International Journal of Hybrid Information Technology
Vol.13, No.1 (2020), pp.19-32
http://dx.doi.org/10.21742/ijhit.2020.13.1.03

Improved Text Classification using Long Short-Term Memory and Word Embedding Technique

Amol C. Adamuthe

Assistant Professor, Rajarambapu Institute of Technology, Rajaramnagar, MS, India
amol.admuthe@gmail.com

Abstract

Text classification is an important problem for spam filtering, sentiment analysis, news filtering, document organizations, document retrieval and many more. The complexity of text classification increases with some classes and training samples. The main objective of this paper is to improve the accuracy of text classification with long short-term memory with word embedding. Experiments were conducted on seven benchmark datasets namely IMDB, Amazon review full score, Amazon review polarity, Yelp review polarity, AG news topic classification, Yahoo! Answers topic classification, DBpedia ontology classification with the different number of classes and training samples. Different experiments are conducted to evaluate the effect of each parameter on LSTM. Results show that 100 batch size, 50 epochs, Adagrad optimizer, 5 hidden nodes, 100-word vector length, 2 LSTM layers, 0.001 L2 regularizations, 0.001 learning rate give the higher accuracy. The results of LSTM are compared with the literature. For IMDB, Amazon review full score, Yahoo! Answers topic classification dataset the results obtained are better than literature. Results of LSTM for Amazon review polarity, Yelp review polarity, AG news topic classification are close to best-known results. For the DBpedia ontology classification dataset the accuracy is more than 91% but less than best known.

Keywords: *Text classification, Long short-term memory, Deep learning, Word2Vec, Word embedding*

1. Introduction

Text classification is to classify the text into the appropriate categories based on the textual content. Due to the large and growing amount of text data, automatic text classification methods are receiving more and more attention from the research community. Although many efforts have been made in this regard, it remains an open question [1]. The World Wide Web needs efficient and effective classification algorithms to help people navigate and browse online documents quickly [2]. Text classification is used in a various area like email classification and spam filtering [3], sentiment analysis [4], opinion and topic detection [5], author identification [6] and language identification [7], news filtering and organizations [8], document organization and retrieval [9], etc.

Various techniques are designed for text classification. Some key methods, usually used for text classification such as support vector machine (SVM) [10], decision trees [3], pattern (Rule)-based, neural network (NN) [11], bayesian (Generative) [12], k. nearest neighbour

Article history:
Received (July 15, 2019), Review Result (December 3, 2019), Accepted (January 24, 2020)

Print ISSN: 1738-9968, eISSN: 2652-2233 IJHIT

(KNN) [13] etc. Compared with other supervised machine learning algorithms, the SVM classifier is one of the most effective text classification methods.

Neural networks are machine learning (ML) models inspired by the human brain. It includes many neurons that form a huge network. Neural networks have a flexible architecture with a distinct number of nodes per layer with a different number of weights and hidden layers. Multiple hidden layers of the neural network are called deep learning [11]. Many types of deep learning can be used for classification such as recurrent neural networks (RNN), convolutional neural networks (CNN), multilayer Perceptron, etc. The advantage of RNN is that the previous state is used to calculate the current state. However, simple RNNs have problems in delivering information in long sequences. LSTM is the solution to this problem. RNN with extra long-term memory was proposed in 1977. LSTM is one of the most successful and developed for controlling robots, natural language text compression, automatic speech recognition, time series prediction, handwriting recognition, document classification and many more. They can equally prove good in the process of document classification [14].

In literature, different text classifiers are investigated for text classification such as Naïve Bayes, SVM, logistic regression, stochastic gradient descent, NN, SVM and hybrid models of these. The main objective of this paper is to improve text classification using long short-term memory. Paper presents results of LSTM with Word2Vec for text classification on seven benchmark datasets. Investigations are conducted on the effect of different parameters of LSTM.

The rest of this paper is organized as follows. Section 2, is about a brief review and some related works. In Section 3, we discuss the proposed methodology. Experimental details, results and discussion are demonstrated in Section 4. Finally, in Section 5 we present our conclusions.

2. Literature Review

This section covers a review of different research carried out for text classification using different machine learning (ML) based approaches and LSTM.

Classification algorithm KNN is given in [13]. The KNN is a frequently used text classification technique. This method works well even when using multi-category documents to process classification tasks. The limitation of KNN is that it needs more time to classify objects when given many training examples. RA has high computational efficiency, fast learning speed [15].

The NB classifier is based on the Bayesian theorem with strong independence assumptions. The algorithm calculates the posterior probabilities that the documents belong to different classes and assign the document to the class with the highest a posteriori probability [12]. It handles numerical and textual data extremely well. The disadvantage of the NB classification method is that the classification performance is relatively low compared with other ML algorithms [3].

The DT text classifier built using a “divide and conquer” strategy. The DT checks if all training examples have the identical label, and if not, selects the term partition from the merged class document with the duplicate term value and places each such item in a separate subtree [16].

SVM is a supervised classification algorithm that deals with a lot of functionalities. SVM is one of the most effective text classification methods as a comparison to other ML algorithms [3]. SVM was originally applied to Joachim's text classification [17]. Joachim verifies the classification performance of SVM in text categorization by comparing it with

SVM and KNN. Drucker uses the SVM to implement a spam filtering system and compares it to NB to implement the system. They show that SVM is a better spam filtering method than NB. Since the analysis, SVM has more parameters than the logistic regression and DT classifiers, the SVM has the highest classification accuracy most of the time, whereas the SVM requires more computation time due to more parameters and is very time-consuming. Logistic regression is computationally efficient compared to SVM.

On comparing DT and NNs, their strengths and weaknesses are almost complementary. For example, people can easily understand the representation of DT, which is not the case of NNs. Decision trees encounter difficulties in handling noise in training data, as well as NNs, DTs learn very quickly. NNs learn relatively slowly. DT learning is used for qualitative analysis, and NN is used for subsequent quantitative analysis.

An NN initialized with a DT is a hybrid approach that can be applied to text classification problems and tested for performance relative to many other text classification algorithms. The method shows that the hybrid decision tree and neural network method improve the accuracy of the text classification task, and the performance of the random text classifier is equivalent to the previous result than the single DT or NN.

The probabilistic neural network is a combination of SVM, KNN, and slightly modified versions of DTs and proposed to better handle multi-label classification problems [18]. BFC is a hybrid method of NB vectorizer. Compared with the simple Bayesian classification method, the SVM classifier improves the classification accuracy. In [19], a hybrid algorithm is proposed, which is based on the variable precision rough set, combined with the strength of KNN and RA techniques to improve the accuracy of text classification and overcome the drawbacks of RA.

Long Short-Term Memory units, also called LSTM, are a variation of Recurrent Neural Networks that are capable of learning long-term dependencies. They were proposed by the German researchers Hochreiter and Schmidhuber as a solution to the error backflow problems [20]. The challenging task of sentiment analysis is a need of required labeled dataset and to solve this issue Qurat Tul Ain et al. [21] combined deep learning technique and sentiment analysis. Deep learning techniques gave an effective performance. Peerapon Vateekul et al. [22] proposed those deep learning techniques for sentiment analysis of Twitter data. LSTM and Dynamic CNN performed well than traditional methods like naïve Bayes and support vector machine. In deep learning techniques, they used word2vec and in the traditional approach bag of the word, the approach was used which occurs difficulty in the training process. LSTM and DCNN gave better accuracy. Dan Li et al. [23] trained the emotional model to find out which sentence belongs to which emotion model using LSTM which was used for better analyzing the long sentences. 10 Unfold layer in backpropagation gave better accuracy rate and recall rate for LSTM than RNN [24]. After analysis of the performance of three RNN methods which are vanilla RNN, LSTM and GRU, one layer with GRU gave better accuracy [25]. Abdalraouf Hassan et al. used a continuous bag of word approach which gave a better performance than a traditional bag of word approach with a single LSTM layer performed well [26]. Extended versions of RNN which was LSTM and Gated Recurrent Unit was proposed by Yong Zhang et al. LSTM and GRU methods achieve better performance compared with RNN [4]. Piotr Semberecki et al. proposed LSTM for classifying English Wikipedia articles. Encoding of a word using a dictionary and Google news pre-trained word vector used for word embedding. Pre-trained word vector achieved better performance [27]. Different ML techniques are used for document classification. Yash R. Ghorpade et al. [14] include text pre-processing, FS, feature extraction and class prediction. LSTM achieves 93% accuracy at 25 epochs.

[Table 1] summarized the contributions for text classification using LSTM with the dataset, parameter details and results.

Table 1. Short summary of LSTM for text classification

Reference	Technique	Dataset	Word Embedding Technique	Parameters studied / used	Results
[22]	LSTM, DCNN	Thai Twitter data	Skip-gram model	50-word vector length, softmax function, stochastic gradient descent with cross-entropy loss function, LSTM layer, Hidden nodes, Filter width, filters, word vector length.	LSTM-96-word vector gave 75.12%, 5 hidden nodes gave 75.30% highest accuracy. DCNN- 48-word vector for DCNN 75.35%, 3-6 and 6-14 filters gave 75.30% highest accuracy.
[23]	LSTM, RNN	JD.COM, Travel comment ctrip and English movie review	Not given	Unfold layer, backpropagation	LSTM produce better accuracy rate and recall rate than RNN.
[24]	Paragraph vector, Deep RNN, and LSTM	IMDB, SST	Skip-gram, Continuous bag of word approach	RMSProp, learning rate, epochs, vocabulary downsampling, LSTM layers, hidden size	IMDB dataset - 150-word vector, 25 epochs, 0.05 learning rate and vocabulary downsampling for CBOW is 1e-4 and for skip-gram 1e-2. paragraph vector got 0.945 higher accuracies than LSTM. SST dataset - 3 LSTM layers, 100 hidden sizes, 0.5 dropouts, 1 learning rate, 0.99 decay, and epsilon of 1e-8 LSTM got 0.843 accuracy which is better than paragraph vector.
[25]	Vanilla RNN, LSTM, GRU	Amazon health product reviews, SST-1, SST-2	Google news pre-trained word vector	RMSProp, word vector, hidden layer, dense layer node, learning rate, mini-batch gradient descent, sigmoid, softmax.	One layer with GRU gave 83.90% accuracy on Amazon health product reviews, 44.61% accuracy on SST-1 and 84.40% accuracy on SST-2
[26]	LSTM	IMDB, SST	Continuous bag of word approach	Stochastic gradient descent, mini batches, rectifier linear units, hidden size, drop out, batches, dropout, LSTM layers.	Single layer LSTM with, 128 hidden sizes, 64 batch size, and 0.5 dropouts performed well.
[4]	RNN, LSTM, GRU	IMDB, Standford sentiment treebank (SST)	Bag of the word, Skip-gram model	Softmax, Stochastic gradient descent, RMSProp, word vector, hidden nodes, learning rate, decay factor, a fuzzy factor of RMSProp.	300-word vector size, 100 hidden nodes, 0.001 learning rate, 0.9 decay factor, and 1e-6 fuzzy factor gave high performance. CA-LSTM and CA-GRU achieve similar and better performance with 90.01 accuracy for CA-LSTM and CA-GRU which is better than RNN recall 89.32.
[27]	LSTM	English Wikipedia article	Encoding of a word using a dictionary, Google news pre-trained word vector	LSTM cells, epochs, Batch size, Adam, learning rate, sigmoid, for binary classification uses binary cross entropy and for multiclass uses sparse categorical cross entropy, truncate length 500 and 1000	For the encoding of a word using dictionary gave 91.52%, 76.53% and 58.93% accuracy for 2, 3 and 7 classes respectively. For the pre-trained vector gave 92.25% and 86.21% accuracy for 2 and 7 classes respectively.
[14]	LSTM	20 Newsgroup	TF-IDF	Epochs	LSTM achieve 93% accuracy at 25 epochs.

3. Methodology

This section presents algorithmic and implementation details of LSTM and word embedding technique (Word2Vec) for text classification.

3.1. Word embedding technique - Word2Vec

LSTM needs the input in numeric format, not text. Different methods are used to convert text to numeric format such as a bag of the word, term frequency-inverse document term frequency, Word2Vec, etc. The bag of word and TFIDF method for obtaining such vectors is based on very simple lexical coding. It works for binary classifications, but as the number of classes (subject categories) increases, its accuracy decreased. Paper [28] suggested two methods for vector representations, CBOW (continuous bag of the word) and the skip-gram. The Word2Vec method is more stable and flexible. Word2Vec is a method of representing words in multidimensional vector space [28]. Word2Vec gave a better performance on different neural network techniques. Word2Vec is a shallow NN that is used to process text and create the vector of words from the vocabulary.

This paper used a skip-gram model for the input of LSTM. It converts the text into a numeric form that the deep network can understand. The Skip-gram model finds the target corpus y from the given word. There are input, output, and hidden layers. The activation function used for the output layer is SoftMax. For configuration of Word2Vec used different parameters that are:

100 batch size - It is the number of words that you process at a time.

20 minimum word frequency - It is the number of word frequencies in the corpus. In the case of a large corpus need to increase the value of word frequency.

100 layer size - It tells the input vector size.

Learning Rate - It is the step size for each update of the coefficient.

Tokenizer - It is used to tokenize the sentences.

3.2. Long Short-Term memory for text classification

LSTM contains one memory cell and three gates, which are the forgotten gate, input gate and output gate and the size of the input vector defined by the word embedding size. Input gate is used to control the flow of input, output gate is used to control the flow of output and forget gate decides the what information going to store in the memory cell and what information going to throw away from the cell. [Figure 1] shows the pseudo-code of LSTM for text classification.

Farhoodi and Yari [29] have listed the performance measures for text classification. Evaluation of the LSTM classifier is tested using two measures namely accuracy and loss. Accuracy is calculated using equation (1).

$$Accuracy = \frac{2*(recall*precision)}{recall+precision} \quad (1)$$

The smaller the MSE value, the higher the accuracy and vice versa [30]. Loss is calculated using the mean square error shown in equation (2) [31].

$$Mean\ square\ error = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2 \quad (2)$$

4. Experimental details, results and discussion

This section presents details of the dataset used, parameters on which LSTM experimentation is carried out, results and discussion. LSTM performance is evaluated with six experiments conducted on batch size, epochs, optimizers, hidden nodes, word vector size, LSTM layers, L2 regularization, and learning rate.

Algorithm: LSTM for text classification.

Input: Word vector size/input size of LSTM network, batch size, number of epochs, train and test set of datasets

Output: Accuracy, class label

Step 1: Load the dataset prepared for word to vector conversion

Step 2: Pre-processing steps - whitespace striping and tokenization

Step 3: Parameters Initialization in Word2Vec – minimum word frequency, number of layers, seed, window size, number of iterations

Step 4: Configuration of LSTM network –number of LSTM layers, number of hidden nodes, optimizer, activation function, loss function, learning rate, L2 regularization

Step 5: Execution of LSTM

    For each epoch

        For each batch fit the LSTM network

            training based on word vector, batch size, truncate length

            save the trained module

        End

    End

Step 6: Testing using the LSTM network

Step 7: Calculate the probability of every class for unseen data for validation

Figure 1. The pseudo-code of LSTM for text classification

All results presented in this section are on the testing dataset. The results of seven datasets are compared with the literature. The experimental datasets are collected from [32]. The details of the dataset to the number of classes, train sets and test sets are shown in [Table 2].

Table 2. Datasets used for text classification

	Name of Dataset	Number of Classes	Number of Train Sets	Number of Test Sets
Dataset 1	IMDB	2	25,000	25,000
Dataset 2	Amazon review full score	2	30,00,000	6,50,000
Dataset 3	Amazon review polarity	2	3,000,000	650,000
Dataset 4	Yelp review polarity	2	560,000	38,000
Dataset 5	AG news topic classification	4	1,20,000	7,600
Dataset 6	Yahoo! Answers topic classification	10	1,400,000	60,000

Dataset 7	DBpedia ontology classification	14	1,800,000	200,000

The deeplearning4j library is used for the implementation of the methodology. It is a Java-based toolkit for building, training and deploying deep neural networks, regressions and KNN [33].

4.1. Experiment 1: Identifying suitable batch size, epochs and optimizer

Experiments are carried out using different combinations of batch size and number of epochs. Results in [Table 3] show that 100 batch size at 50 epochs gives the best accuracy for all seven datasets. The accuracy range is from 91% to 95.3%.

Table 3. Accuracy (in %) on various batch size and epochs

Observation	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Batch Size	10	20	40	60	80	100	10	20	40	60	80	100	10	20	40	60	80	100
No. of epochs	10						50						100					
Dataset 1	74.56	76.2	76.43	75.33	78	79.89	83.55	84.1	86.45	89.76	91.32	95.3	93.4	92.2	92.15	89	90.4	87.89
Dataset 2	74	76.6	77.2	78.34	79.1	79.89	83.55	84.1	86.45	89.7	92.5	94.3	92.2	90.33	89.6	89.76	88	87.3
Dataset 3	77.2	78	79.2	79.56	83.2	86.1	88.16	88.34	89	91.54	93.9	94.26	94	92.4	91	90.33	90	89.23
Dataset 4	77.3	77.87	78.0	78.65	80.98	83.4	85.9	86.4	89.6	92	92.4	94.37	93.5	91.65	91.1	90.76	90	89.76
Dataset 5	77.34	77.89	79.43	81.6	82	83.87	85.1	86	86.76	87.8	89.9	91.43	90.2	90.16	89.3	88.4	88	87.23
Dataset 6	76.3	77	77.43	78.23	79.2	80	81.33	83.8	86	88.76	89.54	92.5	90.5	90.23	89.5	87.45	87	86.65
Dataset 7	73.34	74	76.66	78.97	79.1	82.54	82.9	85.4	86.8	88.2	89.36	91.67	88.89	87.45	87	86.34	87	85.34

We have tested the performance of seven optimizers namely Adagrad, Adadelta, SGD, Adam, RMSprop, Adamax, Nadam. Figure 2 shows that the performance of LSTM differs with optimizers. For all seven datasets Adagrad, Adadelta and SGD are the top three performers respectively. Results show that loss decreases with increasing epochs.

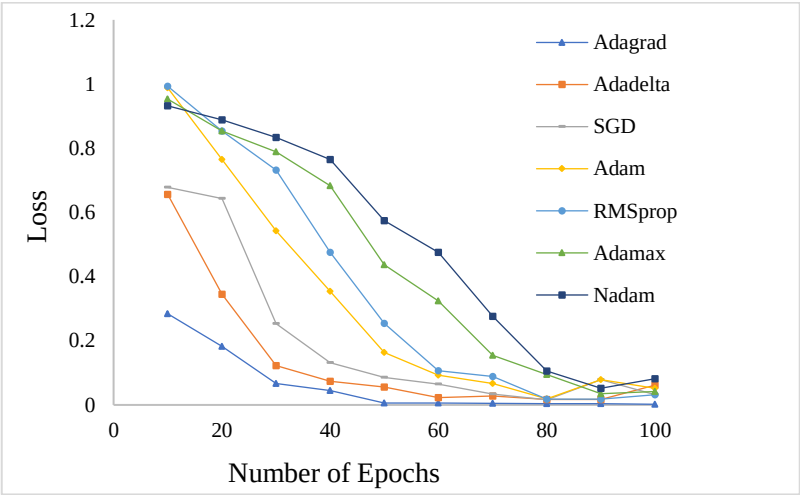

Figure 2. Performance of different optimizers for dataset 6

4.2. Experiment 2: Identifying the suitable number of hidden nodes

The goal of this experiment is to find the optimal output size of the LSTM, which can be thought of as the size of hidden nodes in the network. After experimentation, found that the best accuracy comes from using 5 hidden nodes as shown in [Table 4].

Table 4. Results on a varying number of hidden node in LSTM

	Number of Hidden Nodes			
	5	10	20	50
Dataset 1	95.30	95.20	95.07	95.03
Dataset 2	94.30	93.76	93.12	92.76
Dataset 3	94.26	94.02	93.52	91.87
Dataset 4	94.37	93.65	93.03	92.43
Dataset 5	91.43	91.35	91.00	89.64
Dataset 6	92.50	91.83	91.74	90.28
Dataset 7	91.67	90.23	90.16	89.42

4.3. Experiment 3: Impact of word vector length on accuracy of LSTM

The LSTM used in this experiment receives a word vector sequence and creates an output of length 100 which gives a better result which is shown in [Table 5]. With this word vector, accuracy increases as per vector length increase at the beginning but do not increase after a certain period. All the remaining experiments are conducted with best-fitted vector length.

Table 5. Results on varying word vector length

	Word Vector Length				
	10	20	50	100	300
Dataset 1	93.79	94.79	95.05	95.38	94.88
Dataset 2	92.47	93.48	94.04	94.63	93.45
Dataset 3	92.84	93.75	94.01	94.32	93.63
Dataset 4	91.63	92.67	93.73	94.57	93.52
Dataset 5	89.45	90.54	91.02	91.56	90.73
Dataset 6	89.84	90.26	91.28	92.81	91.36
Dataset 7	88.57	89.91	90.84	91.72	90.65

4.4. Experiment 4: Impact of layers on accuracy of LSTM

The performance of algorithms is always associated with the architecture. [Table 6] shows the results of LSTM with different layers. LSTM with two layers shows the best accuracy for all datasets. Performance is not directly proportional to the number of LSTM layers.

Table 6. Results on a varying number of layers in LSTM

	Number of LSTM Layer				
	1	2	3	4	5
Dataset 1	94.06	95.46	94.38	94.02	93.47
Dataset 2	93.65	94.73	93.72	93.51	92.91
Dataset 3	93.88	94.56	93.68	93.36	92.73

Dataset 4	93.88	94.63	93.71	92.49	91.68
Dataset 5	90.26	91.70	90.28	90.11	89.74
Dataset 6	91.74	92.91	91.58	90.92	89.83
Dataset 7	90.47	91.88	90.82	90.53	89.61

4.5. Experiment 5: Impact of L2 regularization on accuracy of LSTM

L2 regularization is used to avoid overfitting and minimize error. [Table 7] shows that 0.001 value for L2 regularization given the best performance.

Table 7. Results on varying L2 regularization

	L2 Regularization				
	0.001	0.01	0.5	0.6	0.7
Dataset 1	95.62	94.72	94.28	93.82	92.44
Dataset 2	94.82	93.77	92.56	92.35	91.83
Dataset 3	94.63	93.53	93.37	92.64	91.30
Dataset 4	94.73	93.82	93.63	92.72	92.01
Dataset 5	91.74	90.28	90.04	89.74	89.37
Dataset 6	92.97	91.38	91.23	90.38	89.93
Dataset 7	91.92	90.64	90.39	89.83	89.58

4.6. Experiment 6: Impact of learning rate on performance of LSTM

[Table 8] shows that the 0.001 learning rate gives the best performance. If the learning rate is low, then training is more reliable and gives better accuracy, but optimization will take a lot of time because steps towards the minimum of the loss function are tiny.

Table 8. Results on varying learning rate

	Learning Rate				
	0.001	0.01	0.1	0.2	0.3
Dataset 1	95.78	95.47	94.62	93.68	92.17
Dataset 2	94.93	93.26	92.54	92.37	91.76
Dataset 3	94.87	93.63	93.46	92.77	91.79
Dataset 4	94.88	94.27	93.68	93.35	92.78
Dataset 5	91.79	90.67	90.38	89.88	89.62
Dataset 6	93.04	92.69	92.38	91.83	91.63
Dataset 7	91.98	90.83	90.52	89.63	88.89

[Table 9] shows the best results obtained with suitable LSTM parameters.

Table 9. Best results obtained

	Dataset						
	1	2	3	4	5	6	7
100 batch size and 50 epochs	95.3	94.3	94.26	94.37	91.43	92.5	91.67
Adagrad	0.002	0.002	0.002	0.002	0.004	0.002	0.002
5 Hidden Nodes	95.30	94.30	94.26	94.37	91.43	92.50	91.67

100 Word Vector Length	95.38	94.63	94.32	94.57	91.56	92.81	91.72
2 LSTM Layers	95.46	94.73	94.56	94.63	91.70	92.91	91.88
0.001 L2 Regularization	95.62	94.82	94.63	94.73	91.74	92.97	91.92
0.001 Learning Rate	95.78	94.93	94.87	94.88	91.79	93.04	91.98

[Table 10] shows the result comparison for the IMDB dataset with 12 papers from the literature. All these papers experimented with machine learning algorithms. The proposed methodology provides better results with 95.78% accuracy.

Table 10. Results comparison for dataset 1 (IMDB dataset)

Reference	Technique	Accuracy in %
[10]	Support Vector Machine - Unigram	86.4
[34]	Recurrent Neural Tensor Network	87.6
[35]	Bag of Words	87.8
	Full + Bag of Words	88.33
	Full + Unlabelled + Bag of Words	88.89
[4]	Comprehensive Attention-Recurrent Neural Network	89
	Comprehensive Attention - Long short-term memory	90.1
	Comprehensive Attention - Gated Recurrent Unit	90.1
[36]	Long Short-Term Memory + Cognition Based Attention + Local Text Context Based Attention ModelparallelGECO	90.1
[37]	Hybrid Residual Long short-term memory	90.92
[38]	Recurrent Convolution Neural Network-Highway	90.3
[39]	Multinomial Naïve Bayes-Unigram	83.55
	Multinomial Naïve Bayes - Bigram	86.59
	Support Vector Machine-Unigram	86.95
	Support Vector Machine with Naïve Bayes Feature - Unigram	88.29
	Support Vector Machine-Bigram	89.2
	Support Vector Machine with Naïve Bayes Feature - Bigram	91.22
[40]	One Hot Bidirectional-Long short-term memory	91.86
[41]	Paragraph Vector	92.58
[42]	Topic Recurrent Neural Network	93.72
[24]	Averaged Paragraph Vector	88.3
	Long short-term memory	89.1
	Paragraph Vector (Logistic Regression)	94.4

	Paragraph Vector (2 Layer Multilayer Perceptron)	94.5
This Paper	Proposed Methodology	95.78

[Table 11] shows a comparison of LSTM parameters in this paper and literature for IMDB dataset. This result shows the finely tuned parameter values of LSTM for IMDB dataset.

Table 11. Comparison of LSTM parameters of [4], [24], [40] and this paper for IMDB dataset

Parameters	[4]	[24]	[40]	This Paper
Word Embedding	Skip-gram	Glove Vector	One hot	Skip-gram
Word Vector Length	300	100	500	100
Number of Hidden Nodes	100	100	Not Given	5
Optimizer	RMSProp	RMSProp	SGD	Adagrad
Layers	2 LSTM and 1CNN	1 LSTM	Bi-directional LSTM	2 LSTM
Learning Rate	0.001	0.001	Not Given	0.001
Epochs	Not Given	10-17	Not Given	50
Accuracy	90.1	89.1	91.86	95.78

[Table 12] shows a comparison of results for dataset 3 to 7. To best of our knowledge results for dataset 2 (Amazon review full score with 2 classes) is not available. The accuracy obtained for dataset 2 is 94.93%. The accuracy obtained for dataset 3, 4 and 5 are 94.87%, 94.88 and 91.79% respectively, which is close to the best results in the literature. The results are better than many other techniques listed in [Table 12]. Results obtained for dataset 6 are significantly better than literature. The improvement in accuracy is more than 17%. For dataset 7, accuracy is approximately 6% less than the best mentioned in [Table 12].

Table 12. Comparison of previously published results with this paper for dataset 3, 4, 5, 6 and 7

Ref.	Method	Dataset 3	Dataset 4	Dataset 5	Dataset 6	Dataset 7
[43]	Hierarchical Network Averaging	-	-	-	75.2	-
	Hierarchical Network Max Pooling	-	-	-	75.2	-
	Hierarchical Network Attention Model	-	-	-	75.8	-
[44]	Naïve Bayes	-	86	90	68.7	-
	Kneser-Ney Bayes	-	81.8	89.3	69.3	
	Multilayer Perceptron Naïve Bayes	-	73.6	89.9	60.6	-
	Discriminative Long short-term memory	-	92.6	92.1	73.7	-
	Generative Long short-term memory Independent Component	-	90	90.7	70.5	-
	Generative Long short-term memory Shared Component	-	88.2	90.6	69.3	-
[45]	Word Context Region Embedding	95.1	-	92.8	73.7	98.9
	Context Word Region Embedding	95.3	-	92.8	73.4	98.9
[37]	Long short-term memory	-	-	91.76	-	-
	Recurrent Neural Network	-	-	91.19	-	-
	Identity Mapping Skip Connected Long short-term memory	-	-	92.05	-	-

	Parametric Skip Connection Long short-term memory	-	-	92.01	-	-
	Long short-term memory + Gated Recurrent Unit	-	-	91.05	-	-
	Hybrid Residual Long short-term memory	-	-	91.9	-	-
[46]	Very Deep Convolution Network	95.72	95.72	91.33	73.43	98.71
[47]	Long short-term memory	-	-	87.08	-	97.87
	Length Adaptive Recurrent Model	-	-	87.18	-	97.88
This Paper	Proposed Methodology	94.87	94.88	91.79	93.04	91.98

5. Conclusions

Paper presents the effect of the different parameters on the performance of LSTM and Word2Vec for text classification. Text classification accuracy obtained by proposed methodology for dataset 1, 2, 3, 4, 5 and 6 are 95.78%, 94.93%, 94.87%, 94.88%, 91.79%, 93.04% and 91.98% respectively. Six different experimentation shows that 100 batch size, 50 epochs, Adagrad optimizer, 5 hidden nodes, 100-word vector length, 2 LSTM layers, 0.001 L2 regularizations, 0.001 learning rate give better accuracy. Empirical results on IMDB, Amazon review full score, and Yahoo! Answers topic classification dataset demonstrate that the proposed architecture effectively improves the classification performance compared with previously published results on the same dataset. The accuracy of LSTM for dataset Amazon review polarity, Yelp review polarity and AG news topic classification is close to the best results in the literature. For the DBpedia ontology classification dataset, the accuracy is above 91% but 6% less than the best results in the literature. Future work: There is scope to improve the accuracy of LSTM with hybrid architecture.

References

[1] E. Tellez and D. Moctezuma, "An automated text categorization framework based on hyperparameter optimization," Knowledge-Based Systems, pp.110-123, **(2018)** DOI: 10.1016/j.knosys.2018.03.003

[2] B. Parlak and A. Uysal, "The impact of feature selection on medical document classification," in proceedings of the 11th Iberian Conference on Information Systems and Technologies (CISTI), Turkey, pp.1-5, **(2016)** DOI: 10.1109/CISTI.2016.7521524

[3] P. Pawar and S. Gawande, "A comparative study on different types of approaches to text categorization," International Journal of Machine Learning and Computing, vol.2, no.4, pp. 423-426, **(2012)**

[4] Y. Zhang, R. Venkatesan, N. Wang and M. Pratama, "Sentiment classification using comprehensive attention recurrent models," in proceedings of the International Joint Conference on Neural Networks (IJCNN), China, pp.1562-1569, **(2016)** DOI: 10.1109/IJCNN.2016.7727384

[5] R. Rahul, K. Kumar and S. Selvakumar, "Opinion and topic detection using sentiment classifier," International Journal of Engineering and Computer Science, vol.2, no.7, pp.2189-2194, **(2013)**

[6] A. Mohsen, N. El-Makky and Ghanem. Nagia, "Author identification using deep learning," in proceedings of the 15th IEEE International Conference on Machine Learning and Applications (ICMLA), Egypt, pp.898-903, **(2016)** DOI: 10.1109/ICMLA.2016.0161

[7] S. Malmasi and D. Mark, "Language identification using classifier ensembles," in proceedings of the joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects, pp.35-43, **(2015)**

[8] N. Daniel and K. Karthik, "Survey on Text Classification Methods," International Journal of Advanced Research in Computer Science and Software Engineering, vol.6, no.2, pp.585-588, **(2016)**

[9] S. Chakrabarti, B. Dom, R. Agrawal and P. Raghavan, "Using taxonomy, discriminants, and signatures for navigating in text databases," in preceding of the 23rd VLDB Conference Athens, Greece, vol.97, pp.446-455, **(1997)**

[10] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in proceedings of the 42nd annual meeting on Association for Computational Linguistics, pp.271-279, **(2004)** DOI: 10.3115/1218955.1218990
[11] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," in proceedings of the APSIPA Transactions on Signal and Information Processing, vol.3, no.2, pp.1-29, **(2014)** DOI: 10.1017/atsip.2013.9
[12] I. Rish, "An empirical study of the naive Bayes classifier," in IJCAI 2001 workshop on empirical methods in artificial intelligence, New York, vol.3, no.22, pp.41-46, **(2001)**
[13] V. Tam, A. Santoso, and R. Setiono, "A comparative study of centroid-based, neighborhood-based and statistical approaches for effective document categorization," in proceedings of the IEEE 16th International Conference on Pattern Recognition, Hong Kong, vol.4, pp.235-238, **(2002)**
[14] N. Ranjan, Y. Ghorpade, G. Kanthale, A. Ghorpade and A. Dubey, "Document classification using LSTM neural network," Journal of Data Mining and Management, vol.2, no.2, **(2017)**
[15] W. Cohen and Y. Singer, "Context-sensitive learning methods for text categorization," ACM Transactions on Information Systems (TOIS), vol.17, no.2, pp.141-173, **(1999)**
[16] A. Mashat, M. Fouad, S. Philip, and T. Gharib, "A decision tree classification model for university admission system," International Journal of Advanced Computer Science and Applications, vol.3, no.10, pp.17-21, **(2012)**
[17] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in proceedings of the European conference on machine learning, Berlin, pp.137-142, **(1998)**
[18] P. Ciarelli, E. Oliveira, C. Badue, and A. De Souza, "Multi-label text categorization using a probabilistic neural network," International Journal of Computer Information Systems and Industrial Management Applications, vol.1, pp.133-144, **(2009)**
[19] D. Miao, Q. Duan, H. Zhang, and N. Jiao, "Rough set based hybrid algorithm for text classification," Expert Systems with Applications, vol.36, no.5, pp.9168-9174, **(2009)**
[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol.9, no.8, pp.1735-1780, **(1997)**
[21] Q. Ain, M. Ali, A. Riaz, A. Noureen, M. Kamran, B. Hayat, and A. Rehman, "Sentiment analysis using deep learning techniques: a review," International Journal of Advanced Computer Science and Applications, vol.8, no.6, pp.424-433, **(2017)**
[22] P. Vateekul and T. Koomsubha, "A study of sentiment analysis using deep learning techniques on Thai Twitter data," in proceedings of the IEEE 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), Thailand, pp.1-6, **(2016)**
[23] D. Li and J. Qian, "Text sentiment analysis based on long short-term memory," in proceedings of the IEEE International Conference on Computer Communication and the Internet, China, pp.471-475, **(2016)**
[24] J. Hong and M. Fang, "Sentiment analysis with deeply learned distributed representations of variable length texts," Technical report, pp.655-665, **(2015)**
[25] K. Baktha and B. K. Tripathy, "Investigation of recurrent neural networks in the field of sentiment analysis," in proceedings of the IEEE Conference on Communication and Signal Processing, India, April, pp.2047-2050, **(2017)**
[26] A. Hassan and A. Mahmood, "Deep learning for sentence classification," in proceedings of the IEEE Conference on Systems, Applications and Technology, Long Island, USA, pp.1-5, **(2017)**
[27] P. Semberecki and H. Maciejewski, "Deep learning methods for subject text classification of articles," in proceedings of the IEEE Federated Conference on Computer Science and Information Systems, Poland, vol.11, pp.357-360, **(2017)**
[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:(1301)3781, **(2013)**

[29] M. Farhoodi and A. Yari, "Applying machine learning algorithms for automatic Persian text classification," in proceedings of the 6th International Conference on Advanced Information Management and Service, Iran, pp.318-323, **(2010)**

[30] C. Leke, B. Twala and T. Marwala, "Missing data prediction and classification: The use of auto-associative neural networks and optimization algorithms," arXiv preprint arXiv:(1403)5488, **(2014)**

[31] B. Chen, L. Xing, N. Zheng and C. Principe, "Quantized Minimum Error Entropy Criterion," arXiv preprint arXiv:(1710)04089, **(2017)**

[32] https://drive.google.com/drive/u/0/folders/0Bz8a_Dbh9Qhbfll6bVpmNUtUcFdjYmF2SEpmZUZUcVNiMUw1TWN6RDV3a0JHT3kxLVhVR2M

[33] https://deeplearning4j.org/

[34] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in proceedings of the conference on empirical methods in natural language processing, Seattle, Washington, USA, pp.1631-1642, **(2013)**

[35] Maas A., Daly R., Pham P., Huang D., Ng A., and Potts C., "Learning word vectors for sentiment analysis," in proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies, vol.1, pp.142-150, **(2011)**

[36] Y. Long, L. Qin, R. Xiang, M. Li, and C. Huang, "A cognition-based attention model for sentiment analysis," in proceedings of the Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, pp.462-471, **(2017)**

[37] Y. Wang and F. Tian, "Recurrent residual learning for sequence classification," in proceedings of the Conference on Empirical Methods in Natural Language Processing, Austin, Texas, pp.938-943, **(2016)**

[38] Y. Wen, W. Zhang, R. Luo, and J. Wang, "Learning text representation using recurrent convolutional neural network with highway layers," arXiv preprint arXiv:(1606)06905, Pisa, Italy, **(2016)**

[39] S. Wang and D. Christopher, "Baselines and bigrams: Simple, good sentiment and topic classification," in proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Stanford, vol.2, pp.90-94, **(2012)**

[40] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using LSTM for region embeddings," arXiv preprint arXiv:(1602), 02373, **(2016)**

[41] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in proceedings of the International Conference on Machine Learning, Beijing, China, vol.32, pp.1188-1196, **(2014)**

[42] A. Dieng, C. Wang, J. Gao, and J. Paisley, "Topicrnn: A recurrent neural network with long-range semantic dependency," arXiv preprint arXiv:1611, 01702, **(2016)**

[43] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in proceedings of the North American Chapter on Association for Computational Linguistics: Human Language Technologies, Redmond, pp.1480-1489, **(2016)**

[44] D. Yogatama, C. Dyer, W. Ling, and P. Blunsom, "Generative and discriminative text classification with recurrent neural networks," arXiv preprint arXiv:(1703)01898, **(2017)**

[45] C. Qiao, B. Huang, G. Niu, D. Li, D. Dong, W. He, and H. Wu, "A new method of region embedding for text classification," National Engineering Laboratory of Deep Learning Technology and Application, China, **(2018)**

[46] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in proceedings of the 15th Conference on European Chapter of the Association for Computational Linguistics, France, vol.1, pp.1107-1116, **(2016)**

[47] Z. Huang, Z. Ye, S. Li, and R. Pan, "Length adaptive recurrent model for text classification," in proceedings of the ACM on Information and Knowledge Management, Singapore, pp.1019-1027, **(2017)**