European Journal of Information Technologies and Computer Science
www.ej-compute.org

Performance Evaluation of LSTM and RNN Models in the Detection of Email Spam Messages

Elijah John-Africa and Victor T. Emmah

ABSTRACT

Email spam is an unwanted bulk message that is sent to a recipient's email address without explicit consent from the recipient. This is usually considered a means of advertising and maximizing profit, especially with the increase in the usage of the internet for social networking but can also be very frustrating and annoying to the recipients of these messages. Recent research has shown that about 14.7 billion spam messages are sent out every single day of which more than 45% of these messages are promotional sales content that the recipient did not specifically opt-in. This has gotten the attention of many researchers in the area of natural language processing. In this paper, we used the Long Short-Time Memory (LSTM) for classification tasks between spam and Ham messages. The performance of LSTM is compared with that of a Recurrent Neural Network (RNN) which can also be used for a classification task of this nature but suffers from short-time memory and tends to leave out important information from earlier time steps to later ones in terms of prediction. The evaluation of the result shows that LSTM achieved 97% accuracy with both Adams and RMSprop optimizers compared to RNN with an accuracy of 94% with RMSprop and 87% accuracy with Adams optimizer.

Keywords: Adams, LSTM, RMSprop, RNN, unsupervised learning.

Published Online: November 26, 2022

ISSN: 2736-5492

DOI :10.24018/ejcompute.2022.2.6.80

E. John-Africa
Faculty of Engineering, Environment, and Computing, Department of Computer Science, Coventry University, United Kingdom.
(e-mail: africae@uni.coventry.ac.uk)
V. T. Emmah*
Department of Computer Science, Rivers State University, Port Harcourt, Nigeria.
(e-mail: victor.emmah@ust.edu.ng)

****Corresponding Author***

I. Introduction

In recent years the use of email as a means of communication has increasingly developed. With more than 2.6 billion active users and 4.6 billion email accounts in service, email is the most relevant and commonly used Internet communications medium [1]. Just as the use of email has increased and developed the sending of unsubscribed messages known as spam has also increased, mostly for advertising, and increased the return of investment in various businesses. With about 12.5 million spam emails sent out, at least one person responds. That might not sound like much – until you consider that there are more than 14 billion spam messages sent out every day [2]. With only one reply per 12.5 million emails sent, spammers earn approximately $3.5 million over one year from spam mails. Email spam might cost a sender little time to send out, but most of the costs are paid by the recipients of this email. Spam cost can be measured in human loss time, Server loss time, and loss of valuable mail/messages [3]. The server loses time in the sense that when emails are sent, it takes space on the server for storage. If more of these emails are spam, it creates very little space causing the server a valuable space to store ham or useful messages. It can also cause the reader valuable time of reading resourceful messages. Apart from all these losses, there are other indirect losses like bandwidth drain. Not only can email spam affect your productivity and lead to security breaches but if you mistakenly send it to your contacts, it can also cost you a heavy fine.

According to the Privacy and Electronic Communications Regulations of 2003. On the 9th of July 2019, the UK Information Commissioner Office (ICO) announced a £100,000 fine on EE telecoms company for breaching the Privacy and Electronic Communications Regulations 2003. There are various algorithms to determine spam messages but most of the problem leads to overfitting. To overcome such a problem, we implemented the RNN and LSTM models which are deep learning techniques used in sequence classification and document modeling. This paper will be beneficial to researchers in email spam detection and classification Using LSTM or any other models. In this paper, the LSTM model will be used to predict if an email message is a spam or not.

Email Spam proliferation has become a potential challenge to email's credibility as a secure and effective means of Internet communication. A lot of companies including oil multinationals, IT companies, and manufacturing industries to mention but have fallen victims to the menace, leading to a denial of service, directory harvesting, and phishing attacks that directly cause financial losses [4]. A report from ponemon institute in 2016 shows that successful phishing attack on large companies with over 10,000 employees stands at $3.7 million per attack despite the deployment of security solution designed to prevent such attack [5].

Most sophisticated email spam prevention systems can prevent some malicious information from going through, however, if the message does not have the necessary trigger it will just push right past the software detection. According to [6], the Washington Post, companies spent about $6.5

billion every year on anti-spam technology to reduce the effect of email spam in their organization. The Japanese GDP was affected in 2018, causing a decrease in the total amount of 500 billion Japanese Yen due to email spam [7].

These effects and more have been the motivation behind this research, if properly applied, this research will go a long way in reducing unsolicited emails known as spam which has become a major problem to individuals and organizations to a manageable size.

II. Literature Review

Training a model requires a significant amount of training dataset used for categorizing Labeled models. There are a lot of machine learning and ANN techniques suitable for text classification problems such as the one we are looking at. Some of them include Bayes Classifier, nearest neighbor, SVM, C4.5, and decision trees. The Naïve Bayesian spam classification is one of the spam classification algorithm families.

A study by reference [8] shows that spam filtering techniques are based on text or email classification. They developed a simple but highly effective email spam filtering program called '*spamco*' which employs a naïve Bayesian algorithm to determine if an email is spam or not. Among the two papers presented at the event, their filter was the more effective with a success rate of 92% with 1.16% false positives. However, they used all the tokens which can create a vulnerable point for spammers to break through or mislead the algorithm by adding a large chunk of text to counterbalance the spam terms. Also, they were not prejudiced against erroneous positives. Any spam filtering algorithm, in my opinion, should include a handy knob you can turn to reduce the false positive rate at the price of the filtering rate.

False positives are innocent emails that get mistakenly identified as spam. Paul Graham's Algorithm detected 99.5% spam with 0.03% false positives which he claimed a large dataset was the reason behind the 99.5% detection success. [9] in their work, different classifiers were used to determine whether an email is Ham or spam. They separated the dataset into two-part, one part for training and the other part for testing. After using a 10-fold cross-validation method with three classifiers from the WEKA application, an accuracy of 93% with the MLP algorithm was recorded and with a simple logistics algorithm, an accuracy of 92% was recorded.

Reference [10] elucidates and compares the performance of several email spam classification Techniques using different datasets such as the Enron spam corpus dataset, SpamAssassin [14], and UCI machine learning repository Spambase dataset [15]. Some of the methods they applied before comparing their performances are the J48 classifier, K-Nearest Neighbor, Naive Bayes, Artificial Neural Network, Support Vector Machine, and Random Forest. After conducting testing and training on a spam base dataset, random forest with spam base dataset performs better in email spam classification with 87% accuracy than other machine learning methods mentioned in the experiment [11].

Reference [12] compared traditional machine learning algorithms' performance to deep learning using different data from previous research works to ascertain their performance based on their accuracy, precision, recall, and CAP Curve (cumulative accuracy profile). In their comparison, they used Support Vector Machine (SVM), Naive Bayes (NB), Decision Tree (DT), Logistic Regression (LR), Random Forest (RF), Artificial Neural Network (ANN) and Convolutional Neural Network (CNN) to evaluate their performance on different datasets. After a critical evaluation of the methods applied, Convolutional Neural Network (CNN) achieves the highest accuracy of 99.19% and 98.25% and AR (autoregressive) value of 0.9926 and 0.9994 for the two datasets machine learning algorithms.

Reference [13] suggested a brief and concise study for email classification. He added that algorithms like Neural Network, SVM, Naive Bayesian, and J48 classifiers are used to filter spam from the datasets of emails. Neural networks have to go through processes such as data preprocessing, data training, and testing. Feature selection in this process helps to determine the unimportant features and reduces them. The final features are refined and sent to the neural networks, adding weights, and a function is passed into the network to build a classifier. Another algorithm such as SVM is very successful in learning tasks such as generalization. SVM helps to learn from features and the features help to classify the output based on the algorithm. The classes in SVM are separated using the optimum hyperplane. Another tree algorithm that is also very useful in spam filtering is J48 which is a decision tree algorithm. It helps to create a binary tree and is capable to classify the email as spam or ham.

III. Problem and Dataset

This paper is aimed at performing text classification using RNN and LSTM and to determine which of the Neural network algorithms (LSTM and RNN) will perform better with both Adams and RMSprop optimizers in classifying email as spam or not. Fig. 1 shows a sample of the dataset acquired from an open-source database called spam base UCI. The dataset comprises 4825 emails categorized as Ham (Non-spam messages) and 747 spam messages.

	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Fig. 1. Sample of Dataset from Spam-based UCI.

A. Data Pre-Processing

Most data in their state of Acquisition are inadequate for training and testing of any model without pre-processing. An efficient result depends on the good quality of the dataset and efficient mining. Therefore, pre-processing which includes cleaning, integration, transformation, and reduction are major tasks in data pre-processing that must be carried out before making use of the dataset. During the data pre-processing, the Non-ASCII characters were removed and replaced with empty spaces. We also encoded spam and ham to 0 and 1

labels respectively, to give the computer an idea of how to go about fitting the dataset into machine learning. To do that we applied a technique called feature vectorization that transforms all the emails into a feature vector which is a mathematical way of representing strings.

IV. METHODOLOGY

The architecture is shown in Fig. 2 & Fig. 3.

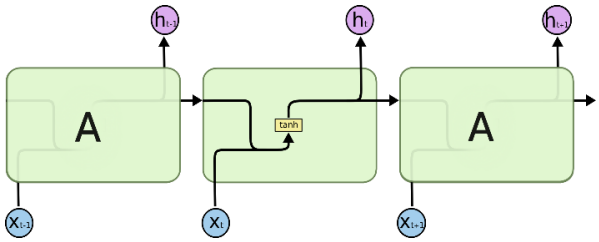


Fig. 2. Simple RNN Architecture.

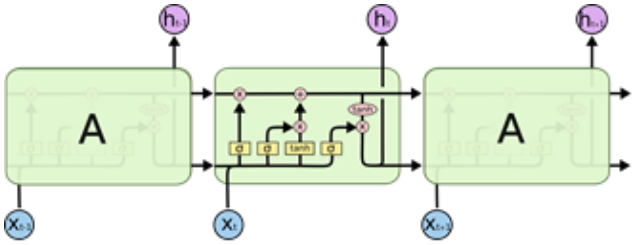


Fig. 3. LSTM Architecture containing four interacting layers.

The architecture consists of a sequence of input marked as email to LSTM and RNN architecture which classifies the last output according to its zero or one as spam or ham message. LSTM was introduced to address the problem of the short time dependence of RNN. LSTM can learn to bridge the time intervals in excess of 1000 steps back and forth. In a condition where there is a small difference between the relevant information and the area it is needed, RNNs can learn to use the past information to predict the present or the future; but in a condition where we need to store more context to predict the present or 20 steps ahead, LSTM comes into the picture. Remembering information for a long period is practically the default behavior of LSTM, not something to learn [1].

Humans don't start their thinking from scratch every second. The understanding of what you are reading right now is based on that of the previous words just like an Episode movie. There are various deep learning algorithms used in sequence modelling. However, in this paper, we used LSTM and RNN models. Fig. 2a shows an RNN structure with loops.

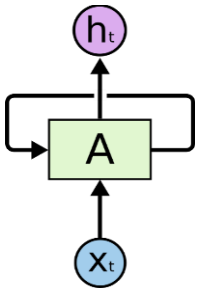


Fig. 2a. An RNN with Loops.

The label section A (loop) receives input from xt and output a value ht. RNN can be seen as multiple copies of the same Network, each passing information to the next network. Fig. 2b shows what happens if we unroll the loop in Fig. 2a.

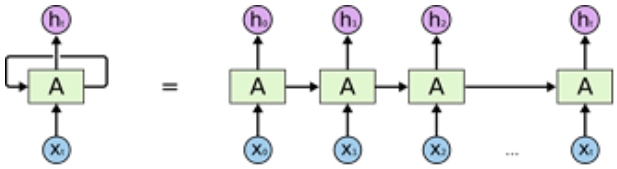
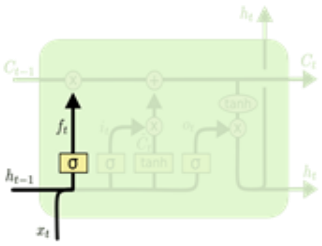


Fig. 2b. An Unrolled RNN.

Many feats can be achieved with RNN. It can be used for text classification which can handle text sequence irrespective of any length and handle long-term dependencies. It can also be used for speech recognition, language modeling, translation, image capturing, the list goes on. However, RNN can give rise to a problem such as exploring gradient or vanishing gradient. For such problems, LSTM or Long Short-Term Memory is used which can be able to remember the sequence for a long time. In sentence modeling, LSTM proved to have received astonishing performance.

Fig. 3a shows an LSTM structure with sigmoid function.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

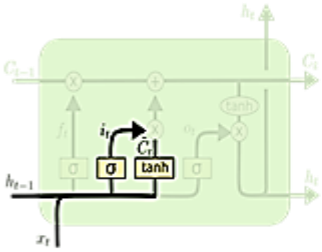
Fig. 3a. An LSTM Structure with Sigmoid Function.

One thing that differentiates LSTM from RNN is a sigmoid layer called the "forget gate". In LSTM the "forget gate" decides what information to discard from the cell state at time t. The forget gate output a number between 0 and 1 for every number in the cell state Ct-1 decided by the sigmoid function considering the following:

ht-1 - the value of the last LSTM unit at time t-1

xt - the current input at time t

An output of 1 means such information should be kept while an output of 0 means such information should be discarded. After deciding what information to discard, the next step is to decide what to store in Ct-1. The sigmoid layer decides what to update while the tanh layer creates a vector of the new information represented as Ct as shown in Fig. 3b.



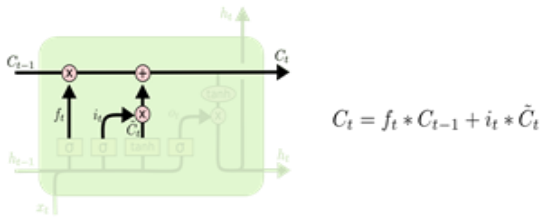
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fig. 3b. An LSTM with tanh & sigmoid functions.

To update Ct-1 (previous state) at time t, to Ct (present state) at time t, we have to multiply Ct-1 by ft. and then we add it *Ct which gives us a new value, scaled as to how much each State value we are trying to improve. where ft is a vector

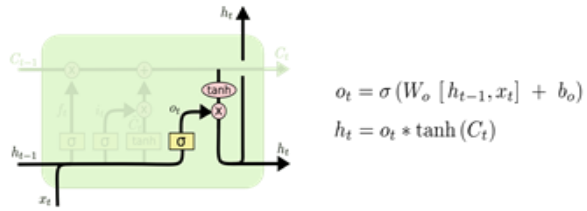
function with values ranging from 0 to 1, σ is a sigmoid function, wf and bf are the weight matrices and bias, respectively as shown in Fig. 3c.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Fig. 3c. An LSTM update state.

The output information (ht) which is the final step, is based on the output cell state (Ot). but a filtered version as shown in Fig. 3d.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Fig. 3d. The Output Information (ht,) filtered by tanh.

The sigmoid layer decides which information makes it to the output. Secondly the output of the sigmoid gate (Ot) is multiplied by a new value created by the tanh layer from the cell state Ct which pushes the value between -1 and 1 and Finally we multiplied by the output of the sigmoid gate so that we only output the section we set out to the output. In this model, we used the maxlen function in python where we provided the maximum sentence length to be considered while training the model.

V. Experimental Procedure

As stated earlier, the data was acquired from an open-source database called spam base UCI [14]. The dataset comprises 4825 emails categorized as Ham (Non-spam messages) and 747 spam messages. It was pre-processed to make it fit for training and validation of the models. The data was divided into two parts, 80% for training and 20% for testing.

In our model, we have defined a term called maxlen where we provided the maximum sentence length to be considered while training the model. As the sequence-based model requires fixed-length input, sequence padding was used to assign symbols to those sentences whose length does not meet maxlen. If the sentence length is greater than the maxlen, then the extra words will be chopped off and considered in the model. Most of the time, deep learning models tend to overfit in small data. Though our data is relatively small, to help prevent overfitting, a regularization technique known as dropout was applied. When dropout is considered, neurons that are randomly picked are often ignored during the training of the data. A small value of dropout is suggested. In this experiment, a dropout of 0.2 is applied.

An increase in losses is often a problem in deep learning training and most of the algorithms are born with it. To reduce losses, two optimizers namely Adam and RMSprop were used to reduce the losses and increase the accuracy rate. The Adam optimizer considers the mean and the uncentered variance of the gradients and it considers the decay rate of the past gradients exponentially. One of the advantages is that this method is quite fast and converges very rapidly. In terms of computational power, it is quite expensive. The RMSprop optimizer works similar to a standard gradient descent algorithm but it tries to prevent oscillation in a vertical direction. We can test with different learning rates to look into the performance and the convergence. To converge faster, we usually increase the learning rate.

VI. Results

To obtain the results, 80% of the dataset was applied to training and 20% was applied to testing and validation of the model. The results of the performance of the RNN model with RMSProp and Adams optimizers is presented and compared with the performance of LSTM with both optimizers.

A. RNN Model

The performance of the RNN Model with RMSprop optimizer is presented in Fig. 4a and Fig. 4b to show their training and validation accuracy.

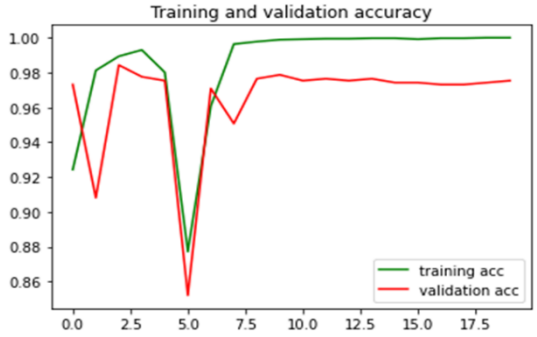


Fig. 4a. RNN training and Validation Accuracy with RMSprop Optimizer.

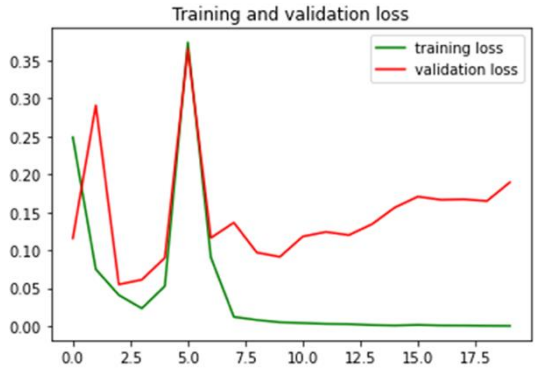


Fig. 4b. RNN training and Validation Loss With RMSprop Optimizer

After nearly 17 epochs we can see that the model is overfitting. Let us see the overall confusion matrix of the model.

```
[[982  15]
 [  3 115]]
```

From the confusion matrix, the rate of misclassification is very low. This is so because the data set is highly imbalanced with very few messages which are spam. The classification report of the RNN model training is shown in Fig. 6c.

	precision	recall	f1-score	support
0	1.00	0.98	0.99	997
1	0.88	0.97	0.93	118
accuracy			0.98	1115
macro avg	0.94	0.98	0.96	1115
weighted avg	0.99	0.98	0.98	1115

Fig. 4c. Classification Report of the RNN Model with Adams Optimizer.

From the classification report, we can see that the RNN model with RMSprop optimizer gives us an overall accuracy of about 94%. Now let us see the performance of the RNN model with the Adam optimizer. Fig. 4d shows the training and validation accuracy of the RNN model with Adams optimizer while Fig. 4e shows the training and validation loss of the RNN model with RMSProp optimizer.

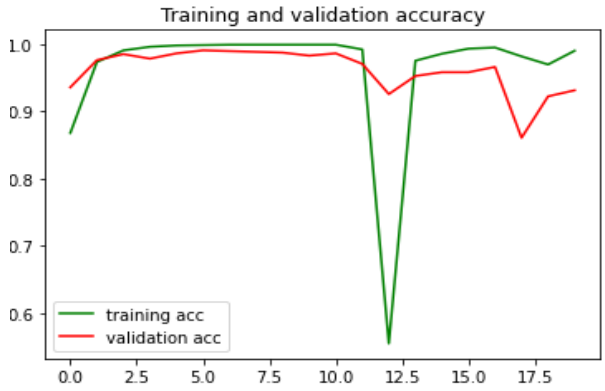


Fig. 4d. RNN Training and Validation Accuracy with Adams optimizer.

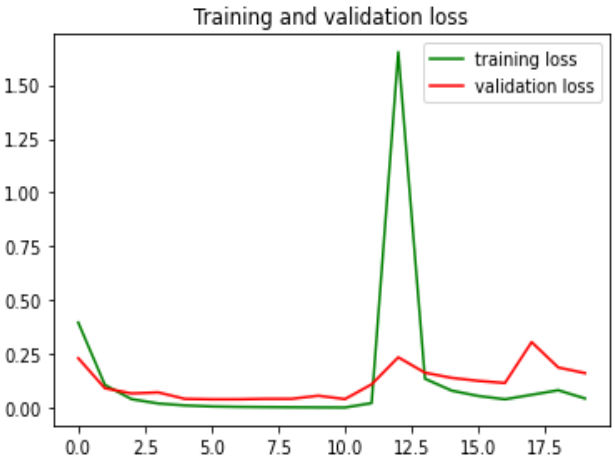


Fig. 4e. RNN Training and Validation Loss with Adams optimizer.

From the graph, we can see the training accuracy suddenly drops to around 60% when the epoch is around 12. Still, we are getting good accuracy on 20 epochs. Let's look into the confusion matrix.

```
[[923  20]
 [ 62 110]]
```

Compared to the model with the RMSprop optimizer, the rate of misclassification is high using the RNN model with the Adam optimizer. The classification report is shown in Fig. 4f.

	precision	recall	f1-score	support
0	0.94	0.98	0.96	943
1	0.85	0.64	0.73	172
accuracy			0.93	1115
macro avg	0.89	0.81	0.84	1115
weighted avg	0.92	0.93	0.92	1115

Fig. 4f. Classification Report of RNN model with RMSprop Optimizer.

We can see that the accuracy dropped to 89% whereas the model performed 94% with RMSprop optimizer.

B. LSTM Model

Now let us see how LSTM performs with RMSprop optimizers. Fig. 5a shows the LSTM training and validation accuracy with RMSprop Optimizer while Fig. 5b shows the LSTM training and validation loss with the same optimizer.

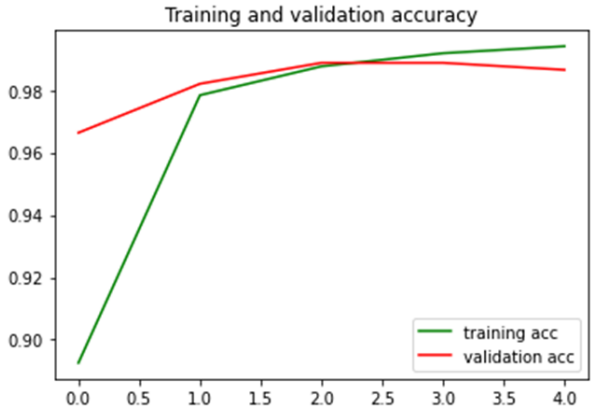


Fig. 5a. LSTM training and Validation Accuracy with RMSprop Optimizer.

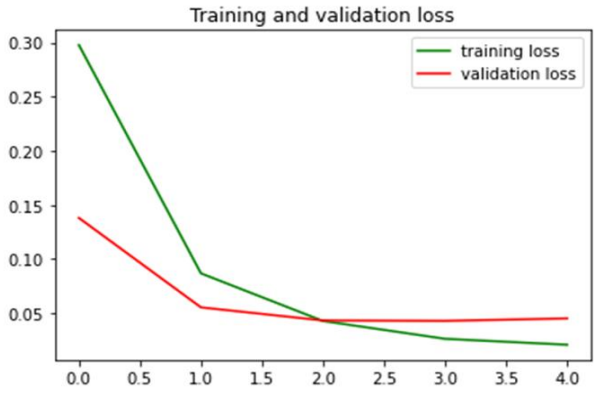


Fig. 5b. LSTM training and Validation Loss with RMSprop Optimizer.

In this case, only 5 epochs with a dropout of 0.2 were set. From the graph, it is evident that the model is giving very high accuracy and is also a case of overfitting.

Let us look into the confusion matrix.

```
[[981   6]
 [  4 124]]
```

The rate of misclassification is much lower compared to an RNN model with both of the optimizers. Hence it is a better model compared to the RNN model. The classification report of the LSTM model training with RMSprop optimizer is shown in Fig. 5c.

	precision	recall	f1-score	support
0	1.00	0.99	0.99	987
1	0.95	0.97	0.96	128
accuracy			0.99	1115
macro avg	0.97	0.98	0.98	1115
weighted avg	0.99	0.99	0.99	1115

Fig. 5c. Classification Report of LSTM model with RMSprop Optimizer.

From Fig. 5c, we see that the LSTM model is 97% accurate which is better when compared to the RNN model.

Now let us see how LSTM performs with Adam optimizers. Fig. 5d shows the training and validation accuracy of the LSTM model with Adams optimizer, while Fig. 5e shows the training and validation loss of the LSTM model with Adams optimizer.

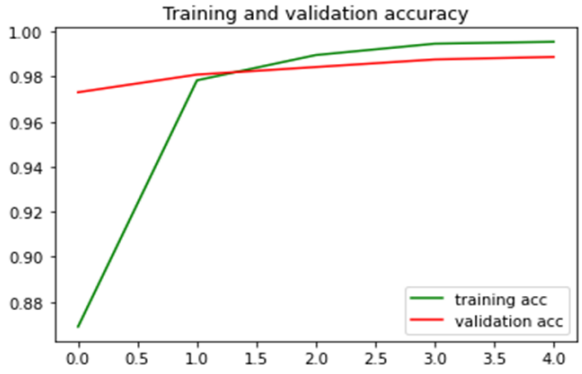


Fig. 5d. LSTM training and Validation Accuracy with Adams Optimizer.

The model performed almost similar to the previous one except for a slight misclassification in ham messages.

Fig. 5e. LSTM training and Validation Loss with Adams Optimizer.

	precision	recall	f1-score	support
0	1.00	0.99	0.99	989
1	0.94	0.97	0.95	126
accuracy			0.99	1115
macro avg	0.97	0.98	0.97	1115
weighted avg	0.99	0.99	0.99	1115

Fig. 5f. Classification Report of LSTM model with Adams Optimizer.

The model is somewhat similar compared to the model with RMSprop optimizer. Let us look into the confusion matrix to validate the same.

```
[[981   8]
 [  4 122]]
```

The model is 97% accurate. The only difference is the precision of spam message is little less compared to the precision of spam message of previous model with RMSprop optimizer.

VII. Discussion of Results

From our experiments, the LSTM model achieved an accuracy of 97% with both Adams and RMSprop optimizers which suggest a considerable improvement compared to the RNN algorithm with an accuracy of 94% with an RMSprop optimizer and dropped to 89% with Adams Optimizer.

Among the two models we have tested with different optimizers, the LSTM model with RMSprop optimizer gave us the best accuracy. Increasing further epochs can lead the model to overfit which is why dropout was used to prevent that. The model can be further improved by performing better text preprocessing methods such as replacing short texts with meaningful words. Such a process can be too hectic as it is quite difficult to perform such preprocessing in such a large amount of text.

VIII. Conclusion

Though we have built a model using different types of optimizers, we can see that the model is overfitting in increasing the number of epochs. The model can be improved only when a large amount of text messages is added. Different machine learning models have also proved to be efficient in giving us good accuracies such as Naïve Bayes and SVM model. Deep learning models such as LSTM are much more effective because of the complex nature to read and memorize the sequence. Different values of regularization can be applied to improve the model but, in this case, the model is still giving us an overfitting case. Though the model is highly imbalanced, more spam messages can be added for better improvement, or else the model could be more biased and more varied. Other optimizers such as AdaDelta or SGD can be used to see if there is an improvement in the model. But it is always a better choice to experiment with the model.

Conflict of Interest

Authors declare that they do not have any conflict of interest.

References

[1] Olah C. Understanding LSTM Networks-colah's blog. [Internet] [cited 2020 June 22] Github.io. Available from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] Hartley A. Spam gets 1 response per 12,500,000 emails. [Internet] [cited 2020 June 22] TechRadar. Available from:

https://www.techradar.com/news/internet/computing/spam-gets-1-response-per-12-500-000-emails-483381

[3] Alsmadi I, Alhami I. Clustering and classification of email contents. *Journal of King Saud University-Computer and Information Sciences*. 2015; 27(1): 46-57.

[4] Banday MT, Jan TR. Effectiveness and limitations of statistical spam filters. arXiv preprint arXiv:0910.2540. 2009 Oct 14.

[5] Sherman R. Financial Losses from Phishing. [Internet] [cited 2022 May 14] Available from: https://resources.infosecinstitute.com/topic/financial-losses-from-phishing/

[6] Plumer B. The Economics of Internet Spam. Washington Post. [Internet] [cited 2020 December 1] Available from: https://www.washingtonpost.com/news/wonk/wp/2012/08/10/the-economics-of-spam/

[7] Ukai Y, Takemura T. Spam mails impede economic growth. *The Review of Socionetwork Strategies.* 2007; 1(1): 14-22.

[8] Kalchbrenner N, Grefenstette E, Blunsom P. A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188. 2014 Apr 8.

[9] Vinitha V, Renuka DK. Feature selection techniques for email spam classification: a survey. *International Conference on Artificial Intelligence, Smart Grid and Smart City Applications.* 2019: 925-935.

[10] Vinitha V.S, Renuka D.K. Performance Analysis of E-Mail Spam Classification using different Machine Learning Techniques. *2019 International Conference on Advances in Computing and Communication Engineering (ICACCE).* 2019: 1-5.

[11] Gupta M, Bakliwal A, Agarwal S, Mehndiratta P. A comparative study of spam SMS detection using machine learning classifiers. *2018 Eleventh International Conference on Contemporary Computing (IC3).* 2018: 1-7.

[12] Bassiouni M, Ali M, El-Dahshan EA. Ham and spam e-mails classification using machine learning techniques. *Journal of Applied Security Research*. 2018; 13(3): 315-31.

[13] Enron spam corpu dataset, SpmAssassin. [Internet] Available from: https://www.kaggle.com/beatoa/spamassassin-public-corpus

[14] UCI machine learning repository Spambase dataset. [Internet] Available from: https://archive.ics.uci.edu/ml/machine-learning-databases/00228/

E. John-Africa studied Computer Science at the Rivers State University of Science and Technology, Nkpolu-Oroworukwo, Rivers State, Nigeria, where he obtained his Bachelor of Science degree. He later obtained his master's in computer science and another Maters in Data Analytics, in Coventry University and UlsterUniversity respectively, both in the United Kingdom. His research interest areas include big data analytics and machine learning.

V. T. Emmah obtained his Bachelor of Science degree (B.Sc.) in 2008 and Master of Science (M.Sc.) degree in 2012 all in Computer Science from the Rivers State University of Science and Technology and University of Port Harcourt, Nigeria respectively. He is currently a lecturer in the Department of Computer science, Rivers State University of Science and Technology, teaching and expanding his knowledge on computer programming and Computer security. He is holds a Doctor of Philosophy (Ph.D.) degree in Computer Science at the University of Port Harcourt. He has published over eighteen papers in different reputable online journals. His research interest includes Computer security and Machine learning. Dr. Emmah is a member of Computer Professional Registration Council (CPN) in Nigeria.