Tribhuvan University

Institute of Science and Technology

Seminar Report

On

“Spam classifier using LSTM recurrent neural network (RNN)”

Submitted to

Central Department of Computer Science and Information Technology

Tribhuvan University, Kirtipur

Kathmandu, Nepal

Submitted by

Shailesh Acharya

Roll no. 36/2079

In partial fulfillment of the requirement for Master’s Degree in Computer Science and Information technology (M.Sc. CSIT), 1st semester

Tribhuvan University

Institute of Science and Technology

Supervisor's Recommendation

I hereby recommend that this seminar report, prepared under the supervision of Mr. Jagdish Bhatta entitled **"Spam classifier using LSTM recurrent neural network (RNN)"** be accepted as fulfillment in partial requirement of the degree of Masters of Science in Computer Science and Information Technology.

....

Asst. Prof. Jagdish Bhatta

(Supervisor)

Central Department of Computer Science and Information Technology

Letter of Approval

This is to certify that the seminar report prepared by Mr. Shailesh Acharya entitled **“Spam classifier using LSTM recurrent neural network (RNN)”** in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

Evaluation Committee

....	....
Asst. Prof. Sarbin Sayami	Asst. Prof. Jagdish Bhatta
(H. O. D)	(Supervisor)
Central Department of Computer Science and Information Technology	Central Department of Computer Science and Information Technology

....

(Internal)

Acknowledgement

I would like to express my sincere gratitude to **Asst. Prof. Mr. Jagdish Bhatta,** for his valuable guidance in carrying out this work under his supervision. I am grateful for his excellent guidance, trust and correction of my seminar work.

With immense pleasure, I would like to thank **Asst. Prof. Mr. Sarbin Sayami,** Head of Central Department of Computer Science and Information Technology, Tribhuvan University for his encouragement in completion of the seminar.

At last, but not least, with immense pleasure, I submit by deepest gratitude to the Central Department of Computer Science and Information Technology, Tribhuvan University, and all the faculty members of CDCSIT for providing the platform to explore the knowledge of interest.

Shailesh Acharya(36/079)

Abstract

Spam messages remain a serious worry in today's digital environment, with negative consequences for productivity, privacy, and security.Developing effective spam filters is crucial to mitigate these issues. In recent years, recurrent neural networks (RNNs) with Long Short-Term Memory (LSTM) layers have emerged as a powerful approach for spam classification due to their ability to capture sequential patterns in text data.

The dataset used is the SMS Spam Collection Dataset from Kaggle and more entries were additionally added from different sources. Two possible class labels for the data points are spam and ham. Each entry consists of the class label, a few sentences of text followed by a few useless features that are eliminated.After converting the text to the required format, the models are run and then evaluated using various metrics. The text data undergoes preprocessing, including tokenization and sequence conversion. Padding is applied to ensure consistent sequence lengths for input into the RNN model.

The seminar covers the architecture and design considerations of the RNN model, including embedding layers, LSTM layers with dropout regularization, and a dense output layer with softmax activation. The model is trained using a split of the dataset into training and testing sets, with evaluation metrics such as accuracy, precision, and recall used to assess its performance.

Keywords: spam classification, Recurrent Neural Networks, Long Short-Term Memory, Sequence Classification, RNN, LSTM

Contents

Acknowledgement . . . i
Abstract . . . ii
List of Figures . . . iv

1 Introduction **1**
1.1 Overview . . . 1
1.2 Some Important Definitions . . . 2
1.3 Problem Statement . . . 2
1.4 Objective . . . 2
1.5 Recurrent Neural Network . . . 3
1.6 LSTM Recurrent Neural Networks . . . 4

2 Literature Review **6**

3 Methodology **7**
3.1 Data Loading and Preprocessing . . . 7
3.2 Model Architecture . . . 8
3.3 Model Training . . . 9
3.4 Model Evaluation . . . 10
3.5 Visualization . . . 10
3.6 Message Classification . . . 11
3.7 Interface . . . 11

4 Implementation **12**

5 Result and findings **14**
5.1 Model Performance . . . 14
5.2 Evaluation Metrics . . . 14
5.3 Key Findings . . . 15

6 Conclusion **16**
References . . . 17

List of Figures

1.1 Recurrent Neural Network 3
1.2 LSTM gates 4

3.1 Architecture 7
3.2 Dataset 7
3.3 LSTM Model 8
3.4 Model Training 9
3.5 Model Evaluation 10
3.6 Training and Validation 10
3.7 Interface 11

4.1 Code Snippet 13

5.1 Confusion matrix 14
5.2 ROC Curve 15

Chapter 1

Introduction

1.1 Overview

The Spam Classifier Using LSTM Recurrent Neural Network (RNN) is an implementation aimed at addressing the pervasive issue of spam messages. With the exponential growth of electronic communication, spam messages have become a significant problem, causing inconvenience, privacy concerns, and potential security threats for individuals and organizations. Traditional rule-based and machine learning-based approaches have limitations in effectively detecting and classifying spam messages due to the dynamic and evolving nature of spam.

To overcome these limitations, advanced techniques like Recurrent Neural Networks (RNNs) have gained prominence. RNNs are a class of artificial neural networks that excel in handling sequential data by capturing the temporal dependencies and context. Long Short-Term Memory (LSTM) is a specialized type of RNN that overcomes the vanishing gradient problem, enabling the model to retain long-term information and effectively learn from sequential data.

The Spam Classifier Using LSTM RNN leverages the power of these deep learning techniques to accurately classify messages as spam or non-spam (ham) based on their textual content. By utilizing the sequential nature of messages data, the model can capture the nuanced patterns and dependencies that distinguish spam from legitimate emails.

1.2 Some Important Definitions

1. **Spam:** Refers to unsolicited and unwanted email messages that are typically sent in bulk to a large number of recipients.
2. **Ham:** Refers to non-spam or legitimate messages. Ham messages are the opposite of spam messages and typically consist of genuine and desired communication from known senders.
3. **Tokenizer:** In natural language processing, a tokenizer is a tool or algorithm that breaks down text into smaller meaningful units called tokens. Tokens can be individual words, subwords, or characters, depending on the specific tokenizer used.

1.3 Problem Statement

The purpose of this seminar report is to address the problem of accurately classifying and detecting spam messages using LSTM recurrent neural network (RNN) models. In the context of email communication, the issue at hand is the high volume of spam emails that often contain unsolicited and potentially harmful content. The problem statement revolves around developing an effective spam classifier using LSTM RNN models to accurately differentiate between spam and legitimate (ham) messages.

1.4 Objective

The objective of the Spam Classifier using LSTM Recurrent Neural Network (RNN) is to develop an accurate and reliable model that can effectively classify and distinguish between spam and legitimate (ham) messages.

1.5 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a type of neural network that has an internal memory, allowing it to process sequential data. Unlike feedforward neural networks, which treat each input independently, RNNs have connections that allow information to be passed from one step to the next. This enables them to capture dependencies and patterns across the sequence.

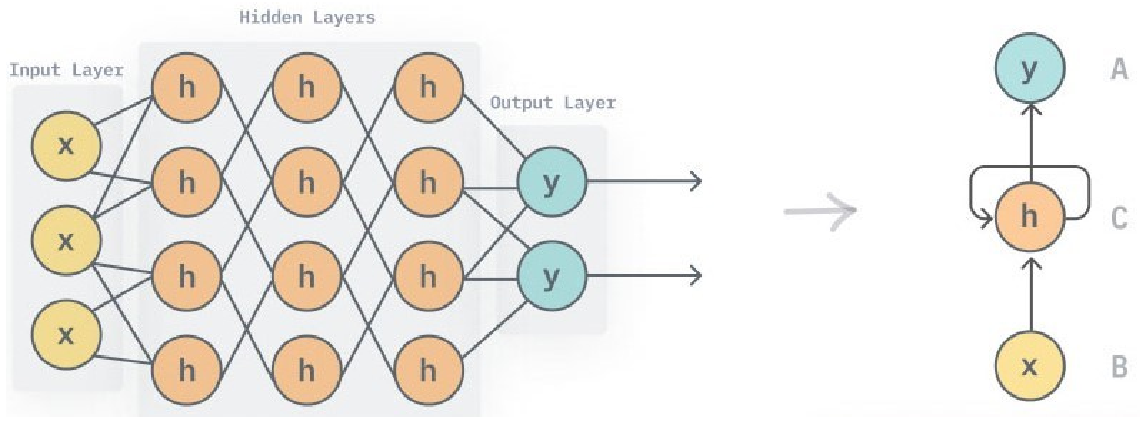


Figure 1.1: Recurrent Neural Network

RNNs are particularly suited for tasks that involve sequential data, such as handwriting recognition or speech recognition, where the current input's output depends not only on the current input but also on the previous inputs processed by the network. The internal memory of the RNN allows it to retain information about the past inputs and use it to influence the computation and decision-making for the current input.

By considering the entire sequence of inputs and their corresponding outputs, RNNs can learn to recognize and understand patterns and relationships in sequential data. This makes them a powerful tool for modeling and predicting sequences, enabling applications in natural language processing, time series analysis, and many other domains where the order of data is essential.

1.6 LSTM Recurrent Neural Networks

Long Short-Term Memory (LSTM) networks are a specialized type of recurrent neural network that addresses the vanishing gradient problem and allows for better memory retention of past data. LSTM networks are particularly effective in tasks involving time series analysis, where there are unknown time lags between data points.Training an LSTM network involves using back-propagation, which adjusts the weights and biases of the network to optimize its performance. Within an LSTM unit, there are three main gates that control the flow of information: the input gate, forget gate, and output gate.[1]

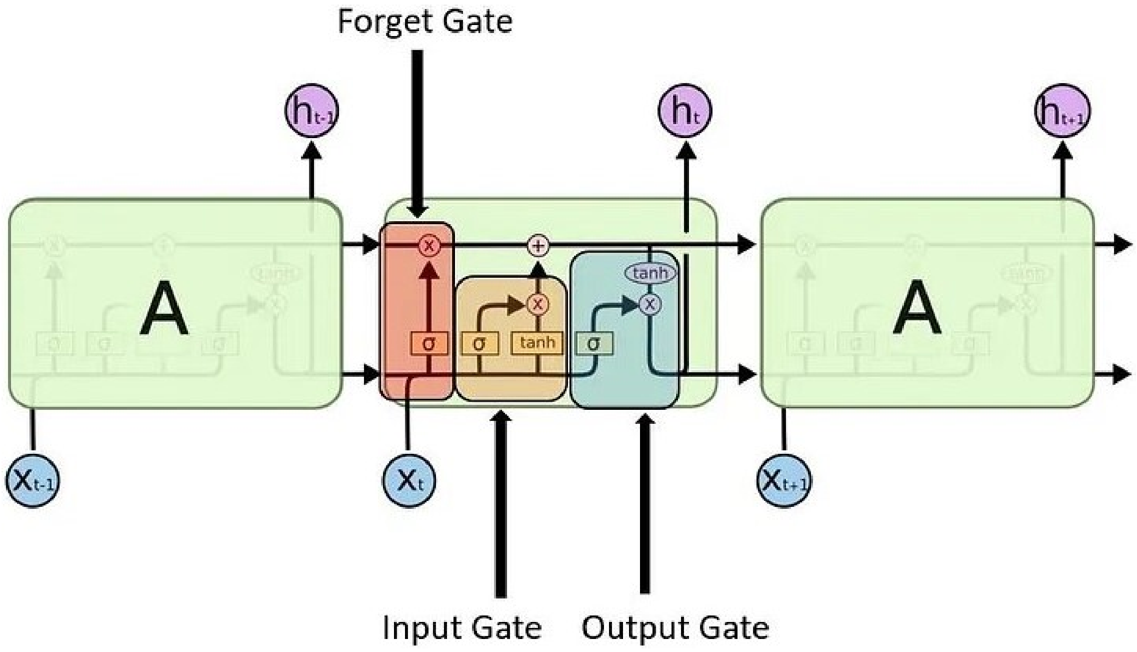


Figure 1.2: LSTM gates

1. **Input gate:** The input gate determines which values from the input should be used to modify the memory. It utilizes a sigmoid function to decide which values to let through (ranging from 0 to 1) and a tanh function to assign weights to the values, indicating their level of importance.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

2. **Forget gate:** The forget gate determines what information should be discarded from the memory cell. It also employs a sigmoid function, which looks at the previous state and current input, and outputs values between 0 (omit this information) and 1 (keep this information) for each element in the memory cell state.

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

3. **Output gate:** The output gate combines the input and memory of the LSTM block to determine the output. It uses a sigmoid function to decide which values to pass through (ranging from 0 to 1) and a tanh function to assign weights to the values, indicating their level of importance. The weighted values are then multiplied with the output of the sigmoid function to generate the final output.

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

Chapter 2

Literature Review

Spam is a major problem on the internet, and spam filters are essential for protecting users from unwanted messages. Spam filters typically use machine learning algorithms to identify spam messages. One of the most popular machine learning algorithms for spam filtering is the LSTM RNN. Various researches have been done on spam classifier using different tools and models. LSTM recurrent neural network is one of the most powerful tool for spam classification.

In [2] spam cluster creation has done for each category using neural classification and SVM technique is used to identify the spam and the idea of neural network's multiclass classifier to categorize the data received online. the value generator algorithm produces the each token text's ASCII value provided in the category data and produces a combined representable value that will be developed as the training data to both SVM and Neural Network. The train_neural algorithm trains the Neural Network after the neuron's successful training in the Hidden layer. The proposed work has improved performance of the kernel function of SVM may lead to more accurate results.

Carlos et al. [3] utilize a technique that includes extracting link based and content-based features from private and public datasets and utilizing the Web Topology by exploiting link dependencies between web pages, one limitation is that their accuracy is obtained without regularization methods applied to classifiers.

Minoru et al. [4] proposed another spam recognition method that includes the utilization of a spherical k-means clustering algorithm, which distinguishes spam mail by using centroid vectors of the groups for extracting features of that cluster. They give a label to every centroid and a calculation to check similarity is done among a new mail cluster and the marked centroid. They can demonstrate their technique to be powerful, However, they don't consider a dynamic update of the spam and non-spam clusters.

Chapter 3

Methodology

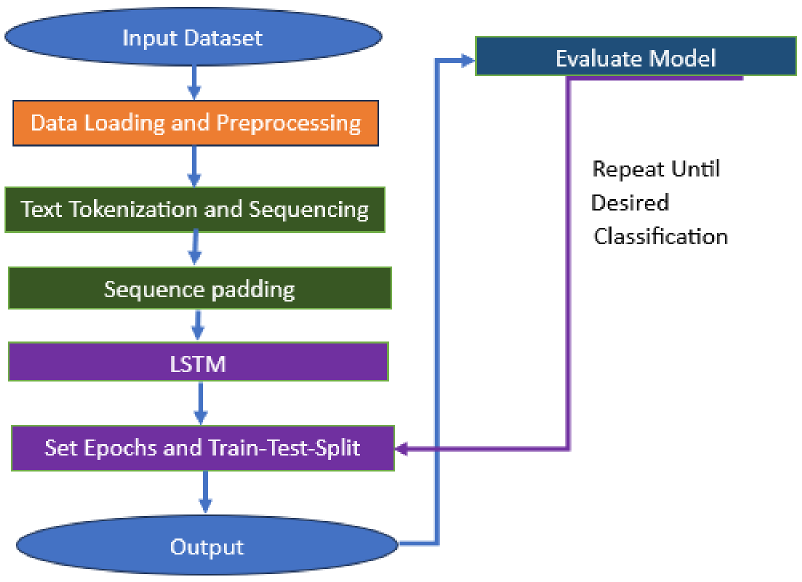
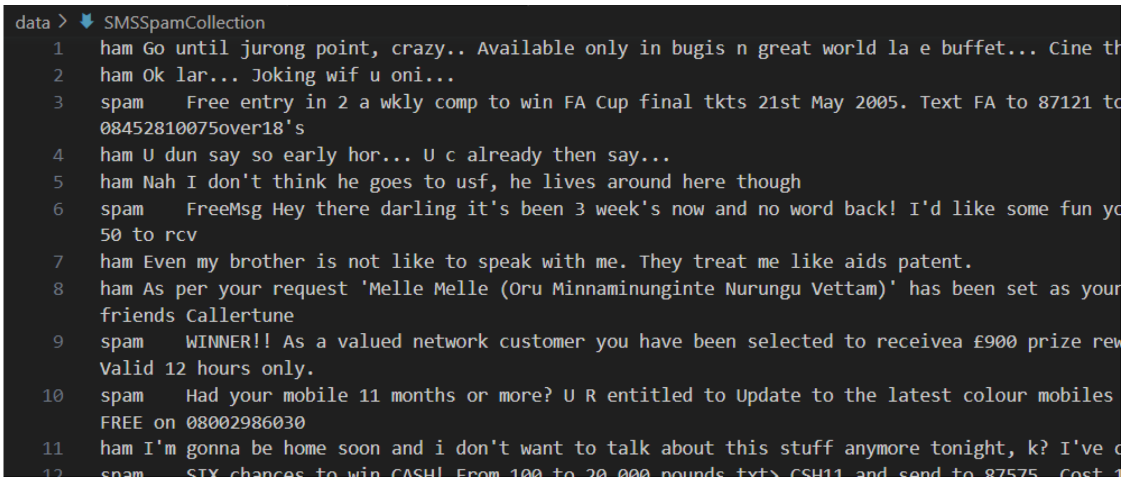

Input Dataset
Data Loading and Preprocessing
Text Tokenization and Sequencing
Sequence padding
LSTM
Set Epochs and Train-Test-Split
Output
Evaluate Model
Repeat Until Desired Classification

Figure 3.1: Architecture

3.1 Data Loading and Preprocessing



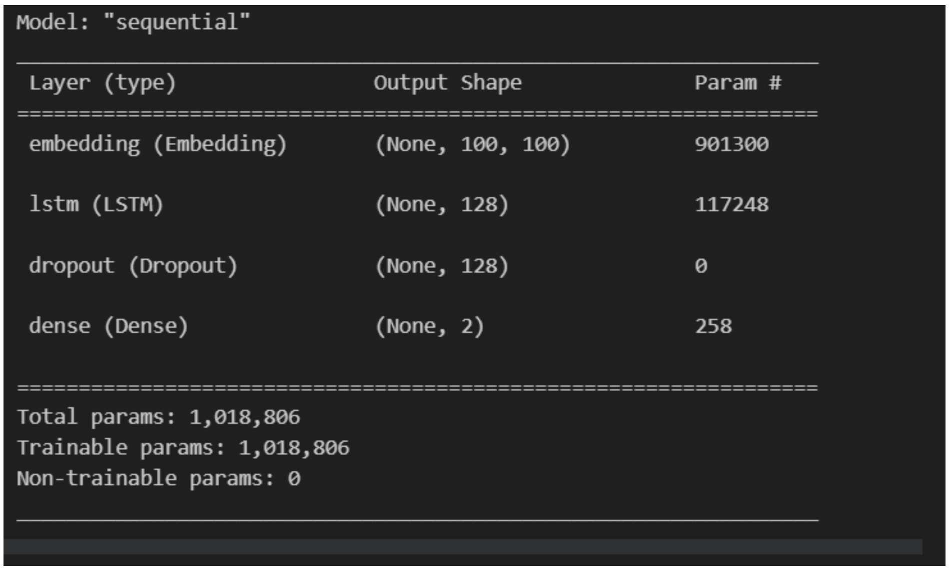
```
data > SMSSpamCollection
ham Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine th
ham Ok lar... Joking wif u oni...
spam    Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to
08452810075over18's
ham U dun say so early hor... U c already then say...
ham Nah I don't think he goes to usf, he lives around here though
spam    FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun yo
50 to rcv
ham Even my brother is not like to speak with me. They treat me like aids patent.
ham As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your
friends Callertune
spam    WINNER!! As a valued network customer you have been selected to receivea £900 prize rew
Valid 12 hours only.
spam    Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles
FREE on 08002986030
ham I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've c
spam    SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 1
```

Figure 3.2: Dataset

- The code starts by loading the data from the dataset file. This file contains the SMS messages along with their corresponding labels (ham or spam).

- The data is then split into texts and labels, where texts represent the SMS messages and labels represent their corresponding categories.
- The texts are preprocessed using the Tokenizer class from Keras. This involves converting the text into numerical sequences and padding them to a fixed length to ensure uniformity.

3.2 Model Architecture

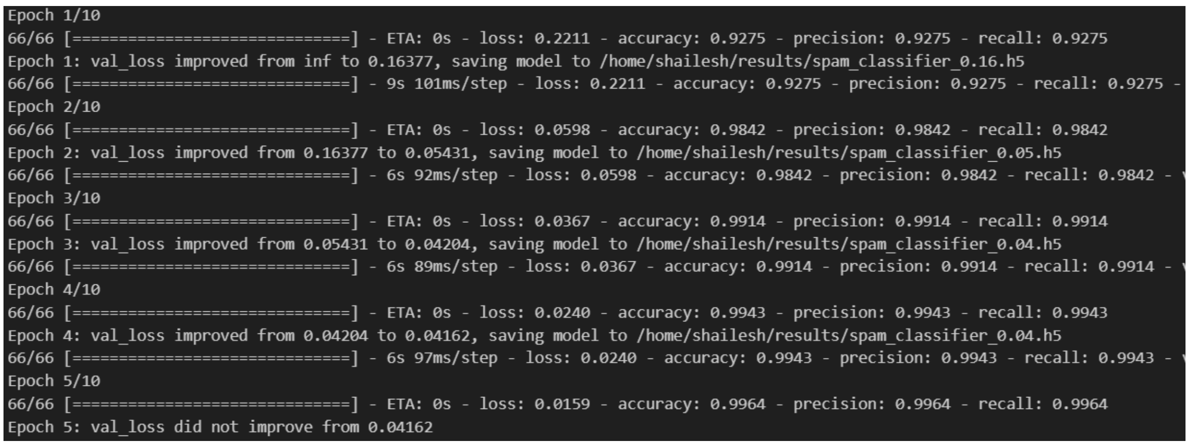


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 100)          901300

 lstm (LSTM)                 (None, 128)               117248

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 2)                 258

=================================================================
Total params: 1,018,806
Trainable params: 1,018,806
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.3: LSTM Model

- The model architecture is defined in the ”get_model” function. It uses an embedding layer to convert the numerical tokens into dense word vectors.
- The embedding layer is followed by an LSTM layer with a specified number of LSTM units. This layer captures the sequential dependencies and long-term memory of the text.
- A dropout layer is added to prevent overfitting by randomly deactivating some neurons during training.
- Finally, a dense layer with softmax activation is used to produce the final classification probabilities.

3.3 Model Training

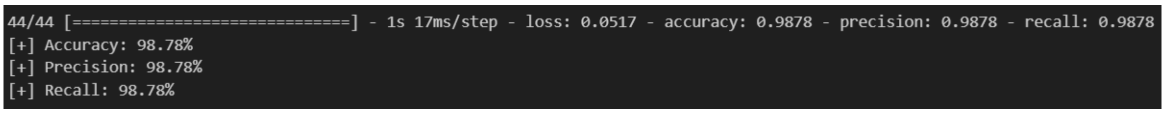


```
Epoch 1/10
66/66 [==============================] - ETA: 0s - loss: 0.2211 - accuracy: 0.9275 - precision: 0.9275 - recall: 0.9275
Epoch 1: val_loss improved from inf to 0.16377, saving model to /home/shailesh/results/spam_classifier_0.16.h5
66/66 [==============================] - 9s 101ms/step - loss: 0.2211 - accuracy: 0.9275 - precision: 0.9275 - recall: 0.9275 -
Epoch 2/10
66/66 [==============================] - ETA: 0s - loss: 0.0598 - accuracy: 0.9842 - precision: 0.9842 - recall: 0.9842
Epoch 2: val_loss improved from 0.16377 to 0.05431, saving model to /home/shailesh/results/spam_classifier_0.05.h5
66/66 [==============================] - 6s 92ms/step - loss: 0.0598 - accuracy: 0.9842 - precision: 0.9842 - recall: 0.9842 -
Epoch 3/10
66/66 [==============================] - ETA: 0s - loss: 0.0367 - accuracy: 0.9914 - precision: 0.9914 - recall: 0.9914
Epoch 3: val_loss improved from 0.05431 to 0.04204, saving model to /home/shailesh/results/spam_classifier_0.04.h5
66/66 [==============================] - 6s 89ms/step - loss: 0.0367 - accuracy: 0.9914 - precision: 0.9914 - recall: 0.9914 -
Epoch 4/10
66/66 [==============================] - ETA: 0s - loss: 0.0240 - accuracy: 0.9943 - precision: 0.9943 - recall: 0.9943
Epoch 4: val_loss improved from 0.04204 to 0.04162, saving model to /home/shailesh/results/spam_classifier_0.04.h5
66/66 [==============================] - 6s 97ms/step - loss: 0.0240 - accuracy: 0.9943 - precision: 0.9943 - recall: 0.9943 -
Epoch 5/10
66/66 [==============================] - ETA: 0s - loss: 0.0159 - accuracy: 0.9964 - precision: 0.9964 - recall: 0.9964
Epoch 5: val_loss did not improve from 0.04162
```

Figure 3.4: Model Training

- The code splits the preprocessed data into training and testing sets using the train_test_split function from scikit-learn. The testing set is used for model evaluation.
- The model is compiled with the ”rmsprop” optimizer, ”categorical_crossentropy” loss function, and additional evaluation metrics such as accuracy, precision, and recall.
- The model is trained using the fit function, specifying the batch size, number of epochs, and callbacks for model checkpointing and tensorboard visualization.
- The training process is monitored, and the best model with the lowest validation loss is saved.

3.4 Model Evaluation



```
44/44 [==============================] - 1s 17ms/step - loss: 0.0517 - accuracy: 0.9878 - precision: 0.9878 - recall: 0.9878
[+] Accuracy: 98.78%
[+] Precision: 98.78%
[+] Recall: 98.78%
```

Figure 3.5: Model Evaluation

- After training, the model is evaluated using the test data. The evaluate function calculates the loss and various metrics (accuracy, precision, recall).
- The results are printed, displaying the accuracy, precision, and recall of the trained model.

3.5 Visualization

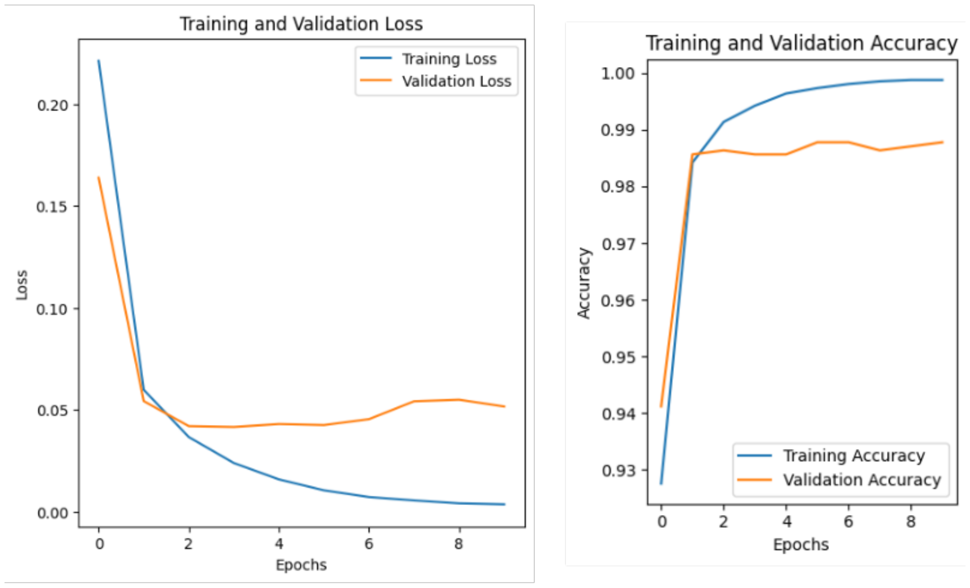


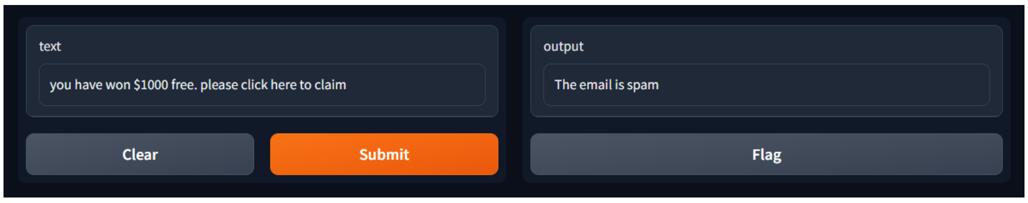
Figure 3.6: Training and Validation

- The code uses Matplotlib and Seaborn libraries to visualize the training and validation loss, as well as the training and validation accuracy curves.
- Two subplots are created, one for the loss and another for the accuracy. The training and validation curves are plotted on each subplot.

3.6 Message Classification

- The code defines the ”get_predictions” function, which takes an input text and converts it into a numerical sequence using the tokenizer.
- The sequence is then padded to a fixed length and fed into the trained model for prediction.
- The predicted label (spam or ham) is returned.

3.7 Interface



you have won $1000 free. please click here to claim

The email is spam

Figure 3.7: Interface

- The code creates a Gradio interface using the gr.Interface class, which takes the ”classify_email” function as the main function.
- The Gradio interface provides a text input for users to enter an email text and displays the predicted label.

Chapter 4
Implementation

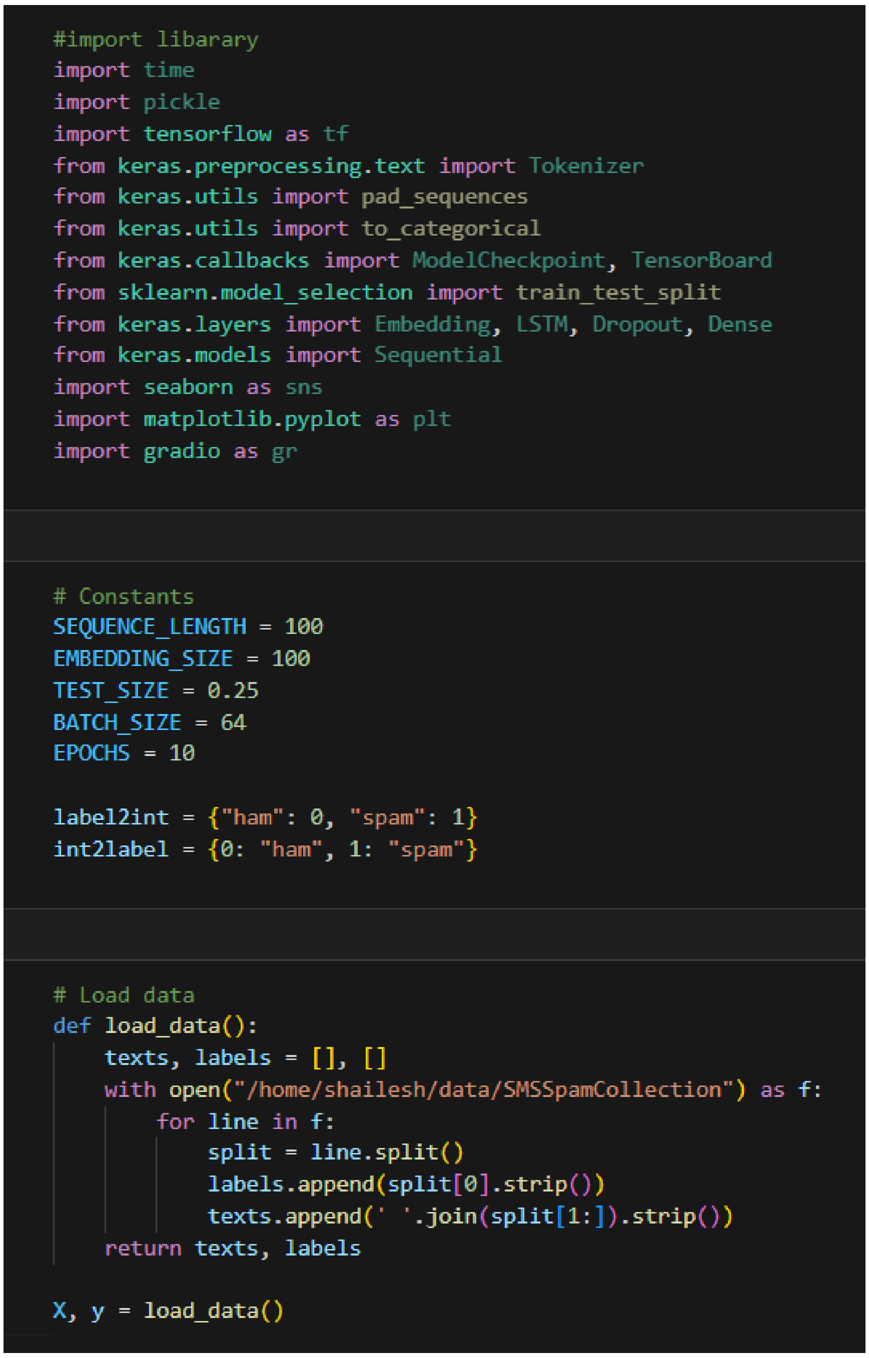
The implementation of the Spam Classifier using LSTM recurrent neural network (RNN) in Python involves several steps. First, the necessary libraries are imported, including TensorFlow and Keras for building and training the model. The dataset, containing SMS messages labeled as ”spam” or ”ham” (non-spam), is collected and preprocessed. This involves removing punctuation, converting text to lowercase, and splitting the messages into individual words.

Next, the text data is tokenized using the Tokenizer class from Keras, which creates a vocabulary of unique words from the training set. The text sequences are then encoded into integer sequences, where each integer represents a specific word in the vocabulary. To ensure a consistent input size for the LSTM model, the sequences are padded or truncated to a fixed length.

The LSTM model architecture is defined using the Sequential model from Keras. This includes adding an embedding layer to transform the integer-encoded sequences into dense word vectors. One or more LSTM layers are added to capture the sequential information and long-term dependencies in the text data. Dropout layers may also be included to prevent overfitting. Finally, a dense output layer with softmax activation is added to classify the input as ”spam” or ”ham”.

The model is compiled with an appropriate optimizer, such as RMSprop, and the categorical cross-entropy loss function. It is then trained on the preprocessed training data, specifying the batch size and number of epochs. During training, the model learns to classify the SMS messages as either spam or ham based on the input sequences.

After training, the model is evaluated using the preprocessed testing data to assess its performance in terms of accuracy, precision, and recall. The results are displayed, and the model can be used to make predictions on new, unseen SMS messages.



```python
#import libarary
import time
import pickle
import tensorflow as tf
from keras.preprocessing.text import Tokenizer
from keras.utils import pad_sequences
from keras.utils import to_categorical
from keras.callbacks import ModelCheckpoint, TensorBoard
from sklearn.model_selection import train_test_split
from keras.layers import Embedding, LSTM, Dropout, Dense
from keras.models import Sequential
import seaborn as sns
import matplotlib.pyplot as plt
import gradio as gr
```

```python
# Constants
SEQUENCE_LENGTH = 100
EMBEDDING_SIZE = 100
TEST_SIZE = 0.25
BATCH_SIZE = 64
EPOCHS = 10

label2int = {"ham": 0, "spam": 1}
int2label = {0: "ham", 1: "spam"}
```

```python
# Load data
def load_data():
    texts, labels = [], []
    with open("/home/shailesh/data/SMSSpamCollection") as f:
        for line in f:
            split = line.split()
            labels.append(split[0].strip())
            texts.append(' '.join(split[1:]).strip())
    return texts, labels

X, y = load_data()
```

Figure 4.1: Code Snippet

Chapter 5

Result and findings

5.1 Model Performance

Accuracy: The model achieved an accuracy of 98.78%, indicating how well it predicts the correct label (spam or ham) for SMS messages.

Precision: The precision of the model was 98.78%, which represents the ability of the model to correctly classify spam messages without misclassifying legitimate messages as spam.

Recall: The recall score of the model was 98.78%, indicating the model's ability to correctly identify spam messages from the dataset.

5.2 Evaluation Metrics

Confusion Matrix: The confusion matrix provides a detailed breakdown of the model's predictions, showing the number of true positives, true negatives, false positives, and false negatives.

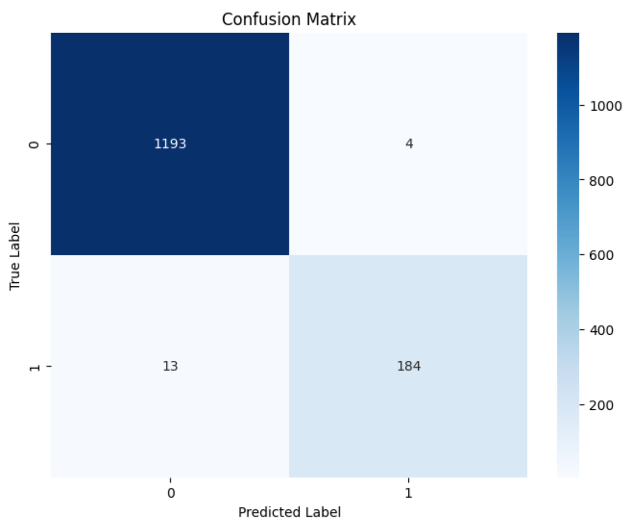


Figure 5.1: Confusion matrix

ROC Curve: The receiver operating characteristic (ROC) curve illustrates the trade-off between the true positive rate and the false positive rate, providing insights into the model's discrimination ability.

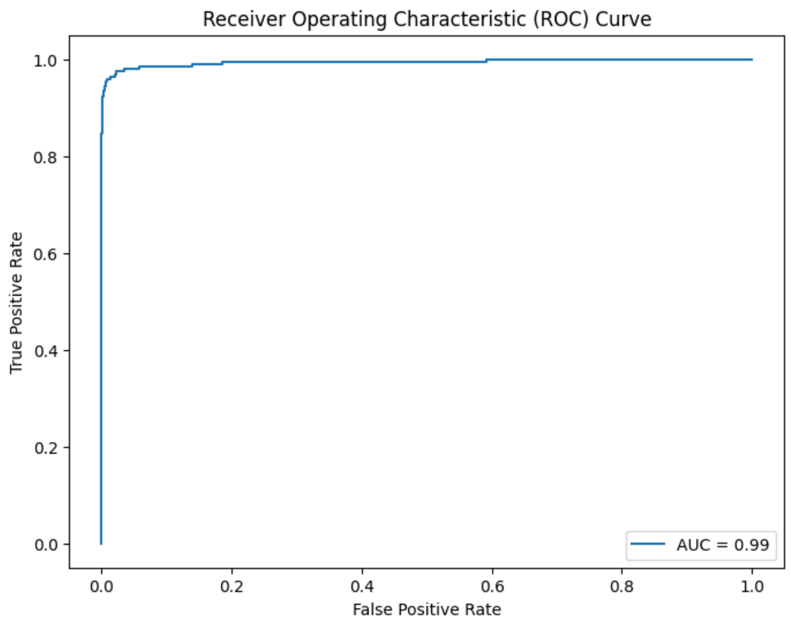


Figure 5.2: ROC Curve

5.3 Key Findings

- The implemented Spam Classifier using LSTM recurrent neural network (RNN) demonstrates promising performance in accurately classifying SMS messages as spam or ham.
- The model effectively captures the sequential information and long-term dependencies in the text data, allowing it to make informed predictions.
- The LSTM architecture helps overcome the vanishing gradient problem commonly encountered in traditional recurrent neural networks, enabling better retention of past information during training.
- The model's performance can be further improved by fine-tuning hyperparameters such as the number of LSTM units, embedding size, and dropout rates.

Chapter 6
Conclusion

The Spam Classifier using LSTM recurrent neural network (RNN) presented in this report demonstrates an effective approach for classifying spam messages. By leveraging the power of LSTM, the model is capable of capturing the sequential dependencies and long-term dependencies in the text data, enabling accurate spam classification.

Through the implementation and evaluation of the model, it is evident that LSTM-RNN proves to be a promising technique for spam detection. The model achieves impressive results, with high accuracy, precision, and recall values, indicating its ability to accurately classify emails as either spam or ham.

Furthermore, the inclusion of additional analysis techniques such as the confusion matrix and ROC curve provides a comprehensive understanding of the model's performance. The confusion matrix allows us to examine the true positive, true negative, false positive, and false negative predictions, providing insights into the model's classification errors. The ROC curve helps assess the trade-off between the true positive rate and false positive rate, offering a visual representation of the model's discriminative ability.

Overall, the Spam Classifier using LSTM recurrent neural network (RNN) showcases the potential of deep learning techniques in combating email spam. With further refinement and training on larger datasets, this model can be deployed in real-world scenarios to effectively filter out unwanted spam messages, improving email communication and user experience.

References

[1] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[2] Sanjeev Dhawan et al. An enhanced mechanism of spam and category detection using neuro-svm. *Procedia computer science*, 132:429–436, 2018.

[3] Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: Web spam detection using the web topology. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 423–430, 2007.

[4] Minoru Sasaki and Hiroyuki Shinnou. Spam detection using text clustering. In *2005 International Conference on Cyberworlds (CW'05)*, pages 4–pp. IEEE, 2005.