Comparative Study of Machine Learning Algorithms for SMS Spam Detection

Amani Alzahrani and Danda B. Rawat
Data Science and Cybersecurity Center (DSC^2)
Department of Electrical Engineering and Compouter Science
Howard University, Washington DC 20059, USA
amani.alzahrani@bison.howard.edu, danda.rawat@howard.edu

***Abstract*— The short message service (SMS) became popular after it was initially provided as a service in the second-generation (2G) terrestrial mobile network architecture (Global System for Mobile Communication – GSM). Its popularity has been exploited by some advertising companies and others to spread unwanted advertising, communicate advertising offers, and send unwanted material to the end users. These undesirable messages, known as spam, make it difficult for the users to receive the desirable messages and make them frustration and irritation. Consequently, there are measures that various experts have implemented in filtering out these spam messages and blocking them from reaching the end users. Most of the solutions have followed the success of email spam filtering and utilized machine learning techniques to filter spam messages. The popular machine learning techniques that have successfully been used include logistical regression, Naïve Bayes algorithms, Support Vector Machine (SVM), and neural networks. The present study adopts these techniques in filtering spam messages and measures their accuracy to determine the most effective method of filtering spam messages. Based on the findings, the neural network performs best as the trained classifier model used to classify incoming messages as ham or spam.**

Keywords—SMS, spam, ham, machine learning, logistical regression, Naïve Bayes, SVM, neural network, classification.

I. INTRODUCTION

The increased penetration of the use of mobile devices can be attributed to the advancements in telecommunication technologies. Another factor behind the increased use of mobile devices is the capabilities that the networks offer to the users [1, 2, 3]. One such capability is the short message service (SMS), which enables mobile users to send messages to other users over the terrestrial mobile network as enabled by the service provider. SMS has become such an integral part of communication in the contemporary society that service providers use it as the primary means of passing information to their subscribers. At the same time, spammers have exploited this opportunity to pass their messages to mobile phone users with the interest of driving their business agenda.

One of the major problems that cell phone users face is the reception of unwanted and mostly annoying SMS messages from their advertisers or other sources. These messages usually adopt a relatively similar pattesrn. They begin with some "catch words" to attract potential "customers" then they provide some kind of contact information – such as a number to call back, SMS reply numbers and/or (malicious) website URLs that the target can click [1]. Many users would wish to have a filter against such messages, with a preference for only useful information from their service providers and their list of contacts [1]. This project entails the comparison between different SMS classifier algorithms in Python – using Google Collab – to find the efficient algorithm that distinguishes between spam and ham.

Spam filtering is a concept that means the necessity to get rid of undesirable messages from email accounts due to the fact that email users could miss important information among the clutter of spam messages [3]. Spam messages that reach mobile devices have become so many, with Asia reporting an approximate 30% of received SMS messages being spam in 2012 [4]. The same concept can be applied in the filtering of SMS from a user's inbox. The difference in filtering between email and SMS messages is in the number of characters, with the SMS limitation on character length providing a better platform for identifying the unwanted messages through search methods as compared to email spam messages. The spam classifier program, therefore, builds on this knowledge and implements it using the Python programming language.

The program collects a specific dataset, visualizes it and cleans it, and splits it into train and test. We have used a dataset collected for SMS spam research, which contains 5,574 SMS messages in English [3]. Out of 5,574, only 747 are Spam which are labelled as 1 whereas the rest are all ham (legitimate) which are labelled as 0 [3]. The files contain one message per line.

II. BACKGROUND

A. Background Study

Spamming of messages has become a common trend especially for email communication and is fast becoming an undesirable trend in mobile text messaging. To address this problem, various authors have developed novel approaches to identify spam messages and filter them out to allow users to obtain only that which they need to read. Mathew and Isaac document the various existing methods for the intelligent filtering of spam messages [1]. One such method is the dual filtering approach that utilizes the KNN classification algorithm to distinguish between spam and ham [5]. The authors discuss various other techniques that other scholars have utilized in SMS spam filtering.

Zhang and Wang implemented machine learning by utilizing Bayesian learning theory for word segmentation and classification of messages [6]. Graham, on the other hand, gives a comprehensive analysis of the similarities and differences between traditional techniques of message filtering that were

978-1-7281-0137-8/19/$31.00 ©2019 IEEE

used at the time and the machine learning technique at the turn of the millennium [7]. His document presented an early marker for the growing significance of machine learning in spam filtering, and many researchers after him developed the idea using various techniques.

For instance, Shirani-Mehr applies different machine learning techniques to the preprocessing and feature extraction processes involving SMS messages from an existing database obtained from the UCI Machine Learning Repositories [4]. He simulated various methods and algorithms for spam classification in the detection process to determine the most effective classifier algorithm. Some of the methods that formed part of his study included the multinomial Naïve Bayes classifier with Laplace smoothing, the Support Vector Machine (SVM) algorithm with liner kernel, the k-nearest neighbor technique, the random forests methodology, and Adaboost with decision trees [4]. Among these techniques, the one that proved to be most effective as an SMS classifier was the multinomial naïve Bayes algorithm followed by the SVM algorithm [4].

Suleiman and Al-Naymat, in their proposal of the detection of SMS spam using the H2O framework, report the existence of several spam detection methods that utilize machine learning classifiers such as random forest, deep learning, SVM, and Naïve Bayes [8]. The H2O framework that they developed experimentally compared the machine learning algorithms to identify the best algorithm to adopt. Their analysis reveals that the Naïve Bayes classifier is the fastest algorithm even though it has a low precision whereas the random forest algorithm has high precision, accuracy, and recall but operates at a slower speed. In the analysis, the authors also reveal that the factors that affect the detection of SMS spam include the number of digits in the messages and the URLs that exist within each text [8].

One of the reasons behind the preference for machine learning algorithms is their successful use in emails spam filtering. For instance, Hidalgo, Bringas, Sanz, and Garcia performed an analysis of the detection and blocking of mobile spam using Bayesian filtering techniques and demonstrated the effectiveness of the techniques in SMS spam filtering [9]. Even though there is still no consensus on the best techniques to use in mobile spam filtering, Delany, Buckley, and Greene acknowledge the efficacy of using some of the methods that have proved effective in filtering email spam [10]. The Naïve Bayes classifier technique has particularly received support by many scholars an efficient and effective spam filtering method [8] [11] [12] [13]

B. Motivation

Spam messages are a bother to many users not only for their "annoying" nature, but also for intruding the end users' devices, occupying memory resources that could have been used for other purposes, and their deception that might lead to huge financial losses [12]. Since there are many mobile phone users who rely on SMS communication, it is important to shield them from the negative impacts that mobile spam presents to them as they use the service. Based on the principles, algorithms, and techniques that have proved successful in filtering and blocking email spam, it is possible to develop solutions that can effectively filter out SMS spam to allow users to receive only the trustworthy information.

III. SYSTEM MODEL

Based on the existing message filtering methods that have been employed to control the reception of SMS spam messages, the present solution made use of machine learning techniques. The techniques included logistical regression, the Naïve Bayes algorithm (Gaussian), the SVM (and support vector clustering – SVC) algorithm, and the neural network model. These techniques were taken through classifier training and used as trained classification models to identify ham and spam messages to enable the filtering of spam messages while allowing the rest to reach the intended recipient.

A. System Model Description

The SMS spam filtering system performs various functions that can be divided into five major sub-systems. The first sub-system is the feature extraction system, which enables the program to obtain the content of the message from a received SMS. The feature extraction receives two inputs: In the present filtering solution, the training data is obtained from a dataset currently found on Google Drive includes the classification of the message into either ham or spam, with the proper labeling of the type of message received. The new SMS is obtained from the same dataset but haven't trained before to simulate a user's reception of a message.

The second sub-section of the system is the classifier training system. By definition, training is the process of obtaining content that has been predetermined to belong to a specific class and creating a classifier based on the known content [13]. In the filtering application, training involves instructing the program to identify ham and spam messages from an already existing dataset to identify the features of both types of messages. The classifier training system receives input from the feature extraction process, uses it to identify the criteria for determining the type of SMS, and creates a classifier based on the input.

The third sub-section of the system is the trained classification system. Classification, on the other hand, refers to using the classifier that has been built from the training content set to determine the class membership of unknown content [13]. In the SMS filtering application, the classification utilizes the trained classifier to categorize unknown incoming text messages into either the ham and spam groups. It is at this sub-system that the trained classifier model is created for use in the classification of unknown incoming messages.

The fourth sub-section is the classification system. In this section of the system, the trained classification model is used on the output of the feature extraction process to identify whether the content makes the message to be ham or spam. The last sub-section is the classification decision-system, which separates the messages into ham and spam, and implements the algorithm to filter out the spam while allowing the user to receive the ham message. The success of the sub-section depends on the accuracy of the adopted filtering technique in identifying the message type based on the classifier training output.

B. Model Flow Diagram

The diagram below shows the basic system model and the various sub-systems.

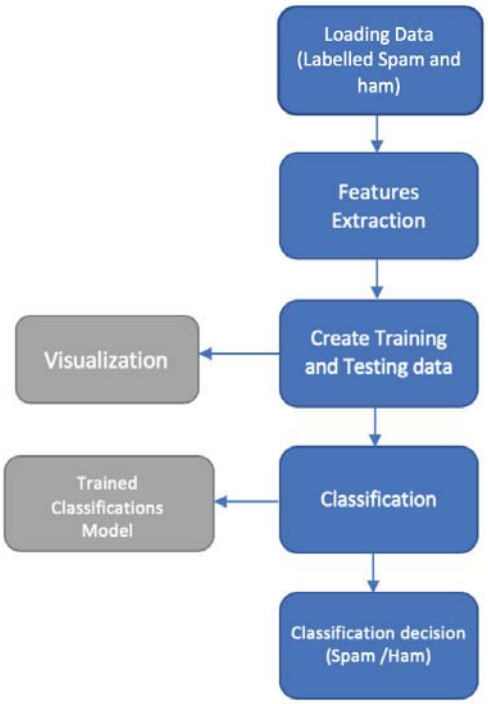

Figure 1: System Model Flow Diagram

IV. Proposed Approaches

Each of the sub-systems presented in the flow diagram (Figure 1) is implemented using Python programs. The new short text message is simulated by importing the libraries and dataset from Google Drive. The feature extraction process is implemented by the parts of the code that browses the dataset, cleans and organizes it, renames the column headers and deletes the unnecessary columns, and visualizes the dataset. The training data is obtained through the code that performs the function of splitting the dataset into train and test.

Based on the split data, the program then trains each classifier to develop a method of classifying the messages into ham and spam based on the downloaded dataset. From the training process, each of the four techniques is used to create a trained classification model that can then be used in identifying whether previously unknown messages belong to the ham or spam categories.

The trained classifier models are then used to classify the received messages into either ham or spam. The success rate at the identification of the type of the message – measured through the accuracy, is determined by the percentage of rightfully identified message type. The error rate is also calculated based on the number of texts that are wrongfully classified as ham or spam

Each of the four algorithms that formed part of the study was tested in terms of accuracy, error on ham messages, and errors on spam messages to determine the best model to utilize in implementing the SMS spam classifier. The system model is implemented through the programming processes detailed below.

A. Importing Libraries and the Data Set

The dataset has been collected for SMS Spam research [14] This paper uses a dataset of UCI Machine Learning repository gathered in 2012 that has 5574 SMS text messages [17]. After saving the dataset in google drive, the first step in the process is the importation of the libraries using the Python programming language. After the importation, the dataset is authenticated, followed by the creation of the PyDrive client. The authentication and creation of the client enabled the addition of the individual files' identities (ID) in Google Drive before downloading it.

The Python codes for performing these functions are as detailed in the screenshots below.

```
[ ] from collections import Counter
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, roc_curve, auc, accuracy_score, classification_report
    from sklearn.linear_model import LogisticRegression
    import string
    import nltk
```

Figure 2: Importing Libraries

1. Authenticate and create the PyDrive client.

```
[ ] from pydrive.auth import GoogleAuth
    from pydrive.drive import GoogleDrive
    from google.colab import auth
    from oauth2client.client import GoogleCredentials
    auth.authenticate_user()
    gauth = GoogleAuth()
    gauth.credentials = GoogleCredentials.get_application_default()
    drive = GoogleDrive(gauth)
```

2. Get the file by Adding the id of the file in Google Drive.

```
[ ]
    downloaded = drive.CreateFile({'id':'15bkz07ptHJ5DIHQVIsni69fVrnWkeWuq'})
    downloaded.GetContentFile('spam.csv')
```

Figure 3: Authenticating Messages

B. Cleaning and Visualizing the Dataset

The dataset thus obtained from Google drive contains the identification number of the message, its classification as either spam or ham, the text message itself, and three other columns that come with default values. The information occurs in a tabular form, with these aspects of each message being in six columns. Since there are some columns that are not useful in the process, they need to be dropped while at the same time renaming the useful columns to values that are relevant to the study.

The screenshot shown in Figure 4 shows the original dataset downloaded from Google Drive with all the columns.

```
[ ] data.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Figure 4: The Dataset

From this table, the columns to be dropped are the "Unnamed: 2", "Unnamed: 3", and "Unnamed: 4". After dropping these columns, the program then renames the data field v1 with the number 0 for ham and 1 for spam for easier analysis.

It then gives each of the retained columns new names, replacing v1 with "type" and v2 with "text". The code for executing these requirements is as shown in the screenshot below.

```
[ ]  #Drop the last three columns
     data.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
     data['v1'] = data['v1'].replace(['ham','spam'],[0,1])

     # Re-name the rest columns
     data.rename(columns={'v1':'type', 'v2':'text'}, inplace=True)

     #Re-Order Columns
     data = data[['text', 'type']]

     #Add one column to see the words count in each message
     data['Word_Count'] = data['text'].apply(len)
```

Figure 5: Column Editing

Finally, the cleaning and visualization process ends with the addition of one more column that is relevant to the understanding of the data. The column "Word Count" is added to establish the number of words that each of the text messages have. The results of all these processes are as contained in the screenshot below.

```
data.head()
```

	text	type	Word_Count
0	Go until jurong point, crazy.. Available only ...	0	111
1	Ok lar... Joking wif u oni...	0	29
2	Free entry in 2 a wkly comp to win FA Cup fina...	1	155
3	U dun say so early hor... U c already then say...	0	49
4	Nah I don't think he goes to usf, he lives aro...	0	61

Figure 6: Resulting Dataset

C. *Visualizing the Data*

This is the analytical phase of the development of the solution. The analysis is based on the number of spam and ham messages as well as the word counts for each of the messages received from the database. From the count of the messages, the mathematical analysis reveals that just 13.4% of them are spam messages with the rest of them being ham. The pie chart below demonstrates this result of the analysis of the received messages.

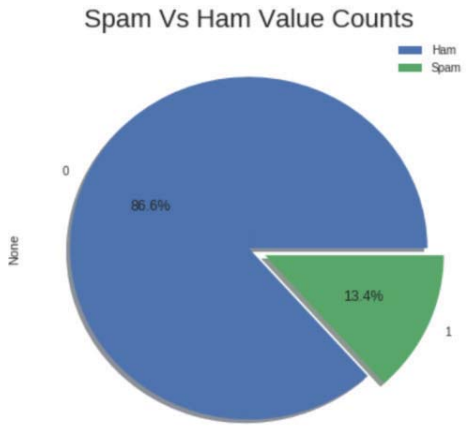

Figure 7: Spam vs. Ham Value Counts

An analysis of the number of words per text showed that the spam messages had more words per text than the ham messages. In essence, the findings reveal that spam messages are averagely longer than normal ham messages as the senders try to cram a lot of information within a single text to beat the word limit. It is, perhaps, worth noting that the average length of spam messages was close to 140, which was twice the average length of ham messages. The bar chart below shows these findings.

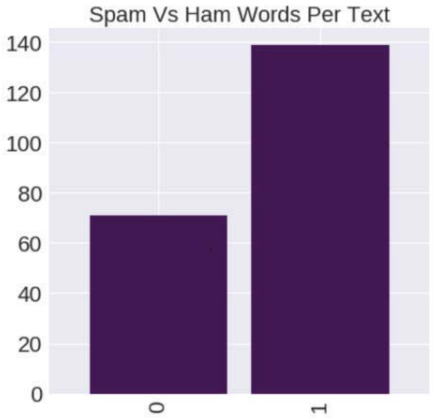

Figure 8: Number of Words

D. *Splitting the Dataset into Train and Test*

Before splitting the data, it was cleaned by removing the words that were not useful – including punctuations, stop words, and stemming words. Next, the TF-IDF of each word for "spam" and "ham" was computed to calculate the difference between them, enabling the selection of the words that are most representative of the "spam" class. The screenshot below shows the computation of the TF-IDF of each word using the vectorizer function.

```
[16] from sklearn.feature_extraction.text import TfidfVectorizer
     from nltk.corpus import stopwords
     sw = stopwords.words("english")
     vectorizer = TfidfVectorizer(stop_words=sw,lowercase=True)
     X = vectorizer.fit_transform(X_text).toarray()
     print('Shape of text: {}'.format(X.shape))
     print('Shape of type: {}'.format(y.shape))

     Shape of text: (5572, 8536)
     Shape of type: (5572,)
```

Figure 9: Splitting of Data

The results of the computation are presented through the variables "shape of text" and "shape of type" as shown above. The data was then split and tested as shown in the screenshot below. Each model was trained based on the split data to understand the degree of accuracy and errors.

```
[17] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y,  test_size=0.20, random_state=2)
     print(X_train.shape)
     print(y_train.shape)
     print(X_test.shape)
     y_test.shape

     (4457, 8536)
     (4457,)
     (1115, 8536)
     (1115,)
```

Figure 10: Results of the Data Splitting

V. Performance Evaluation

The SMS spam classifier model was trained using different machine learning algorithms. The algorithms included logistical regression, Naïve Bayes, SVM, and the neural network.

A. Logistical Regression

When the model was trained using the logistical regression algorithm, the accuracy of the algorithm was found to be 94.26%. The precision for detecting the ham messages was 0.94 while its precision in detecting spam was 0.99. Its recall rate for ham messages was 1.0 as compared to 0.6 for spam, while it provided support of 957 to ham messages as compared to 158 for spam messages. The results of the training are as shown below.

```
confusion_matrix
[[956   1]
[ 63  95]]

The accuracy = % 94.26008968609865
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       957
           1       0.99      0.60      0.75       158

avg / total       0.95      0.94      0.94      1115
```

Figure 11: Logistical Regression Results

B. Naïve Bayes Algorithm

Training with the algorithm yielded an accuracy of 88.16%. Its precision in detecting ham messages was 0.97 as compared to 0.56 for spam, while its recall rate was almost equal for both types of messages – 0.89 for ham and 0.82 for spam. It yielded a similar support result to the logistical regression model, with 957 for ham and 158 for spam. The results are as shown below.

```
[19] confusion_matrix
     [[853 104]
     [ 28 130]]

     The accuracy = % 88.16143497757848
                   precision    recall  f1-score   support

                0       0.97      0.89      0.93       957
                1       0.56      0.82      0.66       158

     avg / total       0.91      0.88      0.89      1115
```

Figure 12: NB Results

C. SVM

The SVM algorithm produced an accuracy of 94.26% when it was used to train the model. Its precision for detecting ham (0.94) was lower than that for detecting spam (0.99). The recall rate for ham was 1.00 and 0.60 for spam, while the support figures remained the same as the two previous models. The results are as shown in the diagram below.

```
confusion_matrix
 [[956   1]
 [ 63  95]]

The accuracy = % 94.26008968609865
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       957
           1       0.99      0.60      0.75       158

avg / total       0.95      0.94      0.94      1115
```

Figure 13: SVM Results

D. Neural Network

The neural network was analyzed using the TensorFlow backend instruction. Of the 137,153 parameters available, the model trained 137,153 – equal to 97.67% success rate. Consequently, it displayed better results as compared to the previous models. The figures below show the findings from the training of the model using the neural network technique.

```
Using TensorFlow backend.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 16)                136592
_________________________________________________________________
dense_2 (Dense)              (None, 16)                272
_________________________________________________________________
dense_3 (Dense)              (None, 16)                272
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 17
=================================================================
Total params: 137,153
Trainable params: 137,153
Non-trainable params: 0


Epoch 21/100
4457/4457 [==============================] - 3s 684us/step - loss: 0.0039 - acc: 0.9998
Epoch 22/100
4457/4457 [==============================] - 3s 686us/step - loss: 0.0035 - acc: 0.9998
Epoch 23/100
4457/4457 [==============================] - 3s 690us/step - loss: 0.0032 - acc: 0.9998
Epoch 24/100
4457/4457 [==============================] - 3s 686us/step - loss: 0.0030 - acc: 0.9998
Epoch 25/100
4457/4457 [==============================] - 3s 682us/step - loss: 0.0028 - acc: 0.9998
Epoch 26/100
4457/4457 [==============================] - 3s 686us/step - loss: 0.0026 - acc: 0.9998
Epoch 27/100
4457/4457 [==============================] - 3s 692us/step - loss: 0.0025 - acc: 0.9998
Epoch 28/100
4457/4457 [==============================] - 3s 685us/step - loss: 0.0024 - acc: 0.9998
Epoch 29/100
4457/4457 [==============================] - 3s 687us/step - loss: 0.0023 - acc: 0.9998
Epoch 30/100
4457/4457 [==============================] - 3s 688us/step - loss: 0.0022 - acc: 0.9998
Epoch 31/100

[[955   2]
 [ 24 134]]


1115/1115 [==============================] - 0s 100us/step

acc: 97.67%
```

Figure 14: Neural Network Results

E. Summary of Findings

The four machine learning algorithms had varying performances on the parameters of accuracy and error of operation. The Naïve Bayes algorithm had the lowest accuracy at 88.16%, but it also had the lowest error in the detection of ham messages – only 28 out of 881 ham messages were in error. Its error on spam messages was relatively high – 104 out of 234 messages.

The logistical regression method, on the other hand, had a fair accuracy rate at 94.26% with an error of 63 out of 1019 ham messages. Consequently, it showed a better performance than the Gaussian NB model since it was more accurate and had a lower error rate for ham messages. Furthermore, it only had an error of 1 in 96 spam messages. The SVM method posted similar results as the logistical regression model in terms of accuracy and errors on ham and spam messages.

Finally, the neural network algorithm presented the best accuracy, which was nearly 98%. It has the lowest error rate for ham messages. In addition, it had only 2 errors out of 979 spam messages. Consequently, the neural network algorithm in detecting spam messages is the best technique for training the model. The table I shows the summary of the findings.

TABLE 1: RESULTS

	ACCURACY	ERROR ON Ham Messages	ERROR ON Spam Messages
Gaussian NB	88.16%	(28 out of 881)	(104 out of 234)
Logistic Regression	94.26%	(63 out 1019)	(1 out of 96)
SVC	94.26%	(63 out of 1019)	(1 out of 96)
Neural Network	97.67%	(24 out of 979)	(2 out of 136

VI. CONCLUSION

Machine learning is the most popular technique used in the classification of messages into spam or ham. Its successful use in producing email spam classification system makes it a viable option for the classification of mobile spam messages. Consequently, it is one of the techniques that can be employed in reducing the burden of SMS spam messages that individual mobile phone users face. Consequently, machine learning techniques have been adopted in implementing the system of spam detection, classification, and blocking. The techniques used in implementing the system include the logistical regression algorithm, the Support Vector Machine (SVM) technique, the Naïve Bayes algorithm, and the neural network model. Each of these techniques was trained and used as classifiers of the incoming SMS messages to determine whether they were ham or spam. From the analysis of their operations, the neural network technique showed the best accuracy in detecting spam messages, which makes it be the most suitable option for implementing an SMS spam filtering system.

ACKNOWLEDGEMENTS

This work is partly supported by the U.S. National Science Foundation (NSF) under grants CNS 1650831 and HRD 1828811. However, any opinion, finding, and conclusions or recommendations expressed in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the NSF.

REFERENCES

[1] K. Mathew and B. Issac, "Intelligent Spam Classifcation for Mobile Text Message," in *2011 International Conference on Computer Science and Network Technology*, Kuching, Malaysia, 2011.

[2] D. B. Rawat and K. Z. Ghafoor, Smart Cities Cybersecurity and Privacy, London: Elsevier Press, November 2018.

[3] D. B. Rawat, R. Doku and M. Garuba, "Cybersecurity in Big Data Era: From Securing Big Data to Data-Driven Security," *IEEE Transactions on Services Computing,* 2019.

[4] P. Navaney, G. Dubey and A. Rana, "SMS Spam Filtering using Supervised Machine Learning Algorithms," in *2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, Uttar Pradesh, 2018.

[5] H. Shirani-Mehr, "SMS Spam Detection using Machine Learning Approach," Stanford University, CA, 2013.

[6] L. a. ,. L. Duan Longzhen, A new spam short message classification, First International Workshop on Education Technology and Computer Science, 2009.

[7] H.-y. Zhang and W. Wang, "Aplication of Bayesian method to spam SMS filtering," in *International Conference on Information Engineering and Computer Science*, 2009.

[8] P. Graham, "A Plan for Spam," August 2002. [Online]. Available: http://paulgraham.com/spam.html. [Accessed 28 January 2019].

[9] D. Suleimana and G. Al-Naymat, "SMS Spam Detection using H2O Framework," in *The 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017)*, Lund, Sweden, 2017.

[10] J. M. G. Hidalgo, G. C. Bringas, E. P. Sanz and F. C. García, "Content based SMS spam filtering," in *Proceedings of the 2006 ACM Symposium on Document Engineering*, Amsterdam, The Netherlands, 2006.

[11] S. J. Delany, M. Buckley and D. Greene, "SMS spam filtering: Methods and data," *Expert Systems with Applications,* vol. 39, no. 10, pp. 9899-9908, 2012.

[12] A. S. Aski and N. K. Sourati, "Proposed efficient algorithm to filter spam using machine learning techniques," *Pacific Science Review A: Natural Science and Engineering,* vol. 18, no. 2, pp. 145-149, 2016.

[13] D. Alorini and D. B. Rawat, "Automatic Spam Detection on Gulf Dialectical Arabic Tweets," in *2019 International Conference on Computing, Networking and Communications (ICNC): Communications and Information Security Symposium*, Honolulu, 2019.

[14] G. Goswami, R. Singh and M. Vatsa, "Automated spam detection in short text messages," *Machine Intelligence and Signal Processing,* vol. 390, pp. 85-98, 2015.

[15] L. Chen, Z. Yana, W. Zhang and R. Kantola, "TruSMS: A trustworthy SMS spam control system based on trust management," *Future Generation Computer Systems,* vol. 49, pp. 77-93, 2015.

[16] Mark Logic, "Training the classifier," Mark Logic, January 2019. [Online]. Available: https://docs.marklogic.com/guide/search-dev/classifier. [Accessed 29 January 2019].

[17] T. Almeida, J. G. Hidalgo and A. Yamakami, "Contributions to the Study of SMS Spam Filtering: New Collection and Results," in *11th ACM Symposium on Document Engineering*, New York, NY, 2011.

[18] T. A. Almeida, "UCI Repository, SMS Spam Collection Data Set from UCI Machine Learning," 2012. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection. [Accessed 2019].

[19] T. Karmali, "Spam classifier in Python from scratch," Towards Data Science, 2 August 2017. [Online]. Available: https://towardsdatascience.com/spam-classifier-in-python-from-scratch-27a98ddd8e73. [Accessed 29 January 2019].