International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 11, No.1 (Feb-2022)

https://dx.doi.org/10.12785/ijcds/110173

A Preliminary Study on Personalized Spam E-mail Filtering Using Bidirectional Encoder Representations from Transformers (BERT) and TensorFlow 2.0

Kashif Iqbal[1], Salman A. Khan[1], Shamim Anisa[1], Ayesha Tasneem[1] and Nazeeruddin Mohammad[2]

[1]*College of Computing and Information Sciences, Karachi Institute of Economics Technology, Karachi, Pakistan*
[2]*Cybersecurity Center, Prince Mohammad Bin Fahd University, Al-Khobar, Saudi Arabia*

Received 15 September 2021, Revised 30 December 2021, Accepted 31 January 2022, Published 15 February 2022

Abstract: Email security has been a major concern in for a long time. One important aspect of e-mail security is effective and efficient detection of spam e-mails, which is an added overhead in the proper functioning of modern email communication systems. In this paper, a method based the Bidirectional Encoders Representations from Transformers (BERT) is proposed that stems from deep learning (DL). Similar to Word Embedding, BERT is a technique for text representation and a combination of various DL methods such as bidirectional encoder LSTM and Transformers. The pre-training phase takes significant computational effort, and to save computational time, we used a pre-trained BERT model. A preliminary analysis of the proposed algorithm is carried out using a standard test suite of Enron dataset. Initial results indicate that the proposed algorithm has a potential of generating promising results.

Keywords: Spam e-mail, Deep learning, Natural language processing, Data security

1. Introduction

E-mails serve as one of the most widely used medium of present day electronic communication. Despite the manifold advantages of e-mails, a major concern is the amount of unsolicited e-mails that end up in a user's email account. These undesired e-mails, known as spam, account for almost 57% of e-mail traffic worldwide [1]. This huge number of spam emails is not only a burden on computing and communication resources, but also cause nuisance to users. Over the years, solutions have been proposed to deal with the issue of spam e-mails, and active research is still being carried out to improve the spam detection and filtering mechanisms. In this context, artificial intelligence based approaches, particularly machine learning, have also received notable attention by researchers. With regard to machine learning methods, several techniques are utilized. Weimiao et. al. [1] proposed a support vector machine (SVM) based Naïve Bayes (NB) e-mail filtering system named SVM-NB. They used a large Chinese e-mail dataset DATAMALL for testing their approach Comparison with SVM and NB showed best performance by SVM. Chakarborty et al. [2] mutually compared NB, Decision Tree, and Logistic Model Tree algorithms. They utilized their own dataset of e-mails collected over a period of 6 months. Results indicated that Logistic Model Tree produced the best results with almost 90% accuracy. Rathi and Pareek [3] carried out performance evaluation of several machine learning techniques including NB, Bayes Net, SVM, function Tree (FT), J48 classifier, Random Tree (RT), and Random Forest (RF). The Spambase dataset was used. Results indicated that RF displayed the best results with an accuracy of 94.82%. Mallampati et. al. [4] have studied and compared NB, J48, and deep neural networks (DNN) based classifiers using the SpamAssasin dataset. Among the three techniques compared, DNN achieved the highest level of accuracy which was over 99%. Rusland et. al. [5] analyzed the performance of NB for e-mail spam detection using two datasets, namely Spam Data and Spambase. SpamData contained 500 attributes and 9324 e-mails, while Spambase has 58 attributes and 4601 e-mails messages. Their comparative results indicated that the performance of NB classifier on Spambase was better than that on SpamData. Srinivasan et. al. [6] analyzed a DL based framework while using four datasets. These datasets were Lingspam, PU, SpamAssasin, and Enron. Hassanpour et. al. [7] developed a DL model while using word2vec. They used an open dataset to assess their algorithm and found over 96% accuracy. Bibi et. al. [8] used NB and compared with SVM. A self-developed

E-mail address: ghani.kashif@gmail.com, sakhan@kiet.edu.pk, shamimanisa07@gmail.com, ashuashi15@outlook.com , nmohammad@pmu.edu.sa

data set consisting of 960 mails was used. The results indicated the NB was able to achieve accuracy of over 96% in comparison with 90% achieved by SVM. AbdunNabi et. al. [9] used the BERT approach and comparisons were made with the baseline deep learning model, NB, and K-NN (K-nearest neighbors). Two open source datasets, namely, Spambase and Spam Filter were used for evaluation. Results indicated that the BERT based approach produced accuracy of over 98% on testing data, which was the best among the other three techniques used for comparisons. The aforementioned studies mainly focused on the use of ML techniques in general. However, the use of DL for spam detection is still evolving. DL gives two advantages over the traditional machine learning techniques. Firstly, DL can efficiently learn about the complex and hierarchical feature representation [6]. Secondly, DL is effective in producing high-quality results even if the data is unstructured [6].

Keeping in view the advantages of DL, the main contributions of this study are enumerated as follows:

1) The impact of LSTM (Long Short Term Memory) model of DL is assessed on the detection of spam email.
2) The Bi-directional Encoder Representations from Transformers (BERT) tokenizer is used as the word embedding approach in conjunction with LSTM.
3) An empirical analysis is carried out using Tensorflow 2.0.

The following sections of this paper are organized as follows. In Section 2, the preliminaries related to the study are discussed. This is followed by the methodology in Section 3. Empirical findings are discussed in Section 4. Conclusive remarks and future directions are given in Section 5.

2. Preliminaries

The email spam detection problem has two important dimensions. One dimension is concerned with the algorithms used in the spam detection process. The other dimension is concerned with the natural language processing (NLP), specifically the text representation in the current case. As such, this section provides a short background on these two dimensions. The first subsection focuses on the deep learning algorithms, while the second subsection provides a brief discussion on different text representations.

A. Deep Learning

The literature [10] has identified a number of DL algorithms. Among them, Convolutional Neural Networks (CNNs), deep belief networks (DBNs), Recurrent Neural Networks (RNNs), De-noising autoencoders (DAEs), and LSTM are the most popular DL methods that have received notable attention by researchers. A detailed discussion on these methods and their applications can be found in a study by Mosavi et al. [10]. Since the current study utilizes the LSTM model, a short background on the model is provided below. The RNNs are a class of DL methods that have a great potential in capturing both the long-term and the short-term temporal dependencies within the time-series data. The LSTM approach is a particular class of RNNs that utilize their inherent memory capabilities, and use the past information to make efficient predictions. The LSTM structure is fundamentally comprised of memory units. The network's internal gating mechanism of LSTM controls the flow of information. One of the main advantages of the LSTM network is that LSTM overcomes the vanishing as well as the exploding gradient problem inherent in the conventional RNNs. An LSTM network accepts a sequential time-series data as input.

B. Text Representation

Text representation is an important topic in the domain of NLP, and serves as a fundamental aspect of email spam detection. Various text representation approaches exist. Machine learning algorithms have limitations in direct handling of raw text. Therefore, the text needs to be transformed into numbers [6]. Three most widely used text representation techniques are One-Hot Encoding, Count Vectors, and Word Embeddings [11]. A detailed discussion on text representation can be found in the work by Srinivasan et. al. [6]

In this work, BERT is adopted which is a recently proposed approach. BERT was developed by in the recent past [12]. The method has shown to be very effective in dealing with various NLP tasks such as text classification, text generation, and text summarization, etc. Further details on BERT are given in the next section.

3. Methodology

A typical task concerned with NLP is comprised of five major phases. These are data collection, data cleaning and preprocessing, feature extraction, and training evaluation of model [9]. Since deep learning is used, feature extraction is carried out automatically within the deep learning model training. The details of the other phases are provided below.

A. Data Collection

In this paper, Enron dataset [13] is used. Our interest in this study is to focus on personalized spam filters. Therefore, following the approach adopted by Metsis et al. [14], we focused on categorized emails from six Enron employees who had large mailboxes. These employees are identified as farmer-d, kaminski-v, kitchen-l, williams-w3, beck-s, and lokay-m [13]. These are referred to as Enron 1 to Enron 6 in this paper, and consist of ham and spam messages, as given in Table I below. A cleaned-up form of the same data is provided by Bekkerman [15], which consists of ham messages only.

B. Data Cleaning and Preprocessing

The data cleaning and preprocessing can be divided into three phases, as explained below.

TABLE I. Details of ham and spam in the Enron dataset

Dataset	Total Number of Emails	Ham Emails	Spam Emails
Enron 1	3146	67%	33%
Enron 2	2297	50%	50%
Enron 3	3540	78%	22%
Enron 4	3540	78%	22%
Enron 5	5176	50%	50%
Enron 6	3747	67%	33%

1) Importing and Preprocessing the Dataset

The data can be pre-processed to remove all punctuation marks and other special characters. In order to do this, a function can be defined that accepts raw text features as input and returns the associated text features in cleaned form.

2) Creation of BERT Tokenizers

Every transformer based model has Consisting of a unique tokenization technique, every transformer model has a different method of utilizing special tokens. The transformer library looks after it and supports tokenization for model which is associated with it. In this paper, we are going to use BERT (Transformer) text embedding as input to train our classification model. In order to do this process, a sentence is divided into words for the input text-based emails. This process is referred to as tokenization. In the present study, BERT tokenizer is employed.

3) Preparing Data for Training

The text emails in the input dataset have different lengths, ranging from very small to very long texts. The model training phase requires that the input sentences should be equal in length. One way to generate sentences of uniform length, shorter sentences are padded by s. This can cause a sparse matrix having huge number of s. Alternatively, sentences can be padded within each batch. In this study, the model is trained in batches. Therefore, sentences can be padded locally within the training batch, while taking into consideration the sentence with the maximum length.

C. Baseline model

The baseline model is the latest BiLSTM model [16]. A class named as Text_Model is created that mainly inherits from tf.keras.model (LSTM / BILSTM). We define model layers inside the class. Our model takes input from embedding layer and fed to a convolutional neural network (CNN). The CNN contains three layers initialized with filter, kernel values and activation function respectively and Global max pooling layer is applied to the output of each CNN layer. Afterward, the outputs of these CNN layers are fed to first dense layer. Then, the next layer is dropout layer to avoid the over-fitting. Lastly, another dense layer is used that employs the sigmoid / softmax function to predict the output is used. Figure 1 illustrates the structure of layers and baseline model.

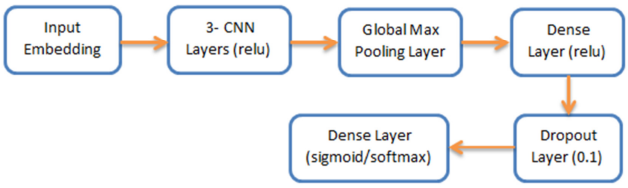

Figure 1. Baseline Model Structure

D. Transformer model

Like word tokenizer and embedding technique, BERT is a text representation mechanism that combines a variety of DL algorithms, such as bidirectional encoder LSTM and Transformers. BERT is used by Google as a core module in their search algorithm for better understanding of queries. In this study, we do not focus on the theoretical details regarding the implementation of BERT. Rather, the focus of this study is the application of BERT for email classification, implemented with the BERT Tokenizer. The BERT Tokenizer with Tensorflow 2.0 can be employed to build more efficient and effective NLP models.

TensorFlow is an ML platform available as open source. The platform is used with training data for developing ML models in a short duration of time [17]. TensorFlow enables users to create intricate models from scratch through the use of wide-range of methods and classes. The current version, TensorFlow 2.0, has significant enhancements compared to the basic version. Due to its several features, TensorFlow has received notable attention by the machine learning researchers. These features include ease of use, flexibility in deploying models on web browsers and mobile devices, and python interface [17]. While the primary use of TensorFlow is for machine learning applications, the platform is also actively used for tasks that do not directly fall in the domain of machine learning, such as numerical computing using data flow graphs [17].

In this study, we used a pre-trained BERT model (BERT encoder model from SavedModels from TF Hub in TensorFlow 2.0) and applied the transfer learning technique as a hidden layer in a deep neural network in order to execute the pre-trained model on our dataset. The pre-training phase takes significant computational time. Due to this reason, it is very useful to use pre-trained model and later fine-tune it on defined dataset.

Figure 2 shows the overall flow of our proposed algorithm. During the first stage, the raw email corpus (containing Enron 1 to Enron 6) is transformed into CSV format document. This is passed to feature engineering stage where

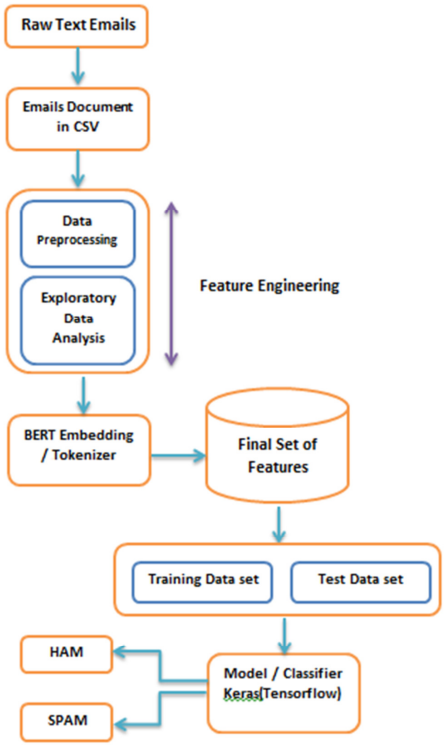

Figure 2. Classification Model of Email filtering with BERT + TF 2.0

data pre-processing and FDA techniques are applied on it to get pre-processed data. Furthermore, pre-processed data is fed to BERT tokenizer along with Embedding (transformer model) stage to generate final set of features. The next step is to apply padding on our final set of features and divide into test and training data sets. Finally, the training data sets are forwarded into the classifier to train the model which segregates the ham and spam messages. This segregation is then used to assess the performance of our proposed model on the test dataset.

E. Performance Evaluation

Accuracy and *Loss* are used as the performance measures. Accuracy gives a measure of the correct classification. The Loss function tells how good the model is in predictions. A low value of Loss indicates that the model predictions are closer to the actual values, and vice versa.

In order to measure the accuracy and loss, the input data need to be found. The input data is given by the following equation:

$$D = \sum_{i=1}^{n} C_i \tag{1}$$

where C is dataset corpus which contain raw email in text format, i is number of raw text file in dataset corpus, and D is a list of raw text files.

Similarly, in order to take output and find Loss, we use binary cross entropy [6] as follows:

$$Loss(P, z) = -\frac{1}{n}\sum_{i=1}^{n}[z_i \ log(P_i) + (1 - z_i) \ log(1 - P_i)] \tag{2}$$

where z_i is the corresponding target value, P_i is a predicted probability vector for all e-mails in testing dataset, and n is the number of lines in the tested e-mail. Here, *Loss* is showing the difference between predicted value by our model and the true value of data.

In our model, Accuracy is one of the metrics to measure the evaluation and performance, and is mathematically represented as

$$Accuracy = \frac{Number \ of \ Correct \ Classifications}{Total \ Number \ of \ Classifications} \tag{3}$$

4. Results and Discussion

The data used for training was divided into 80% for training, while 20% data was used for validation. The Adam optimizer [18] was used. The Adam optimization approach is fundamentally an extension of the stochastic gradient descent algorithm and has been widely adopted for the optimization of deep learning methodologies. The input of dropout layer specified rate size at each step is arbitrarily equal to zero, consequently avoiding the model from over-learning and data over-fitting. Dropout of 0.2 was used. Two different input parameters were used during the performance evaluation. These parameters were batch sizes and number of epochs. Table II gives the details of values used for these parameters. This was done to see the impact of parameter sensitivity and to fine-tune the parameters. Tables III, IV, and V show the results of training and testing datasets using the proposed algorithm (BERT with TensorFlow 2.0). Columns two and three of the tables show the loss and accuracy of training data, respectively. Similarly, columns four and five provide loss and accuracy of the testing data, respectively.

In Table III, a batch size of 64 and 5 epochs were used. As seen from the table, the loss was very low, ranging between 0.000115 and 0.000867, while the accuracy was 100% for the training data. With regard to the loss and accuracy of the testing data, the loss ranged between 0.064 and 0.290 as seen from column seven of the table. More importantly, the accuracy ranged between 86.56% and 97.27%. For the same batch size (i.e. 64), when the epochs were increased to 40, a reduction in loss of training data was observed. However, for the testing data, the results for

TABLE II. Combinations of input parameters

Name	Batch size	Epochs
Set 1	64	5
Set 2	64	40
Set 3	32	10

TABLE III. Loss and Accuracy rates for training and testing data using batch size = 64, Epochs = 5

Dataset	Loss (Train)	Accuracy % (Train)	Loss (Test)	Accuracy % (Test)
Enron 1	7.85×10^{-4}	100	0.29	86.56
Enron 2	1.60×10^{-4}	100	0.17	93.75
Enron 3	1.15×10^{-4}	100	0.16	89.38
Enron 4	8.67×10^{-4}	100	0.19	88.75
Enron 5	2.80×10^{-4}	100	0.06	97.27
Enron 6	4.44×10^{-4}	100	0.17	94.06

loss and accuracy were more or less the same as those of batch size 64 and 5 epochs, as seen in Table IV.

In Table V, when the batch size was reduced to 32, and the number of epochs was increased to 10, the loss in training data ranged between 0.0000167 and 0.000091, while the accuracy in the training data was maintained at 100%. Furthermore, the loss in testing data varied between 0.0692 and 0.3812, while the accuracy ranged between 89.20% and 95.54%.

Table VI summarizes the best performer in the different categories for each dataset. For example, column 2 of Table VI indicates that with regard to the loss in training data, Set 3 (batch size =32 and epochs = 10) showed the lowest values in loss for all six datasets. However with regard to loss in training data, the trends were different. As seen in column 3 of the table, Set 1 was dominant, showing the lowest loss in four datasets (Enron 1, Enron 2, Enron 3, and Enron 6). For Enron 4 and Enron 5, the best results were displayed by Set 3 and Set 2, respectively. As far as accuracy in testing data is concerned, Set 2 and Set 3 showed the highest accuracy for three datasets each. Set 2 (batch size = 64, epochs = 40) was the best for Enron 3, Enron 5, and Enron 6, whereas Set 3 (batch size = 32, epochs 10) produced the best accuracy for Enron 1, Enron 2, and Enron 4 datasets.

TABLE IV. Loss and Accuracy rates for training and testing data using batch size = 64, Epochs = 40

Dataset	Loss (Train)	Accuracy % (Train)	Loss (Test)	Accuracy % (Test)
Enron 1	1×10^{-4}	100	0.45	84.69
Enron 2	1×10^{-4}	100	0.22	93.75
Enron 3	1×10^{-4}	100	0.16	95.94
Enron 4	1×10^{-4}	100	0.18	87.84
Enron 5	5×10^{-4}	100	0.06	97.66
Enron 6	1×10^{-4}	100	0.33	94.27

TABLE V. Loss and Accuracy rates for training and testing data using batch size = 32, Epochs = 10

Dataset	Loss (Train)	Accuracy % (Train)	Loss (Test)	Accuracy % (Test)
Enron 1	5.16×10^{-5}	100	0.38	86.76
Enron 2	9.10×10^{-5}	100	0.18	95.54
Enron 3	5.31×10^{-5}	100	0.18	89.20
Enron 4	5.06×10^{-5}	100	0.17	90.30
Enron 5	1.67×10^{-5}	100	0.07	90.27
Enron 6	2.11×10^{-5}	100	0.23	92.90

TABLE VI. Best performers for different datasets

Dataset	Loss (Training)	Loss (Testing)	Accuracy (Testing)
Enron 1	Set 3	Set 1	Set 3
Enron 2	Set 3	Set 1	Set 3
Enron 3	Set 3	Set 1	Set 2
Enron 4	Set 3	Set 3	Set 3
Enron 5	Set 3	Set 2	Set 2
Enron 6	Set 3	Set 1	Set 2

In this paper, the impact of epochs and batch size on training dataset is evaluated. Both batch size and epochs are critical hyper-parameters used in tuning deep learning and transformer model. In our model, we tuned the epochs (increase or decrease) to overcome the problem of under-fitting, overfitting and to achieve optimum solution. While increase in epochs added training time, it also helped in enhancing the accuracy of our model (i.e. epochs of 10 and 40 producing higher accuracy than epoch 5). With regard to loss, a small epoch size (in testing phase for loss) was more effective. With regard to the impact of batch size, the effect was somewhat mixed; both batch sizes of 32 and 64 were equally good, and the results do not favor a specific batch size. Note that the current study is preliminary and more experimentation with different batch sizes and the number of epochs is required.

Figures 3 and 4 demonstrate the trends for accuracy for Set 1 and Set 3 respectively. Each plot in the figures compares the trends of training and testing data. It can be seen from Figure 3 that the trends for training and testing data for batch size of 32 are quite similar to each other for Enron 2 and Enron 5. For batch size of 64, similar trends for Enron 2, Enron 5, and Enron 6 can be observed in Figure 4. Thus, it can be stated that the proposed algorithm did fairly good for half of the test cases, and further improvement is desired and possible.

With regard to the loss metric, Figures 5 and 6 depict the trends for Set 1 and Set 3 respectively. In Figure 5, the trends for batch size of 32 are observed, where the

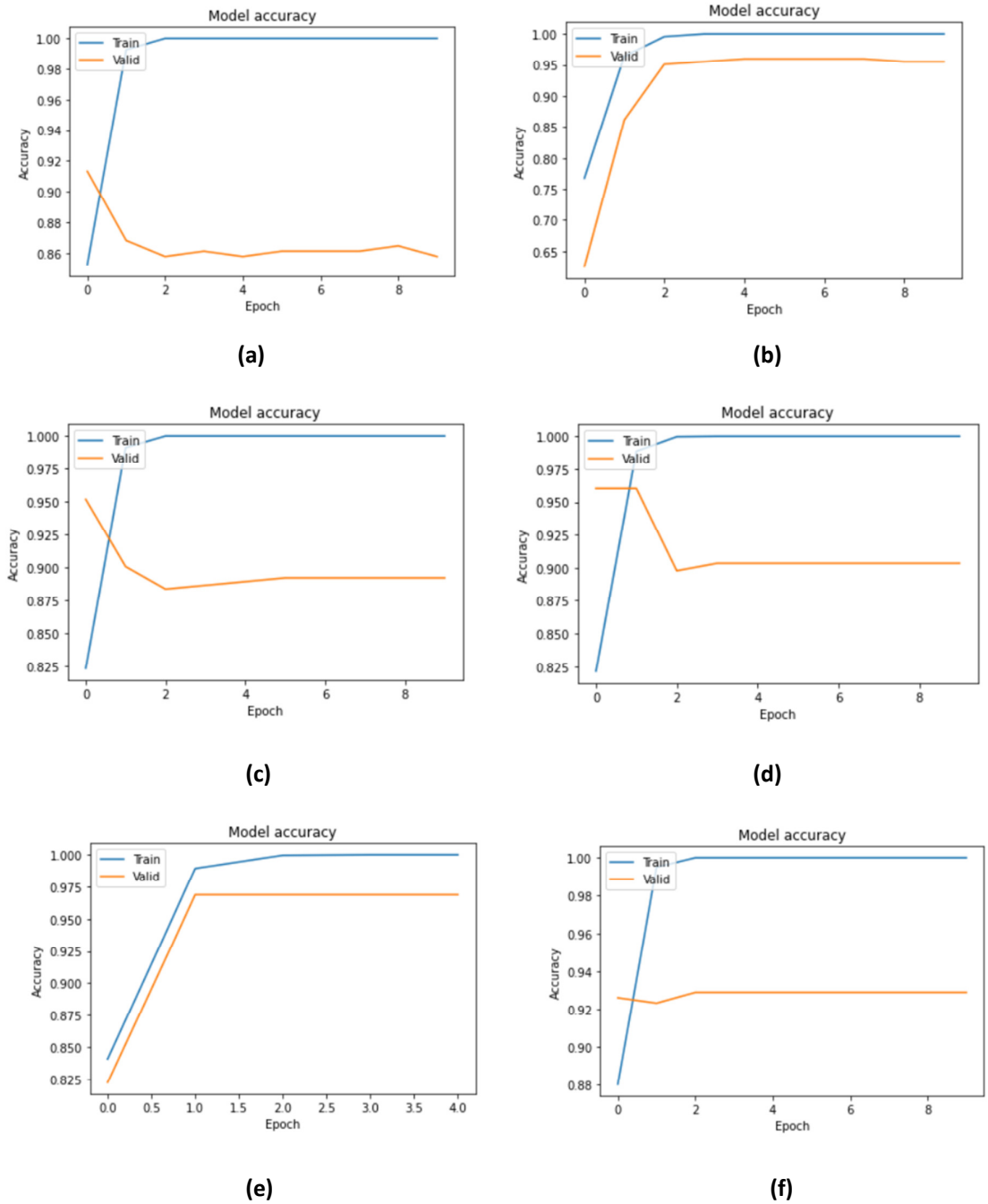

Figure 3. Plots for accuracy for batch size = 32, epochs = 10 (a) Enron 1 (b) Enron 2 (c) Enron 3 (d) Enron 4 (e) Enron 5 (f) Enron 6

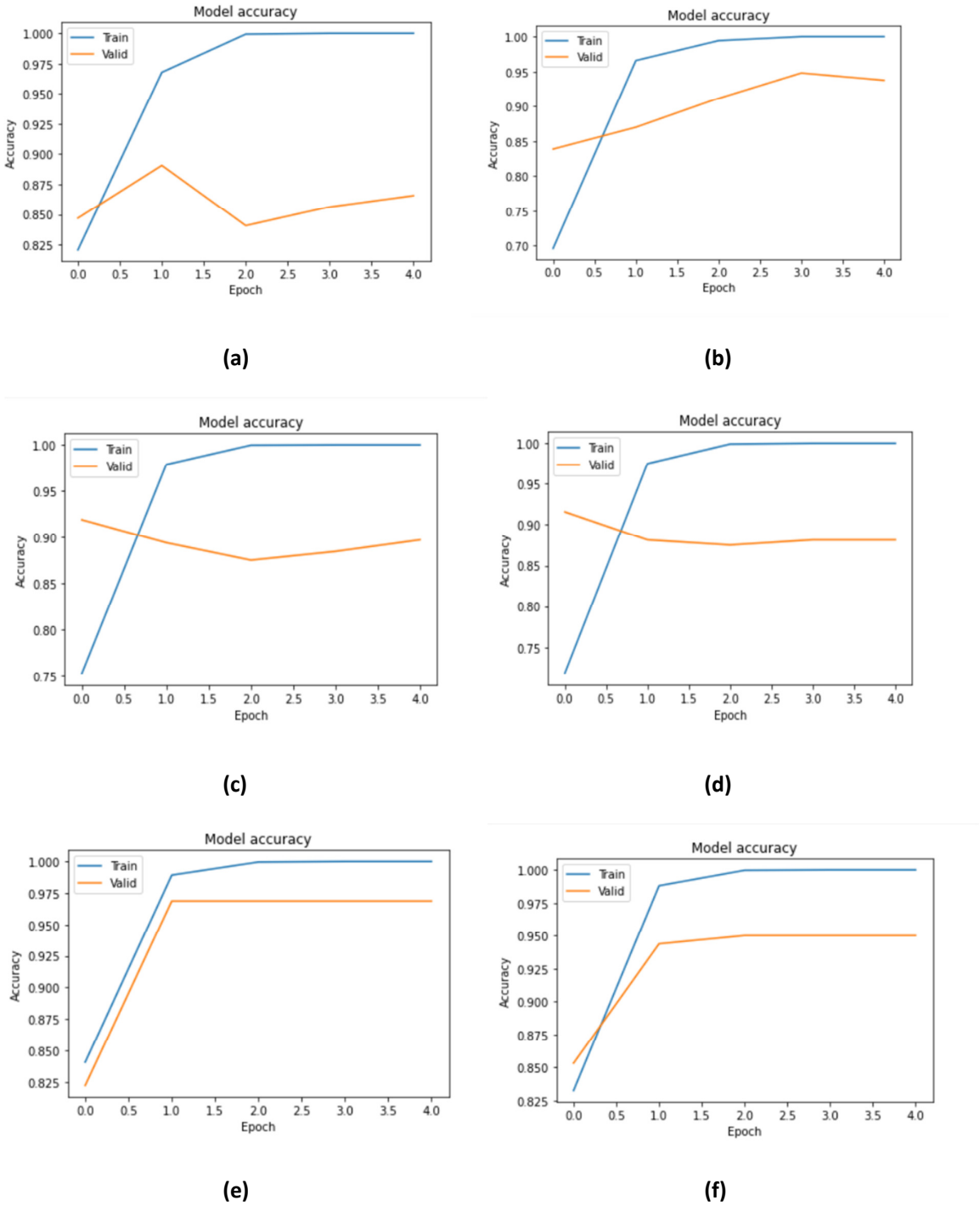

Figure 4. Plots for accuracy for batch size = 64, epochs = 5 (a) Enron 1 (b) Enron 2 (c) Enron 3 (d) Enron 4 (e) Enron 5 (f) Enron 6

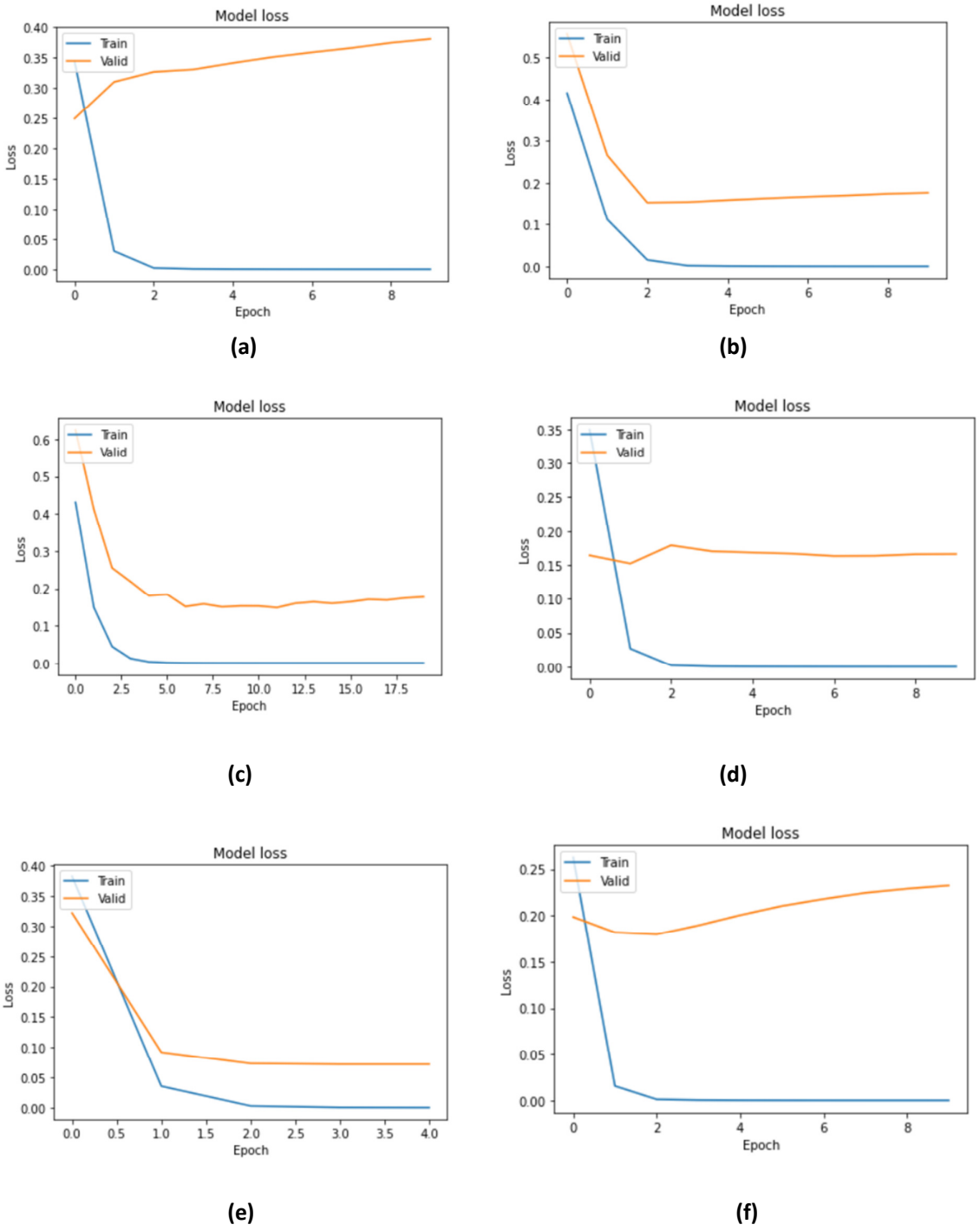

Figure 5. Plots for loss for batch size = 32, epochs = 10 (a) Enron 1 (b) Enron 2 (c) Enron 3 (d) Enron 4 (e) Enron 5 (f) Enron 6

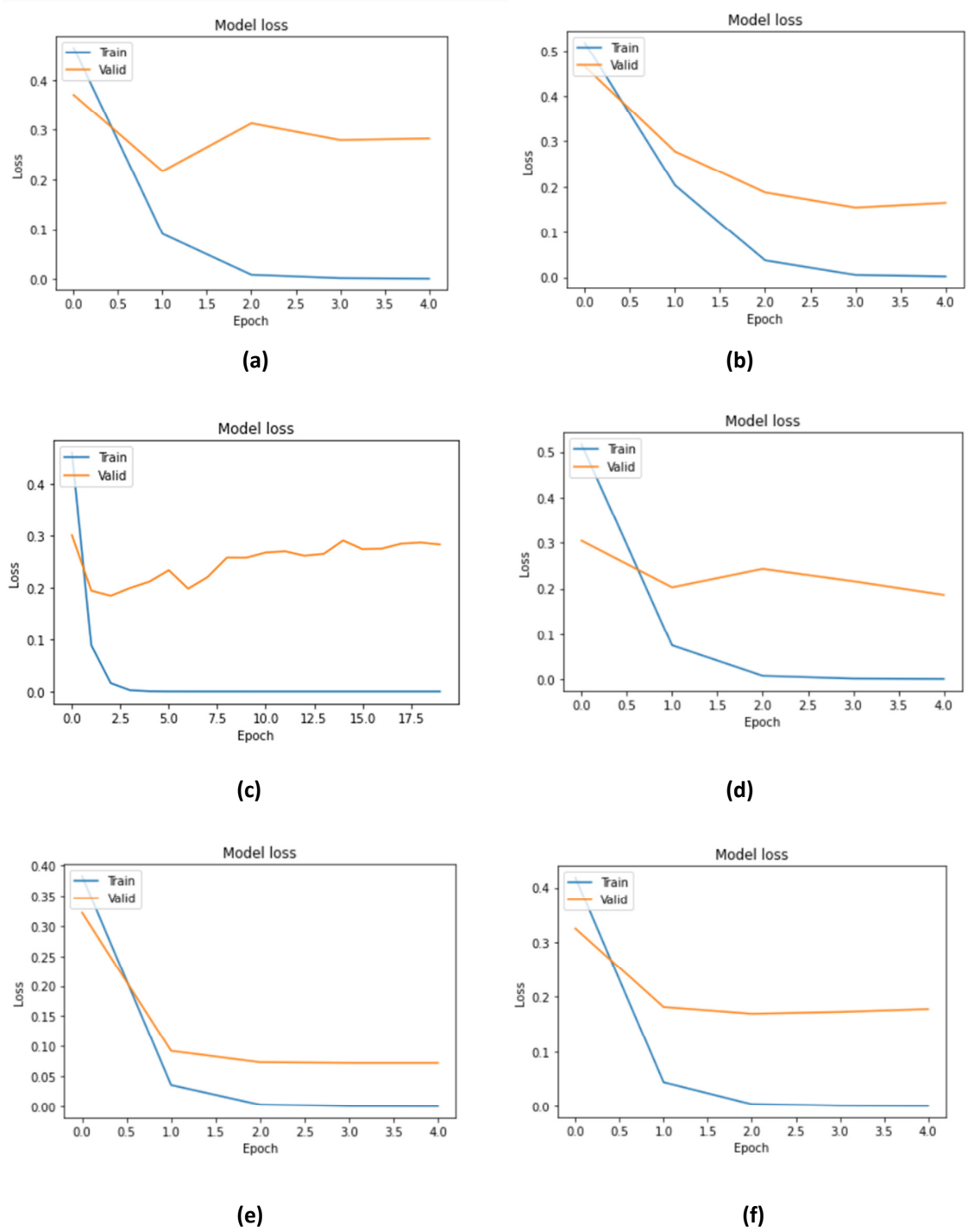

Figure 6. Plots for loss for batch size = 64, epochs = 5 (a) Enron 1 (b) Enron 2 (c) Enron 3 (d) Enron 4 (e) Enron 5 (f) Enron 6

deviation in loss for training and testing data was quite low for Enron 2, Enron 3, and Enron 5 datasets. When the batch size was increased to 64, similar trends were observed; the deviation for Enron 2 and Enron 5 was also low. Therefore, it can be fairly claimed that the proposed algorithm showed a reasonable performance with regard to the loss metric, although better results are desired.

5. Conclusion

E-mail spam detection is a significant area of research in the domain of data security. Various techniques have been proposed over the years to effectively address the issue of email spam detection. Deep learning has shown promising results for various ML applications, including NLP. This study has addressed the issue of effective email detection using a deep learning technique known as BERT. This technique is combined with TensorFlow 2.0 and the proposed algorithm is assessed using Enron dataset, while using *accuracy* and *loss* as the assessment metrics. Preliminary results with different batch sizes and number of epochs are promising, though a more in-depth analysis is required.

In terms of future studies, several research directions can be highlighted. The impact of batch size and number of epochs on the algorithm performance can be evaluated by taking a variety of values for the two parameters. Also, several performance metrics, such as precision, recall, and F1 score can also be considered for a more comprehensive performance evaluation. Furthermore, apart from Enron, several other datasets can be considered. A comparative analysis with other algorithms is also an appealing direction of research.

Acknowledgment

This research was supported by Cyerbersecurity Center at Prince Mohammad bin Fahd University under grant PCC-Grant-202116.

References

[1] W. Feng, J. Sun, L. Zhang, C. Cao, and Q. Yang, "A support vector machine based naive bayes algorithm for spam filtering," in *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2016, pp. 1–8.

[2] S. Chakraborty and B. Mondal, "Spam mail filtering technique using different decision tree classifiers through data mining approach-a comparative performance analysis," *International Journal of Computer Applications*, vol. 47, no. 16, 2012.

[3] M. Rathi and V. Pareek, "Spam mail detection through data mining-a comparative performance analysis," *International Journal of Modern Education and Computer Science*, vol. 5, no. 12, p. 31, 2013.

[4] D. Mallampati and N. P. Hegde, "A machine learning based email spam classification framework model: Related challenges and issues," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 4, pp. 3137–3144, 2020.

[5] N. F. Rusland, N. Wahid, S. Kasim, and H. Hafit, "Analysis of naïve bayes algorithm for email spam filtering across multiple datasets," in *IOP conference series: materials science and engineering*, vol. 226, no. 1. IOP Publishing, 2017, p. 012091.

[6] S. Srinivasan, V. Ravi, M. Alazab, S. Ketha, A.-Z. Ala[U+0092]M, and S. K. Padannayil, "Spam emails detection based on distributed word embedding with deep learning," in *Machine Intelligence and Big Data Analytics for Cybersecurity Applications*. Springer, 2021, pp. 161–189.

[7] R. Hassanpour, E. Dogdu, R. Choupani, O. Goker, and N. Nazli, "Phishing e-mail detection by using deep learning algorithms," in *Proceedings of the ACMSE 2018 Conference*, 2018, pp. 1–1.

[8] A. Bibi, R. Latif, S. Khalid, W. Ahmed, R. A. Shabir, and T. Shahryar, "Spam mail scanning using machine learning algorithm." *J. Comput.*, vol. 15, no. 2, pp. 73–84, 2020.

[9] Q. Yaseen *et al.*, "Spam email detection using deep learning techniques," *Procedia Computer Science*, vol. 184, pp. 853–858, 2021.

[10] A. Mosavi, S. Ardabili, and A. R. Varkonyi-Koczy, "List of deep learning models," in *International Conference on Global Research and Education*. Springer, 2019, pp. 202–214.

[11] J. Hu, "An overview of text representation in nlp," in *https://towardsdatascience.com/an-overview-for-text-representations-in-nlp-311253730af1*, 2020, p. Accessed 3 Jan 2022.

[12] in *https://albertauyeung.github.io/2020/06/19/bert-tokenization.html*, 2020, p. Accessed 3 Jan 2022.

[13] in *https://www.cs.cmu.edu/ enron/*, p. Accessed 22 Dec 2021.

[14] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam filtering with naive bayes-which naive bayes?" in *CEAS*, vol. 17. Mountain View, CA, 2006, pp. 28–69.

[15] R. Bekkerman, "Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora," 2004.

[16] G. Liu and J. Guo, "Bidirectional lstm with attention mechanism and convolutional layer for text classification," *Neurocomputing*, vol. 337, pp. 325–338, 2019.

[17] P. Janardhanan, "Project repositories for machine learning with tensorflow," *Procedia Computer Science*, vol. 171, pp. 188–196, 2020.

[18] A. Botchkarev, "Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology," *arXiv preprint arXiv:1809.03006*, 2018.

Kashif Iqbal is doing Masters in Computer Science from Karachi Institute of Economics Technology, Karachi, Pakistan. .He is working as a Lecturer at Roots / TMUC. His research interests lie in Machine Learning, Deep Learning, NLP, Sentiment Analysis, Automated Reasoning and Quantum Science.

Salman A. Khan received BS and MS in Computer Engineering from KFUPM, Saudi Arabia, He has a Ph.D. in computer science from University of Pretoria, South Africa. He is a Professor in College of Computing and Information Sciences, Karachi Institute of Economics Technology (KIET), Pakistan. He has more than 60 publications in peer-reviewed journals and conferences. His expertise is in evolutionary computation, optimization, and fuzzy logic. His interests are in application of these areas in cyber security, renewable energy, and smart cities.

Shamim Anisa is doing Masters in Computer Science from KIET, Pakistan. She is also working as Senior Software Engineer at Gaditek. She has over 3+ years of experience in Web Development and Data Management. Her research interests are ML, DL, and Optimization Algorithms.

Ayesha Tasneem is doing Masters in Software Engineering from KIET, Pakistan. She is experienced Senior Business Intelligence Developer and is skilled in Data Analytics, Business Intelligence and SQL.

Nazeeruddin Mohammad is an Associate Professor in Computer Engineering Department and the Director of Cybersecurity Center at Prince Mohammad bin Fahd University, Saudi Arabia. He received MS from KFUPM, Saudi Arabia, Ph.D. from University of Ulster, UK. He has over 20 years of experience in industrial and academic research, university-level teaching, Infrastructure design and management and proto-type development in simulation real world test-bed environments.