ISSN 2347 - 3983
Volume 9. No. 9, September 2021
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter11992021.pdf
https://doi.org/10.30534/ijeter/2021/11992021

Spam text classification using LSTM Recurrent Neural Network

Yeshwanth Zagabathuni
India, zyeshwanth@gmail.com

Received Date : August 05, 2021 Accepted Date : August 25, 2021 Published Date : September 07, 2021

ABSTRACT
Sequence Classification is one of the on-demand research projects in the field of Natural Language Processing (NLP). Classifying a set of images or text into an appropriate category or class is a complex task that a lot of Machine Learning (ML) models fail to accomplish accurately and end up under-fitting the given dataset.
Some of the ML algorithms used in text classification are KNN, Naïve Bayes, Support Vector Machines, Convolutional Neural Networks (CNNs), Recursive CNNs, Recurrent Neural Networks (RNNs), Long Short Term Memory (LSTM), etc. For this experimental study, LSTM and a few other algorithms were chosen for a more comparative study.
The dataset used is the SMS Spam Collection Dataset from Kaggle and 150 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.
In experimental studies, the LSTM gives much better classification accuracy than the other machine learning models. F1-Scores in the high nineties were achieved using LSTM for classifying the text. The other models showed very low F1-Scores and Cosine Similarities indicating that they had underperformed on the dataset. Another interesting observation is that the LSTM had reduced the number of false positives and false negatives than any other model.

Key words: Long Short-Term Memory, Recurrent Neural Networks, Sequence Classification, Text Classification

1. INTRODUCTION

Sequence classification is a branch of Natural Language Processing that 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 others. "Conventional text classifiers often rely on many human-built features, such as dictionaries, knowledge bases, thesaurus, index terms and special tree kernels"[1].
What makes this problem difficult is the dynamic nature of input sequences or variable-length input sequences, often "comprised of a very large number of input symbols which may require the model to learn the long-term dependencies between symbols"[2]. Several deep learning algorithms were introduced to overcome each one of the issues.
CNN may end up classifying structured data with excellent accuracy but there is a problem with unstructured data, especially text. This is where it may underperform.
"Recently, the rapid development of pre-trained word embeddings and deep neural networks has brought new inspiration to various Natural Language Processing tasks"[1]. Word embedding is referred to as a distributed representation of words and deals with the data sparsity problem to a large extent[3]. "One of the most commonly used word embedding systems is Word2Vec, which is essentially a computationally-efficient neural network prediction model that learns word embeddings from the text"[4]. It comprises of the Continuous Bag of Words (CBOW) which predicts the missing word from given context words (Ex: I am *climbing* the tree.) and the Skip-Gram model (SG) which predicts the context words based on target words (Ex: *I am* climbing *the tree*.)[4].
An interesting development on neural networks working with word embeddings was the Recurrent Neural Network, introduced in the 1980s. However, even it had its limitations. The problem of limitation in the amount of data it can store did affect its performance. For example, consider "*My name is Suresh. I'm from Karnataka. I speak …..*". In this example, the RNN can fill the gap as Kannada. However, let us imagine around 500 sentences in-between the first two sentences and the last sentence. Now, based on "*I speak*", the RNN will identify that it is a language. But the question is what language? To

know the answer the RNN needs to trace back 500 sentences to know. On the way, it may need to add the necessary information and eliminate unnecessary information.

All of the above problems called for a much more complex architecture that functions via complex coordination between complex logical structures called gates. This model came to be known as the LSTM, introduced in the late 1990s. It is used in numerous sequence classification applications. Furthermore, GRU was later introduced as a simpler structure compared to the LSTM. Given below in Figure 1, are a few examples of sequence classification.

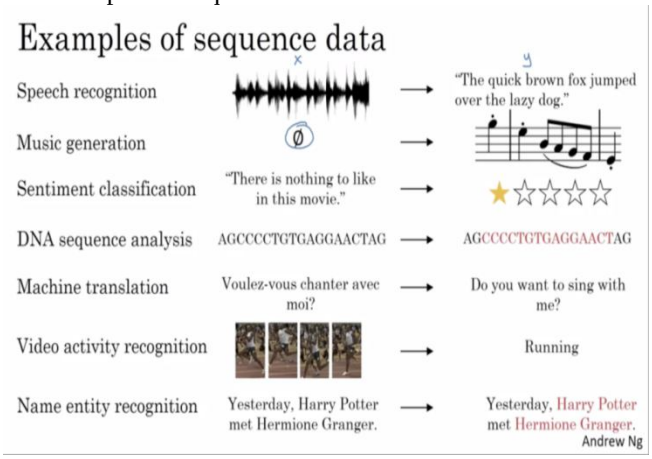


Figure 1: **Source:**[5] The above image courtesy, the Google Brains lead and Associate Professor at Stanford University, Andrew Ng, shows various applications of sequence classification. Some of the notable applications from the image are DNA sequence analysis which may be used to find whether a person 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 how a certain movie or product was reviewed by a viewer.

2. LITERATURE SURVEY

Generally, text classification has two types: "topic-based classification and genre-based classification"[6]. If we classify a certain text based on a certain topic, then it is called topic-based classification. "Texts can also be written in many genres, such as opinion polls, news reports, tweets, product reviews, articles, papers and advertisements"[6]. Such classification is called genre classification. "Previous work on genre classification recognized that this task differs from topic-based categorization"[7].

An interesting architecture for text classification was the C-LSTM (Chunting Zhou). The CNN and LSTM were combined to form this architecture. "To benefit from the advantages of both CNN and RNN, a simple end-to-end, unified architecture by feeding the output of a layer of CNN into LSTM"[8]. Evaluations showed that C-LSTM achieved much better results compared to a wide range of other models.

A more sophisticated architecture for document classification involved data pre-processing, stop-word removal, tokenization, lemmatization, and then the vectorization followed by encoding[9]. Firstly, the pre-processing stage would do tokenization, stopping, and stemming. Next, the vital feature selection process which selects a subset of features to best represent the original data. Saving a lot of memory and obtaining efficient and accurate classification are some of the perks of this process[9]. Finally, we have the LSTM to perform the classification.

Two models introduced by Adithya Rao and Nemanja Spasojevic were applied on two different datasets and were also notable contributions[10]. One of them was used to classify messages into one of the class labels: *actionable* or *non-actionable* to help company agents providing customer support. The other was to classify a set of social media messages into *Democrat* or *Republican*[10]. The process involved Word Embedding first. Next, a layer with multiple LSTM cells was introduced. The Dropout layer was next used to randomly drop a part of the units to avoid over-fitting. Finally, a Fully-Connected layer to learn from previous data and a Loss Layer to record the loss of data in the Dropout layer.

Khushbu Khamar discussed various models to classify short text such as Twitter messages, blogs, chat massages, book and movie summaries, forum, news feeds, and customer reviews[11]. Hiroshi Shimodaira discussed two variations of the Naïve Bayes algorithm to classify text which are Bernoulli and Multinomial Document models. "We shall look at two probabilistic models of documents, both of which represent documents as a bag of words, using the Naive Bayes assumption"[12].

As most of the Machine Learning classifiers needed work on labelled data, a lot of time is spent on manually labelling each sample which can be tiring and not necessarily error-free. Mohamed Goudjil along with three other researchers proposed and active learning strategy using SVM for text classification. "The main objective of active learning is to reduce the labeling effort, without compromising the accuracy of classification, by intelligently selecting which samples should be labelled"[13].

Penghua Li and three other researchers proposed a semi-supervised approach to text classification using CNN. Un-labelled data and a small part of labelled training data are integrated in the dataset. "For effective use of word order for text categorization, we use the feature of not low-dimensional word vectors but high-dimensional text data, that is, a small text regions is learned based on sequences of one-hot vectors"[14].

An interesting approach to text-classification is the Semantics Aware Random Forest (SARF). "SARF extracts the features used by trees to generate the predictions and selects a subset of the predictions for which the features are relevant to the predicted classes"[15].

A. P. Marathe and A. J. Agarwal proposed a two-stage cascaded CNN to classify text. One CNN is trained on spam samples and the other is trained on non-spam samples. The output of each CNN will be sent to a decision-making unit which predicts the Class Label[16].

3. PROPOSED METHODOLOGY

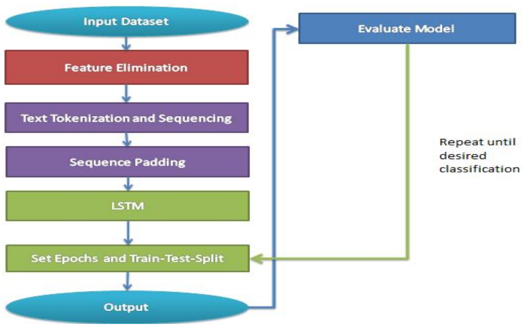
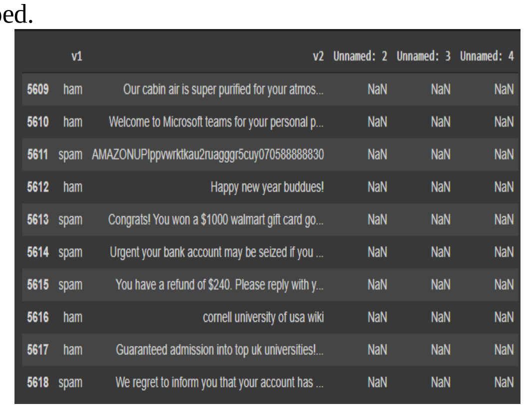


Figure 2: Proposed Architecture

Each step in the architecture illustrated in Figure 2 is explained in brief as follows.

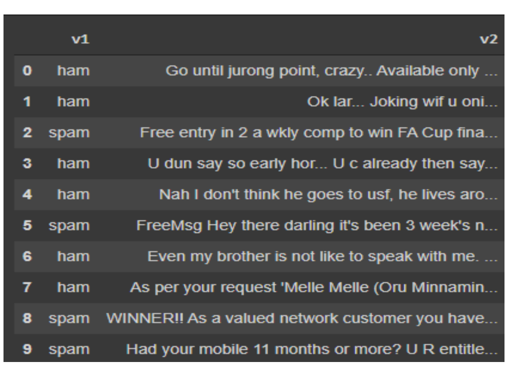
Feature Elimination: The first step is to eliminate features that are of least significance towards the classification process. The dataset consisted of 5 features initially which are v1-v5. Out of these, v3-v5 are useless features and will thus be dropped.



	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
5609	ham	Our cabin air is super purified for your atmos...	NaN	NaN	NaN
5610	ham	Welcome to Microsoft teams for your personal p...	NaN	NaN	NaN
5611	spam	AMAZONUPIppvwrktkau2ruagggr5cuy070588888830	NaN	NaN	NaN
5612	ham	Happy new year buddues!	NaN	NaN	NaN
5613	spam	Congrats! You won a $1000 walmart gift card go...	NaN	NaN	NaN
5614	spam	Urgent your bank account may be seized if you ...	NaN	NaN	NaN
5615	spam	You have a refund of $240. Please reply with y...	NaN	NaN	NaN
5616	ham	cornell university of usa wiki	NaN	NaN	NaN
5617	ham	Guaranteed admission into top uk universities!...	NaN	NaN	NaN
5618	spam	We regret to inform you that your account has ...	NaN	NaN	NaN

Figure 3: Dataset before feature elimination,

Figure 3 shows a preview of how the dataset looks like before initially. After eliminating features v3-v5, the dataset would like the one shown below in Figure 4.



	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...

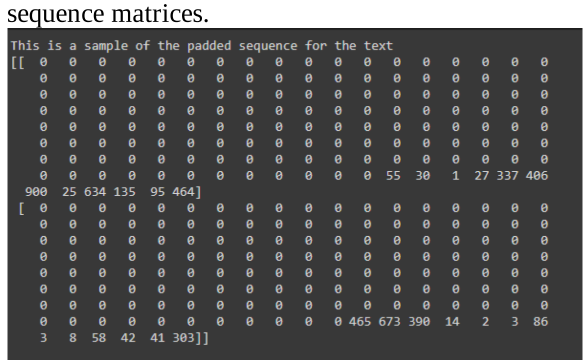
Figure 4: Dataset after feature elimination

Table 1: Dataset Summary

Samples	Train-size	Test-size	Split
5,722	4005	1,717	70:30

Text Tokenization and Sequencing: We now need to transform the text in each entry before feeding it to the LSTM model. The first of the transformation is tokenizing words. Considering the separators as special characters or spaces and splitting words accordingly, this process is achieved. Such characters are removed and individual words are tokenized in maximum word counts of 1000. Afterward, the text sequences are formed.

Padding: After forming the text sequences these are to be converted into matrices called text sequence matrices of max-sizes of 150. Other than the text sequences, entries in the matrix are filled with zeros. Figure 5 gives a visualization of the sequence matrices.



```
This is a sample of the padded sequence for the text
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0  55  30   1  27 337 406
  900  25 634 135  95 464]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0 465 673 390  14   2   3  86
    3   8  58  42  41 303]]
```

Figure 5: Example of Sequence Matrix

LSTM: The LSTM makes use of 2 activation functions which are “relu" and "sigmoid". Since there are only 2 classes, the loss function is chosen as binary-cross-entropy. The optimizer is chosen as RMSProp. The text also undergoes embedding with input dimension as 1000, the output dimension as 50, and the input sequence
length as 150. Then, the model is passed for training.

Set Epochs: Before we jump to conclusions, we need to test whether the model is over-fitting or under-fitting the dataset. For this reason, we vary the number of epochs until the desired classification result is achieved.

Evaluate the model: We make use of metrics from the confusion matrix. Furthermore, we also make use of similarity measures cosine similarity and Jaccard similarity. If we see the values like 0% or 100%, that means the model may have underperformed or over-performed on the dataset respectively.

4. EXPERIMENTAL RESULTS

For the academic study, a dataset available on Kaggle, an online repository that has a bulk of datasets, was used. The SMS Spam Collection Dataset contains a total of 5,722 entries out of which 4,949 were, non-spam, and the remaining

773 are spam text[17]. Out of these, 150 entries were additionally added to the dataset from different online sources. Table 1 gives a summary of the dataset and Figure 6 gives the counterplot:

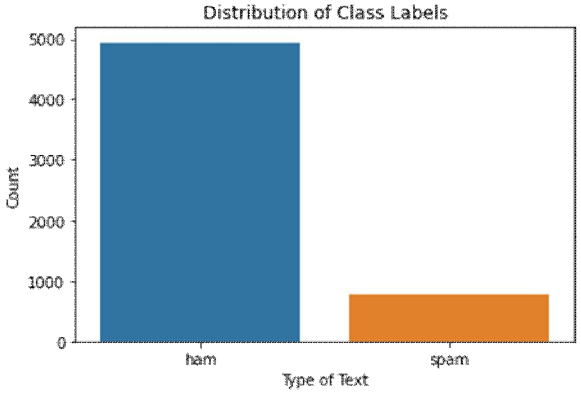


Figure 6: Counterplot for the Dataset

The LSTM was implemented considering the train-test-split as 70:30. The metrics from the confusion matrix such as Accuracy, Precision, and F1-Score were tabulated in Table 2. Additionally, the similarity measures Jaccard Similarity and Cosine Similarity were also tabulated and are considered much more accurate measures. The same metrics for other models considered in this study were also tabulated in Table 3.

Table 2: LSTM: Results in different epochs

Epochs	Accuracy	Precision	Recall	F1-score
12	0.98	0.96	0.92	0.94
11	0.98	0.97	0.89	0.93
10	0.98	0.97	0.89	0.93
9	0.98	0.95	0.93	0.94
8	0.98	0.98	0.89	0.93
7	0.98	0.97	0.91	0.94
6	0.93	1.00	0.48	0.65
5	0.95	0.99	0.69	0.81

Jaccard index	Cosine similarity
0.88	0.94
0.86	0.93
0.87	0.93
0.88	0.94
0.87	0.93
0.88	0.94
0.48	0.69
0.68	0.82

Table 3: Other Algorithms

Algorithm	Accuracy	Precision	Recall	F1-score
Random Forest	0.93	0.92	0.53	0.67
Support Vector Machine	0.89	0.79	0.31	0.44
Naïve Bayes	0.16	0.13	1.00	0.24
KNN	0.88	0.58	0.33	0.42
Multi-Layer Perceptron	0.87	0.49	0.44	0.46

Jaccard index	Cosine similarity
0.51	0.70
0.28	0.49
0.13	0.37
0.26	0.44
0.30	0.46

We start at epochs=5. At epochs=5, we see low recall and cosine similarity, and thus the F1-score. In that case, we need to keep increasing the epochs to make sure the LSTM learns a bit more. As shown above, the highest F1-Score and Cosine Similarity is obtained when epochs=9. Overtraining the model any further may not make a difference. Hence, we stop the training.

As for the other algorithms used for the study, Random Forest appeared to have performed well with decent Accuracy and Cosine Similarity but is still not satisfactory. Although the number of trees (n_estimators) were increased up to 500, the results did not vary by much. The other algorithms were also ruled out as they showed poor metric scores.

ACKNOWLEDGMENT

The author thanks his colleague Manoj Kumar M, without whom the paper would not have been formatted and furnished to the highest quality. He would also like to thank Mrs. Jyostna Devi B, Assistant Professor at Vignan's Foundation for Science, Technology, and Research for providing valuable assistance and guidance throughout the project, wherever necessary.

REFERENCES

[1] S. Lai, L. Xu, K. Liu, J. Z.-T. A. conference on artificial, and undefined 2015, "Recurrent convolutional neural networks for text classification,"

aaai.org, Accessed: Aug. 25, 2021. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/viewPaper/9745.

[2] M. Aghaei, M. Dimiccoli, C. Ferrer, P. R.-C. V. and Image, and undefined 2018, "Towards social pattern characterization in egocentric photo-streams," *Elsevier*, 2018, Accessed: Aug. 25, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314218300675.

[3] Y. Bengio *et al.*, "A Neural Probabilistic Language Model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[4] L. Zhang, S. Wang, B. L.-W. I. R. Data, and undefined 2018, "Deep learning for sentiment analysis: A survey," *Wiley Online Libr.*, vol. 8, no. 4, Jul. 2018, doi: 10.1002/widm.1253.

[5] "rnn-applications.png (1848×1032)." https://cbare.github.io/images/rnn-applications.png (accessed Aug. 25, 2021).

[6] M. Ikonomakis, S. Kotsiantis, V. T.-W. transactions on, and undefined 2005, "Text classification using machine learning techniques.," *Citeseer*, Accessed: Aug. 25, 2021. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.9153&rep=rep1&type=pdf.

[7] B. Kessler, G. Nunberg, and H. Schuetze, "Automatic Detection of Text Genre," pp. 32–38, Jul. 1997, Accessed: Aug. 25, 2021. [Online]. Available: https://arxiv.org/abs/cmp-lg/9707002v1.

[8] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM Neural Network for Text Classification," 2015, [Online]. Available: http://arxiv.org/abs/1511.08630.

[9] M. Ranjan, Y. Ghorpade, … G. K.-J. of D. M., and undefined 2017, "Document classification using lstm neural network," *core.ac.uk*, Accessed: Aug. 25, 2021. [Online]. Available: https://core.ac.uk/download/pdf/230492600.pdf.

[10] A. Rao and N. Spasojevic, "Actionable and Political Text Classification using Word Embeddings and LSTM," 2016, [Online]. Available: http://arxiv.org/abs/1607.02501.

[11] K. Khamar, "Short Text Classification Using kNN Based on Distance Function," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 2, no. 4, pp. 1916–1919, 2013, [Online]. Available: http://ijarcce.com/upload/2013/april/58-Khushbu Khamar-Short Text Classification USING.pdf.

[12] H. Shimodaira, "Note 7 Informatics 2B-Learning Text Classification using Naive Bayes."

[13] M. Goudjil, M. Koudil, M. Bedda, and N. Ghoggali, "A Novel Active Learning Method Using SVM for Text Classification," *Int. J. Autom. Comput.*, vol. 15, no. 3, pp. 290–298, 2018, doi: 10.1007/s11633-015-0912-z.

[14] P. Li, F. Zhao, Y. Li, and Z. Zhu, "Law text classification using semi-supervised convolutional neural networks," *Proc. 30th Chinese Control Decis. Conf. CCDC 2018*, pp. 309–313, Jul. 2018, doi: 10.1109/CCDC.2018.8407150.

[15] M. Zahidul Islam *et al.*, "A Semantics Aware Random Forest for Text Classification," 2019, doi: 10.1145/3357384.3357891.

[16] A. P. Marathe and A. J. Agrawal, "Improving the accuracy of spam message filtering using hybrid CNN classification," *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 5, pp. 2194–2198, 2020, doi: 10.30534/ijeter/2020/116852020.

[17] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of SMS spam filtering: New collection and results," *DocEng 2011 - Proc. 2011 ACM Symp. Doc. Eng.*, pp. 259–262, 2011, doi: 10.1145/2034691.2034742.