**SMS SPAM DETECTION**

Pavani Billapati([Pbill1@unh.newhaven.edu](mailto:Pbill1@unh.newhaven.edu))

Trilok kumar pidikiti([tpidi1@unh.newhaven.edu](mailto:tpidi1@unh.newhaven.edu))

Harianth Kancherla([hkanc1@unh.newhaven.edu](mailto:hkanc1@unh.newhaven.edu))

**Abstract:**

In the era of pervasive mobile communication, the surge in SMS spam presents a formidable threat, with malicious actors exploiting unsuspecting individuals through deceptive messages for extracting sensitive information. This research addresses the urgent need for effective SMS spam detection, aiming to curtail the risk of individuals falling victim to phishing attacks and incurring financial losses. We propose an innovative SMS spam detection system, leveraging advanced Natural Language Processing (NLP) techniques and employing Deep Learning architectures, including Dense Neural Networks, Long Short-Term Memory (LSTM), and Bidirectional LSTM.

Concentrating on English language SMS spams, our methodology encompasses meticulous data preparation involving word tokenization, padding, truncating, and word embedding to enhance the dimensionality of the data. The model development phase incorporates a Dense Neural Network for baseline understanding and compares it with the sequence modeling capabilities of LSTM and the bidirectional nature of Bi-LSTM.

This research contributes a robust solution to SMS spam detection, emphasizing the efficacy of advanced NLP and Deep Learning techniques in fortifying individuals against phishing attempts and preserving their financial well-being. The diverse model architectures considered in our approach signify a comprehensive exploration of strategies to address the evolving landscape of SMS spam threats.

**Introduction:**

The state of communication technology is currently quickly advancing. This implies that with the use of email, text messaging, and web surfing, anyone may now obtain or receive information more easily than in the past. By sending a false notice of a bank transaction or a misleading advertisement, a hacker can use these technologies' benefits to obtain sensitive information from users, including phone numbers, credit card numbers, and online banking account information [1]. Furthermore, the unsuspecting individuals who received the message in a panic then obeyed the hacker's instructions, providing them with their private information. The hacker then utilizes this data to obtain assets from individuals.

Spam that is sent via mobile devices and targets users' sensitive information is known as short messaging service (SMS) spam [2], [3]. These days, a variety of people who follow the hacker's instructions and believe the information in the message are severely impacted by an SMS spam attack. Having a technology that can efficiently identify spam messages can help to mitigate this issue.

Deep Learning is currently a widely utilized data analysis technique since it often yields excellent prediction accuracy [4], [5]. Recurrent Neural Networks (RNNs) are the algorithm that is frequently used for evaluating sequential data, such as text data [5]. The Long Short-Term Memory (LSTM), Bi-LSTM and Dense Model algorithms are adjusted for RNNs. Many published studies assert that LSTM Bi-LSTM and Dense Model perform better in deep learning than other methods, especially for sequential data analysis. For instance, Kraus and Feuerriegel developed a decision support system to help investors feel more safe about their investments by using economic news and long short-term memory (LSTM). Moreover, they employ word tokenization, transfer learning, and word embedding to prepare data for their

In this study, we make use of Keras, a Python toolkit for developing deep learning models with a Tensorflow backend. Its data preparation toolset includes word tokenization, data padding and truncation, and word embedding. Using the word tokenization process, text inputs are converted into sequential data using the words' index values. While the word embedding approach is used to add extra dimensions to the sequence into the vector, the padding and truncating data techniques are utilized to make all sequences have the same length. Following the data preparation step, we use the LSTM, Bi-LSTM and Dense Model algorithms to train the model. Next, we assess the models' performance.

**Dataset:**

We use an SMS spam dataset that was downloaded from UCI datasets. This collection has roughly 5,572 records. It includes English-language SMS text messaging chats with text and numbers in varying sentence lengths. Every record in this dataset has a label already. The label for the spam messages is ‘spam’ (747 records), while the label for the regular mails is ‘ham’(4,825 records).

A screenshot of a computer

Description automatically generated

**Preparing Data:**

Natural Language Process (NLP) is used in this procedure to pre-process natural language data. The goal of natural language processing (NLP) is to enable computers to comprehend natural language in the same way that humans do. It offers a plethora of methods for preprocessing data into a machine-readable format. In this study, we use natural language processing (NLP) approaches to convert SMS text input into sequential data, which we then use to build SMS classification models using the LSTM, Bidirectional LSTM and Dense Model algorithms. For pre-processing data, we additionally employ word tokenization, padding, truncating, and word embedding algorithms. The following is a description of each method we employ in the data pre-processing step.

**Word Tokenization:**

The process of turning words in a sentence into index values represented by a number is called word tokenization. To develop a word tokenizer, we established a lot of intriguing vocabulary terms in this process. We utilize the word tokenizer to turn words in a sentence into sequence data after creating the tokenizer. When a word is unknown, the tokenizer sets the index to 0 and converts the word to index. Furthermore, we generate a tokenizer by assigning a fixed 10,000 vocabulary items. Additionally, we employ the tokenizer to transform text data into an index-number word sequence.

A screenshot of a computer code

Description automatically generated

**Padding and Truncating Data:**

Data Truncation and Padding During this procedure, we use the LSTM, BI-LSTM and Dense Model algorithms to ensure that every sequence in the dataset has the same length for training. Based on (1), we determine the optimal message length. Once the message's length has been optimized, we pad any data that is shorter than the ideal length by adding 0 to the start of the sequence until the length of the data matches the ideal length. We truncate the data starting at the beginning until its length equals the ideal length if the data's length exceeds that of the sample. Optimize is equal to mean (len(xi)) + 2×std(len(xi)).

where xi represents the dataset's records, mean(x) represents the mean of the data, std(x) represents the standard deviation of the data, and len(x) represents the message length function. Based on (1), we determine the optimal message length. After computation, the ideal length for cover data is 200, which accounts for 97.95% of datasets.

Consequently, we regulate padding and truncating in sequence using this ideal length.

**Word Embedding:**

The word embedding technique developed by Pennington et al. is employed in this study. Using this method, a preprocessed word sequence is transformed into a vector representation known as embedding space, which has more dimensions than the typical word data and is used to train LSTM, BI-LSTM and Dense Model algorithms. We utilize the word embedding approach to create additional dimensions for the data in sequence after padding and truncating it. We set the embedding size to 32.

**Modeling**

In this experiment, we develop the SMS spam classification models based on the three deep learning algorithms including LSTM, Bidirectional LSTM and Dense Model algorithms. The details of each algorithm described as follows.

**Long Short-Term Memory (LSTM):**

LSTM is developed by Hochreiter and Schmidhuber in 1997. It improves the basic RNN algorithm that solves the vanishing problem by adding cell states for remembering or forgetting data. The cell states contain structure called cell gates. The cell gates consist of four parts including input gate, forget gate, memory-cell state gate, and output gate. The input is gate used to control the input data that is worthwhile to keep or not. The forget gate is used to control the previous hidden state that is to be kept in the memory cell of the current hidden state. The memory-cell state gate is used to update the data based on the information of the input gate and the forget gate. The output gate is used to compute the output data from the network based on the memory-cell state.

A close-up of a screen

Description automatically generated

The LSTM architecture excels in capturing sequential patterns and dependencies, making it well-suited for tasks involving natural language understanding. In the context of SMS spam detection, it enables the model to effectively discern the nuanced characteristics of spam messages and differentiate them from legitimate ones. Adjusting hyperparameters, such as the number of LSTM units, can influence the model's capacity to capture long-term dependencies and optimize its performance.

This code implements an LSTM-based neural network for SMS spam detection. The architecture involves an embedding layer to represent words, an LSTM layer to capture sequential patterns, and a dense layer for binary classification. The model is trained using the Adam optimizer and binary cross-entropy loss, with early stopping to prevent overfitting. The trained model is then saved for future use. Adjusting hyperparameters and model architecture can be explored to optimize performance further.

A screenshot of a document

Description automatically generated

**BIDIRECTIONAL LSTM:**

Bidirectional Long Short-Term Memory (Bi-LSTM) is a variant of the traditional Long Short-Term Memory (LSTM) neural network architecture, designed to capture contextual information from both past and future elements in a sequence. In many sequential data tasks, information processing traditionally occurs in a unidirectional manner, from the beginning to the end of the sequence. This unidirectional approach may limit the model's understanding of context and dependencies within the sequence.

A screen shot of a computer code

Description automatically generated

A white background with black text

Description automatically generated

Bi-LSTM addresses this limitation by incorporating two separate LSTM layers: one processing the sequence in the forward direction (from the beginning), and the other processing it in the backward direction (from the end). This bidirectional processing enables the model to consider both past and future context for each element in the sequence.

The forward LSTM layer captures information from the past elements of the sequence, while the backward LSTM layer captures information from the future elements. By combining both directions, the model gains a more comprehensive understanding of the context and dependencies within the sequence.

The outputs from both LSTM layers are typically combined, often by concatenation, before being passed to subsequent layers in the neural network. This concatenated representation incorporates information from both directions.

**Dense Model:**

A dense model architecture, often referred to as a dense neural network or fully connected neural network, is a type of artificial neural network where each node (or neuron) in one layer is connected to every node in the next layer. In other words, the information flows in both forward and backward directions without skipping any layer. This architecture is characterized by dense connectivity between layers.

A screenshot of a computer code

Description automatically generated

In a dense model, the input layer receives input data, and subsequent hidden layers process this information through weighted connections and activation functions. The output layer produces the final of the model. The term "dense" comes from the dense connections between neurons in adjacent layers.

The model's parameters include the weights associated with each connection and biases for each neuron. Training involves adjusting these weights and biases to minimize the difference between the predicted output and the actual target values, typically using optimization algorithms like gradient descent.

Dense architectures are versatile and can be used for various tasks, including classification, regression, and feature learning. However, they may become computationally expensive and prone to overfitting on complex tasks, leading to the development of more sophisticated architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) for specific applications.

A screenshot of a document

Description automatically generated

The Dense model for SMS spam detection is defined with an Embedding layer for word mapping, followed by a Global Average Pooling 1D layer to reduce spatial dimensions. Two Dense layers are included, the first with 24 units and ReLU activation, followed by a Dropout layer for regularization. The final Dense layer with a sigmoid activation produces binary predictions. The model is compiled using binary cross-entropy loss, Adam optimizer, and accuracy metric. Training involves 50 epochs with early stopping after 3 epochs of no improvement in validation loss. The model achieves accurate spam detection, and the trained model is saved for future use. Adjustments to hyperparameters and model architecture can be explored for optimization.

**Evaluation:**

All Dense, LSTM and Bi-LSTM models are comparable in terms of loss and accuracy. The validation loss for these three models are 0.14, 0.16 and 0.18, respectively. And, the validation accuracy are 95.99%, 95.32% and 94%, respectively.

we select Dense architecture as a final model for classifying the text messages for spam or ham. The dense classifier has simple structure and the loss and accuracy over epochs are more stable than in LSTM and Bi-LSTM.

A screenshot of a computer code

Description automatically generated

**Predictions:**

**Scenario 1: Using raw text from our data:**

First and second messages below are ham whereas the third one is a spam message. We’ve used the same tokenizer that we created earlier in the code to convert them into the sequences. This makes sure the new words will have the same token as in the training set. Once tokenized, we use padding as we did earlier and provide the same dimension as in training set.

A screenshot of a computer

Description automatically generated

the model correctly predicts first two sentences as not spam whereas the third one as spam. There is 99% chance that the third sentence is spam.

A computer screen shot of a program

Description automatically generated

**Scenario 2: Using newly created text message and see how the model classifies them.**  
Below, first sentence is more like a spam whereas the rest of the two sentences are more like ham.

**A screenshot of a computer

Description automatically generated**

Our model correctly classifies the first message as spam (78% chance to be spam) were as the rest as ham.

**REFERENCES:**

1. N. Jindal and B. Liu, “Review spam detection,” in Proc. of the 16th International Conference on World Wide Web, 2007, pp. 1189-1190.

2. T. A. Almeida, J. M. G. Hidalgo, and T. P. Silva, “Towards SMS spam filtering: Results under a new dataset,” International Journal of Information Security Science, vol. 2, No. 1, pp. 1-18, 2013.

3 T. A. Almeida, T. P. Silva, I. Santos, and J. M. G. Hidalgo, “Text normalization and semantic indexing to enhance instant messaging and SMS spam filtering,” Knowledge-Based Systems, vol. 108, pp. 25-32, 2016.

4. J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural Networks, vol. 61, pp. 85-117, 2015.

5. Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning,” Nature, vol. 521, no. 7553, pp. 436-444, 2015.