A PROJECT REPORT ON

“Text Classification using LSTM RNN”

**Submitted**

*In the partial fulfilment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

In

**COMPUTER SCIENCE & ENGINEERING**

By

**Yeshwanth Zagabathuni (171FA04512)**

**Gokul Krishna Sesank (171FA04546)**

Under the esteemed guidance of

**Mr. K.Pavan Kumar, Assistant Professor**

**Dr. Dondeti Venkatesulu, HOD Department of CSE**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH**

(**Accredited by NAAC “A” grade**) **Vadlamudi, Guntur.**

A PROJECT REPORT ON

“Text Classification using LSTM RNN”

**Submitted**

*In the partial fulfilment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

In

**COMPUTER SCIENCE & ENGINEERING**

By

**Yeshwanth Zagabathuni (171FA04512)**

**Gokul Krishna Sesank (171FA04546)**

Under the esteemed guidance of

**Mr. K.Pavan Kumar, Assistant Professor**

**Dr. Dondeti Venkatesulu, HOD Department of CSE**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH**

(**Accredited by NAAC “A” grade**) **Vadlamudi, Guntur.**

**VIGNAN’S FOUNDATION FOR SCIENCE TECHNOLOGY AND RESEARCH**

**(Accredited by NAAC “A” grade)**



**CERTIFICATE**

This is to certify that the Project Report entitled **“Text Classification using LSTM RNN”** that is being submitted by **Yeshwanth Zagabathuni (171FA04512), Gokul Krishna Sesank (171FA04546)** in partial fulfilment for the award of B.Tech degree in Computer Science and Engineering to the Vignan’s Foundation for Science, Technology and Research, Deemed to be University, is a record of bonafide work carried out by them under our supervision.

**Mr. K. Pavan Kumar External Examiner Dr. Dondeti Venkatesulu Assistant Professor, (Ph.D.) Professor, HOD**

# ACKNOWLEDGEMENT

We are very grateful to our beloved Chairman **Dr. Lavu. Rathaiah**, and Vice Chairman

**Mr. Lavu. Krishna Devarayalu,** for their love and care.

It is our pleasure to extend our sincere thanks to Vice-Chancellor **Dr. M.Y.S. Prasad** and Dean Engineering & Management, **Dr. M. Santhi Sree Rukmini,** for providing an opportunity to do my academics in VFSTR.

It is a great pleasure for me to express my sincere thanks to **Dr. Dondeti Venkatesulu HOD, CSE** of VFSTR**,** for providing required infrastructure to do my Project.

We extend our whole hearted gratitude to **Mr. K. Pavan Kumar**, under whose valuable guidance that the project came out successfully after each stage.

We feel it our responsibility to thank **Mrs. D. Radha Rani** and **Mr. Vijay Babu,** for helping in all required aspects during the course.

We thank all our **Faculty** members and **Programmers** of Department of Computer Science and Engineering who provided technical support to us in our academics throughout course.

Finally, we wish to express thanks to our family members for the love and affection overseas and forbearance and cheerful depositions, which are vital for sustaining effort, required for completing this work.

With Sincere regards,

**Zagabathuni Yeshwanth (171FA04512)**

**Gokul Krishna Sesank (171FA04546)**

Date:

# ABSTRACT

Sequence Classification is one of the on-demand research projects in the field of natural language processing. Classifying a set of images or text into an appropriate category or class is a complex task that many machine learning models failed to accomplish and ended up under-fitting or over-fitting the dataset. Some of the algorithms used in text classification are Naïve Bayes, SVM, CNN, RCNN, LSTM and the recently popularized GRUs and Transformers. In this project, we have chosen the LSTM RNNs for text-classification. The dataset we used is from Kaggle consisting of over 5000 entries and have added a 150 more entries ourselves. The two possible class labels are spam and ham. Each entry consists of the class label, a few sentences of text followed by a few useless features.

As a part of data pre-processing we have removed the attributes other than class-label and the text as they play little or no part in the classification process. For classification simplicity we have converted the label Ham into 0 and Spam into 1 using a standard encoding. Next, the text has to be transformed before calling the LSTM model. It is passed into a tokenizer first and is then converted into sequences each of length 150. The gaps in the sequences are first padded with zeros and they are thus transformed into sequence matrices. These matrices are passed into the LSTM model with a train-test-split of 7:3.

In our experimental studies we observed that our proposed LSTM architecture gives better classification accuracy than the standard machine learning models. We achieved 98.22% Accuracy and 93.69% of F1-Score whereas a traditional CNN achieved a mere 25.8% Accuracy and 23% F1-Score. Another interesting observation is that proposed model has reduced number of false positives and false negatives than any other model.

**Keywords:** Text Classification, Sequence Classification, Convolutional Neural Network (CNN), Machine Learning Models (ML), Long Short-Term Memory Recurrent Neural Networks (LSTM RNNs).

TABLE OF CONTENTS

**Acknowledgement**………………………………………………………………………….4

**Abstract**…………………………………………………………………………………….5

**1.Introduction**10

1.1.Sequence Classification10

1.2.Text Classification11

1.3.Hardware and Software Requirements……………………………………………………....11

**2.Literature Survey**11

2.1.Text Classification using Machine Learning Techniques12

2.2.A C-LSTM model for Text Classification13

2.3.Document Classification using LSTM Neural Network………………………………….....14

2.4.Actionable and Political Text Classification using Word Embeddings and LSTM…………15

**3.Proposed Methodology**.....................................................................................................16

3.1.Feature Elimination………………………………………………………………….……..16

3.2.Text Tokenization and Sequencing………………………………………………………...17

3.3.Padding…………………………………………………………………………………….18

3.4.LSTM……………………………………………………………………………………….18

3.5.Set Epochs…………………………………………………………………………………18

3.6.Evaluate the model…………………………………………………………………………19

**4.Implementation**………………………………………………………………………..20

**5.Experimental Results**………………………………………………………………….24

5.1.Dataset Summary…………………………………………………………………………..24

5.2.Experimental Results………………………………………………………………………24

**6.Conclusion**……………………………………………………………………………..25

**References**………………………………………………………………………………….26

**LIST OF FIGURES AND TABLES**

**FIGURES**

Figure 1.1.Examples of Sequence Classification…………………….10

Figure 2.1.Text Classification using Machine Learning Techniques...12

Figure 2.2.A C-LSTM Model for Text Classification………………..13

Figure 2.3.Document Classification using LSTM Neural Network….14

Figure 2.4.Actionable and Political Text Classification using Word Embeddings and LSTM………………………………………….............................15

Figure 3.1.Proposed Architecture…………………….…….………...16

Figure 3.2.Dataset before Pre-Processing……………….…...……….17

Figure 3.3.Dataset after Pre-Processing…………….…………...……17

Figure 3.4.Sequence Matrix after Padding……………………...…….18

Figure 3.5.Confusion Matrix….............................................................19

**TABLES**

Table 5.1.Dataset Summary………………………………………….24

Table 5.2.Experimental Results………………………………………24

**LIST OF ABBREVATIONS**

ML – Machine Learning

SVM – Support Vector Machines

CNN – Convolutional Neural Networks

RNN – Recurrent Neural Networks

LSTM - Long Short Term Memory

NLP – Natural Language Processing

GRU – Gate Recurrent Units

KNN – K Nearest Neighbors

C – LSTM – Convolutional Long Short Term Memory

NLTK – Natural Language Tool Kit

FP – False Positive

FN – False Negative

**CHAPTER - 1** **INTRODUCTION**

## Sequence Classification

Sequence classification is a branch of Natural Language Processing which deals with classifying a set of images or text into a certain class. This may be a simple binary classification problem in certain cases and a multi-class classification problem in the others.

What makes this problem difficult is that the sequences can vary in length, be comprised of a very large number of input symbols and may require the model to learn the long-term dependencies between symbols in a given input sequence.

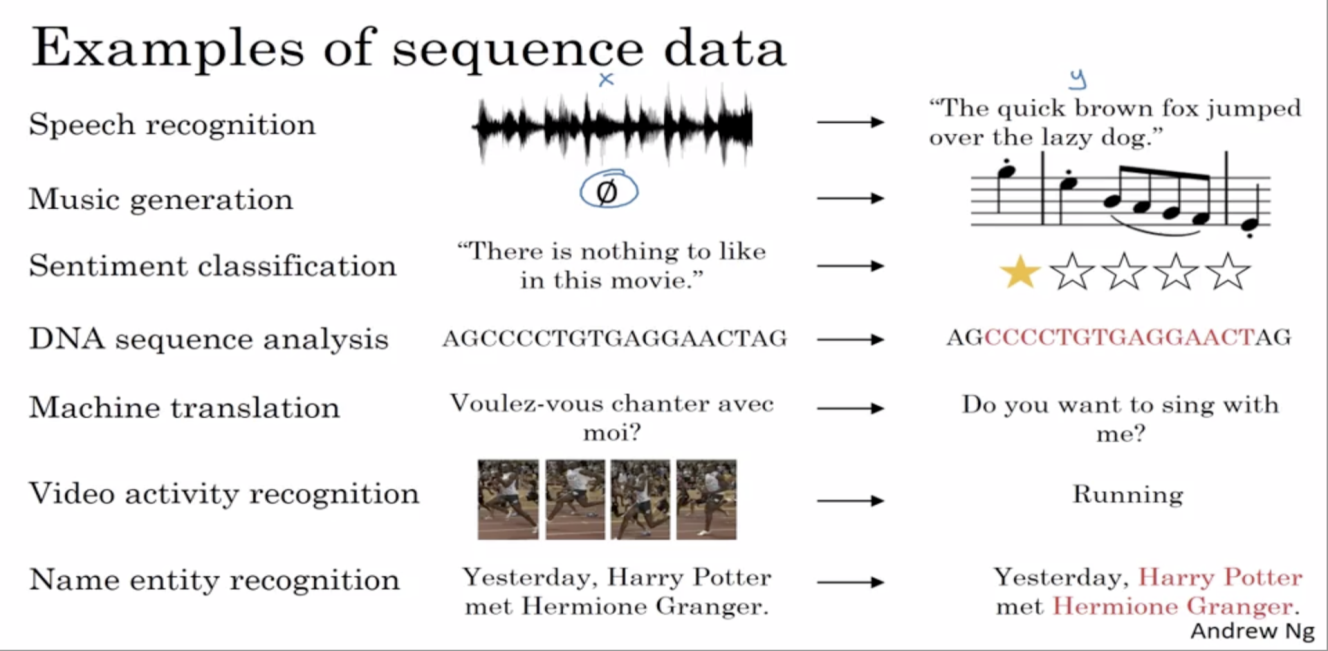


Figure 1.1

The above image courtesy, the Google Brains lead Prof. Andrew Ng, shows the various applications of sequence classification. Some of the notable applications from the image are DNA sequence analysis which may be used to find whether a person is affected by a certain disease, Speech recognition which is used to identify the voice of a certain individual in a noise, Video activity recognition which involves training a model to identify the events that took place at every minute or second of the video and Sentiment Classification which involves finding out the sentiment of an individual based on his review of a certain movie or product.

## Text Classification

## Text Classification is one of the many kinds of sequence classification. It is the process of predicting a class label for a given set of text. It is an essential component majorly in search engine based applications such as information filtering, web searching and sentiment analysis.

## There are many models used in text classification although not necessarily do all models fit a given dataset. While some models may end up showing drastically low levels of accuracy, others may be a highly accurate fit.

## Text classification is definitely not an easy task. Considering its unstructured nature, extracting useful insights from it and eliminating the useless ones can be hard and time consuming.

## Many of the conventional ML algorithms such as SVM, Naïve Bayes and CNNs may end up under-performing on high volumes of text data. One of the many reasons for this may be their limitation in terms of memory. However a few RNN algorithms such as LSTMs, GRUs, Transformers and Vanilla RNNs may be a much better fit. The complex structures involved in these neural networks help to first analyze and structure the text effectively. This structuring makes it much easier for classification tasks be it binary or multi-class.

## The development of pre-trained word embedding and deep neural networks has brought new inspiration to various NLP tasks. Word embedding is a distributed representation of words and greatly alleviates the data sparsity problem. These embeddings can capture meaningful syntactic and semantic regularities. Further processing of the text may involve tokenizing, vectorization, stop words removal, Feature Selection and so on.

* 1. **Hardware and Software Requirements**

**Hardware Requirements**

* RAM : 8GB or higher
* Intel Core : i5 or higher
* Minimum memory card required : 8GB
* Minimum Graphic Card : NVIDIA GeForce GTX 970

**Software Requirements**

* Windows OS (8 or higher) or any Linux OS.
* Python (3.6 or higher).
* CSV file support.
* High speed internet connectivity.
* MS-Office support.
* Google Co-lab
* Jupyter Notebook or Anaconda Navigator

**CHAPTER - 2**

**LITERATURE SURVEY**

## Text Classification using Machine Learning Techniques

## 

Figure 2.1

The above architecture gives a comprehensive overview of text classification:

* After a text document is read, it is broken down into a series of tokens.
* Then the words of similar context or meaning are grouped by the process of stemming.
* In the process, many functional words may be generated which may be useful to forming a meaningful sentence but not to the text classification task. Such words are called stop words and are eliminated.
* After stemming and stopwords removal, the tokens are used to form word vectors or the vectorized representation of text.
* After that we decide on the useful features, useless features and features that may need to be transformed. Feature selection and feature transformation process is carried out.
* Finally, the desired machine learning algorithm is implemented and the processed data and features are fed to it.
* The machine learning algorithm used were SVM, KNN and Naïve Bayes.
* Several measures from the confusion matrix such as Accuracy, Precision, Recall and Fß were used to evaluate the results.

* 1. **A C-LSTM Neural Network for Text Classification**

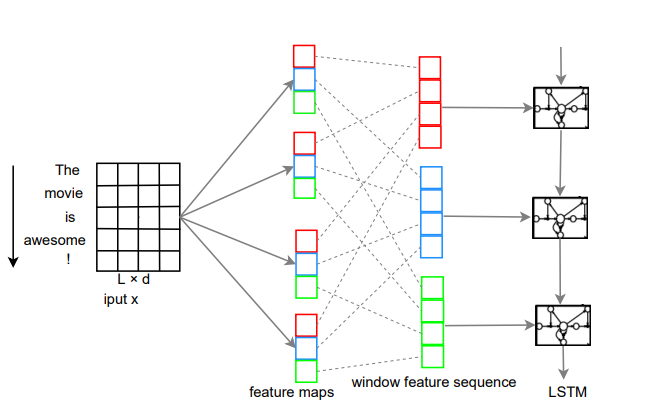
****

Figure 2.2

* The C-LSTM architecture is as shown above. It contains 2 major components: CNN and LSTM.
* The CNN involves finding a feature map over each window vector. If we consider a window vector w with filter m with bias b, then the overall weighted summation passes through a non-linear activation function and can be used to find the feature map as follows:

Cj = f(wj\*m+b)

* The LSTM is responsible to capture long-term dependencies over window feature sequences.
* It has a range of repeated modules and the output is controlled by a set of gates as a function of hidden state ht-1 and input of current time step xt: the forget gate ft, the input gate it and output gate ot.
* The above architecture is for sentence modelling. All the set of sentences are fit into matrices of size **L**x**d**. As we can clearly see in the figure, each word is fitted row-wise each of length **L** and there are **d** such words.
* Blocks of the same color in the feature map layer and window feature sequence layer correspond to features for the same window.
* The dashed lines connect the feature of a window with the source feature map. The final output of the entire model is the last hidden unit of LSTM.
  1. **Document Classification using LSTM Neural Network**

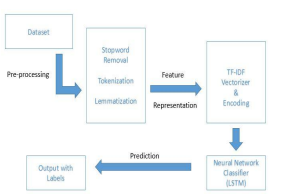
****

Figure 2.3

* This is an approach in the field of data mining. The basic process involved is as follows.
* **Data pre-processing:** This stage prepares the document for the categorization process. The pre-processing stage includes the common steps Tokenization, Removing stop words and Stemming. These steps are used to process the document so that features can be successfully extracted through them.
* **Stop word removal:** The stop words are filtered out after processing of natural language data. A general strategy would be to sort the words with the most frequency. Such words are available in the NLTK library and can be imported.
* **Tokenization:** It relies mostly on separation by whitespace, punctuations or line breaks to separate tokens. Errors generated in this stage are critical.
* **Lemmatization:** It is similar to stemming. A stemmer cannot recognize the difference between words that have different meanings as it does not depend on the knowledge of the context.
* Lemmatization refers to the process of removal of only inflectional endings and to return the dictionary form of the word. Ex: walk may appear as walks, walked and walking. The base form walk is the lemma. NLTK has a built-in lemmatizer for the same.
* **Feature Selection:** This helps in improving the efficiency, scalability and accuracy of a document classifier. It is used to select a subset of features with minimal information loss or entropy.
* **Categorization Algorithm**: The LSTM RNN is applied on the transformed dataset and is evaluated on its performance.
* **Evaluation:** The metrics from the confusion matrix Precision, Recall and F-score are used for evaluating the model.
  1. **Actionable and Political Text Classification using Word Embeddings and LSTM**

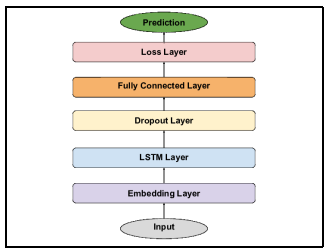
****

Figure 2.4

* This was another approach which was used for classification in two problem domains. One of them was to classify social media messages as *Action-able* or *Non-Actionable*. The other was with respect to political learning, where social media messages were classified as *Democratic* or *Republican.* The various layers used can be explained as follows:
* **Embedding Layer:** It turns positive integer indices in the input into dense real-valued vectors of fixed size, determined by the number of units in this layer.
* **LSTM Layer:** A layer with multiple LSTM units is used. It consists of 4 main components: an input gate, a self-recurrent connection, a forget gate and an output gate.
* **Dropout Layer:** It is a regularization technique to avoid overfitting. It is achieved by randomly dropping a fraction of units while training a neural network.
* **Fully-Connected Layer:** It has full connections to all activations in the previous layer. This layer is used to learn non-linear combinations of higher level features learnt by the previous layers in the network.
* **Loss Layer:** The final layer in the network is the loss layer that determines how deviations between predicted labels and actual labels are penalized. Since this project used binary classification, the binary *cross-entropy* function is used.

**CHAPTER - 3**

**PROPOSED METHODOLOGY**

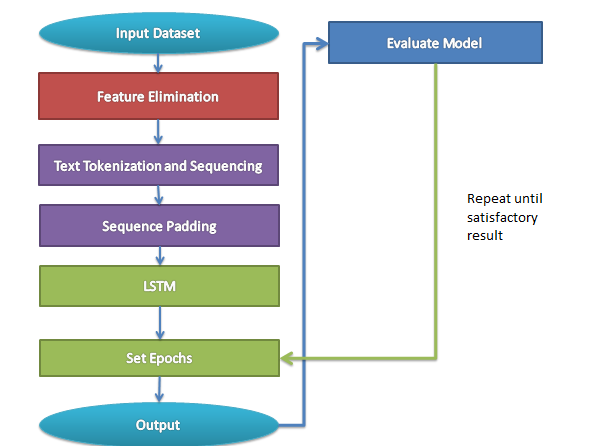


Figure 3.1

## 

## Our proposed methodology is as shown above. Each step is explained separately below:

## Feature Elimination

## The first step is to eliminate features which are of the least significance in the dataset. These features cannot be of any use in the classification task and are thus eliminated to save memory.

## The dataset consists of 5619 entries and 5 features initially. The features v1 to v2 are essential in the classification task and hence they are unchanged.

## Whereas, the features v3 to v5 do not carry any useful information that contribute to the LSTM model for class-label prediction. Hence, these features are dropped or eliminated.

## The initial dataset and the transformed dataset can both be visualized in the figures below:

## 

Figure 3.2

## The transformed dataset is as shown below:

## 

Figure 3.3

## Text Tokenization and Sequencing

## After splitting the dataset into a train test split of 7:3, we now need to transform the text in each entry before feeding it to the LSTM.

## The first of the transformations is called tokenizing. In this process the words are separated by spaces or special characters. Such characters are removed and individual words are tokenized in maximum word counts of 1000. This is done by fitting a tokenizer object to the train dataset.

## 3.3 Padding

## The next step is to create text sequences and this is done by the texts\_to\_sequences() function. Afterward the remaining spaces are padded with zeros using the pad\_sequences() function.

## Finally, sequence matrices of max length 150 are formed. A sample of such matrices is as shown below.

## 

Figure 3.4

## LSTM

## The LSTM function is called next. It makes use of a total of 2 activation functions namely “relu” and “sigmoid”.

## Since, there are only 2 classes, the loss function is chosen as binary cross-entropy. Then the train data-set is passed for training.

## After training, our next task is to evaluate the dataset. For this purpose, the test dataset is to be used. It undergoes the same transformations as the train set.

## Set Epochs

## Before we begin jumping to conclusions, we need to decide if the number of epochs for the model are too less or too much.

## This number should be chosen such that the model is just perfect for the dataset and does not end up overfitting the dataset as that is one of the risks of LSTM.

## When we see perfect values such as 0% or 100%, it may mean that we have over-trained the model and we need to reduce the number of epochs accordingly.

## Evaluate the Model

## The model makes use of the metrics from the confusion matrix which are accuracy, precision and F1-score. The confusion matrix is visualized as follows:

## 

Figure 3.5

## Other than the metrics shown above, there are several other metrics that can be used to assess the performance of the model.

## The similarity measures such as Jaccard Quotient, Asymmetric binary similarity and Cosine similarity also serve as good evaluation metrics.

## Furthermore, distance based metrics such as adjusted r-score, MSE and RMSE also serve as really good evaluation metrics.

## In our project, we have used Precision, Recall, F1-Score as the confusion matrix metrics and Jaccard Similarity and Cosine Similarity as similarity measures.

## The epochs are initially set to 10 and increased or decreased based on under-performing or over-performing on the dataset respectively.

## 

**CHAPTER-4 IMPLEMENTATION**

* Import necessary modules
* import pandas as pd
* import numpy as np
* import matplotlib.pyplot as plt
* import seaborn as sns
* from sklearn.preprocessing import LabelEncoder
* from sklearn.model\_selection import train\_test\_split
* from keras.models import Model
* from keras.layers import LSTM,Activation,Dense,Dropout,Input,Embedding
* from keras.optimizers import RMSprop
* from keras.preprocessing.text import Tokenizer
* from keras.preprocessing import sequence
* from keras.callbacks import EarlyStopping
* %matplotlib inline
* Load dataset
* df=pd.read\_csv('/content/spam.csv',delimiter=',',encoding='latin-1')
* df.tail(10)
* Eliminate useless features
* df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1,inplace=True)
* print(df.head(10))
* print('\nThe size of the dataset in tuples and attributes is ',df.shape)
* View the summary of dataset in a counterplot.
* sns.countplot(df.v1)
* plt.xlabel('Type of Text')
* plt.ylabel('Count')
* plt.title('Distribution of Class Labels')
* View a few examples of spam and ham messages.
* print("The following sentences are examples of spam ",df[df["v1"]=="spam"].head(),end='\n\n')
* print("The following sentences are examples of non-spam or ham ",df[df["v1"]=="ham"].head())
* Encode the labels as spam: 1 and ham: 0 respectively.
* X=df.v2
* Y=df.v1
* le=LabelEncoder()
* Y=le.fit\_transform(Y)
* Y=Y.reshape(-1,1)
* Now tokenize the text. Then perform necessary padding to form a sequence matrix. View the first 2 matrices of the sequence matrix.
* Xtrain,Xtest,Ytrain,Ytest=train\_test\_split(X,Y,test\_size=0.3)
* #Next let's tokennize the data
* max\_words=1000
* max\_len=150
* tok=Tokenizer(num\_words=max\_words)
* tok.fit\_on\_texts(Xtrain)
* sequences=tok.texts\_to\_sequences(Xtrain)
* seq\_matrix=sequence.pad\_sequences(sequences,maxlen=max\_len)
* print('This is a sample of the padded sequence for the text ',seq\_matrix[:2][:],sep='\n')
* Now, let us implement the LSTM function before we call it.
* def RNN():
* inputs=Input(name='Inputs',shape=[max\_len])
* layer=Embedding(max\_words,50,input\_length=max\_len)(inputs)
* layer=LSTM(64)(layer)
* layer=Dense(256,name='FC1')(layer)
* layer=Activation('relu')(layer)
* layer=Dropout(0.5)(layer)
* layer=Dense(1,name='out\_layer')(layer)
* layer=Activation('sigmoid')(layer)
* model=Model(inputs=inputs,outputs=layer)
* return model
* Compile the function with specifying the loss function and optimizer.
* model = RNN()
* model.summary()
* model.compile(loss='binary\_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
* Fit the model with required parameters and most importantly epochs.
* md1=model.fit(seq\_matrix,Ytrain,batch\_size=128,epochs=4,
* validation\_split=0.2,callbacks=[EarlyStopping(monitor='val\_loss',min\_delta=0.0001)])
* Now let us transform the test set as we did for the train set.
* tst\_sequences=tok.texts\_to\_sequences(Xtest)
* tst\_sequences\_matrix=sequence.pad\_sequences(tst\_sequences,maxlen=max\_len)
* Now on, let us evaluate the model.
* accr = model.evaluate(tst\_sequences\_matrix,Ytest)
* print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(accr[0],accr[1]))
* To perform evaluation we first convert the series of continuous values in YPred to discrete values which are the class labels.
* predictions\_nn\_test = model.predict(tst\_sequences\_matrix)
* for i in range(len(predictions\_nn\_test)):
* if predictions\_nn\_test[i][0] < 0.5:
* predictions\_nn\_test[i][0] = 0
* else:
* predictions\_nn\_test[i][0] = 1
* Now let us evaluate the model.
* from sklearn.metrics import accuracy\_score,confusion\_matrix,precision\_score,recall\_score,f1\_score
* print('Test accuracy = ', accuracy\_score(Ytest, predictions\_nn\_test))
* print('Test precision = ', precision\_score(Ytest, predictions\_nn\_test))
* print('Test Recall = ',recall\_score(Ytest, predictions\_nn\_test))
* print('Overall F1-Score = ',f1\_score(Ytest,predictions\_nn\_test))
* cnf\_matrix\_test = confusion\_matrix(Ytest, predictions\_nn\_test)
* print(cnf\_matrix\_test)
* Let us visualize the confusion matrix.
* import pylab as pl
* pl.matshow(cnf\_matrix\_test)
* pl.title('Confusion Matrix Visualization\n')
* pl.colorbar()
* pl.show()
* Let us view a view similarity measures as well.
* from sklearn.metrics import mean\_squared\_error,jaccard\_score
* from math import sqrt
* def square\_rooted(x):
* return round(sqrt(sum([a\*\*2 for a in x])),3)
* def cosine\_similarity(x,y):
* numerator=sum(a\*b for a,b in zip(x,y))
* denominator=square\_rooted(x)\*square\_rooted(y)
* return numerator/float(denominator)
* print('The RMSE =',round(sqrt(mean\_squared\_error(Ytest,predictions\_nn\_test)),3))
* print('The Jaccard Similarity =',round(jaccard\_score(Ytest,predictions\_nn\_test),3))
* print('The Cosine Similarity =',round(cosine\_similarity(Ytest,predictions\_nn\_test)[0],3))
* The model is to be trained with different number of epochs each time based on under-fitting or over-fitting. Optionally, we can implement the CNN to understand why LSTM is much better.
* from sklearn.neural\_network import MLPClassifier
* clf = MLPClassifier(solver='lbfgs', activation='relu', alpha=1e-5,hidden\_layer\_sizes=(5, 2), random\_state=1)
* clf.fit(seq\_matrix, Ytrain)
* from sklearn.metrics import accuracy\_score,confusion\_matrix,precision\_score,recall\_score,f1\_score
* predictions\_nn\_test = clf.predict(tst\_sequences\_matrix)
* print('Test accuracy = ', accuracy\_score(Ytest, predictions\_nn\_test))
* print('Test precision = ', precision\_score(Ytest, predictions\_nn\_test))
* print('Overall F1-Score = ',f1\_score(Ytest,predictions\_nn\_test))
* pl.matshow(cnf\_matrix\_test)
* pl.title('Confusion Matrix Visualization\n')
* pl.colorbar()
* pl.show()

**CHAPTER - 5** **EXPERIMENTAL RESULTS**

In this Chapter, we give the details of the experimental studies we have performed to understand the limitations of the proposed algorithm. First, we introduce the dataset used for our studies. Later we present the results of proposed experiment.

## Dataset Summary

## The brief summary of the dataset is as follows. We can clearly view the number of spam and ham samples.

|  |  |
| --- | --- |
| **Class Label** | **# Samples** |
| Ham | 4859 |
| Spam | 760 |
| **Total** | **5619** |

Table 5.1

Next, let us view the results of the experiment.

**5.2 Experimental Results**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| epochs | Accuracy (%) | Precision (%) | F1-score (%) | Jaccard similarity (%) | Cosine similarity (%) |
| 10 | 95.19 | 100 | 78.62 | 64.80 | 80.50 |
| 9 | 98.10 | 96.65 | 92.66 | 86.30 | 92.70 |
| 8 | 98.22 | 99.01 | 93.08 | 87.10 | 93.30 |
| 7 | 98.45 | 93.88 | 94.29 | 89.20 | 94.30 |
| 6 | 97.98 | 93.36 | 92.54 | 86.00 | 92.00 |
| 5 | 98.51 | 98.09 | 94.27 | 89.20 | 94.40 |

Table 5.2

* As we can see in the results above, if we give the number of epochs as 10, the model has over-fit the dataset with a precision of 100%.
* The ideal number of epochs must be below 10. In our case we have the highest F1-Score at epochs=7.

**CHAPTER - 6**

**CONCLUSION**

Firstly, LSTM was a revolution in the field of NLP since its introduction in the 1990s. Having implemented the algorithm using the pre-defined modules and functions in Keras, we now know of its capabilities in the field of text classification. On evaluating its performance we also came to know of a few of its limitations and thus the need for GRUs, Transformers, C-LSTMs and so on.

Secondly, the dataset we dealt with was also furnished well. Having added over 150 entries ourselves as a part of the test set, we still achieved highly accurate classification with the minimum FPs and FNs.

Although LSTM turned out to be highly accurate in our experiment, there were cases were it over-fitted the dataset and the number of epochs had to be reduced accordingly. And also considering its complex structure which is not that easy to understand, the need for a simpler and efficient structure was inevitable.

# REFERENCES

1. M. Ikonomakis, S. Kotsiantis and V. Tampakas, “Text Classification using Machine Learning Techniques,” WSEAS Transactions on Computers, Issue 8, Volume 4, August 2005, pp 966-974.
2. Yash R. Ghorpade, Gauri R. Kanthale, Prof. Nihar M. Ranjan, Adishree R. Ghorpade and Abhishek S. Dubey, “Document Classification using LSTM neural network,” Journal of Data Mining and Management, Volume 2, Issue 2.
3. Chunting Zhou, Chonglin Sun, Zhiyuan Liu and Francis C.M. Lau, “A C-LSTM Neural Network for Text Classification”, arXiv:1511.08630v2 [cs.CL] 30 Nov 2015.
4. Adithya Rao and Nemanja Spasojevic “Actionable and Political Text Classification using Word Embeddings and LSTM”, arXiv:1607.02501v2 [cs.CL] 13 Jul 2016.
5. Lei Zhang, Shuai Wang, Bing Liu, “Deep Learning for Sentiment Analysis: A Survey”, Supported by National Science Foundation grant no. IIS1407927 and IIS1650900 and Huawei Technologies Co. Ltd
6. “Sentiment Classification using Document Embeddings trained with Cosine Similarity”, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pages 407-414, Florence, Italy, July 28-August 2,2019.