SMS Spam Classification–Simple Deep Learning Models With Higher Accuracy Using BUNOW And GloVe Word Embedding

Surajit Giri, Sayak Das, Sutirtha Bharati Das, and Siddhartha Banerjee*

Department of Computer Science, Ramakrishna Mission Residential College, Narendrapur, West Bengal, India

Corresponding author. E-mail: sidd_01_02@yahoo.com

Received: Mar. 16, 2022; Accepted: Nov. 30, 2022

Unwanted text messages are called Spam SMSs. It has been proven that Machine Learning Models can categorize spam messages efficiently and with great accuracy. However, the lack of proper spam filtering software or misclassification of genuine SMS as spam by existing software, the use of spam detection applications has not become popular. In this paper, we propose multiple deep neural network models to classify spam messages. Tiago's Dataset is used for this research. Initially, preprocessing step is applied to the messages in the data set, which involves lowercasing the text, tokenization, lemmatization of the text, and removal of numbers, punctuations, and stop words. These preprocessed messages are fed in two different deep learning models with simpler architectures, namely Convolution Neural Network and a hybrid Convolution Neural Network with Long Short-Term Memory Network for classification. To increase the accuracy of these two simple architectures, BUNOW and GloVe word embedding techniques are incorporated with deep learning models. BUNOW and GloVe are popular choices in sentiment analysis, but in this work, these two-word embedding techniques are tried in the context of text classification to improve accuracy. The best accuracy of 98.44% is achieved by the CNN LSTM BUNOW model after 15 epochs on a 70% - 30% train-test split. The proposed model can be used in many practical applications like real-time SMS spam detection, email spam detection, sentiment analysis, text categorization, etc.

Keywords: SMS Spam; Machine Learning; CNN; CNN-LSTM; Word Embedding; GloVe; BUNOW

http://dx.doi.org/10.6180/jase.202310_26(10).0015

1. Introduction

Short Message Service (SMS) is one of the most popular forms of telecommunication service around the world because of its affordability. According to [1], around 5 billion people can send and receive SMS and more than 200 thousand SMS are sent every second. 83% of SMS messages are read within 90 seconds. On average, SMS messages have a 98% open rate compared to 20% of emails. Text messages have a 209% higher response rate than emails. Because of these statistics, marketers prefer SMS as the primary form of advertising which leads to spam-ming. SMS spamming has become a serious problem for mobile subscribers. Spam SMS refers to unwanted text messages that are usually either someone promoting a product or service or someone attempting to scam a subscriber into providing personal information. It incurs substantial costs in terms of lost productivity, network bandwidth usage, management, and raid of personal privacy. The main reason to stop spamming is that it costs a lot more to its receivers than its senders. Because of these reasons and many others, SMS spam detection is very necessary.

SMS spam is a kind of problem that doesn't have an algorithmic definite solution. Existing SMS spam filtering methods are not very robust. The machine learning method comes to be the most popular choice for spam classification;

several researchers have utilized supervised machine learning methods for comparative results of spam classification. Plenty of research has been carried out in this direction making use of machine learning techniques such as Naïve Bayes, Random Forest, Support Vector Machine, and Decision Trees. Using traditional machine learning methods mentioned above, feature engineering is a time-consuming process with an extra computational expense. It is also difficult to extract all the information from the short length of the text.

Among the recent solutions that have proven to be effective in solving these kinds of problems is the use of deep neural networks. Deep neural network-based architecture, such as Convolution Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) has been used for many classification problems for images, videos, and texts. Deep learning models are empowered with automatic pattern recognition as well as traditional clustering machine learning techniques (unsupervised techniques).

It has been seen from the existing work, the deep learning models achieved better accuracies than traditional machine learning models. But the researchers use complex architecture of deep learning models to increase the performance of the model. In this work, deep learning models with simpler architectures have been tried in search of better accuracy by using word embedding techniques. The proposed method classifies spam and non-spam (ham) messages using two deep learning models, namely Convolution Neural Network (CNN) and a Convolution Neural Network – Long Short-Term Memory (CNN-LSTM). These two models are embedded with two indexed vocabularies – BUNOW [2] and GloVe [3] as input vectors. The models are tested with Tiago's data set [4] and the accuracies of these models are compared with existing methods. The performances of the models are also evaluated on Youtube spam collection dataset [5] to judge the versatility of the proposed method. The contribution of the authors can be listed as:

1. Use of word embedding techniques in spam message classification for achieving increased accuracy.
2. Application of ablation experiment to show the necessity of word embedding techniques.
3. Performance analysis of the proposed models for different train-test splits.
4. Use of paired comparison for statistical evaluation of the proposed models.

The organization of the paper is as follows. After this brief introduction, the existing methodologies are described in Section 2. The proposed methodology for spam classification is discussed in section 3 and the results are listed in Section 4. The paper is concluded in Section 5 by mentioning future scopes of this research.

2. Related work

Machine learning techniques have been used extensively for SMS spam detection. Some researchers use traditional machine learning approaches while others use deep learning models for this purpose. Among these, which use Tiago's data set for testing and performance evaluation have been discussed here.

2.1. Traditional Machine Learning Approaches

Almeida et al. [6], the creators of Tiago's dataset, tested several machine learning algorithms and laid the groundwork for further research. In this work two tokenizers have been used, one for preserving domain names, and mail addresses and another one to preserve symbols that can have a crucial part in classification. SVM, Boosted NB, Boosted C4.5 and PART achieved the best accuracies. SVM got the highest accuracy of 97.64%. Three machine learning algorithms, viz., Naïve Bayes, Random Forest, and Logistic Regression are used by Sethi et al. [7]. Frequencies of the words are used as the input vector. Naïve Bayes achieved the best accuracy of 98.445% while Random Forest and Logistic Regression got 97.009% and 94.312% accuracy respectively. Navaney et al. [8] compared the results of Naïve Bayes, SVM, and Maximum Entropy algorithm for SMS spam detection. A document term matrix is used as an input vector. In this research, Naïve Bayes and Maximum Entropy Algorithm achieved accuracies of 94.55% and 91.95% respectively while SVM achieved the maximum accuracy of 97.4%. Alzahrani et al. [9] compared the performance of spam detection using a neural network model, Gaussian NB, Logistic Regression, and SVM. The neural network had three dense layers and one output layer with 137,153 total parameters. After training it for 30 epochs, the neural network model achieved the best accuracy of 97.67% among these four models. Gaussian NB got the worst accuracy of 88.16% while Logistic Regression and SVM both achieved an accuracy of 94.26%. Xia et al. [10] used a hidden Markov model to detect spam SMS where every word is processed directly. They extend their research [11] by assigning weight for each word that indicates the probability of the message to be detected as spam or ham SMS. The experimental results show that the weighted feature gives better accuracy. The weighted feature model achieves

96.9% accuracy. Initially, Diallo et al. [12] tried to find a similarity matrix in the multi-view document clustering algorithm and further enhanced their proposed model [13] by considering Cosine similarity, Euclidean distance, and magnitude difference. Their proposed model gives better performance for small dimensional but for larger dimensions, it consumes more space and time.

2.2. Deep Learning Models

In deep neural category, Taheri et al. [14] implemented an RNN-based classification model on Tiago's dataset. The data set is divided into training and testing purpose which includes 70% and 30% respectively. The size of the RNN was 100 and each word in a training vector got a size of 50. The batch size for each epoch and word sequence was 25 for both. After training the model for 200 epochs the best accuracy achieved was 98.11%. The semantic LSTM model has been tried for spam detection by Jain et al. [15]. For vectorization of the vocabulary, the missing words are searched from WordNet and ConceptNet, instead of passing each word through Google Word2Vec. If the missing word is not found on WordNet and ConceptNet, a random value is assigned to that word. The LSTM model consisted of 100 units, the dropout rate was 0.1 and the Sigmoid activation function was used. The model was trained with 10 epochs. One interesting observation from this research was that as the feature count went up from 5000 to 6000, the accuracy increased but increasing the features even more resulted in lesser accuracy. The best accuracy, 98.92% was achieved when the number of features was 6000. Popovac et al. [16] implemented a CNN model for spam detection. For preprocessing, several steps had been carried out like lowercasing, tokenizing, stop word removing, and finally converting the messages into a matrix of TF-IDF features. The CNN model was composed of two convolution layers with a filter size of 32, and a kernel size of 3 with a ReLU activation function. After that MaxPooling of pool size, 2 were added. After a flattened layer, a fully connected dense layer of 128 units with ReLU activation was used. Finally, the output layer consisted of one unit with a sigmoid activation function. Due to the nature of the problem Adam optimizer and binary cross-entropy loss were used. After training the model for 10 epochs an accuracy of 98.4% was achieved. CNN and RNN models are tried and their performances are compared by Annareddy et al. [17]. The text_to_sequence() function of the Keras library had been used to assign a unique integer value to every word in the vocabulary for the input vector. For both models, the ReLU activation function was used and the dropout was initialized at 0.2. After training two models for 10 epochs, the accuracy of the testing data for the CNN model was 96.4% with a loss of 0.090 while the RNN model achieved an accuracy of 97.8% with a loss of 0.143. Roy et al. [18] implemented several deep learning models on Tiago's dataset. The best accuracy was achieved on a 3-channel CNN with different dropout values for each CNN. After doing a 10-fold cross-validation on the multi-channel CNN, 99.44% accuracy was achieved. Chandra et al. [19] proposed an RNN-LSTM model for classification. As preprocessing steps stop words were removed from the corpus and the words were converted into a TF-IDF vector. The data set was divided into 70% training and 30% testing data sets. RMSprop optimizer, ReLU, and Sigmoid activation function are used in the model and 98% accuracy was achieved after 8 epochs. On the same TF-IDF vector, Naïve Bayes and SVM models had been applied and achieved accuracies of 80.54% and 97.81% respectively. Konti et al. [20] use the different versions of BERT models and analyze the performance of these models for detecting SMS spam. The highest accuracy (95.02%) is achieved by the DistilBERT model. Abayomi-Alli et al. [21] used different machine learning algorithms like Naïve Bayes, BayesNet, Self Organizing Maps, decision tree, and a deep learning model BiLSTM to detect spam SMS. The experiment achieves 98.6% highest accuracy using the BiLSTM model. Diallo et al. [22] introduce a deep learning based document clustering technique can work better than the existing clustering algorithms. Their proposed model can learn document representation in a better manner and they proved it by considering images and text datasets. Shaaban et al. [23] proposed a deep ensemble approach for spam detection. For feature extraction, they used convolution and pooling layers, and their base classifier consists of random forests and extremely randomized trees. Their model achieves 98.38% accuracy. Ghourabi et al. [24] proposed a hybrid CNN-LSTM model for the classification of spam messages. They merge the two data sets collected from the UCI repository and the Arabic messages collected from a different source. Their proposed model achieved 98.3% accuracy.

Although all these mentioned works achieved a great accuracy using existing techniques, it is clearly visible that deep learning models achieved better accuracies than traditional machine learning models. Some of these deep learning models have a very complex architecture. In this work deep learning models with simpler architectures have been tried in search of better accuracy. For this purpose, BUNOW and GloVe word embedding techniques are incorporated with deep learning models. BUNOW and GloVe are the popular choices in sentiment analysis [25, 26], but in this work, these two word embedding techniques are tried in the context of text classification to improve accuracy.

3. Materials and methods

The proposed methodology for SMS spam classification is given in Fig. 1. Initially, preprocessing is done on the messages in the SMS corpus. After preprocessing, two different word embedding techniques are applied to extract the context of each word in the messages. This context information is used in the training phase of two deep learning models. Finally, testing is applied to evaluate the performance of the proposed model.

3.1. SMS Corpus

The dataset used in this research is known as Tiago's Dataset [4]. It consists of 5,574 English, real and non-encoded messages labeled as ham (non-spam) or spam. The dataset is a collection of 4 subsets:

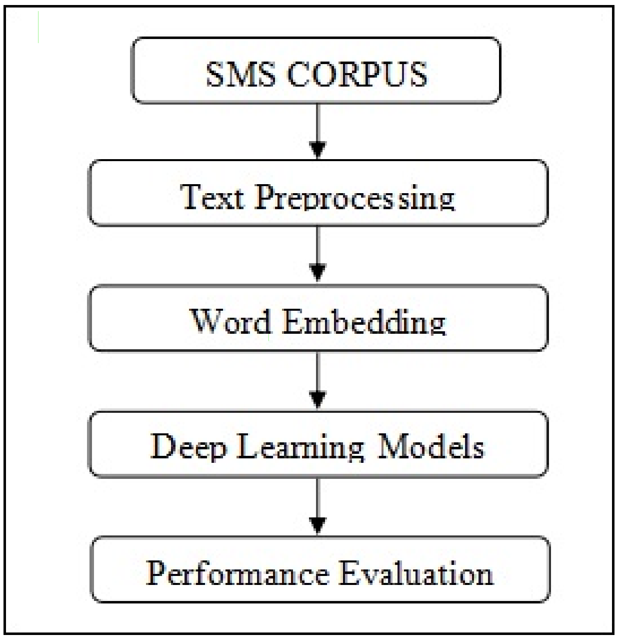


Fig. 1. Proposed Methodology.

- A subset extracted from the Grumble text Web site. It contains 425 spam SMS messages.
- A collection of 3,375 SMS non-spam messages was provided by NUS SMS Corpus (NSC) and collected for research at the Department of Computer Science at the National University of Singapore.
- SMS Spam Corpus v.01 Big subsets. It consists of 322 spam and 1,002 ham messages.
- A collection of 450 SMS ham messages from Caroline Tag's Ph.D. Thesis.

The data set contains 86.60% ham messages and the remaining 13.40% of spam messages as shown in Fig. 2(a). A snapshot of the data set is shown in Fig. 2b).

3.2. Preprocessing

One of the essential steps for creating a good machine learning model for classification is to preprocess the text data [27, 28]. Preprocessing converts the raw data to something understandable by an algorithm. Preprocessing is a complex process that requires a lot of steps. In the proposed model, the following steps are performed in preprocessing.

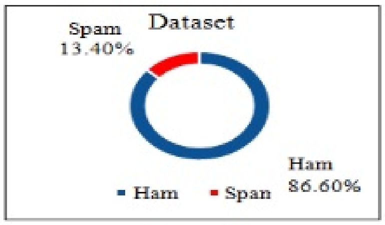
In the first stage, every word in the data set is converted into lower case. Generally, text messages often have a variety of capitalizations, which have no big impact on the final model. So, every word is reduced to lower case for simplicity. In the next step, tokenization was applied. A word tokenizer splits each lowercased text message into a set of words. For example, the tokenizer breaks the message "alright i have a new goal now" into a set of seven words { 'alright', 'i', 'have', 'a', 'new', 'goal', 'now'}. Thus it produces a data frame of individual words for further processing. After tokenization, all the words with lengths less than two are removed. The alphanumeric words i.e. words with the mixing of letters and digits are also removed from the data set as numbers, symbols and punctuations did not contribute much to the learning. In the next step, stop words were removed. Stop words are frequently used common words like 'or', 'and', 'this', 'that' etc. These stop words do not contribute to the knowledge source and can be removed from the textual data. The final step of preprocessing was lemmatization. Lemmatization is the process of reducing a word to its base form by removing inflectional endings only, which is also known as a lemma. It was done to get the simplest form of each word. For example, a lemmatization reduces the words car, cars, car's, cars' to the car. Lemmatizer uses vocabulary and morphological analysis of words and returns the base or dictionary form of a word.

The most frequent words in the spam messages in the data set with their frequencies are shown in Figure 3.

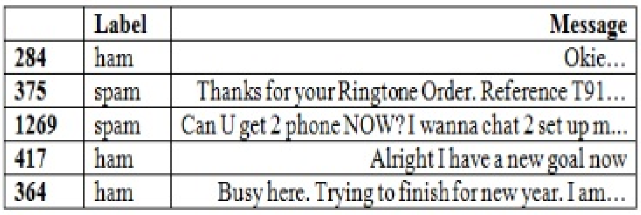
3.3. Word Embedding

The context of a word in a document, the words with similar semantics, and the relation of words in a text can be captured using word embedding. In word embedding, words that have the same meaning have similar representation. Each word is represented as real-valued vectors in a predefined vector space. In this paper, two types of word embedding techniques have been used.

The first one is the Binary Unique Number of Word



(a)



	Label	Message
284	ham	Okie...
375	spam	Thanks for your Ringtone Order. Reference T91...
1269	spam	Can U get 2 phone NOW? I wanna chat 2 set up m...
417	ham	Alright I have a new goal now
364	ham	Busy here. Trying to finish for new year. I am...

(b)

Fig. 2. (a) Distribution of Spam and Ham messages in the data set. (b) A Snapshot of the data set.

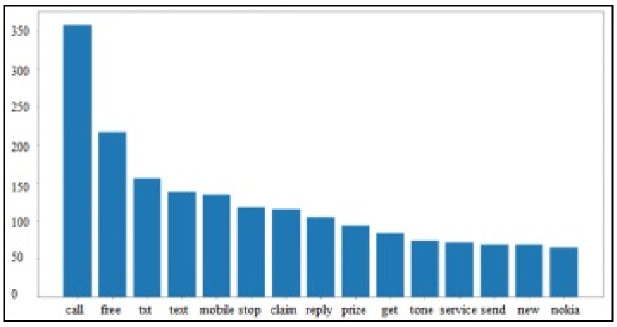


Fig. 3. Most Frequent Words in Spam messages with their frequencies.

(BUNOW) method [2]. In this technique, each distinct word (Wi) in the training corpus (T) is assigned a unique integer ID(ID_{Wi}). For this purpose, a vocabulary of distinct words is created from the training data set. After that, a unique serial integer ID has been assigned for each distinct word. The word (Wi) is represented by a fixed dimensional vector of size k where (2^k=*number of distinct words in Vocabulary*) by converting the assigned unique integer ID (ID_{Wi}) into its binary equivalent(B_{Wi}). Maximum message length (LWmax) is set with the number of words in the longest message in the training data set. The input feature vector (IV) of each message (M) is created by concatenating the binary vector of each word that exists in that message as given in Equation 1.

$$\mathrm{IV_M} = \mathrm{B_{W1}} + \mathrm{B_{W2}} + \mathrm{B_{W3}} + \mathrm{B_{W4}} + \ldots + \mathrm{B_{LW\,max}} \quad (1)$$

Where (+) is the concatenation operator. Any message with less than LWmax words is padded with all-zero vectors.

The second one is Stanford's Global Vectors (GloVe) word embedding technique [3]. GloVe word embedding considers the global context of the word rather than considering only local meaning. It is a pre-trained word embedding model that combines the advantages of two major model families: global matrix factorization and local context window methods. The model was trained on five corpora of varying sizes: 2010 Wikipedia dump with 1 billion tokens; 2014 Wikipedia dump with 1.6 billion tokens; Gigaword5 which has 4.3 billion tokens; the combination Gigaword5 and Wikipedia2014, which has 6 billion tokens; and 42 billion tokens of web data, from Common Crawl. Each corpus was tokenized and lowercased. A co-occurrence matrix X was constructed with a vocabulary of 400,000 most frequent words. In the construction of X, a context window was initialized to distinguish the left context from the right context, and a decreasing weighting function was used in the context window so that word pairs that are d words apart contribute $1/d$ to the total count. This way, much distant word pairs contributed less to the relevant information about word relationships.

The cost function for the experiment is described as,

$$J = \sum_{i=1}^{V}\sum_{j=1}^{V} f\left(X_{ij}\right)\left(\tilde{w}_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \quad (2)$$

where V is the size of the vocabulary, X_{ij} is the co-occurrence matrix. w_i represents a particular word in the message and w_j represents the set of context words of w_i.$\tilde{w}_i$ and $\tilde{w}_j$ are the vector representation of w_i and w_j respectively. $b_i, \tilde{b}_j$ are biases and $f\left(X_{ij}\right)$ is a weighting function that should follow the following properties,

1. $f(0) = 0$. If f is viewed as a continuous function, it should vanish as $x \to 0$ fast enough that the

$$\lim_{x\to 0} f(x)log^2 x$$

is finite.

2. $f(x)$ should be non-decreasing so that rare co-occurrences are not overweighted.

3. $f(x)$ should be relatively small for large values of x, so that frequent co-occurrences are not over weighted.

$$f(\mathrm{x}) = \begin{cases} \left(\frac{\mathrm{x}}{x_{\max}}\right)^{\alpha} & \text{if } \mathrm{x} < x_{\max} \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

The performance of the model depends weakly on the cutoff, which was fixed to $x_{\max} = 100$ for all experiments.

3.4. Deep Learning Models

Two deep learning models namely Convolution Neural Network (CNN) [29] and Convolution Neural Network – Long Short Term Memory (CNN-LSTM) [29] are used for the classification of messages.

In this proposed method, CNN model consists of two 1-dimensional convolution layers with filter size 32, kernel size 3 and swish activation function. Between these two convolution layers, a Max Pooling layer of pool size 3, and Dropout layer with a rate of 0.2 are used. A Global Max Pooling layer and another Dropout layer with a rate of 0.2 followed the second convolution layer. A fully connected dense layer of 128 units with swish activation function and another Dropout layer with a rate of 0.2 preceded the output layer. The output layer consists of two units with sigmoid function. Weights of CNN network are randomly initialized. The model is shown in Fig. 4.

Due to imbalanced class distribution, cost-sensitive classification is executed and class weights are assigned accordingly, 0 for ham messages and 1 for spam messages. Hyper parameters were assigned based on tests of multiple models with different values. Due to its computational and space efficiency, Adam optimizer was used with a learning rate of 0.001 along with binary cross entropy loss for compiling the model. Models were trained through 20 epochs while one of them having BUNOW embedding and the other one having GloVe embedding.

The CNN-LSTM model has exactly the same architecture as the CNN model, except the fully connected layer was preceded by a LSTM layer with 256 units with recurrent dropout rate of 0.3, and the Global Max Pooling layer was replaced with a Max Pooling layer of pool size 3. The proposed model is shown in Fig. 5.

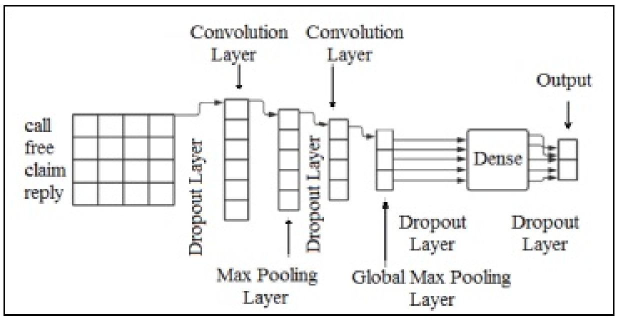


Fig. 4. Proposed CNN Architecture.

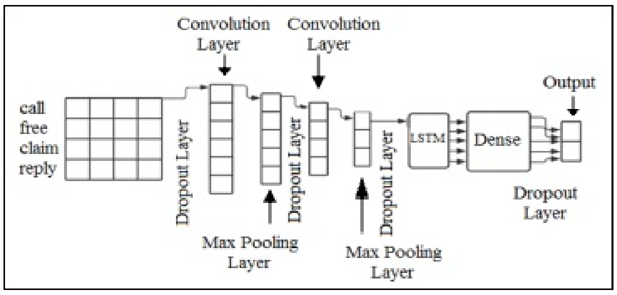


Fig. 5. Proposed CNN LSTM Architecture.

3.5. Working Principle

Tiago's data set [4] consists of 5,574 English, real and non-encoded messages labeled as ham (non-spam) or spam. The assigned labels of the messages are represented by numeric values 0 and 1 for indicating ham and spam messages respectively. These messages are preprocessed by the steps described in section 3.2.

Preprocessed messages are split into two parts for training and testing purposes. In this experiment, different training testing ratios have been used. The distinct words (Wd) within the messages in the training set are identified. In the case of BUNOW word embedding, each of these distinct words is represented by a unique vector by the method described in Section 3.3. On the other hand, in the case of GloVe word embedding, each distinct word is represented by a vector following the method described in Section 3.3. Next, the longest message from the training set is identified and the number of words in that message (LWmax) is calculated. Each message in the training set is represented by a fixed length vector of size LWmax. The vector representation of the message is created by concatenating the vector representation of the words that constitute the message. If the number of words in the message is less than LWmax, the remaining components in the fixed length vector are set to zero. After representing the messages in the training set by fixed length vector, two different deep learning models, namely the CNN model and CNN LSTM model are trained. One dimension convolution neural network has been applied in this experiment since each word is represented as one dimensional input vector. The dimension of the input layer in both the models is equal to the number of distinct words (Wd) and the output dimension is set to 300. A polling layer has been applied to decrease the dimension of the features without losing essential features. Finally, a dropout layer has been applied to prevent overfitting. The detailed structures of the models have been described in section 3.4.

For detecting a message as spam or ham, initially, the message is represented as fix length vector of size LWmax by a similar approach described above. If the length of the message is more than LWmax, the first LWmax words are considered. If the message consists of new words which are not present in the vocabulary of the training dataset, the word is eliminated from the message before representing it in the vector. The vector representation is fed into the trained model to classify the message as spam or ham.

3.6. Performance Evaluation

To evaluate the accuracies of the models, several performance measures exist in the literature. True positive, false positive, true negative and false negative are calculated to build the confusion matrix. Accuracy, precision, recall, specificity and f1 score are used to evaluate the performance of each model.

True positive (TP): It measures the number of spam messages classified as spam messages in the entire SMS corpus.

False positive (FP): It measures the number of ham messages classified as spam in the entire SMS corpus.

False negative (FN): It measures the number of spam messages classified as ham messages in the entire SMS corpus.

True negative (TN): It measures the number of ham messages classified as ham messages in the entire SMS corpus.

Accuracy (A): It measures the overall rate of correct prediction. It is defined by Equation 4.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{4}$$

Precision: It measures the rate of instances correctly detected as spam messages concerning all instances detected as spam. It is defined by Equation 5.

$$\text{Precision} = \frac{TP}{(TP + FP)} \tag{5}$$

Recall or True Positive Rate (TPR): It measures the proportion of spam messages which are iden-tified correctly, as defined by Equation 6

$$\text{Recall} = \frac{TP}{(TP + FN)} \tag{6}$$

Specificity or True Negative Rate (TNR): It measures the proportion of ham messages that are identified correctly, defined by Equation 7

$$TNR = \frac{TN}{TN + FP} \tag{7}$$

f1 Score: It is the harmonic mean of Precision and Recall. It is defined by Equation 8.

$$f1\text{ Score} = \frac{(2 * \text{ Precision } * \text{ Recall })}{(\text{ Precision } + \text{ Recall })} \tag{8}$$

4. Results and discussions

As mentioned earlier, four different models are proposed for spam message classification. The performance of these four models, namely, CNN with BUNOW word embedding, CNN with GloVe Embedding, CNN-LSTM with BUNOW word embedding, and CNN-LSTM with GloVe Embedding are evaluated using Tiago's Dataset [4]. All the models are implemented using Python and evaluated in Colab platform. To measure the accuracy, four models are trained with five different train-test splits (90% – 10%, 85% – 15%, 80% – 20%, 75% – 25% and 70% – 30%). The confusion matrix and the different performance measures are shown in Table 1 and Table 2 respectively for these five train-test splits.

From Table 2, it can be observed that both CNN BUNOW and CNN GloVe perform best for 90% - 10% train-test with 99.46% accuracy. These two models classify all 489 ham messages correctly. Out of 69 spam messages, these two models predict 66 messages as spam, hence having the highest sensitivity (TPR) of 100%. But as the training percentage is decreased, CNN- LSTM BUNOW perform best among these four models with accuracy 99.04%, 99.01%, 98.92% and 98.44% for 85% – 15%, 80% – 20%, 75% – 25% and 70% – 30% train-test splits respectively. Since Long Short-Term Memory (LSTM) is capable of remembering information in its memory cell for a long period, the CNN-LSTM model achieves better accuracy as expected. Though GloVe word embedding considers the global context of words, but fails to achieve the best performance with the CNN-LSTM model in this context.

All of these accuracies and confusion matrices are measured after the 15th epoch. Although, all of the models achieved a better accuracy in earlier epochs and because of over-fitting their accuracies decreased. The best accuracy of each model and the epoch at which this best accuracy is achieved are listed in Table 3 for each train-test split. The best accuracies achieved by each of these four models are compared in Fig. 6.

For statistical evaluation of the results among four proposed models, paired comparison [30] has been carried out. The p-values of the paired comparison are given in Table 4. Since all the pairs except pair CNN LSTM BUNOW and CNN LSTM GloVe exhibit p-value more than 5% significance level, all the proposed models significantly equal in terms of statistical measures. Since the pair CNN LSTM BUNOW and CNN LSTM GloVe shows p-value 0.002, which is less than 5% significance, it can be concluded CNN LSTM BUNOW perform better with respect to CNN LSTM GloVe model.

To judge the versatility of the proposed method, the ex-

Table 1. Confusion Matrices for all models.

			Predicted							
	Train-Test Split		Ham	Spam	Ham	Spam	Ham	Spam	Ham	Spam
Actual	90% – 10%	Ham	489	0	487	2	489	0	487	2
		Spam	3	66	2	67	3	66	3	66
	85% – 15%	Ham	722	1	720	3	721	2	722	1
		Spam	9	104	5	108	8	105	9	104
	80% – 20%	Ham	966	2	968	0	962	6	962	6
		Spam	9	138	11	136	8	139	6	141
	75% – 25%	Ham	1207	1	1206	2	1199	9	1202	6
		Spam	16	169	13	172	14	171	12	173
	70% – 30%	Ham	1436	6	1437	5	1438	4	1436	6
		Spam	26	204	21	209	28	202	24	206
			CNN BUNOW		CNN-LSTM BUNOW		CNN GloVe		CNN- LSTM GloVe	

Table 2. Performance Measures for four models.

Train-Test Split		CNN BUNOW	CNN-LSTM BUNOW	CNN GloVe	CNN-LSTM GloVe
90% – 10%	TPR	0.9565	0.9710	0.9565	0.9565
	TNR	1.000	0.9959	1.000	0.9959
	Precession	1.000	0.9710	1.000	0.9705
	fl score	0.9778	0.9710	0.9778	0.9635
	Accuracy	0.9946	0.9928	0.9946	0.9910
85% – 15%	TPR	0.9204	0.9558	0.9292	0.9204
	TNR	0.9986	0.9959	0.9972	0.9986
	Precession	0.9905	0.9730	0.9813	0.9905
	fl score	0.9541	0.9643	0.9545	0.9541
	Accuracy	0.9880	0.9904	0.9880	0.9880
80% – 20%	TPR	0.9388	0.9251	0.9456	0.9592
	TNR	0.9979	1.000	0.9938	0.9938
	Precession	0.9857	1.000	0.9586	0.9592
	fl score	0.9617	0.9611	0.9521	0.9592
	Accuracy	0.9901	0.9901	0.9874	0.9892
75% – 25%	TPR	0.9135	0.9297	0.9243	0.9351
	TNR	0.9992	0.9983	0.9925	0.9950
	Precession	0.9941	0.9885	0.9500	0.9665
	fl score	0.9521	0.9582	0.9370	0.9505
	Accuracy	0.9878	0.9892	0.9835	0.9871
70% – 30%	TPR	0.8870	0.9087	0.8783	0.8956
	TNR	0.9958	0.9965	0.9972	0.9958
	Precession	0.9714	0.9766	0.9806	0.9717
	fl score	0.9273	0.9414	0.9266	0.9321
	Accuracy	0.9809	0.9844	0.9809	0.9821

periment has also been applied on Youtube spam collection dataset [5]. It consists of 1956 different data items, out of these 1005 (51.38%) data items are spam and 951 (48.62%) data items are ham. The experimental results are shown in Table 5 for 80% - 20% train-test split as a representative. To show the importance of BUNOW and GloVe Word embedding techniques, the ablation test has been carried out on the Youtube dataset by eliminating word embedding process and the performance metrics of CNN and CNN-LSTM models are shown in Table 6 for same train-test split. It can be observed from the results, by eliminating word embedding process, CNN and CNN-LSTM models achieve 60.46% and 68.88% accuracy whereas all the models achieve above 92% accuracies when combined with word embeddings.

To measure the performance, the accuracies of the proposed CNN-LSTM BUNOW model (achieved after 15 epochs) are compared with the accuracies of other exiting models (Table 4). For comparing the performance, the accuracies achieved in a given train-test split on Tiago's Dataset [4] are considered. It can be concluded from Table 7, the proposed CNN-LSTM BUNOW model outperforms all of the state-of-the-art machine learning algorithms mentioned in [5, 7, 8, 12–14] and [16] by achieving better accuracy. The model proposed by Roy et al. [15] achieves better accuracy than the proposed CNN-LSTM BUNOW model. But the authors proposed a complex multichannel CNN

Table 3. Best Accuracies achieved by four models.

Train-Test Split	Model	Accuracy	Epoch
90% − 10%	CNNLSTM BUNOW	0.9928	7
	CNN BUNOW	0.9946	11
	CNN GloVe	0.9946	8
	CNNLSTM GloVe	0.9910	15
85% − 15%	CNNLSTM BUNOW	0.9904	12
	CNN BUNOW	0.9880	6
	CNN GloVe	0.9880	9
	CNNLSTM GloVe	0.9880	11
80% − 20%	CNNLSTMBUNOW	0.9901	10
	CNNBUNOW	0.9901	11
	CNN GloVe	0.9874	7
	CNNLSTM GloVe	0.9892	10
75% − 25%	CNN BUNOW	0.9878	11
	CNNLSTM BUNOW	0.9892	5
	CNN GloVe	0.9835	14
	CNNLSTM GloVe	0.9871	11
70% − 30%	CNN BUNOW	0.9809	7
	CNNLSTM BUNOW	0.9844	5
	CNN GloVe	0.9809	14
	CNNLSTM GloVe	0.9821	10

Table 4. Results of Paired comparisons.

	CNN BUNOW	CNN LSTM BUNOW	CNN GloVe
CNN LSTM BUNOW	0.300	−	−
CNN GloVe	0.192	0.110	−
CNN LSTM GloVe	0.369	0.002	0.643

Table 5. Performance measures of the Youtube data set for 80% – 20% train-test split.

Train-Test Split		CNN BUNOW	CNN-LSTM BUNOW	CNN GloVe	CNN- LSTM GloVe
80% − 20%	TPR	0.9121	0.9024	0.9073	0.8585
	TNR	0.9679	0.9786	0.9733	0.9947
	Precession	0.9689	0.9788	0.9738	0.9944
	F1	0.9396	0.9390	0.9393	0.9215
	Accuracy	0.9387	0.9388	0.9388	0.9234

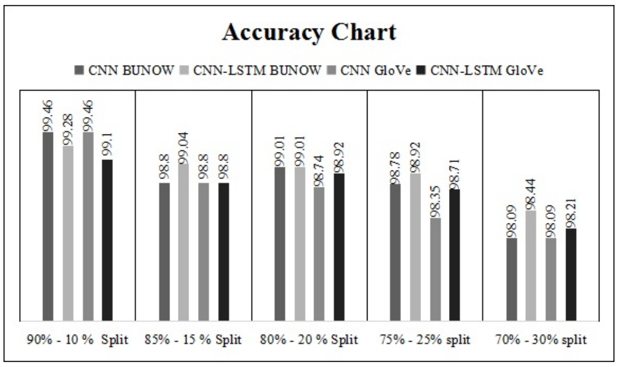


Fig. 6. Best Accuracies achieved by four models.

Table 6. Result of ablation test on the Youtube data set.

Train-Test Split		CNN	CNN-LSTM
80% − 20%	TPR	0.3902	0.5024
	TNR	0.8396	0.8930
	Precession	0.7273	0.8374
	F1	0.5079	0.6280
	Accuracy	0.6046	0.6888

architecture and perform a 10-fold cross-validation which takes huge time on training the model. The CNN-LSTM BUNOW model is much simpler and achieves acceptable accuracy in less training time.

5. Conclusion

In this paper four different neural network models viz., CNN BUNOW, CNN-LSTM BUNOW, CNN GloVe, and CNN-LSTM GloVe are proposed. The models were applied to Ti-ago's dataset to distinguish spam from non-spam messages. The dataset is mostly composed of ham messages.

Table 7. Comparison of CNN-LSTM BUNOW model with other existing models.

Existing Models	Year of Publication	Train-Test Split Train-Test Split	Accuracy Achieved by Existing models	Accuracy of Proposed CNN-LSTM BUNOW model
Almeida et al. [5]	2011	70% – 30%	97.64%.	98.44%
Taheri et al. [7]	2017	70% – 30%	98.11%	98.44%
Navaney et al. [8]	2018	75% – 25%	97.4%	98.92%
Popovac et al. [12]	2018	70% – 30%	98.4%	98.44%
Alzahrani et al. [13]	2019	80% – 20%	94.26%	99.01%
Annareddy et al. [14]	2019	80% – 20%	97.8%	99.01%
Roy et al. [15]	2019	66.7% – 33.3%	99.44%	98.44%
Chandra et al. [16]	2019	70% – 30%	97.81%	98.44%

To obtain a good model, preprocessing of the data is performed. Preprocessing steps include lowercasing the text, tokenization, lemmatization, and removal of stop words, symbols, numbers, and words with lengths less than 2. To achieve better accuracy, the texts were converted into two different types of embeddings, BUNOW embedding, and GloVe embedding. Proposed models are trained and tested on different train-test splits. The accuracies of the proposed models are calculated on these different train-test splits. It should be noted that though both CNN BUNOW and CNN GloVe achieve best accuracy on 90% - 10% train-test splits but CNN-LSTM BUNOW perform best among four models with accuracy 99.04%, 99.01%, 98.92% and 98.44% for 85% – 15%, 80% – 20%, 75% – 25% and 70% – 30% train-test splits respectively. Though GloVe word embedding considers the global context of words but does not succeed to achieve the best accuracy in this context. The accuracy of CNN-LSTM BUNOW is compared with other existing Machine Learning models and achieves better accuracy in most cases. This research work could be extended by incorporating more complex preprocessing techniques like N-grams, spelling correction, etc. In the future, the performance of complex deep neural structures with word embedding techniques can also be evaluated for spam message classification.

References

[1] *What is Text Message Marketing*. https://www.tatango.com/.

[2] A. A. Helmy, Y. M. Omar, and R. Hodhod. "An innovative word encoding method for text classification using convolutional neural network". In: *2018 14th international computer engineering conference (ICENCO)*. IEEE. 2018, 42–47. DOI: 10.1109/ICENCO.2018.8636143.

[3] J. Pennington, R. Socher, and C. D. Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, 1532–1543.

[4] T. Almeida and J. Hidalgo. *SMS Spam Collection v.1*. http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/.

[5] https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection.

[6] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami. "Contributions to the study of SMS spam filtering: new collection and results". In: *Proceedings of the 11th ACM symposium on Document engineering*. 2011, 259–262. DOI: 10.1145/2034691.2034742.

[7] P. Sethi, V. Bhandari, and B. Kohli. "SMS spam detection and comparison of various machine learning algorithms". In: *2017 international conference on computing and communication technologies for smart nation (IC3TSN)*. IEEE. 2017, 28–31. DOI: 10.1109/IC3TSN.2017.8284445.

[8] 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)*. IEEE. 2018, 43–48. DOI: 10.1109/CONFLUENCE.2018.8442564.

[9] A. Alzahrani and D. B. Rawat. "Comparative study of machine learning algorithms for SMS spam detection". In: *2019 SoutheastCon*. IEEE. 2019, 1–6. DOI: 10.1109/SoutheastCon42311.2019.9020530.

[10] T. Xia and X. Chen, (2020) *"A discrete hidden Markov model for SMS spam detection"* **Applied Sciences *10***(14): 5011. DOI: 10.3390/app10145011.

[11] T. Xia and X. Chen, (2021) *"A weighted feature enhanced Hidden Markov Model for spam SMS filtering"* **Neurocomputing *444***: 48–58. DOI: 10.1016/j.neucom.2021.02.075.

[12] B. Diallo, J. Hu, T. Li, G. Khan, and C. Ji. "Concept-enhanced multi-view clustering of document data". In: *2019 IEEE 14th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*. IEEE.

2019, 1258–1264. DOI: 10 . 1109 / ISKE47853 . 2019 . 9170436.

[13] B. Diallo, J. Hu, T. Li, G. A. Khan, and A. S. Hussein, (2022) *"Multi-view document clustering based on geometrical similarity measurement"* **International Journal of Machine Learning and Cybernetics *13***(3): 663–675. DOI: 10.1007/s13042-021-01295-8.

[14] R. Taheri and R. Javidan. "Spam filtering in SMS using recurrent neural networks". In: *2017 Artificial Intelligence and Signal Processing Conference (AISP)*. IEEE. 2017, 331–336. DOI: 10.1109/AISP.2017.8515158.

[15] G. Jain, M. Sharma, and B. Agarwal, (2019) *"Optimizing semantic LSTM for spam detection"* **International Journal of Information Technology *11***(2): 239–250. DOI: 10.1007/s41870-018-0157-5.

[16] M. Popovac, M. Karanovic, S. Sladojevic, M. Arsenovic, and A. Anderla. "Convolutional neural network based SMS spam detection". In: *2018 26th Telecommunications Forum (TELFOR)*. IEEE. 2018, 1–4. DOI: 10.1109/TELFOR.2018.8611916.

[17] S. Annareddy and S. Tammina. "A comparative study of deep learning methods for spam detection". In: *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE. 2019, 66–72. DOI: 10 . 1109 / I - SMAC47947 . 2019 . 9032627.

[18] P. K. Roy, J. P. Singh, and S. Banerjee, (2020) *"Deep learning to filter SMS spam"* **Future Generation Computer Systems *102***: 524–533. DOI: 10.1016/j.future.2019.09.001.

[19] A. Chandra and S. K. Khatri. "Spam SMS filtering using recurrent neural network and long short term memory". In: *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*. IEEE. 2019, 118–122. DOI: 10.1109/ISCON47742.2019.9036269.

[20] S. Kotni, D. Chandrasekhar Potala, and L. Sahoo, (2022) *"Spam Detection Using Deep Learning Models"* **International Journal of Advanced Research in Engineering and Technology *13***(5): 55–64. DOI: 10 . 17605/OSF.IO/NT4.

[21] O. Abayomi-Alli, S. Misra, and A. Abayomi-Alli, (2022) *"A deep learning method for automatic SMS spam classification: Performance of learning algorithms on indigenous dataset"* **Concurrency and Computation: Practice and Experience *34***(17): 1–15. DOI: 10.1002/cpe.6989.

[22] B. Diallo, J. Hu, T. Li, G. A. Khan, X. Liang, and Y. Zhao, (2021) *"Deep embedding clustering based on contractive autoencoder"* **Neurocomputing *433***: 96–107. DOI: 10.1016/j.neucom.2020.12.094.

[23] M. A. Shaaban, Y. F. Hassan, and S. K. Guirguis, (2022) *"Deep convolutional forest: a dynamic deep ensemble approach for spam detection in text"* **Complex & Intelligent Systems**: 1–13. DOI: 10.1007/s40747-022-00741-6.

[24] A. Ghourabi, M. A. Mahmood, and Q. M. Alzubi, (2020) *"A hybrid CNN-LSTM model for SMS spam detection in Arabic and english messages"* **Future Internet *12***(9): 156. DOI: 10.3390/fi12090156.

[25] Z. Jianqiang, G. Xiaolin, and Z. Xuejun, (2018) *"Deep convolution neural networks for twitter sentiment analysis"* **IEEE access *6***: 23253–23260. DOI: 10 . 1109 / ACCESS.2017.2776930.

[26] S. M. Rezaeinia, R. Rahmani, A. Ghodsi, and H. Veisi, (2019) *"Sentiment analysis based on improved pre-trained word embeddings"* **Expert Systems with Applications *117***: 139–147. DOI: 10.1016/j.eswa.2018.08.044.

[27] A. K. Uysal and S. Gunal, (2014) *"The impact of preprocessing on text classification"* **Information processing & management *50***(1): 104–112. DOI: 10.1016/j.ipm.2013.08.006.

[28] J. Camacho-Collados and M. T. Pilehvar, (2017) *"On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis"* **arXiv preprint arXiv:1707.01780**: DOI: 10.48550/arXiv.1707.01780.

[29] S. Weidman. *Deep learning from scratch: building with python from first principles*. O'Reilly Media, 2019.

[30] A. C. Michalos. *Encyclopedia of quality of life and well-being research*. Springer Netherlands Dordrecht, 2014.