information

Article

Malicious Text Identification: Deep Learning from Public Comments and Emails

Asma Baccouche *,†, **Sadaf Ahmed** *,†, **Daniel Sierra-Sosa** and **Adel Elmaghraby**

Department of Computer Science and Engineering, University of Louisville, Louisville, KY 40292, USA; d.sierrasosa@louisville.edu (D.S.-S.); adel.elmaghraby@louisville.edu (A.E.)

* Correspondence: asma.baccouche@louisville.edu (A.B.); sadaf.ahmed@louisville.edu (S.A.)

† These authors contributed equally to this work.

Received: 16 April 2020; Accepted: 5 June 2020; Published: 10 June 2020

Abstract: Identifying internet spam has been a challenging problem for decades. Several solutions have succeeded to detect spam comments in social media or fraudulent emails. However, an adequate strategy for filtering messages is difficult to achieve, as these messages resemble real communications. From the Natural Language Processing (NLP) perspective, Deep Learning models are a good alternative for classifying text after being preprocessed. In particular, Long Short-Term Memory (LSTM) networks are one of the models that perform well for the binary and multi-label text classification problems. In this paper, an approach merging two different data sources, one intended for Spam in social media posts and the other for Fraud classification in emails, is presented. We designed a multi-label LSTM model and trained it on the joint datasets including text with common bigrams, extracted from each independent dataset. The experiment results show that our proposed model is capable of identifying malicious text regardless of the source. The LSTM model trained with the merged dataset outperforms the models trained independently on each dataset.

Keywords: spam text filter; text mining; content-based classification; natural language processing; multi-label classification; LSTM

1. Introduction

Spam is a trending internet dysfunction that has been affecting social networks and websites [1,2]. Replying with out-of-context comments on social media is, in general, a sign of an attempt to induce users to open malicious links or disturb the reader with marketing. Information phishing was initially used for marketing, but it degenerated into harmful internet interactions that lead users into serious security threats using means such as emails, comments, blogs, and messages [3]. Detecting spam has several purposes including security and creating better user experiences on the communication platforms [4]. Several effective tools have been used for spam filtering that relies on techniques such as heuristic rules and logistic regression combination [5], and baseline classifiers with hybrid ensemble of features selection [6].

Phishing is common in spam and fraud communications. These communications include emails, social media, and video streaming services, among others. Filtering these malicious messages could be as simple as a binary text classification aiming to determine whether a text is harmful or legitimate. In many cases, text classification requires transforming the unstructured text into a standardized numerical representation for ease of analysis [7–9]. Usually, texts are projected by word embedding models. The most frequently used are known as Word2vec models that work by preserving semantic meaning between words [10–12].

Deep Learning models have historically proven to be effective for email spam classification, provided their adaptable nature and capacity to maximize the potential of modern hardware and computational limits. Deep learning techniques show great promise in the advancement of spam filtering [13]. Various different architectures such as Convolutional Neural Networks (CNN), Multi-Layer Perceptron (MLP) and Long Short Term Memory (LSTM) have been successfully employed for this purpose [14].

The problem that is the focus of this work is to develop a robust and reliable spam detection model which can determine a given comment or email as spam or ham. In this paper, we have focused on identifying YouTube spam comments and Nigerian fraudulent emails, by designing a binary text classification model based on LSTM architecture with pre-trained word embeddings Word2vec model. The spam text was extracted from comments on YouTube videos [15]. Aside from structural and compositional differences between the text from the two datasets, for the purposes of this paper, we considered phishing as a type of fraud with the intent to illegally exploit a users financial or personal data. We labeled a text as "spam" if it contained any commercial links and words that deviated from the context of the videos. Fraudulent text, for training and testing, was derived from a dataset of phishing emails [16]. For the rest of this paper, we considered a text "fraud" if it shared structural and compositional similarities with the Nigerian fraudulent emails along with phishing intent. We considered a text "spam" if it shared structural and compositional similarities with the spam labeled YouTube comments. "Spam" text did not necessarily possess phishing intent.

The framework is extended to present a joint LSTM architecture to conduct a multi-label classification. The joint dataset used is the data collection including associations of two words (bigrams) that were present in both datasets. This framework constitutes a phishing detection tool, based on multi-source text classification. In addition, our contribution has the aim a more comprehensive classification model that predicts the nature of similar domain texts (i.e., Harmful or Normal) and its malicious style (i.e., Spam Comments or Fraud Emails).

This paper is organized as follows. Section 2 presents a summary of state-of-the-art techniques on phishing detection and text classification methods. Section 3 introduces the background details of the implemented research methods. Section 4 details the methodology of our proposed framework, and explains the setup and the preprocessing for the conducted experiments. The results and analysis of our approach are discussed in Section 5. Finally, Section 6 concludes the paper.

2. Background

This section introduces recent research on text analysis techniques for NLP tasks and phishing detection. We specifically highlight advances in information security, text classification, and neural networks and their applications in malicious text filtering and multi-domain learning.

2.1. Security Information

In recent years, the user's information has become an invaluable tool in explaining behavior [17], opinion and emotional influence [18,19] towards micro-blogging websites and e-commerce networks. The exponential growth of shared information on social networks has given rise to the problem of user security and privacy. Many studies have researched this issue in order to contribute to technical solutions against these cyber threats [20]. From the perspective of users, public information on popular websites and social networks need more credibility and validation because it can be spam, misleading or inaccurate. Therefore, several mechanisms have been introduced to analyze the relevance of the information before being considered for important decisions in varying domains [21].

Phishing detection has been a challenging problem that treats several sub-problems such as fraudulent inbox emails or spam and irrelevant comments in social media blogs. For several years, many tools and algorithms were implemented for specific domains and for different purposes, such as financial gain, identity theft, identity trafficking, industrial espionage, malware distribution, and password harvesting. A study proposed PhishCatch algorithm for phishing emails detecting [22]. It is a heuristic-based rule algorithm that alerts the user about identified suspicious links, achieving an 80% detection rate of phishing emails. The work of [23] also implemented a recognition model for

URL phishing sites in mobile messages, which is based on neural networks. It achieved an accuracy rate of 98.2% and a recall rate of 96.9%.

2.2. Text Classification and Malicious Text Filtering

Text filtering can be achieved by classification. Labeling texts allows for the identification of text groups and provides differential handling. For example, in phishing identification, the texts could be classified into harmful (i.e., spam, fraud, etc.) or harmless (i.e., non-spam, normal, non-fraud, etc.). As a data mining problem, several machine learning algorithms were suggested to effectively solve this problem. In [24], a semantic-based Decision Tree classifier was proposed to help to classify a public Chinese spam detection. The work of [25,26] introduced a spam detection tool using an ensemble model of traditional classifiers (i.e., KNN, Naïve Bayes, SVM, etc.) trained independently on five subsets of datasets and achieved accuracy scores that range from 90.64% to 94.9%.

Moreover, the spam detection problem was approached with string matching algorithms such as the longest common string, bigram, and Jaro-Winkler [27]. These aimed to find similarities among spam phrases and effectively assign the text labels compared to the well-known classifiers. Moreover, features extraction and dimensionality reduction methods were proven in [28,29] to highlight the performance of text classification for phishing detection in emails.

2.3. Neural Networks for Text Filtering

Neural network models have been employed for a variety of NLP tasks [30–32]. As text representation has become a challenging problem, the neural network presented an efficient way of modeling written language by a human into a continuous representation of words, phrases, and paragraphs [33,34]. The application helped in preserving the semantic meaning of a document and projecting the words into a continuous vector space. In this context, a convolutional neural network model (CNN) for document-level representation was suggested to extract features from a collection of opinion spam and then a gated recurrent neural network model (GRU) was used for deceptive spam detection [35].

The CNN architecture was also implemented for multi-label classification of short texts and showed promising results using domain-specific word embeddings [36]. Nevertheless, email classification algorithms based on deep neural networks (DNN) outperformed the results of the limited statistical-based classification algorithm [22]. The work of [37] presented a new implementation of LSTM for document-level sentiment classification and proved the effectiveness of the maximum number of texts in each document for the running time optimization. Other neural network models were also implemented such as a multi-layer perceptron neural network for spam detecting [38] and Bi-Gated Recurrent Unit for hot news classification [39].

Dhingra and Mittal [40] approached the challenge of spam classification on live tweets from Twitter via an API through the implementation of a Multi-Layer Perceptron (MLP). Raw text extracted from tweets was preprocessed by removing special characters, tokenization, and stop word removal, stemming and word separation. Additionally, content-based features such as unique hashtags and URLs were extracted for training. Naïve Bayes was implemented for comparison of accuracy, precision and recall values with the deep learning model. Naïve Bayes produced an accuracy of 79%, a precision of 71% and a recall of 75%, compared with MLP results of 82%, 75%, and 81%, respectively. The MLP model outperformed the Naïve Bayes classifier for all three metrics.

2.4. Multi-Domain Learning

Text mining problems usually deal with a specific domain and focus on enhancing the generalization of the model toward the domain of the dataset. However, with the growing availability of information online, new domains have been introduced seeking the adaptability of machine learning solutions. For example, the work presented in [41] proposed a domain adaptation algorithm for classifying text reviews using maximum entropy and point-wise mutual information, aiming to transfer knowledge without training on a labeled dataset. In [42], multi-domain learning was enhanced by using an adversarial training methodology to prove the effectiveness of feature sharing between different domains. The use of stacked auto-encoders was also suggested as a deep learning-based approach for solving the domain adaptation problem by providing a new representation for the domains [43]. We have selected some literature from which to tabulate results of experiments and methods that were used for either Natural Language Processing or similar tasks in Table 1.

Table 1. Literature Review.

Reference	**Applied Method**	**Task**	**Results**	**Application**
Dhingra & Mittal [40]	Multi-Layer Perceptron	Spam classification in tweets from Twitter. MLP had not been applied to tweets for this purpose before	82% accuracy, 75% precision, 81% recall	MLP outperformed Naïve Bayes in classification of tweets as spam
Jain et al. [14]	CNN-LSTM based architecture	Detecting spam in noisy and short-text messages such as those found in social media	95% accuracy, 95% precision, 98% recall, 97% F1-score in tweet spam detection	High performing method for spam detection in short texts using machine learning
Ding et al. [5]	Heuristic Rule and Logistic Regression	Detecting phishing websites based on URLs	98% accuracy, 98% recall, 98% F1-score, 97% precision	A method to detect phishing websites through obfuscation techniques processing
Chiew et al. [6]	Random Forest	Preprocessing data to perform spam classification in emails	96.17% accuracy	Hybrid Ensemble Feature Selection for preprocessing data that works best with Random Forest classifier
Hua [44]	BERT ensemble	Classifying propaganda texts at the sentence level	63% precision, 69% recall, 66% F1-score	A set of illustrative experiments to understand the performance of BERT on propaganda classification
Aggarwal et al. [45]	BERT	Classifying text articles as fake news	97% accuracy	A method for classifying fake news for long-text articles (avg. 731 words)

3. Research Methods

In our efforts to contribute to the state of the art in classification models for detection of spam comments and fraudulent emails, neural network models were tested as described in this section. Neural network models (including Deep Learning) were first inspired by the human brain and are applied in many fields. In particular, several methods were designed for NLP applications, to learn complex motifs from large datasets [46]. The following paragraphs describe the word embeddings for text representation and two basic text classification deep learning models: RNN and LSTM.

3.1. Word Embeddings

In NLP applications, the use of conventional features like term frequency-inverse document frequency (TF-IDF) was proven to be less efficient than word embeddings [47]. Therefore, word embeddings were suggested for text representation by converting words into real-valued vectors. The vectorization is made after training neural networks on a text corpus. Words, as discrete atomic symbols, require a continuous space projection, where semantically related words have similar and homogeneous vector representation. Word embeddings preserve the semantic meaning and the syntax of the words based on their context in the documents. In many NLP tasks such as machine translation, speech recognition, and text classification, word embeddings were applied with pre-trained neural network models.

Word2vec is a word embedding model that offers two variations: the Continuous Bag-of-Words model (CBOW) [48] and the Skip-Gram model [49]. The CBOW model works to predict the current target word from a window of surrounding source-context words. The skip-gram model, however, weights the surrounding context words more heavily by predicting the source-context words from the target words. Word vectors pre-trained on datasets from several domains are available and they are created with unsupervised learning on a large text corpus.

3.2. Basic Text Classification Architectures

Text data is naturally sequential, thus it requires an architecture's design that makes use of sequential data, because sequential information is clearly important while conducting a text classification task. The dependencies between words and symbols in the input sequence is important as the meaning can be misinterpreted or the order of words can be incorrect if sequential information is not used. In this context, Recurrent Neural Networks (RNN) and Long-Short Term Memory (LSTM) networks are suggested as the most widely used techniques for text classification. These models are designed with a feed-forward and backward propagation structure, which makes use of sequential information and contains directed loops that represent the propagation of activation to future input. The traditional neural network is composed of independent input and output; however, this structure had limitations for sequentially dependent data. A deep learning model is composed of multiple hidden layers with different linear and non-linear activation functions. Every model is a unique manifestation of the general neural network framework. Most of them incorporate an output layer that is generated with a dense layer, commonly known as a fully-connected layer, which predicts the final class labels. Through every batch of the training dataset, the model updates parameters and optimizes the error using some advanced adaptive techniques. Since the text dataset has a sequential dependency, RNN models are highly robust and suitable for learning from these datasets, efficiently performing classification tasks. The correlation between the front and back of the data is connected through a sequence of nodes that can learn from word vectors.
The RNN model follows this set of steps:

- The input of the hidden layer at time t: W_t
- The output of the step t: where f is the activation function such as sigmoid or ReLu
- Final output: where g is the output activation function such as softmax

The traditional RNN model cannot capture long-distance dependent information between words and output and thus, the gradient descendent can dramatically decrease until reaching zero. The LSTM model suggests solving the problem of gradient vanishing by introducing an input gate, i, an output gate, o, a forget gate, f, and a memory cell. The forget gate decides what information to discard in the memory cell. As Equation (1) explains, the LSTM cell at time, t, takes three inputs: x_t and two previous outputs h_{t-1} and C_{t-1}. The forget gate is a calculated value between 0 and 1.

$$f_t = g(W_f x_t + U_f h_{t-1} + V_f C_{t-1} + b_f) \tag{1}$$

The LSTM cell connects the input, x_t, and the forget gate, f, through the weight of the previously hidden layer, h_{t-1}. V_f connects the weight of the previous state of the memory cell, C_{t-1}. U_f connects h_{t-1} and forget gate, f. The equation uses a bias term, b_f, and a non-linear transformation, g, which must be either ReLu or sigmoid. The input gate, i, updates the memory cell at time, t, as explained in the following Equations (2)–(4).

$$i_t = g(W_i x_t + U_i h_{t-1} + V_i C_{t-1} + b_i) \tag{2}$$

$$n_t = tanh(W_c x_t + U_c h_{t-1} + V_c C_{t-1} + b_c) \tag{3}$$

$$C_t = f_t C_{t-1} + i_t n_t \tag{4}$$

The input, x_t, and the input gate, i_t, are connected through the weight, W_i. U_i also connects the input gate, i_t, and h_{t-1}. C_{t-1} is connected with it through V_i. W_c connects x_t with n_t, which is connected with h_t through U_c. The previous equations use bias terms, b_i, b_c. The following Equations (5) and (6) explain the computation of the LSTM output gate.

$$o_t = g(W_o x_t + U_o h_{t-1} + V_o C_{t-1} + b_o) \tag{5}$$

$$h_t = o_t tanh(C_t) \tag{6}$$

The input, x_t, and o_t are connected through W_o. h_{t-1} and o_t are connected through a weight, U_o. C_{t-1} and o_t are also connected by U_o. The previous equations also use a bias term, b_o.

4. Methodology and Experimental Setup

This section describes our training datasets and presents the preprocessing and parameterization for conducting the experiments.

4.1. Datasets

A major part of our contribution of a robust spam comment and fraudulent email classifier is the preprocessing and combination of the data and selected datasets. The proposed models are trained over two datasets: one containing samples of spam comments and the other containing fraudulent emails. The spam samples are derived from a collection of user comments on YouTube videos for five popular music artists, extracted from [15,50]. The comments were labeled either "spam" or "non-spam". The files are combined into 2394 comments, evenly distributed between the two labels. The average length of the comments is 11 words, with short text format containing words, symbols, hyperlinks, and punctuation.

The fraud dataset is formed from a publicly available collection of emails, known as the "419 Fraud" or "Nigerian Letter" dataset, defined at [51]. It is formed by 11,000 body of emails that are nearly balanced between "fraud" or "non-fraud" labels.

4.2. Data Preparation

Raw data is cleaned and prepared prior to input for the classification training models. We applied the following steps to remove unrelated characters and symbols for every comment and email. The following list outlines the implemented methodologies to obtain a preprocessed dataset [52]:

- Lower-casing all the words because our models are not case-sensitive
- Tokenizing the texts using NLTK Python Library
- Removing the stop words using an enhanced version of NLTK English corpus
- Removing URLs and links to websites that start with www.* or http://*
- Removing repeating characters from words
- Removing numbers and punctuation marks
- Removing strange characters that were utilized from the keyboard
- Word lemmatization, semantic reconstruction of misspelled words, mapping the Emojis to their expression, and replacing the slang by their original meaning

Each class was balanced in both datasets through random under-sampling. The training dataset for the joint model was derived from records containing text that is common in both Spam and Fraud datasets. In order to construct this, we used n-gram (i.e., association of n linked words) mapping across the two different datasets. We extracted two lists of bigrams (i.e., association of two words) from both the Spam and Fraud datasets and any duplicates were thrown out.

Next, the intersection of bigrams from both datasets was marked and each bigram was tracked using the text ID. We then retrieved the records containing only common bigrams and compared it to all the existing bigrams in the texts.

Each record from either Spam or Fraud dataset that contains a common bigram is assigned a new "artificial" label, which defines a class from both datasets (i.e., if the bigram originally belonged to the Spam dataset, an "artificial" Fraud label is assigned to it). The artificial label is assigned based on the class frequency of the common bigram in the dataset opposite to which the original text belongs. For example, if a record with a true spam label has a common bigram occurring in both the fraud and non-fraud subsets, we will assign it a "fraud" label if it occurs more frequently in the fraud subset than in the non-fraud subset. The resulting dataset will have records containing only common text between the two datasets with a true label from the parent dataset and an artificial label from the opposite dataset.

The following equations detail the joint dataset generation, where bigrams were extracted from the original datasets and compared with a list of the common bigrams in each dataset. Records were retrieved through their original Id if the record's bigrams are present in both the list of common bigrams and the bigrams list for each class. Equations (7)–(10) describe the retrieval of the records for each independent class: fraud, non-fraud, spam and non-spam.

$$record_{\text{fraud}} = \{x \subseteq cB, y \subseteq fraud;\ \text{substring(x, y)}\} \tag{7}$$

$$record_{\text{spam}} = \{x \subseteq cB, y \subseteq spam;\ \text{substring(x, y)}\} \tag{8}$$

$$record_{\text{non\_fraud}} = \{x \subseteq cB, y \subseteq non\_fraud;\ \text{substring(x, y)}\} \tag{9}$$

$$record_{\text{non\_spam}} = \{x \subseteq cB, y \subseteq non\_spam;\ \text{substring(x, y)}\} \tag{10}$$

where

$$cB = \text{collection of common bigrams} \tag{11}$$

$$\text{x and y} = \text{set of bigrams extracted from original texts of each dataset} \tag{12}$$

$$\text{substring(x,y)} = \text{subset of biagrams x and y} \tag{13}$$

The Equation (14) shows the collection of intersection between the list A that includes the bigrams of the Spam dataset and the list B that includes the bigrams of the Fraud dataset. Therefore, each list of records regroups the substrings which are texts including bigrams that belong to the collection cB and the bigrams of texts in each corresponding dataset.

$$cB = \{x : x \subseteq A \cap B | A = bigrams_{\text{Spam Dataset}}; B = bigrams_{\text{Fraud Dataset}}\} \tag{14}$$

4.3. Models

We present a two-part system that is based on LSTM neural network models for text classification. The framework of our proposed approach is described in Figure 1 where two individual models are designed and a third joint model was suggested and trained on a new joint dataset.

We implemented three classification models based on LSTM architecture. The models share the same architecture design but differ in the last output layer. Spam Model and Fraud Model (i.e., independent models) are designed for binary text classification. Spam Model is used for classifying the YouTube comments into Spam and Non-Spam, and equivalently, Fraud Model is designed for classifying fraudulent emails into Fraud and Non-Fraud. However, Joint Model is designed for multi-label text classification of four different class labels that are not mutually exclusive. In Figure 1 the detailed labels from Joint Model output are presented, where they are formed using the mutually exclusive binary labels (Spam, Fraud, Non-Spam, and Non-Fraud).

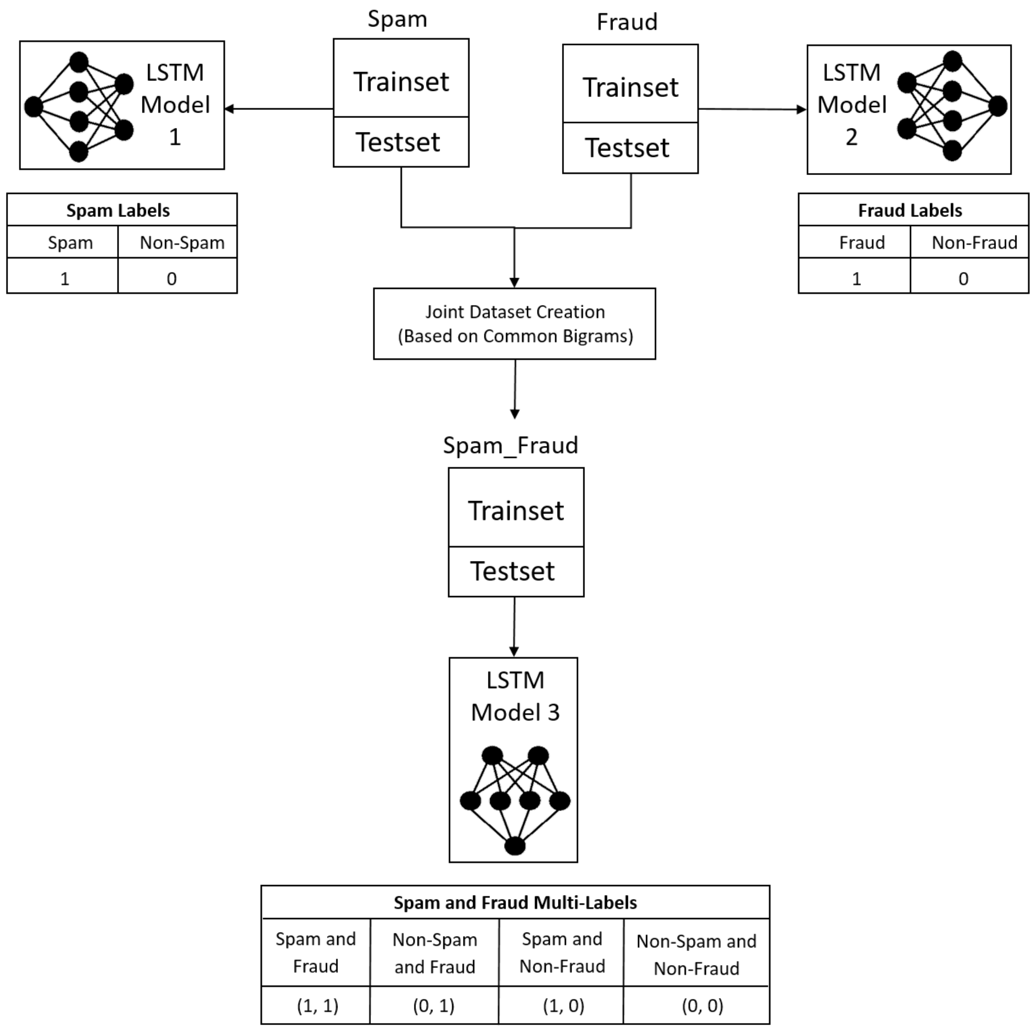


Figure 1. The framework of Spam and Fraud text classification.

Figure 2 shows the stacked architecture, which is composed of sequential layers to add levels of abstraction to the sequential input over time. The first layer is the Embedding layer, it is fed with a matrix created using vocabulary words extracted from each dataset and transformed through the embedding model. The output of this layer is a two-dimensional vector with an embedding for each word in the input sequence. After that, the model stacks a block of LSTM layer followed by a dropout L2-regularization, in order to avoid the over-fitting problem. For label classification, a dense layer with a softmax activation function was added to the architecture. Finally, binary labels are obtained by using a fully connected layer. This layer applies an optimization of the weights and evaluates the quality of the predictions using the performance evaluation metrics. The binary classification Spam Model and Fraud Model are designed with a last fully connected dense layer that applies a softmax activation function. However, the multi-label classification Joint Model presents its last layer along with the sigmoid activation. The training objective is to reduce a binary cross-entropy loss between the predicted and the actual true class labels.

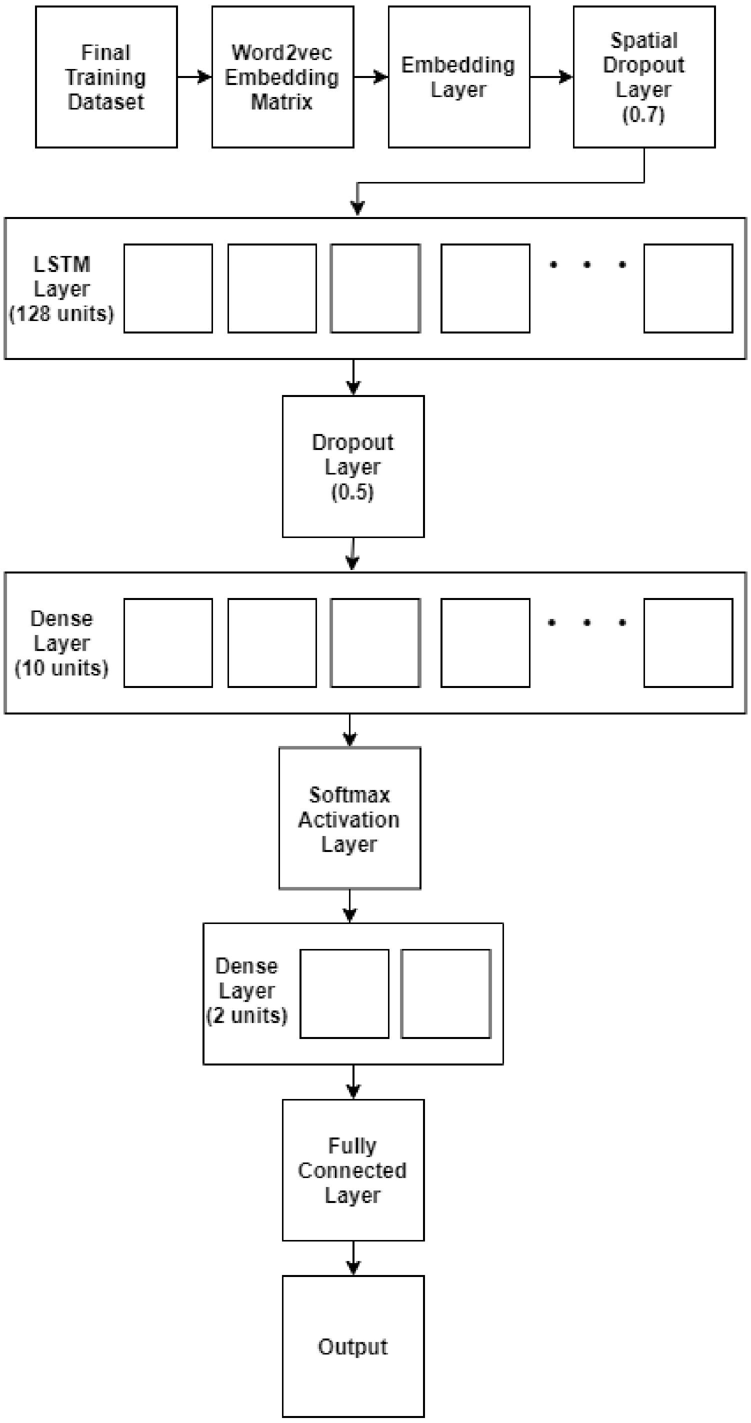


Figure 2. Our proposed LSTM architecture design.

4.3.1. Spam Model and Fraud Model Parameters

Since Spam Model and Fraud Model are designed for binary classification and share the same architecture, we adopted the same parameters that were partially inspired by the works [53–55] on text classification using LSTM model and word2vec model. Experiments were conducted on the hidden state dimension of the LSTM layer and the number of epochs, and we concluded the appropriate setting for the best performance.

We also evaluated the choice of word2vec parameters using the literature settings. Therefore, we used word vector size, which is the word embedding dimension d = 300 from the pre-trained Wor2vec model on the Google news dataset. This defines the size of the embedding matrix that was loaded into the embedding layer of our model. The LSTM layer was applied with a number of units = 128 which are the hidden units that represent the depth of the LSTM and its learning capacity to memorize during the training. The LSTM was preceded by a spatial dropout 1D layer.

Before stacking the last fully connected dense layer with two units, the additional dense layer was applied with 10 units. The two models are trained using Adam optimizer and a number of epochs between 10 and 30, which determines the number of times of selecting the training set once to update the weights. We used a batch size of 64 and a dropout parameter at the embedding layer with a probability p = 0.7 and at the LSTM layer with a probability p = 0.5 and L2-regularization parameter value with 0.5. We used early stopping criteria with a minimum delta = 0.0001, which stopped the training in case of no improvement after a patience = 3. This helped to monitor the performance measure and to stop the process if it did not achieve an improvement on minimizing the loss function with more than 0.0001 during 3 iterations. To normalize the input text matrix, we added padding of value zero with a max length of all the vector texts. This technique helps to avoid reducing the size of the text matrix due to the inequality of the vector texts' size. As the two main datasets have different text sizes and structures, the input matrix of the Spam dataset was padded with pad_length = 150, however, the input matrix of the Fraud dataset was padded with pad_length = 207.

4.3.2. Joint Model Parameters

The joint Model is designed for multi-label classification and thus it requires different parameters. It is trained over the joint dataset 'Spam_Fraud Dataset', where the texts are labeled with an original class and an inherited class. The training objective of this model is to reduce a binary cross-entropy loss between the predicted and the actual true class labels. A threshold T was set in order to convert the sigmoid output probability vector into two classes label. We tried to run the algorithm iteratively and we found that T = 0.6 gave the best results. We adopted the same parameters of the independent models except for a spatial dropout 1D layer at the embeddings layer with p = 0.5. The padding length was set to pad_length = 400. All models' parameters in our experiments are summarized in Table 2.

Table 2. Parameter values in our proposed models.

Parameter	**Value**
Word embedding dimension	300
Number of LSTM units	128
Dropout probability at embedding layer	0.5 and 0.7
Dropout probability at the output layer	0.5
L2 regularization rate	0.5
Early stopping min delta	0.0001
Number of epochs	10, 20 and 30
Padding length	150, 207 and 400

4.4. Word Representation: Word2vec

Our experiments used the same word embeddings for all three models. We implemented a word-embedding matrix using the Google pre-trained vectors with a dimension of 300 [56]. The model is trained on a part of Google News dataset, which contains approximately 100 billion words. The model contains 300-dimensional vectors for 3 million words and sentences that can be used to create word embeddings for a specific dataset. The suggested algorithm summarizes the creation of the embedding matrix, which takes two main inputs extracted from each dataset. A binary file of Google pre-trained vectors is converted into 300-dimensional real-valued vectors using a model from Python's Gensim library and a vocabulay_inverse of words that contains a set of words sorted by their frequency in the collections of texts.

4.5. The Performance Evaluation Metrics

The experiments were conducted for binary and multi-label classification, so we reported the evaluation of all the models using four evaluation metrics: accuracy rate (ACC), precision rate (P), recall rate (R), and F1 score (F1) as detailed below, where the metrics are computed using the elements of the confusion matrix as explained in Table 3.

Table 3. Confusion matrix parameters.

	True Class 1	**True Class 2**
Predicted Class 1	TP (True Positive)	FP (False Positive)
Predicted Class 2	FN (False Negative)	TN (True Negative)

Class 1 indicates the Spam (resp. Fraud) class and class 2 indicates the Non-Spam (resp. Non-Fraud).

4.5.1. Accuracy

Accuracy is the ratio between the number of correctly classified-instances and the total number of instances. It is also defined as the ratio of true positive (TP) and true negative (TN) over the total number of instances as shown in the following Equation (15):

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TM} + \text{FP} + \text{FN}} \tag{15}$$

where TP is the number of positive instances that are predicted correctly as positive, TN is the number of negative instances that are predicted correctly as negative, FP is the number of positive instances that are predicted incorrectly as negative, and FN is the number of negative instances that are predicted incorrectly as positive.

4.5.2. Precision

Precision represents the proportion of the correctly predicted positive instances TP to the total predicted positive instances. The Equation (16) for calculating the precision rate P of the positive class is as follows:

$$\text{P} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{16}$$

4.5.3. Recall

Recall refers to the proportion of correctly predicted positive instances to all instances in the actual class. The Equation (17) for the recall rate R of the positive class is as follows:

$$\text{R} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{17}$$

4.5.4. F1 Score

To balance the accuracy rate, F1 value is used to measure the effect of a certain class in the classification process. This score is the weighted average of the precision rate and the recall rate; therefore it takes into account both the false positives and false negatives. Intuitively, F1 is usually more useful than accuracy, especially in case of slightly unbalanced-classes distribution. Hence, the accuracy gives the same value of F1 score if the false positives and false negatives have similar values. The Equation (18) for the F1 score is as follows:

$$\text{F1 score} = \frac{2 \times (\text{P} \times \text{R})}{\text{R} + \text{P}} \tag{18}$$

5. Results and Analysis

This section presents the results of the experiments conducted using our proposed spam and fraud detection models: LSTM Spam Model for spam comments classification, LSTM Fraud Model for fraud emails classification and LSTM Joint Model for Spam_Fraud classification.

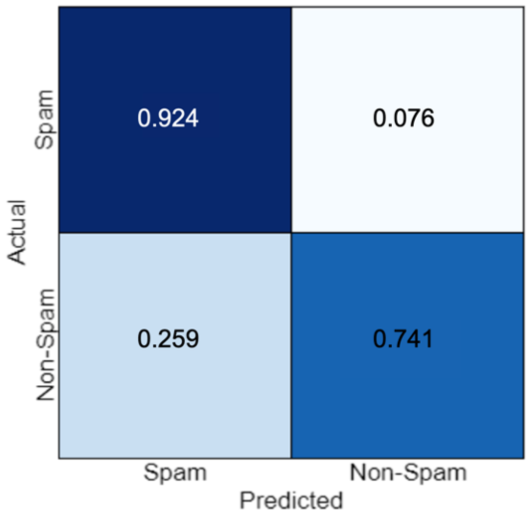
5.1. Spam Model Results

Before applying the Spam Model, we balanced the dataset so that the train and test datasets have equally-distributed class labels. We split the data into 80% for the training dataset and 20% for the test dataset. A validation dataset was randomly selected as 10% of the training dataset while training the model.

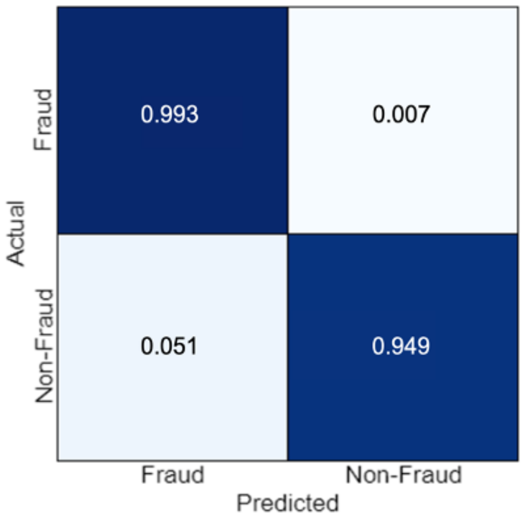
The proposed methodology was compared with three baseline classification models: Support Vector Machines (SVM), Naïve Bayes and Stochastic Gradient Descent (SGD). We first tested the performance of independent models to classify the individual datasets and we also tuned the parameters to report their best performance. SVM model was reported with a regularization parameter 1 and a linear kernel. Naïve Bayes model was reported with a smoothing parameter 0.01 and SGD model was reported with a regularization coefficient 0.001. Table 4 shows the evaluation of training simple classification models on the Spam dataset, along side with the Fraud dataset and the Joint dataset that will be discussed later. We reported the best performance using Naïve Bayes classifier on the Spam dataset, with an accuracy score of 69%, a precision rate of 0.67, a recall rate of 0.75 and a F1 score of 71% for the Spam model.

After that, we conducted experiments using our model and results showed an 83% accuracy value, with a 0.91 precision rate, a 0.75 recall rate, and an 0.82 F1 score. Figure 3 is a confusion matrix that summarizes the accuracy of our deep learning classifier over the prediction vs. the expectation of each class.

With the results presented in Figure 3a, it is apparent that our LSTM model is capable of correctly predicting the spam class better than the non-spam class with a proportion difference of 20%. This difference may be attributed to the quality of the cleaning and preprocessing of the original dataset. In fact, the model converges rapidly after the 4th epoch, and there is no further drop in loss values.



(**a**)



(**b**)

Figure 3. Performance of the Independent Models. (**a**) Confusion matrix of the Spam Model; (**b**) Confusion matrix of the Fraud Model.

Table 4. Performance Evaluation of the Baseline Models.

Model	Classifier	Dataset	Accuracy	Precision	Recall	F1 Score
Spam Model	SVM	Spam Dataset	0.67	0.69	0.61	0.64
	Naïve Bayes		0.69	0.67	0.75	0.71
	SGD		0.66	0.81	0.42	0.56
Fraud Model	SVM	Fraud Dataset	0.55	0.55	0.72	0.62
	Naïve Bayes		0.67	0.63	0.85	0.72
	SGD		0.52	0.53	0.43	0.48
Joint Model	SVM	Joint Dataset	0.71	0.86	0.71	0.72
	Naïve Bayes		0.77	0.85	0.75	0.78
	SGD		0.73	0.80	0.72	0.74

5.2. *Fraud Model Results*

As in Spam Model, the original dataset is not balanced, so we equally extracted the same number of instances for both train and test datasets with a split of 80–20%. A 10% of the training dataset was randomly-selected and dedicated to the validation dataset while training the model.

Similarly, we compared our proposed model with three baseline models on the Fraud dataset, as shown in Table 4 where We reported the best results with Naïve Bayes model, having an accuracy score of 67%, a precision rate of 0.63, a recall rate of 0.85 and a F1 score of 71% for the Fraud model. However, the conducted experiment of our model for Fraud Model achieved a 96% accuracy rate value, with a 0.99 precision rate and a 0.92 recall rate, ending with an 0.92 F1 score. Figure 3b shows an outstanding confusion matrix that summarizes the accuracy of our deep learning classifier over the predicted class vs. the expected class. It can be observed from Figure 3b that our LSTM model is capable of correctly predicting the fraud class as well as the non-fraud class. In fact, the model performs well overall for the two classes, does not suffer from over-fitting, and clearly converges rapidly after the 7th epoch.

5.3. *Joint Model Results*

After extracting the common dataset as described in Section 4, we conducted experiments using the Joint Model that is designed for the multi-label classification as described above. The experiment was conducted after splitting randomly the dataset into 90% for training and 10% for testing, which resulted in an unbalanced distribution of the four categories. As before, 10% of the training dataset was dedicated to the validation of the model.

We first trained the baseline models on the Joint dataset and we also reported the best results using Naïve Bayes classifier with an accuracy score of 77%, a precision rate of 0.85, a recall rate of 0.75 and a F1 score of 78% as shown in Table 4. After that, we conducted experiment for our Joint Model and results showed an accuracy value of 92%, with a precision rate of 0.84 and a recall rate of 0.86, and with an F1 score of 0.85. The confusion matrix for this model is summarized in Figure 4, where it is noticed that our proposed classification model is doing well for correctly categorizing the four classes.

The experiments validated that our proposed LSTM models could show better results than standard baseline classification models that performed poorly for all the individual models with a maximum 77% accuracy score.

As the model was designed to classify two classes simultaneously, we ended up comparing the prediction of the four joint classes. We used the same threshold T = 0.6 to transform the last output sigmoid layer with two independent class probabilities into four-cross classes. Furthermore, the performance of the proposed model shows good learning of the model without over-fitting because of using the early stopping methods, where the training stopped after the 30th iteration.

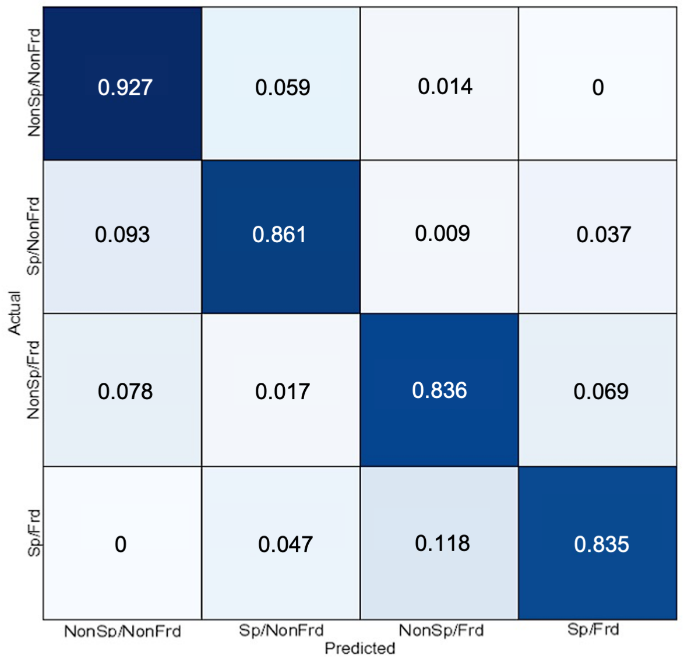


Figure 4. Confusion matrix of the Joint Model.

It can be observed from Figure 4 that our joint LSTM model is capable of performing well for the four different categories. The model has the highest classification performance for the Non-Spam/Non-Fraud class. This is explained by the semantic similarity of the Non-Spam comments and Non-Fraud emails. However, the Spam/Fraud, Spam/Non-Fraud, and Non-Spam/Fraud classes perform similarly with an approximate difference of 1–3% due to the unbalanced distribution of the classes. The Spam/Non-Fraud class and the Non-Spam/Fraud classes have the most satisfying performance amongst the aforementioned categories. This can be interpreted with the short Spam (resp. Non-Spam) comments toward the long Non-Fraud (resp. Fraud) emails and their detected common bigrams between the two different sources of each class. The Spam/Fraud category has the lowest result, as the quality of the two classes is semantically related but with different sizes of texts. Even though common bigrams of the two classes were detected in many records, every dataset has particular text format and structure, that the model may be confused between them.

5.4. Cross-Datasets Evaluation of the Joint Model Results

We present a comparative analysis of the Joint Model with the independent models on two different test samples from the two main datasets: Spam test samples and Fraud test samples. We conducted the following experiments by predicting the different samples using not only the main Spam Model (resp. Fraud Model) but also using the Joint Model after discarding the unconcerned Fraud/Non-Fraud class (resp. Spam/Non-Spam class).

We first conducted a cross-datasets evaluation using the three baseline models, by training them on Samples 1 and Samples 2 as shown in Table 5, and we noticed that Naïve Bayes classifier achieved the best results among the three classifiers. However, we reported a low enhancement of the performance for Samples 1, with only 2% of accuracy score and 0.06 of the recall rate. We also reported a low enhancement of the results for Samples 2, with only 2% of accuracy score and 0.03 of the precision rate.

Table 5. Cross-Datasets Evaluation of the Baseline Models.

Model	Classifier	Dataset	Accuracy	Precision	Recall	F1 Score
Spam Model	SVM	Spam test samples	0.49	0.50	0.94	0.65
	Naïve Bayes		0.56	0.52	0.82	0.67
	SGD		0.45	0.52	0.76	0.59
Joint Model	SVM	Spam test samples	0.50	0.50	1.00	0.66
	Naïve Bayes		0.58	0.52	0.88	0.67
	SGD		0.47	0.55	0.79	0.60
Fraud Model	SVM	Fraud test samples	0.50	0.50	1.00	0.67
	Naïve Bayes		0.53	0.52	0.89	0.70
	SGD		0.50	0.52	0.89	0.69
Joint Model	SVM	Fraud test samples	0.54	0.50	1.00	0.68
	Naïve Bayes		0.55	0.55	0.89	0.70
	SGD		0.50	0.56	0.81	0.69

After that, we applied our proposed models and as shown below in Table 6, our Joint Model outperformed the independent Spam Model on Samples 1 with more than 14% accuracy score, 0.19 on precision rate, 0.04 on recall rate and 0.13 on F1 score. Additionally, our Joint Model outperformed the independent Fraud Model on the Samples 2 with a 2% accuracy score, 0.03 on precision rate and 0.02 on F1 score.

Table 6. Performance Evaluation of the Joint Model.

Model	Dataset	Accuracy	Precision	Recall	F1 Score
Spam Model	Spam test samples	0.79	0.72	0.93	0.81
Joint Model		0.93	0.91	0.97	0.94
Fraud Model	Fraud test samples	0.97	0.95	1.00	0.97
Joint Model		0.99	0.98	1.00	0.99

Therefore, the enhancement of the prediction is more noticeable for Spam Model than Fraud Model, where it can be observed that the Joint Model performed better on Sample 1 rather than Sample 2. This can be explained by the resemblance of the Fraud dataset with the joint dataset more than with the Spam dataset, as shown in Figure 5, where the texts in the joint dataset have a maximum length of 1500 words, which is close to the maximum text length of the Fraud dataset having 2500 words. Further, this enhancement specifically addresses the generally harder problem of classifying short texts as opposed to longer texts. Longer texts have NLP benefits including more context and denser data which leads to better performance in NLP tasks as seen in [57,58].

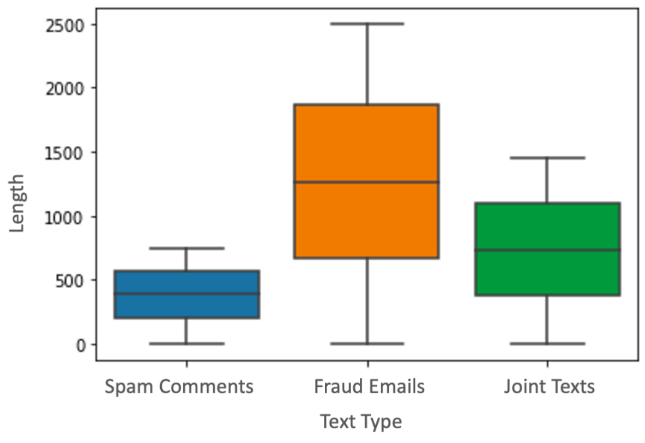


Figure 5. Comparison of the text length in the three different datasets.

The cross-datasets evaluation shows that training a text classification LSTM model on a joint dataset is capable of outperforming the models trained by the individual datasets. The datasets were extracted from two different sources "YouTube" and "Mailbox", and despite having different structures and formats, our proposed joint model is capable of correctly classifying the texts when tested using their original labels regardless of the source.

Moreover, this experiment validates our hypothesis about the enhancement of the performance of the Joint model to predict nature of the texts and their source. However, it is noticeable that our proposed LSTM models outperformed the baseline models, and this can be explained by the fact that with the emergence of word embedding models (i.e., word2vec model), it is expensive for standard classification models, such as Naïve Bayes, to build more complex text representations recursively because word embedding presents elements of hierarchy and should be useful with sequential models such as LSTM. Even though standard models showed promising results on NLP tasks, they usually employ bag-of-words model that are computationally heavy to carry along the learning process, thus they are not able to earn structure from the sequential dataset because they do not preserve the order of the words [54,59].

6. Conclusions

Binary text classification is applied with conventional approaches and deep learning algorithms. The advanced neural network models outperform simple techniques and LSTM models showed the highest classification performance among those reported in the literature. In this work, we first propose an implementation of two LSTM models for classifying a collection of text from a Spam and Fraud dataset into two categories. The models are trained on the representation of the datasets using pre-trained word embeddings models, which preserve the semantic information between the words. Second, we present a joint LSTM model for transforming the problem into a multi-label text classification problem. The joint model is trained on a joint dataset that regroups text, which share the same bigrams, from the two different datasets. This generated dataset that presents text with four different non-exclusive labels, where a proposed model classifies text simultaneously into two binary exclusive labels.

Apart from the challenge that multi-label classification presents, our proposed joint LSTM model outperformed the classification results of the independent LSTM models for Spam and Fraud classification. Despite having different sizes and formats, joining text semantically from different sources of datasets enhanced the performance of the original classification models. We empirically showed that different sources of datasets but within a similar domain could be grouped into a joint dataset that is suitable for a multi-label text classification task.

One of the strengths of this work is present in the experimental assessment conducted, where the validation of our different neural network models showed a consistent high evaluation performance for the two independent binary models and satisfying results for the joint model.

The significance of this framework as it relates to information security rests on the ability of our joint model to distinguish between the two sources of text within similar contexts. This work offers a more accurate interpretability for phishing detection. We showed that text can be appropriately predicted with more than one domain label. Even though there are no similar works that presented the idea of multi-label classifying two types of non-malicious texts, we achieved the highest results of identifying text as non-malicious (i.e., the non-exclusive label "Non-Spam/Non-Fraud") with an accuracy rate of 92.7% in our test case, which is higher than the work conducted by Dhingra, A. et al. in [40] to detect spam on similar short texts dataset using an MLP model that achieved an 81% accuracy rate. Another similar work applied by Yu, W. et al. in [22] for phishing emails detection using a matching and heuristic algorithm performed with only an 80% catch rate.

The application of the LSTM joint classification model, which has a slight change in the neural network design, showed robust results, compared to the independent LSTM classification models.

The idea behind combining two different datasets using one of the NLP methods was limited to the checking of the existing common bigrams.

Author Contributions: Conceptualization, A.B. and S.A.; Data curation, A.B. and S.A.; Methodology, A.B. and S.A.; Project administration, D.S.-S.; Supervision, D.S.-S. and A.E.; Validation, D.S.-S. and A.E.; Writing—original draft, A.B. and S.A.; Writing—review & editing, A.B., S.A., D.S.-S. and A.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chiew, K.L.; Yong, K.S.C.; Tan, C.L. A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Syst. Appl.* **2018**, *106*, 1–20. [CrossRef]
2. Curtis, S.R.; Rajivan, P.; Jones, D.N.; Gonzalez, C. Phishing attempts among the dark triad: Patterns of attack and vulnerability. *Comput. Hum. Behav.* **2018**, *87*, 174–182. [CrossRef]
3. Parsons, K.; Butavicius, M.; Delfabbro, P.; Lillie, M. Predicting susceptibility to social influence in phishing emails. *Int. J. Hum. Comput. Stud.* **2019**, *128*, 17–26. [CrossRef]
4. Laorden, C.; Ugarte-Pedrero, X.; Santos, I.; Sanz, B.; Nieves, J.; Bringas, P.G. Study on the effectiveness of anomaly detection for spam filtering. *Inf. Sci.* **2014**, *277*, 421–444. [CrossRef]
5. Ding, Y.; Luktarhan, N.; Li, K.; Slamu, W. A keyword-based combination approach for detecting phishing webpages. *Comput. Secur.* **2019**, *84*, 256–275. [CrossRef]
6. Chiew, K.L.; Tan, C.L.; Wong, K.; Yong, K.S.; Tiong, W.K. A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. *Inf. Sci.* **2019**, *484*, 153–166. [CrossRef]
7. Gao, L.; Zhou, S.; Guan, J. Effectively classifying short texts by structured sparse representation with dictionary filtering. *Inf. Sci.* **2015**, *323*, 130–142. [CrossRef]
8. Ren, Y.; Wang, R.; Ji, D. A topic-enhanced word embedding for Twitter sentiment classification. *Inf. Sci.* **2016**, *369*, 188–198. [CrossRef]
9. Stein, R.A.; Jaques, P.A.; Valiati, J.F. An analysis of hierarchical text classification using word embeddings. *Inf. Sci.* **2019**, *471*, 216–232. [CrossRef]
10. Nalisnick, E.; Mitra, B.; Craswell, N.; Caruana, R. Improving document ranking with dual word embeddings. In Proceedings of the 25th International Conference Companion on World Wide Web, Montréal, QC, Canada, 11–15 April 2016; pp. 83–84.
11. Kusner, M.; Sun, Y.; Kolkin, N.; Weinberger, K. From word embeddings to document distances. In Proceedings of the International Conference on Machine Learning, Lille, France, 12 July 2015; pp. 957–966.
12. Kim, D.; Seo, D.; Cho, S.; Kang, P. Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec. *Inf. Sci.* **2019**, *477*, 15–29. [CrossRef]
13. Dada, E.G.; Bassi, J.S.; Chiroma, H.; Adetunmbi, A.O.; Ajibuwa, O.E. Machine learning for email spam filtering: Review, approaches and open research problems. *Heliyon* **2019**, *5*, e01802. [CrossRef] [PubMed]
14. Jain, G.; Sharma, M.; Agarwal, B. Spam detection in social media using convolutional and long short term memory neural network. *Ann. Math. Artif. Intell.* **2019**, *85*, 21–44. [CrossRef]
15. Alberto, T.C.; Lochter, J.V.; Almeida, T.A. Tubespam: Comment spam filtering on youtube. In Proceedings of the 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 9–11 December 2015; pp. 138–143.
16. Nizamani, S.; Memon, N.; Glasdam, M.; Nguyen, D.D. Detection of fraudulent emails by employing advanced feature abundance. *Egypt. Inform. J.* **2014**, *15*, 169–174. [CrossRef]
17. Guan, W.; Gao, H.; Yang, M.; Li, Y.; Ma, H.; Qian, W.; Yang, X. Analyzing user behavior of the micro-blogging website Sina Weibo during hot social events. *Phys. A Stat. Mech. Its Appl.* **2014**, *395*, 340–351. [CrossRef]
18. Serrano-Guerrero, J.; Olivas, J.A.; Romero, F.P.; Herrera-Viedma, E. Sentiment analysis: A review and comparative analysis of web services. *Inf. Sci.* **2015**, *311*, 18–38. [CrossRef]
19. Zhao, Y.; Kou, G.; Peng, Y.; Chen, Y. Understanding influence power of opinion leaders in e-commerce networks: An opinion dynamics theory perspective. *Inf. Sci.* **2018**, *426*, 131–147. [CrossRef]

20. Rathore, S.; Sharma, P.K.; Loia, V.; Jeong, Y.S.; Park, J.H. Social network security: Issues, challenges, threats, and solutions. *Inf. Sci.* **2017**, *421*, 43–69. [CrossRef]
21. Urena, R.; Kou, G.; Dong, Y.; Chiclana, F.; Herrera-Viedma, E. A review on trust propagation and opinion dynamics in social networks and group decision making frameworks. *Inf. Sci.* **2019**, *478*, 461–475. [CrossRef]
22. Yu, W.D.; Nargundkar, S.; Tiruthani, N. Phishcatch-a phishing detection tool. In Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference, Washington, DC, USA, 20–24 July 2009; pp. 451–456.
23. Sun, X.X.; Dai, S.; Wang, Y.X. A platform for automatic identification of phishing URLs in mobile text messages. *J. Phys. Conf. Ser.* **2018**, *1087*, 042009. [CrossRef]
24. Hu, W.; Du, J.; Xing, Y. Spam filtering by semantics-based text classification. In Proceedings of the 2016 Eighth International Conference on Advanced Computational Intelligence (ICACI), Chiang Mai, Thailand, 14–16 February 2016; pp. 89–94.
25. Harikrishnan, N.B.; Vinayakumar, R.; Soman, K.P. 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), Tempe, AZ, USA, 21 March 2018; pp. 455–468.
26. Sharmin, S.; Zaman, Z. Spam detection in social media employing machine learning tool for text mining. In Proceedings of the 2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Jaipur, India, 4–7 December 2017; pp. 137–142.
27. Varol, C.; Abdulhadi, H.M.T. Comparision of String Matching Algorithms on Spam Email Detection. In 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), Ankara, Turkey, 3–4 December 2018; pp. 6–11.
28. Hassan, M.A.; Mtetwa, N. Feature Extraction and Classification of Spam Emails. In Proceedings of the 2018 5th International Conference on Soft Computing & Machine Intelligence (ISCMI), Nairobi, Kenya, 21–22 November 2018; pp. 93–98.
29. Zareapoor, M.; Seeja, K.R. Feature extraction or feature selection for text classification: A case study on phishing email detection. *Int. J. Inf. Eng. Electron. Bus.* **2015**, *7*, 60. [CrossRef]
30. Zhang, Y.; Zhang, Z.; Miao, D.; Wang, J. Three-way enhanced convolutional neural networks for sentence-level sentiment classification. *Inf. Sci.* **2019**, *477*, 55–64. [CrossRef]
31. Yaghoobzadeh, Y.; Schutze, H. Multi-level representations for fine-grained typing of knowledge base entities. *arXiv* **2017**, arXiv:1701.02025. Available online: www.arxiv.org/abs/1701.02025 (accessed on 10 January 2020).
32. Zhang, X.; Zhao, J.; LeCun, Y. Character-level convolutional networks for text classification. In Proceedings of the Advances in Neural information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 649–657.
33. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NE, USA, 5–8 December 2013; pp. 3111–3119.
34. Mikolov, T.; Yih, W.T.; Zweig, G. Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, GA, USA, 9–14 June 2013; pp. 746–751.
35. Ren, Y.; Ji, D. Neural networks for deceptive opinion spam detection: An empirical study. *Inf. Sci.* **2017**, *385*, 213–224. [CrossRef]
36. Parwez, M.A.; Abulaish, M.; Jahiruddin, J. Multi-Label Classification of Microblogging Texts using Convolution Neural Network. *IEEE Access* **2019**. [CrossRef]
37. Rao, G.; Huang, W.; Feng, Z.; Cong, Q. LSTM with sentence representations for document-level sentiment classification. *Neurocomputing* **2018**, *308*, 49–57. [CrossRef]
38. Alghoul, A.; Al Ajrami, S.; Al Jarousha, G.; Harb, G.; Abu-Naser, S.S. Email Classification Using Artificial Neural Network. *Int. J. Acad. Dev.* **2018**, *2*, 8–14.
39. Yawen, W.; Fan, Y.; Yanxi, W. Research of Email Classification based on Deep Neural Network. In Proceedings of the 2018 Second International Conference of Sensor Network and Computer Engineering (ICSNCE 2018), Xi'an, China, 27–29 April 2018.
40. Dhingra, A.; Mittal, S. Content based spam classification in twitter using multi-layer perceptron learning. *Int. J. Latest Trends Eng. Technol.* **2015**, *5*, 9–19.

41. Deshmukh, J.S.; Tripathy, A.K. Mining multi domain text reviews using semi-supervised approach. In Proceedings of the 2016 IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, India, 17–18 March 2016; pp. 788–791.
42. Ding, X.; Shi, Q.; Cai, B.; Liu, T.; Zhao, Y.; Ye, Q. Learning Multi-Domain Adversarial Neural Networks for Text Classification. *IEEE Access* **2019**, *7*, 40323–40332. [CrossRef]
43. Jiang, W.; Gao, H.; Lu, W.; Liu, W.; Chung, F.L.; Huang, H. Stacked Robust Adaptively Regularized Auto-Regressions for Domain Adaptation. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 561–574. [CrossRef]
44. Hua, Y. Understanding BERT performance in propaganda analysis. In Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda, Hong Kong, China, 3–7 November 2019; pp. 135–138.
45. Aggarwal, A.; Chauhan, A.; Kumar, D.; Mittal, M.; Verma, S. Classification of Fake News by Fine-tuning Deep Bidirectional Transformers based Language Model. In *EAI Endorsed Transactions on Scalable Information Systems Online First*; EAI: Ghent, Belgium, 2020.
46. Rusk, N. Deep learning. *Nat. Methods* **2016**, *13*, 35. [CrossRef]
47. Kulkarni, A.; Shivananda, A. Converting text to features. In *Natural Language Processing Recipes*; Apress: Berkeley, CA, USA, 2019; pp. 67–96.
48. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781. Available online: ww.arxiv.org/abs/1301.3781 (accessed on 10 January 2020).
49. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
50. YouTube Spam Collection. Available online: http://dcomp.sor.ufscar.br/talmeida/youtubespamcollection/ (accessed on 15 November 2019).
51. Radev, D. CLAIR Collection of Fraud Email, ACL Data and Code Repository 2008, ADCR2008T001. Available online: http://aclweb.org/aclwiki (accessed on 21 March 2019).
52. Jianqiang, Z.; Xiaolin, G. Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access* **2017**, *5*, 2870–2879. [CrossRef]
53. She, X.; Zhang, D. Text Classification Based on Hybrid CNN-LSTM Hybrid Model. In Proceedings of the 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 8–9 December 2018; pp. 185–189.
54. Li, C.; Zhan, G.; Li, Z. News Text Classification Based on Improved Bi-LSTM-CNN. In Proceedings of the 2018 9th International Conference on Information Technology in Medicine and Education (ITME), Hangzhou, China, 19–21 October 2018; pp. 890–893.
55. Xiao, L.; Wang, G.; Zuo, Y. Research on Patent Text Classification Based on Word2Vec and LSTM. In Proceedings of the 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 8–9 December 2018; pp. 71–74.
56. Trausan-Matu, S. Intertextuality detection in literary texts using Word2Vec models. In Proceedings of the 21st International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 8–10 October 2017; pp. 262–265.
57. Xu, J.; Cai, Y.; Wu, X.; Lei, X.; Huang, Q.; Leung, H.F.; Li, Q. Incorporating context-relevant concepts into convolutional neural networks for short text classification. In *Neurocomputing*; Elsevier: Amsterdam, The Netherlands, 2019.
58. Zheng, Y.; Haixun, W.; Xuemin, L.; Min, W. Understanding short texts through semantic enrichment and hashing. *IEEE Trans. Knowl. Data Eng.* **2015**, *28*, 566–579.
59. Kowsari, K.; Jafari Meimandi, K.; Heidarysafa, M.; Mendu, S.; Barnes, L.; Brown, D. Text classification algorithms: A survey. *Information* **2019**, *10, 150*. [CrossRef]