is discussed in section 1.2.

3.5 Classification

The rich set of vectors are generated from the vectorization module, these vectors are amazingly powerful because they allow identifying similarity across different words in a continuous vector space. The generated vectors from word embedding techniques are classified into phishing or legitimate emails using five machine learning classification algorithms. The machine learning algorithms used in the proposed work are Random Forest (RF), Decision Tree (DT), Support Vector Machine (SVM), XGBoost, and Logistic Regression (LR). These algorithms are very efficient in classifying emails as phishing or legitimate.

4 Implementation

The proposed model uses Pandas, nltk, sklearn, gensim, and numpy, libraries. Pandas library used for database processing and reshaping, nltk used for statistical natural language processing. Sklearn is used for classification, regression, clustering, and dimensionality reduction. Gensim model is an open-source library used for converting word to vector, document to vector, and finding text-similarity in word embeddings.

4.1 Dataset preparation

The Datasets are prepared by two methods, 1. Using open-source corpus, and 2. In-house corpus generation.

4.1.1 Open-source corpus

The dataset preparation is one of the major tasks of the current work. Most of the works published till today used two well known open-source datasets available in the open access repositories for phishing and legitimate emails called phishing corpus[1] and ham corpus[2]. These two corpora contain a periodically updated email repository of legitimate and phishing emails. These two corpora have duplicate emails due to periodic updates of emails to the same repository. We have selected unique 6295 legitimate[2] and 9135 phishing[1] emails as **Dataset-1** by eliminating duplicate emails from the repository. For **Dataset-2** we have used the phishing emails from **Dataset-1** and legitimate emails from an in-house dataset of size 18270.

4.1.2 In-House corpus

The dataset creation by examining individual emails is one of the most important steps in email phishing. The innovative researchers require updated new real-time data to understand the day-to-day activities of the phisher. Most of the works carried out by researchers have used existing open-source datasets as in Dataset-1. The selected SpamAssasin[2] datasets are collected in the year 2002, and Phishing corpus[1] datasets are collected during the period 2004 to 2007 and 2015 to 2017. The ham email datasets are older than phishing corpus emails collected after November 2004. Both open-source datasets mismatch with their period of recording the repository. The duration mismatch may lead to the failure of phishing email detection. Phishers may modify unnoticeable parameters to lure victims. The phisher's behaviors and techniques are changing every day to trick victims by obtaining sensitive credentials to defraud users. To overcome the above problem, and tackle the current tricks, **Dataset-3** is created using real-time in-house phishing and legitimate datasets. The new repositories are collected from institution students, research scholars, family members, and friends to understand the behavior of fraudsters with a diversified set of users. The selected emails are analyzed manually and labeled individual emails as phishing and legitimate. The selected datasets and their size are shown in Table 5. In the process of dataset creation, some basic steps are followed to identify emails as phishing or legitimate and they are given below.

Table 5 Datasets used

Dataset	Legitimate Emails	Phishing Emails	**Total**
Dataset – 1	6295	9135	**15430**
Dataset – 2	18270	9135	**27405**
Dataset – 3	18270	8986	**27256**

- Analyse the behavior of suspicious emails.
- Analyse the source code of the original email message.
- Analyse the Google warning indicators.
- Using MxTOOLBOX[3] online tool to analyze email headers.

4.2 Evaluation metrics

Evaluation metrics are used to analyze the performance of the proposed model. The evaluation metrics used to evaluate our proposed model are given below.

- The sensitivity or recall is known as true positive rate (TPR):

$$TPR = \frac{TP}{(TP + FN)} * 100 \tag{4}$$

  where, *TP = No. of phishing emails classified as phishing*, and *(TP + FN) = Total no. of phishing emails*.
- Specificity as true negative rate (TNR):

$$TNR = \frac{TN}{(TN + FP)} * 100 \tag{5}$$

  where, *TN = No. of ham emails classified as ham*, and *TN + FP) = Total no. of ham emails*.
- Accuracy (Acc):

$$Acc = \frac{(TP + TN)}{(TP + FP + TN + FN)} * 100 \tag{6}$$

  where, *(TP + TN) = No. of correctly classified phishing and ham emails*, and *(TP + FP + TN + FN) = Total no. of emails.*

[3]https://mxtoolbox.com/Public/Tools/

- Precision (P):
$$P = \frac{TP}{(TP + FP)} * 100 \tag{7}$$
where, *TP = No of phishing emails classified as phishing*, and *(TP + FP) = Total no. of emails classified as phishing.*
- F-score (F) :
$$F = 2 * \frac{P\ *\ TPR}{P\ +\ TPR} \tag{8}$$
- Matthews Correlation Coefficient (MCC): This measure is considered as a balanced measure, used for different class size datasets. MCC provides a correlation coefficient between predicted and observed outcomes.

$$MCC = \frac{TP * TN\ --FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{9}$$

The equations (4) (5) (6) (7) (8), and (9) represent an evaluation of the input sample for their rate of positive and negative rates. The metric TP is truly positive, which represents the number of phishing emails classified as phishing, TN is true negative represented as the number of legitimate emails classified as legitimate. The other two basic metrics used are false positive (FP) and false-negative (FN), the FP represents the number of legitimate emails classified as phishing, and FN represents the number of phishing emails classified as legitimate. These basic metrics are used to calculate recall (4), specificity (5), accuracy (6), precision (7), F-measure (8), and Matthews Correlation Coefficient (9).

4.2.1 System requirements

The basic system configuration used to run the experiments are, CPU – ThinkStation, processor – Intel Xeon(R) CPU E5-2650 v3 @ 2.30GHz x 40, Memory – 64GiB. The other basic setup required to run the word embedding and machine learning algorithms are Ubuntu 18.04, Pycharm professional tool to run python scripts, and required datasets.

5 Results and discussion

To conduct experimentation, the three different datasets as mentioned in Table 5 is used. The training and testing are performed with 70% & 30% of each dataset's total size with a window size of 10. Before performing the main experiments, a series of prerequisite tasks are needed. To minimize as much noise as possible, python scripts are written to remove empty spacing, angle brackets, single and double quotes, and so on. The parsed email header heuristics are then saved as a CSV file. The training on these selected CSV files is conducted to train the proposed model with word embedding and ML algorithms. The highest training accuracy achieved from dataset-1 is 100% with Word2Vec (CBOW & SkipGram) and RF for vector size 300. In the similar training experiments conducted on dataset-2, the highest training accuracy achieved is 99.88% with Word2Vec (CBOW) and RF for vector size 200. The training is also conducted on a purely in-house repository i.e. dataset-3, the achieved highest training accuracy is 99.87% with Word2Vec (CBOW & SkipGram) and RF for vector size 150. From the training results, it is observed that Word2Vec with RF consistently performed the best with all three datasets.

5.1 Experiment-1

Dataset-1 is the input to our proposed model. To perform this experiment the following word embedding algorithms such as TF-IDF, Count Vectorization, Word2Vec (CBOW), Word2Vec (SkipGram), FastText (CBOW), and FastText (SkipGram) are used. The output vector size of these algorithms is varied from 50 to 300 to study the performance of each algorithm. The results are tabulated in Table 6. It may be observed that TF-IDF with RF, DT, XG Boost, and LR achieves an accuracy of 99.37%, 99.13%, 98.62%, and 99.07% respectively for the vector size 300. Similarly, TF-IDF with SVM achieves an accuracy of 98.16% for the vector size of 150. Count Vectorizer also performed the same as TF-IDF. The RF, DT, XG Boost, LR, and SVM achieve an accuracy of 99.33%, 98.92%, 98.42%, 98.77%, and 97.17% for vector sizes 300 and 150 respectively. From the above results, TF-IDF and CV efficiency improve as the vector size increases in four cases except for SVM.

The results of Word2Vec (CBOW) with RF & DT are observed that the accuracy achieved are 99.26% & 98.79% respectively for vector size 300. Similarly, Word2Vec (CBOW) with SVM achieves an accuracy of 98.90% for vector sizes 100 & 200. XG Boost and LR achieve an accuracy of 98.92% and 99.16% for the vector sizes of 100 and 200 respectively. Word2Vec (SkipGram) achieves an accuracy of 99.35%, 98.96%, and 99.26% with RF, DT, and LR respectively for vector size 200. Similarly, SVM and XG Boost achieve an accuracy of 98.90% and 98.92% for vector sizes 50 and 150 respectively. The results observed from Word2Vec are not uniform for the selected vector size.

Table 6 Selection of vector size with Dataset-1

Word Embedding	Machine Learning	Testing Accuracy(%) of Vectors				
		50	**100**	**150**	**200**	**300**
TF-IDF	RF	98.92	98.59	99.07	99.16	**99.37**
	DT	98.49	98.03	98.64	98.66	**99.13**
	SVM	97.08	97.73	**98.16**	98.08	97.97
	XG Boost	98.25	97.64	98.27	98.27	**98.62**
	LR	98.21	98.16	98.62	98.92	**99.07**
Count Vectorizer	RF	98.90	98.59	98.98	99.29	**99.33**
	DT	98.55	98.08	98.55	98.87	**98.92**
	SVM	96.11	97.08	**97.17**	97.10	96.37
	XG Bosst	98.33	97.86	98.34	98.38	**98.42**
	LR	98.18	97.77	98.53	98.64	**98.77**
Word2Vec (CBOW)	RF	98.81	98.94	98.87	99.09	**99.26**
	DT	97.73	98.25	98.49	98.40	**98.79**
	SVM	98.72	**98.90**	98.85	**98.90**	98.75
	XG Boost	98.70	**98.92**	98.40	98.79	98.79
	LR	99.15	98.90	98.92	**99.16**	98.90
Word2Vec (SkipGram)	RF	98.94	99.00	98.98	**99.35**	98.79
	DT	98.29	98.23	98.44	**98.96**	98.68
	SVM	**98.90**	98.75	98.77	98.85	98.46
	XG Boost	98.75	98.57	**98.92**	98.77	98.66
	LR	99.11	99.07	99.16	**99.26**	98.92
FastText (CBOW)	RF	99.42	**99.50**	**99.50**	99.31	99.16
	DT	98.92	99.03	**99.11**	98.94	98.64
	SVM	**98.38**	98.29	97.95	97.79	97.45
	XG Boost	98.87	**99.29**	98.64	98.94	98.70
	LR	98.66	**99.05**	98.72	98.94	98.55
FastText (SkipGram)	RF	99.44	99.33	99.39	99.37	**99.46**
	DT	98.75	98.94	**99.24**	98.94	98.79
	SVM	**98.87**	98.46	98.31	97.86	97.79
	XG Boost	**99.26**	98.96	99.18	98.85	98.96
	LR	**99.44**	99.16	99.03	99.37	99.24

Facebook proposed a vectorization algorithm called FastText with two word learning techniques as in Word2Vec called CBOW and SkipGram. It may be observed that FastText (CBOW) with RF achieves an accuracy of 99.50% for vector sizes 100 and 150. Similarly, DT and SVM achieve an accuracy of 99.11% and 98.38% for vector sizes of 150 and 50 respectively. XG Boost and LR achieve an accuracy of 99.29% and 99.05% for vector size of 100 respectively. Similarly, FastText (SkipGram) with RF achieves an accuracy of 99.46% for vector size 300. The algorithm vectors applied to DT achieved an accuracy of 99.24% for vector size 150. Similarly, SVM,

Table 7 Selection of vector size with Dataset-2

Word Embedding	Machine Learning	Accuracy(%) of Vector size				
		50	100	150	**200**	300
TF-IDF	RF	95.99	98.47	99.05	**99.39**	99.28
	DT	95.51	97.86	98.42	**98.77**	98.50
	SVM	94.96	97.20	97.80	**97.94**	97.74
	XG Boost	95.44	97.91	98.36	98.66	**98.67**
	LR	95.00	97.38	98.34	98.83	**98.94**
Count Vectorizer	RF	95.96	98.51	99.01	**99.37**	99.37
	DT	95.78	98.02	98.40	**98.62**	98.56
	SVM	94.93	96.69	97.33	**97.53**	96.68
	XG Bosst	95.42	98.0	98.40	98.70	**98.77**
	LR	95.10	97.14	98.28	98.73	**98.75**
Word2Vec (CBOW)	RF	98.39	98.58	98.47	98.42	**98.62**
	DT	97.44	97.43	**97.88**	97.70	97.60
	SVM	97.46	**98.60**	98.40	98.33	98.40
	XG Boost	98.03	97.88	**98.12**	98.10	98.09
	LR	97.91	98.16	98.21	98.13	**98.38**
Word2Vec (SkipGram)	RF	98.68	98.58	**98.72**	98.38	98.56
	DT	97.78	97.50	97.44	**97.88**	97.86
	SVM	97.97	**97.99**	97.87	97.35	97.53
	XG Boost	**98.23**	98.06	98.02	97.84	98.11
	LR	98.61	98.77	98.68	98.70	**98.96**
FastText (CBOW)	RF	**98.72**	98.62	98.58	98.39	98.64
	DT	97.94	**98.00**	97.98	97.44	97.64
	SVM	**97.98**	97.61	97.35	96.82	96.78
	XG Boost	**98.15**	98.08	97.85	97.61	97.83
	LR	**98.00**	97.85	97.87	97.30	97.46
FastText (SkipGram)	RF	98.48	98.47	**98.71**	98.59	98.50
	DT	97.54	97.75	97.48	**97.77**	97.49
	SVM	**97.24**	97.16	97.05	96.82	96.65
	XG Boost	98.05	98.03	97.93	**98.11**	98.10
	LR	98.41	98.36	98.43	**98.50**	98.30

XG Boost, and LR achieved an accuracy of 99.87%, 99.26%, and 99.44% for vector size 50 respectively. Overall, FastText (CBOW) with RF achieves the best accuracy of 99.50% for vector size 100.

5.2 Experiment-2

The experiment-1 is repeated with Dataset-2, and the results are tabulated in Table 7. To validate our proposed technique, an experiment is conducted using the in-house generated legitimate dataset. It may be observed that TF-IDF with RF, DT, and SVM achieved an accuracy of 99.39%, 98.77%, and

97.94% respectively for a vector size of 200. Similarly, XG Boost and LR achieve an accuracy of 98.67% and 98.94% for vector size 300 respectively. Count vectorizer also achieved the highest accuracy with RF, DT, and SVM of 99.37%, 98.62%, and 97.53% for vector size 200. XG Boost and LR achieved an accuracy of 98.77% and 98.75% respectively. As observed, TF-IDF and count vectorizer performed better for vector sizes 200 and 300. The obtained result of Word2Vec (CBOW) with RF is 98.62% accuracy for vector size 300. Similarly, DT, SVM, XG Boost, and LR achieved an accuracy of 97.88%, 98.60%, 98.12%, and 98.38% for vector sizes 150, 100, 150, and 300 respectively. Word2Vec (SkipGram) with RF achieves an accuracy of 98.72% for vector size 150, similarly, DT, SVM, XG Boost, and LR achieve an accuracy of 97.88%, 97.99%, 98.23%, and 98.96% for vector sizes 200, 100, 50. and 300 respectively. The achieved accuracy for CBOW with RF, SVM, XG Boost, and LR is 98.72%, 97.98%, 98.15%, and 98.00% for vector size 50. DT achieves an accuracy of 98.00% for vector size 100. The FastTxet (SkipGram) achieves an accuracy of 98.71% with RF for vector size 150. Similarly, FastText (SkipGram) with DT, SVM, XG Boost, and LR achieves an accuracy of 97.77%, 97.24%, 98.11%, and 98.50% for vector sizes 200, 50, 200, and 200 respectively. The overall results observed from the experiments with dataset-2 are different for different vector sizes. The best accuracy achieved is 99.39% with TF-IDF and RF for vector size 200.

5.3 Experiment-3

Dataset-3 is a purely in-house prepared repository. The in-house repository is an input for the proposed model in the current experiment. The results are tabulated in Table 8. It may be observed that TF-IDF with RF achieves an accuracy of 99.12% for the vector size of 150 and 200. Similarly, TF-IDF with DT achieves an accuracy of 99.03% for the vector size of 200. SVM, XG Boost, and LR achieve an accuracy of 97.60%, 98.74%, and 98.34% for the vector sizes of 100 and 150 respectively. For the Count Vectorizer, the RF, DT, SVM, XG Boost, and LR achieve an accuracy of 99.18%, 99.09%, 97.63%, 98.86%, and 98.49% for the vector sizes of 150, 200, 150, 200, and 150 respectively. The results of Word2Vec (CBOW) with RF and SVM achieve an accuracy of 98.86% and 98.50% for a vector size of 150. DT, XG Boost, and LR achieve an accuracy of 98.69%, 98.52%, and 98.54% for vector sizes of 100, 50, and 200 respectively. Similarly, Word2Vec (SkipGram) with RF achieved an accuracy of 99.06% for the vector size of 100. DT, XG Boost, and LR achieved an accuracy of 98.69%, 98.76%, 98.86% for a vector size of 300. Similarly, SVM achieves an accuracy of 98.69% for the vector size 50.

Table 8 Selection of vector size with Dataset-3

Word Embedding	Machine Learning	Accuracy(%) of Vector size				
		50	100	**150**	200	300
TF-IDF	RF	98.73	99.06	**99.12**	**99.12**	99.08
	DT	98.58	98.71	98.70	**99.03**	98.93
	SVM	96.46	**97.60**	**97.60**	97.05	97.03
	XG Boost	98.24	**98.74**	98.52	98.69	98.64
	LR	97.64	98.00	**98.34**	98.16	98.08
Count Vectorizer	RF	98.75	99.06	**99.18**	99.17	99.12
	DT	98.67	98.92	98.79	**99.09**	98.93
	SVM	96.32	97.49	**97.63**	97.04	96.93
	XG Boost	98.49	98.72	98.73	**98.86**	98.79
	LR	97.71	98.08	**98.49**	98.09	98.09
Word2Vec (CBOW)	RF	98.75	98.80	**98.86**	98.74	98.80
	DT	98.57	**98.69**	98.60	98.52	98.57
	SVM	96.83	98.09	**98.50**	98.22	98.26
	XG Boost	**98.52**	98.46	98.47	98.02	98.08
	LR	98.18	97.94	98.24	**98.54**	98.19
Word2Vec (SkipGram)	RF	98.81	**99.06**	99.02	98.63	98.99
	DT	98.30	98.58	98.50	98.43	**98.69**
	SVM	**98.69**	98.31	98.27	98.03	98.12
	XG Boost	98.04	98.38	98.59	98.46	**98.76**
	LR	98.38	98.47	98.78	98.78	**98.86**
FastText (CBOW)	RF	98.79	98.82	**98.86**	98.75	98.80
	DT	98.50	98.63	98.64	**98.76**	98.68
	SVM	**98.16**	97.98	97.75	97.88	97.89
	XG Boost	98.26	98.46	98.26	**98.62**	98.32
	LR	98.11	98.35	98.31	**98.46**	98.27
FastText (SkipGram)	RF	**98.85**	98.60	98.84	98.73	98.84
	DT	98.49	98.49	98.53	98.51	**98.62**
	SVM	**98.11**	97.42	97.48	97.16	97.22
	XG Boost	**98.52**	98.37	98.46	98.41	98.26
	LR	98.46	98.42	98.47	98.47	**98.67**

FastText (CBOW) with RF achieved an accuracy of 98.86% for a vector size 150. DT, XG Boost, and LR achieve an accuracy of 98.76%, 98.62%, and 98.46% for a vector size of 200. And also SVM achieved an accuracy of 98.16% for a vector size of 50. Similarly, FastText (SkipGram) with RF, SVM, and XG Boost achieve an accuracy of 98.85%, 98.11%, and 98.52% for the vector size 50. DT and LR achieve an accuracy of 98.62% and 98.67% for the vector size 300. Overall Count vectorizer with RF achieved the best accuracy of 99.18% for the vector size 150.

Table 9 Performance Evalution with Dataset-1

Word Embeddings	Algorithm	Acc	MCC	Preci-sion	TPR	TNR	F-Score
TF-IDF	RF	99.37	98.71	99.96	98.98	99.95	99.47
	DT	**99.13**	98.22	99.48	99.05	99.26	99.26
	SVM	97.97	95.84	99.63	96.99	99.45	98.30
	XGBoost	98.62	97.15	99.30	98.36	98.99	98.83
	LR	99.07	98.09	99.70	98.72	99.57	99.21
Count Vectorizer	RF	99.33	98.62	99.85	99.01	99.79	99.43
	DT	98.92	97.77	99.26	98.90	98.95	99.08
	SVM	96.37	92.58	99.08	94.95	98.61	96.97
	XGBoost	98.42	96.75	99.22	98.11	98.88	98.66
	LR	98.77	97.46	99.37	98.54	99.10	98.95
Word2Vec (CBOW)	RF	99.26	98.50	100	98.75	100	99.37
	DT	98.79	97.52	98.95	98.95	98.56	98.95
	SVM	**98.75**	97.44	99.89	97.98	99.84	98.93
	XGBoost	98.79	97.53	99.77	98.16	99.68	98.96
	LR	98.90	97.74	99.40	98.70	99.17	99.05
Word2Vec (SkipGram)	RF	98.79	97.53	100	97.95	100	98.96
	DT	98.68	97.30	99.29	98.44	99.01	98.86
	SVM	98.46	96.88	99.92	97.49	99.89	98.69
	XGBoost	98.66	97.27	99.89	97.84	99.84	98.85
	LR	98.92	97.79	99.85	98.31	99.79	99.07
FastText (CBOW)	RF	99.16	98.26	99.82	98.77	99.73	99.29
	DT	98.64	97.18	99.05	98.65	98.61	98.85
	SVM	97.45	94.72	98.39	97.33	97.63	97.86
	XGBoost	98.70	97.32	99.60	98.24	99.40	98.91
	LR	98.55	97.00	98.98	98.58	98.51	98.78
FastText (SkipGram)	**RF**	**99.46**	98.88	99.89	99.20	99.84	99.54
	DT	98.79	97.50	98.76	99.19	98.21	98.97
	SVM	97.79	95.48	99.67	96.70	99.50	98.15
	XGBoost	**98.96**	97.85	99.56	98.69	99.36	99.12
	LR	**99.24**	98.44	99.71	99.02	99.57	99.36

5.4 Performance Evalution with Dataset-1

The proposed model is evaluated by using individual datasets selected in the current study. The results with Dataset-1 are tabulated in Table 9 with six metrics. Vector size 300 was selected after analyzing the results in section 5.1 and Table 6. According to the input Dataset-1 results, the RF classifier with FastText (SkipGram) achieved the highest accuracy of 99.46%, and the related metrics MCC, Precision, TPR, TNR, and F-Score are 98.99%, 99.89%, 99.20%, 99.84%, and 99.54%, respectively. Similarly, Table 9 tabulates and highlights the greatest individual classifier accuracies. SVM with

Word2Vec (CBOW) attained an accuracy of 98.75%, while XG Boost and LR with FastText (SkipGram) obtained their best individual accuracies of 98.96% and 99.24%, respectively.

5.5 Performance Evaluation with Dataset-2

Testing procedures with dataset-2 are the same as with dataset-1 discussed in previous section 5.4. Vector size of 200 is selected based on the results obtained in experiment-2 as given in Table 7. The testing results of dataset-2 are tabulated in Table 10 with seven different metrics. Among all results, TF-IDF performed well with RF, DT, and LR by achieving 99.39%, 98.77%, and 98.83% accuracies. SVM and XG Boost performed well with FatsText (SkipGram) and Count vectorizer by achieving accuracies of 98.82% and 98.70% as highlighted in Table 10. The corresponding metrics for the highest accuracy observed are MCC 98.63%, Precision 99.19%, TPR 98.97%, TNR 99.60%, and F-score of 99.08 are achieved for TF-IDF with RF.

5.6 Performance Evaluation with Dataset-3

The corpus-3 is tested similar way as discussed in the above sections 5.1 & 5.2, and the results are tabulated in Table 13. The obtained results are nearly the same as with corpus-1 and corpus-2. The achieved results proved that the procedures followed to create in-house datasets are inline. The obtained results for the generated corpus are nearly similar as with open access corpus. This is one of our novel contributions by producing a real-time repository, and sole property of our Institution. As previously stated, the best result obtained with the new corpus is 99.18% accuracy with Count Vectorizer and RF, with corresponding metrics of 98.11%, 98.05%, 99.38%, 99.09%, and 98.71% for MCC, Precision, TPR, TNR, and F-score. In the resulting Table 11, the classifiers with the highest accuracy are highlighted.

5.7 Model Validation

The validation method used for email phishing classification is a Train/test split of rate 70%/30% of total dataset size for all three datasets. This method uses random partitions of 70% training data and 30% of test data. The Confusion matrix for the obtained results of individual datasets are listed in Tables 11 – (a), (b), and (c). The total size of Dataset-1 is 15430, out of which 10801 emails are used for training and 4629 emails are used for testing. Dataset-2 has a total of 27405 emails, out of which 19183 emails are used for

Table 10 Performance Evaluation with Dataset-2

Word Embeddings	Algorithm	Acc	MCC	Preci-sion	TPR	TNR	F-Score
TF-IDF	RF	**99.39**	98.63	99.19	98.97	99.60	99.08
	DT	**98.77**	97.23	98.16	98.13	99.09	98.14
	SVM	97.94	95.35	96.55	97.22	98.29	96.88
	XGBoost	98.66	96.98	98.31	97.66	99.16	97.98
	LR	**98.83**	97.36	98.24	98.24	99.13	98.24
Count Vectorizer	RF	99.37	98.57	99.30	98.79	99.65	99.05
	DT	98.62	96.89	97.76	98.08	98.89	97.92
	SVM	97.93	94.41	95.37	97.12	97.72	96.24
	XGBoost	**98.70**	97.07	98.38	97.70	99.20	98.04
	LR	98.73	97.14	98.09	98.09	99.05	98.09
Word2Vec (CBOW)	RF	98.42	96.50	99.78	95.60	99.89	97.64
	DT	97.70	94.80	97.07	95.97	98.56	96.52
	SVM	98.33	96.27	98.96	96.08	99.48	97.50
	XGBoost	98.10	95.78	99.15	95.26	99.57	97.17
	LR	98.13	95.76	97.44	96.87	98.75	97.15
Word2Vec (SkipGram)	RF	98.38	96.43	99.52	95.77	99.76	97.61
	DT	97.88	95.25	97.43	96.24	98.72	96.83
	SVM	97.35	94.21	99.12	93.31	99.55	96.13
	XGBoost	97.85	95.22	98.42	95.25	99.20	96.81
	LR	98.70	97.08	98.57	97.53	99.28	98.05
FastText (CBOW)	RF	98.39	96.48	99.18	96.23	99.57	97.68
	DT	97.44	94.33	96.79	95.76	98.33	96.27
	SVM	96.82	93.01	96.79	94.07	98.31	95.41
	XGBoost	97.61	94.77	98.25	94.93	99.08	96.56
	LR	97.30	94.04	97.29	94.92	98.58	96.09
FastText (SkipGram)	RF	98.59	96.87	99.56	96.31	99.78	97.90
	DT	97.77	94.98	96.77	96.52	98.40	96.64
	SVM	**98.82**	93.09	98.82	92.16	99.39	95.38
	XGBoost	98.11	95.82	99.01	95.47	99.50	97.21
	LR	98.50	96.63	98.05	97.45	99.03	97.75

training and 8222 emails are used for testing. The total size of Dataset-3 is 27256, out of which 19079 emails were used for training and 8177 emails are used for testing the proposed system. The list of evaluation matrix rates computed from the confusion matrix for the binary classifier is tabulated in Table 12.

Also, it may be noted that the training time for different algorithms ranges from 67.15 seconds (TF-IDF) to 425.02 seconds (Word2Vec-SkipGram) for the vector size of 200. The testing time for different algorithms ranges from 50.44 seconds (TF-IDF) to 328.56 seconds (Word2Vec-SkipGram) for the vector size of 200.

Table 11 Confusion Matrix (a). Dataset-1, (b). Dataset-2, (c). Dataset-3

	True Positive	True Negative
False Positive	**2741**	**1**
False Negative	**22**	**1865**

(a)

	TP	TN
FP	**2701**	**22**
FN	**28**	**5471**

(b)

	TP	TN
FP	2652	54
FN	13	5458

(c)

Table 12 Confusion matrix computed rates

Measure (%)	Dataset-1	Dataset-2	Dataset-3
Accuracy	99.50	99.39	99.18
MCC	98.97	98.62	98.15
Precision	99.96	99.19	98.00
TPR	99.20	98.97	99.51
TNR	99.95	99.60	99.02
F-Score	99.58	99.08	98.75
FPR	0.05	0.40	0.98

5.8 Performance of Individual Features

In the proposed work only four header labels are used for the classification of emails. The performance of individual labels is tabulated in Table 14. The Return-Path achieves an accuracy of 99.34% with FastText and RF. The second-highest performing feature is Subject with an accuracy of 98.71% for Word2Vec SkipGram with RF. The other two features ”From” and ”Message-Id” achieve the highest accuracy of 97.75% and 95.23% respectively. Hence it may be observed that all the four selected features are very efficient in classifying the emails.

5.9 Comparison Study

5.9.1 Using Common datasets

In this section, a comparison of our work with other existing works executed on the same open-source dataset is given. The results of other existing works are directly taken from the respective papers and tabulated in Table 15 along with results of our proposed work using the same repository. According to the summary Table 15, some of the existing works used multiple hybrid features and achieved less accuracy compared to our proposed work by using only four header heuristics. The publicly available datasets used by different researchers are variable in size. Similarly, we have used the same publicly

Table 13 Performance Evaluation with Dataset-3

Word Embeddings	Algorithm	Acc	MCC	Precision	TPR	TNR	F-Score
TF-IDF	RF	99.12	97.97	97.86	99.38	99.00	98.61
	DT	98.70	97.03	98.28	97.68	99.19	97.98
	SVM	97.60	94.52	92.62	99.87	96.64	96.11
	XGBoost	98.52	96.59	97.17	98.18	98.67	97.67
	LR	98.33	96.17	95.76	99.01	98.03	97.36
Count Vectorizer	RF	**99.18**	98.11	98.05	99.38	99.09	98.71
	DT	**98.79**	97.22	98.20	98.01	99.15	98.11
	SVM	97.63	94.58	92.59	100	96.63	96.15
	XGBoost	**98.73**	97.07	97.63	98.38	98.89	98.00
	LR	98.49	96.55	95.64	99.64	97.99	97.60
Word2Vec (CBOW)	RF	98.86	97.39	98.51	97.95	99.29	98.23
	DT	98.60	96.82	98.59	97.11	99.33	97.84
	SVM	**98.51**	96.57	96.53	98.79	98.38	97.65
	XGBoost	98.47	96.49	97.49	97.75	98.81	97.62
	LR	98.24	95.98	97.90	96.65	99.00	97.28
Word2Vec (SkipGram)	RF	99.02	97.78	98.51	98.51	99.27	98.51
	DT	98.49	96.61	98.48	97.00	99.24	97.73
	SVM	98.27	96.12	94.84	99.92	97.52	97.31
	XGBoost	98.59	96.82	98.03	97.71	99.03	97.87
	LR	**98.78**	97.24	98.40	97.89	99.21	98.15
FastText (CBOW)	RF	58.86	97.40	96.97	99.49	98.57	98.21
	DT	98.64	96.89	97.54	98.24	98.83	97.89
	SVM	97.75	94.88	93.10	99.92	96.82	96.39
	XGBoost	98.26	96.02	96.78	97.82	98.47	97.29
	LR	98.31	96.13	95.72	99.02	97.99	97.34
FastText (SkipGram)	RF	98.84	97.38	96.79	99.69	98.43	98.22
	DT	98.53	96.69	97.49	98.07	98.76	97.78
	SVM	97.48	94.37	92.41	100	96.36	96.05
	XGBoost	98.46	96.52	96.09	99.24	98.09	97.64
	LR	98.47	96.55	96.54	98.83	98.30	97.67

available dataset of size 9135 phishing emails and 6295 legitimate emails from the open-source corpus. The proposed model outperformed all other existing works by achieving an accuracy of 99.50%.

5.9.2 Using Common methods

The Comparison of the proposed work with other word embedding based techniques is given in Table 16. It may be noted that the proposed technique outperforms the other techniques with accuracies of 99.50%, 99.39%, and

Table 14 Performance of individual features

Feature	Classifier	TF-IDF Accuracy(%)	CV Accuracy(%)	W2V-CBOW Accuracy(%)	W2V-SG Accuracy(%)	FT-CBOW Accuracy(%)	FT-SG Accuracy(%)
From	RF	86.73	86.73	97.72	97.75	94.35	94.11
	DT	86.68	86.60	97.75	97.72	94.35	94.11
	SVM	85.07	83.63	94.30	93.98	74.88	85.50
	XG-Boost	85.74	85.79	96.49	96.60	93.87	93.42
	LR	86.09	86.09	96.01	96.30	87.77	89.94
Message-ID	RF	85.07	85.10	94.29	95.23	90.89	89.92
	DT	85.02	84.99	94.18	95.07	90.91	89.92
	SVM	80.65	79.69	75.54	75.86	53.97	54.37
	XG-Boost	84.91	84.86	93.27	93.83	90.11	89.52
	LR	84.40	84.35	86.33	91.48	51.11	73.02
Return-Path	**RF**	**97.89**	**97.89**	**99.33**	**99.34**	**98.63**	**98.61**
	DT	97.83	97.86	99.30	99.51	98.63	98.58
	SVM	96.58	96.12	95.94	96.05	86.84	87.42
	XG-Boost	97.60	97.51	98.85	99.19	98.50	98.26
	LR	97.81	97.75	98.40	98.42	93.61	95.11
Subject	**RF**	**96.29**	**96.20**	**98.68**	**98.71**	**98.16**	**98.62**
	DT	96.26	96.17	97.46	97.46	97.40	97.81
	SVM	94.40	93.67	96.26	96.93	92.68	94.86
	XG-Boost	94.69	94.83	96.58	96.87	96.24	97.34
	LR	95.18	95.12	97.95	98.36	93.58	96.23

Table 15 Summary of the works implemented on the publicly available dataset.

Author	**No.of Features**	**Type of features**	**Dataset size**	**Accuracy (%)**
Smadi et al [27]	50	Hybrid	Phishing-4559 Ham-4559	98.63
Islam and Abawajy [40]	21	Hybrid	Unknown	97
Almomani et al [46]	21	Hybrid	Phishing-4300 Ham-6000	99
Khonji et al [41]	47	Hybrid	Phishing-4116 Ham-4150	99.37
Gansterer and P¨olz [23]	30	Hybrid	Phishing-5000 Ham-5000	97
Toolan and Carthy [19]	5	Hybrid	Phishing-4202 Ham-4563	99.31
Hamid and Abawajy [44]	7	Hybrid	Total-4594	96
Toolan and Carthy [21]	22	Hybrid	Phishing-4202 Ham-4563	97
Fette et al [18]	10	Hybrid	Phishing-860 Ham-6950	96
Proposed work	**4**	**Header**	**Phishing-9135 Ham-6295**	**99.50**

Table 16 Summary of the works used Word Embedding and Natural language processing

Author	Features	Dataset (s)	Dataset size	Accuracy (%)	Precision (%)	F-score (%)
Nguyen et al [28]	Hybrid	IWSPA-AP 2018	Legit:4082 Phish:503	–	99.0	99.1
Bagui et al [32]	Hybrid	Private	Legit:14950 Phish:3416	98.89	–	–
Castillo et al [33]	Body	Enron, APWG, and Non-public	Legit:84111 Phish:30776	95.68	–	–
Ra et al [34]	Body	IWSPA-AP 2018	Legit:5088 Phish:612	99.1	90.59	93.07
Hiransha et al [35]	Body	IWSPA-AP 2018	Legit:5088 Phish:612	96.8	–	–
Harikrishnan et al [36]	Hybrid	IWSPA-AP 2018	Legit:5088 Phish:612	90.29	92.5	94.6
Verma et al [37]	Hybrid	PhishCatch	Legit:1000 Phish:2000	97	–	–
Gutierrez et al [38]	Hybrid	Purdue university's Sophos	Legit:158000 Phish:425870	96.5	–	–
Valecha et al [39]	Hybrid	Enron Millersmile	Legit:19153 Phish:17902	96.52	98.53	96.31
Proposed Model	**Header**	**Dataset – 1**	Legit:6295 Phish:9135	**99.50**	**99.96**	**99.58**
		Dataset – 2	Legit:18270 Phish:9135	**99.39**	**99.19**	**99.08**
		Dataset – 3	Legit:18270 Phish:8986	**99.18**	**98.00**	**98.35**

99.18% for different datasets. Also, the proposed technique achieved the accuracies with only four header features of the email.

6 Conclusion and future work

In this paper, we have presented novel techniques for phishing email detection using word embedding and machine learning classifiers. The presented techniques use only four email header features for the classification. The FastText-CBOW algorithm with RF classification achieves the highest accuracy of 99.50% with the publicly available datasets. Also, the RF classifier consistently performed well with all the word embedding algorithms. Hence, the RF classifier is more suitable for the classification of phishing emails with word embedding techniques.

As future work, we would like to extend this work with additional heuristics with the body of the email using word embedding and machine learning techniques. Also, the performance of these features may be evaluated using deep learning techniques.

References

[1] M Somesha, Alwyn Roshan Pais, Routhu Srinivasa Rao, and Vikram Singh Rathour. Efficient deep learning techniques for the detection of phishing websites. *Sādhanā*, 45(1):1–18, 2020.

[2] Riittakerttu Kaltiala-Heino, Tomi Lintonen, and Arja Rimpelä. Internet addiction? potentially problematic use of the internet in a population of 12–18 year-old adolescents. *Addiction Research & Theory*, 12(1):89–96, 2004.

[3] Martha Shaw and Donald W Black. Internet addiction. *CNS drugs*, 22(5):353–365, 2008.

[4] D J Kuss, M D Griffiths, Laurent Karila, and Jöel Billieux. Internet addiction: A systematic review of epidemiological research for the last decade. *Current pharmaceutical design*, 20(25):4026–4052, 2014.

[5] Tracii Ryan, Andrea Chester, John Reece, and Sophia Xenos. The uses and abuses of facebook: A review of facebook addiction. *Journal of behavioral addictions*, 3(3):133–148, 2014.

[6] Ying-ying Zhang, Jian-ji Chen, Hai Ye, and Lupe Volantin. Psychological effects of cognitive behavioral therapy on internet addiction in adolescents: A systematic review protocol. *Medicine*, 99(4), 2020.

[7] APWG. Apwg 2019 phishing activity trends reports, third quarter 2019. https://docs.apwg.org//reports/apwg_tre-nds_report_q3_2019.pdf, 2019. Accessed: 2019-11-04.

[8] APWG. Apwg 2019 phishing activity trends reports, fourth quarter 2019. https://docs.apwg.org/reports/apwg_tre-nds_report_q4_2019.pdf, 2020. Accessed: 2020-02-24.

[9] APWG. Apwg 2018 phishing attack trends reports, fourth quarter 2018. https://docs.apwg.org/reports/apwg_tre-nds_report_q4_2018.pdf, 2019. Accessed: 2019-03-04.

[10] APWG. Apwg 2015 phishing activity trends reports, first-to-third quarter 2015. https://docs.apwg.org//reports/apwg_tre-nds_report_q1-q3_2015.pdf, 2015. Accessed: 2015-12-23.

[11] APWG. Apwg 2015 phishing activity trends reports, fourth quarter 2015. https://docs.apwg.org//reports/apwg_tre-nds_report_q4_2015.pdf, 2015. Accessed: 2016-03-22.

[12] Mimecast. Mimecast – the state of the email security report 2019. https://www.mimecast.com/globalassets/documents/ebook/state-of-email-security-2019.pdf?v=2, 2019.

[13] Kaspersky. Spam and phishing in q3 2019. https://securelist.com/spam-report-q3-2019/95177/, 2019.

[14] Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey. *arXiv preprint arXiv:1901.09069*, 2019.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[17] Ammar Almomani, Brij B Gupta, Samer Atawneh, Andrew Meulenberg, and Eman Almomani. A survey of phishing email filtering techniques. *IEEE communications surveys & tutorials*, 15(4):2070–2090, 2013.

[18] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th international conference on World Wide Web*, pages 649–656, 2007.

[19] Fergus Toolan and Joe Carthy. Phishing detection using classifier ensembles. In *2009 eCrime researchers summit*, pages 1–9. IEEE, 2009.

[20] André Bergholz, Jan De Beer, Sebastian Glahn, Marie-Francine Moens, Gerhard Paaß, and Siehyun Strobel. New filtering approaches for phishing email. *Journal of computer security*, 18(1):7–35, 2010.

[21] Fergus Toolan and Joe Carthy. Feature selection for spam and phishing detection. In *2010 eCrime Researchers Summit*, pages 1–12. IEEE, 2010.

[22] Mahmoud Khonji, Andrew Jones, and Youssef Iraqi. A study of feature subset evaluators and feature subset searching methods for phishing classification. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, pages 135–144, 2011.

[23] Wilfried N Gansterer and David Pölz. E-mail classification for phishing defense. In *European Conference on Information Retrieval*, pages 449–460. Springer, 2009.

[24] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. Distributed phishing detection by applying variable selection using bayesian additive regression trees. In *2009 IEEE International Conference on Communications*, pages 1–5. IEEE, 2009.

[25] Madhusudhanan Chandrasekaran, Krishnan Narayanan, and Shambhu Upadhyaya. Phishing email detection based on structural properties. In *NYS cyber security conference*, volume 3. Albany, New York, 2006.

[26] Aviad Cohen, Nir Nissim, and Yuval Elovici. Novel set of general descriptive features for enhanced detection of malicious emails using machine learning methods. *Expert Systems with Applications*, 110:143–169, 2018.

[27] Sami Smadi, Nauman Aslam, and Li Zhang. Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. *Decision Support Systems*, 107:88–102, 2018.

[28] Minh Nguyen, Toan Nguyen, and Thien Huu Nguyen. A deep learning model with hierarchical lstms and supervised attention for anti-phishing. *arXiv preprint arXiv:1805.01554*, 2018.

[29] Qi Li, Mingyu Cheng, Junfeng Wang, and Bowen Sun. Lstm based phishing detection for big email data. *IEEE Transactions on Big Data*, 2020.

[30] Areej Alhogail and Afrah Alsabih. Applying machine learning and natural language processing to detect phishing email. *Computers & Security*, 110:102414, 2021.

[31] Yong Fang, Cheng Zhang, Cheng Huang, Liang Liu, and Yue Yang. Phishing email detection using improved rcnn model with multilevel vectors and attention mechanism. *IEEE Access*, 7:56329–56340, 2019.

[32] Sikha Bagui, Debarghya Nandi, Subhash Bagui, and Robert Jamie White. Classifying phishing email using machine learning and deep learning. In *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–2. IEEE, 2019.

[33] Esteban Castillo, Sreekar Dhaduvai, Peng Liu, Kartik-Singh Thakur, Adam Dalton, and Tomek Strzalkowski. Email threat detection using distinct neural network approaches. In *Proceedings for the First International Workshop on Social Threats in Online Conversations: Understanding and Management*, pages 48–55, 2020.

[34] Vinayakumar Ra, Barathi Ganesh HBa, Anand Kumar Ma, Soman KPa, Prabaharan Poornachandran, and A Verma. Deepanti-phishnet: Applying deep neural networks for phishing email detection. In *Proc.*

1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA), pages 1–11. Tempe, AZ, USA, 2018.

[35] M Hiransha, Nidhin A Unnithan, R Vinayakumar, K Soman, and ADR Verma. Deep learning based phishing e-mail detection. In *Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA)*. Tempe, AZ, USA, 2018.

[36] NB Harikrishnan, R Vinayakumar, and KP Soman. A machine learning approach towards phishing email detection. In *Proceedings of the Anti-Phishing Pilot at ACM International Workshop on Security and Privacy Analytics (IWSPA AP)*, volume 2013, pages 455–468, 2018.

[37] Rakesh Verma, Narasimha Shashidhar, and Nabil Hossain. Detecting phishing emails the natural language way. In *European Symposium on Research in Computer Security*, pages 824–841. Springer, 2012.

[38] Christopher N Gutierrez, Taegyu Kim, Raffaele Della Corte, Jeffrey Avery, Dan Goldwasser, Marcello Cinque, and Saurabh Bagchi. Learning from the ones that got away: Detecting new forms of phishing attacks. *IEEE Transactions on Dependable and Secure Computing*, 15(6):988–1001, 2018.

[39] Rohit Valecha, Pranali Mandaokar, and H Raghav Rao. Phishing email detection using persuasion cues. *IEEE Transactions on Dependable and Secure Computing*, 2021.

[40] Rafiqul Islam and Jemal Abawajy. A multi-tier phishing detection and filtering approach. *Journal of Network and Computer Applications*, 36(1):324–335, 2013.

[41] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Enhancing phishing e-mail classifiers: A lexical url analysis approach. *International Journal for Information Security Research (IJISR)*, 2(1/2):40, 2012.

[42] Venkatesh Ramanathan and Harry Wechsler. phishgillnet—phishing detection methodology using probabilistic latent semantic analysis, adaboost, and co-training. *EURASIP Journal on Information Security*, 2012(1):1–22, 2012.

[43] Liping Ma, Bahadorrezda Ofoghi, Paul Watters, and Simon Brown. Detecting phishing emails using hybrid features. In *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, pages 493–497. IEEE, 2009.

[44] Isredza Rahmi A Hamid and Jemal Abawajy. Hybrid feature selection for phishing email detection. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 266–275. Springer, 2011.

[45] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[46] A Almomani, TC Wan, A Manasrah, A Altaher, M Baklizi, and S Ramadass. An enhanced online phishing e-mail detection framework based on evolving connectionist system. *International Journal of Innovative Computing, Information and Control (IJICIC)*, 9(3):169–175, 2013.

Biographies

Somesha M. is an Assistant Professor & HOD, Department of Computer Science and Engineering, Government Engineering College, Karwar, Karnataka, India. He completed his B.E.(CSE) from Bangalore University, India and M.Tech.(CNE) from NIE Mysore, India. He is currently pursuing Ph.D. from the Department of Computer Science & Engineering, National Institute of Technology Karnataka (NITK), Surathkal. His areas of interest include Information Security, Computer Networks, and Cyber security.

Alwyn R. Pais is an Associate Professor and Research Guide, Department of Computer Science and Engineering, National Institute of Technology Karnataka (NITK). He completed his B.Tech.(CSE) from Mangalore University, India, M.Tech. (CSE) from IIT Bombay, India, and Ph.D. (CSE) in NITK, Surthkal. His area of interest includes Information Security, Image Processing and Computer Vision.