Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)
Vol. 5, No. 2, December 2019, pp. 85~100
ISSN: 2338-3070, DOI: 10.26555/jiteki.v5i2.15021

Text Classification Using Long Short-Term Memory with GloVe Features

Winda Kurnia Sari[1], Dian Palupi Rini[2], Reza Firsandaya Malik[3]
[1] Master of Computer Science, Universitas Sriwijaya, Palembang 30128, Indonesia
[2] Informatics Departement, Universitas Sriwijaya, Palembang 30128, Indonesia
[3] Communication Network and Information Security Research Lab, Palembang 30128, Indonesia

ARTICLE INFO

Article history:

Received 19 December 2019,
Revised 30 January 2020,
Accepted 04 February 2020.

Keywords:

Recurrent Neural Network
Long Short-Term Memory
Multilabel Classification
Text Classification
GloVe

ABSTRACT

In the classification of traditional algorithms, problems of high features dimension and data sparseness often occur when classifying text. Classifying text with traditional machine learning algorithms has high efficiency and stability characteristics. However, it has certain limitations concerning large-scale dataset training. In this case, a multi-label text classification technique is needed to be able to group four labels from the news article dataset. Deep Learning is a proposed method for solving problems in text classification techniques. This experiment was conducted using one of the methods of Deep Learning Recurrent Neural Network with the application of the architecture of Long Short-Term Memory (LSTM). In this study, the model is based on trial and error experiments using LSTM and 300-dimensional word embedding features with Global Vector (GloVe). By tuning the parameters and comparing the eight proposed LSTM models with a large-scale dataset, to show that LSTM with features GloVe can achieve good performance in text classification. The results show that text classification using LSTM with GloVe obtain the highest accuracy is in the sixth model with 95.17, the average precision, recall, and F1-score are 95. Besides, LSTM with the GloVe feature gets graphic results that are close to good-fit on average.

Corresponding Author:

Dian Palupi Rini
Universitas Sriwijaya, Jl. Srijaya Negara Bukit Besar, 30139, Palembang, South Sumatera, Indonesia
Email: dprini@unsri.ac.id

1. INTRODUCTION

Text classification is an important part of Natural Language Processing with many applications [1], such as sentiment analysis [2][3], information search [4], ranking [5], and document classification [6]. The text classification model is generally divided into two categories: machine learning and deep learning. Much research on text classification has involved traditional machine learning algorithms such as k-Nearest Neighbors [7][8], Naive Bayes [9][10], Support Vector Machine [11][12], Logistic Regression [13]. Also, compared to traditional machine learning classification algorithms have high efficiency and stability characteristics. However, it has certain limitations in the case of large-scale dataset training [14].

Recently, neural network-based models are becoming increasingly popular [15][16][17]. These models achieve excellent performance in practice, tend to be relatively slow both during training and testing, limiting their use to very large datasets [14]. Several recent studies have shown that the success of deep learning about text classification is highly dependent on the effectiveness of word embedding [17]. Specifically, Shen et al. 2018 quantitatively show that the task of text classification based on word embedding can have the same level of difficulty regardless of the model used, using the concept of intrinsic dimension [1].

Some applications of deep learning methods used for text classifications include convolutional neural networks [16][17], autoencoder [19][20], deep belief network [21]. Recurrent Neural Network (RNN) is one of the most popular architectures used in natural language processing (NLP) because the recurrent structure is

suitable for variable length text processing. One of the deep learning methods proposed in this study is RNN with the application of the Long Short-Term Memory (LSTM) architecture. RNN can use a distributed word representation by first changing the token consisting of each text into a vector, which forms a matrix. Whereas, LSTM was developed to solve exploding and vanishing gradient problems that can be faced when training traditional RNN [22]. In addition to expanding memory, the classification of texts using LSTM in this study because the structure of LSTM is a sequence in which an integrated whole or cannot be cut as well as the structure of text documents that if cut will change the meaning of the sentence. The use of word embedding will be an input feature on LSTM before classifying text.

2. RESEARCH METHOD

2.1 Methodology

In general, the steps in the research methodology used to assist in the preparation of this research proposal require a clear framework for the stages. The research framework used as in Figure 1, which consists of literature review by research in the past 1 year and 5 years, in preparing data the dataset used in this study is AGNews consist of 400,000 data samples, after preparing the dataset is pre-processing data by removing punctuation and tokenization, do the classification process with LSTM, and analyzing the results, and make conclusions. The classification process with LSTM consists of 3 sub-processes, namely the training process, validation, and testing.

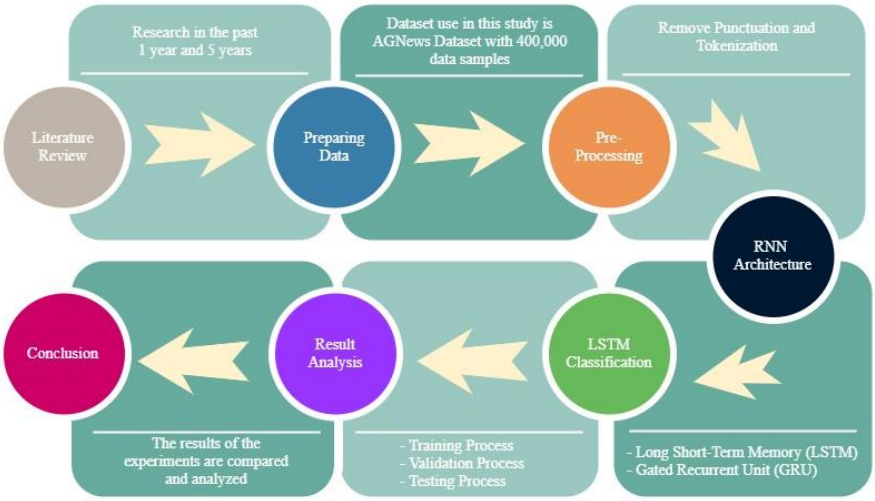

Fig. 1. Research Methodology

2.2 Feature Extraction

Feature extraction is an important part of machine learning, especially for text data. Text dataset is the most unstructured data which is necessary to produce meaning and structure used by machine learning algorithms. Recently, T. Mikolov introduced a better technique for extracting features from text using the concept of embedding or placing words into vector spaces based on context. This approach to word embedding, called Word2Vec, solves the problem of representing contextual word relationships in computable feature space [23]. J. Pennington in 2014 developed a vector representation of learning spaces from words called GloVe and placed them in Stanford's NLP lab [24]. In this study use 300 embedding dimensions of GloVe to be an input feature in LSTM.

2.3 Recurrent Neural Network

RNN is a type of neural network with a memory status for processing sequence inputs. Traditional RNN has a problem called gradient vanishing and exploding during training [25]. Recurrent node activation consists of feedback for itself from one time-step to the next. RNN is included in the deep learning category because data is processed automatically and without defining features [26]. RNN can use the internal states (memory) to process the input sequence. This makes it applicable to tasks such as Natural Language Processing (NLP) [15], speech recognition [25], music synthesis [27], and time-series financial data processing [28]. There are two implementations of RNN i.e Backpropagation Through Time (BPTT) algorithm for calculating gradients and Vanishing Gradient problems that have led to the development of LSTM and GRU, the two most popular and powerful models currently used in NLP.

The basic equation of RNN,

$$s_t = \tanh(U_{x_t} + W_{s_{t-1}}) \quad (1)$$

$$\hat{y}_t = softmax(V_{s_t}) \quad (2)$$

2.2 Long Short-Term Memory

Long short-term memory (LSTM) has recently become a popular tool among NLP researchers for their superior ability to model and learn from sequential data. These models have shown state-of-the-art results on various public benchmarks ranging from the classification of sentences [29] and various tagging problems [30] for language modeling [16][17], and sequence-to-sequence predictions [26]. LSTM aims to solve the RNN problem called gradient vanishing and exploding. LSTM replaces hidden vectors from recurrent neural networks with memory blocks equipped with gates. This can maintain long-term memory in principle by practicing appropriate gating weights and has proven to be very useful in achieving state-of-the-art for various problems, including speech recognition [31]. LSTM was proposed by Hochreiter and Schmidhuber, 1997 to specifically address this problem of learning long-term dependency. LSTM stores separate memory cells in it which can update and display their contents only if necessary [32]. The LSTM gates mechanism implements three layers; (1) inputs gate, (2) forget gate, and (3) output gate [33].

Each LSTM unit, can be seen in Figure 2 has a memory cell, and the states at time t are represented as ct. Reading and modifying are controlled by the sigmoid gate and affect the input gate i_t, forget gate f_t and output gate o_t. LSTM is calculated as follows: At the moment of the moment, the model receives input from two external sources (h_{t-1} and x_t). The hidden states ht is calculated by the xt input vector the network received at time t and the previous hidden states h_{t-1}. When calculating the hidden layer node states, input gate, output gate, forget gate and x_t will simultaneously affect the state of the node.

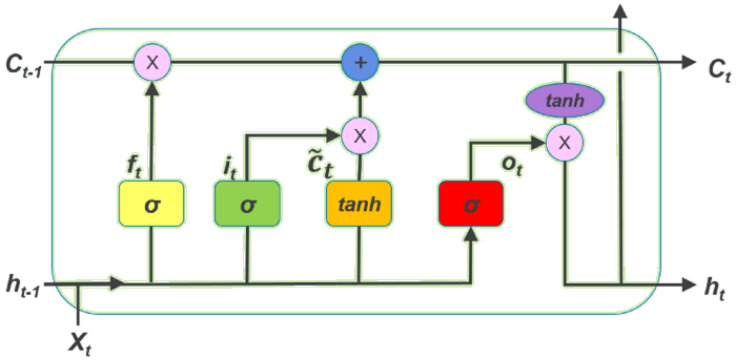

Fig. 2. LSTM Architecture

A step-by-step explanation of the LSTM cell and its gates is provided below:

1) Input Gate:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (3)$$

$$\check{C}_t = \tanh(W_C.[h_{t-1}, x_t] + b_C \quad (4)$$

2) Forget Gate:

$$f_t = \sigma\big(W_f.[h_{t-1}, x_t + b_f\big) \quad (5)$$

3) Memory State:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (6)$$

4) Output Gate:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = o_t * \tanh(C_t) \quad (8)$$

2.3 Evaluation

The multi-label evaluation steps of the confusion matrix in the following equations:

$$Acc = \frac{\sum_{i=1}^{l} \frac{TP_i + TN_i}{TP_i + FN_i + TN_i + FP_i}}{l} * 100\% \quad (9)$$

$$Presisi = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l} (FP_i + TP_i)} * 100\% \quad (10)$$

$$Recall = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l}(TP_i + FN_i)} * 100\% \quad (11)$$

$$F1\ score = \frac{2 * Presisi * Recall}{Presisi + Recall} \quad (12)$$

2.4 Optimization

There are some types of optimizers for deep learning models such as SGD, Adam, RMSProp, etc. This paper applied Adam and RMSProp for training the data. Adam Optimizer can control sparse gradient issues [34]. It is an expansion to stochastic gradient descent that has currently seen wider adoption for deep learning applications such as Natural Language Processing.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (13)$$

$$v_t = \beta_2 v_{t-1} + (1 + \beta_2) g_t^2 \quad (14)$$

where m and v refer averages the first two moments of the gradient, g indicates gradient on current mini-batch. RMSProp can adapt the learning rate for each of the parameters. It aims to divide the learning rate for weight by a running average of the magnitudes of recent gradients for that weight [35].

$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2 \quad (15)$$

Where γ is the forgetting factor.

And the parameters are updates as,

$$w := w - \frac{n}{\sqrt{v(w,t)}} \nabla Q_i(w) \quad (16)$$

2.5 Pre-Processing

2.5.1 One-hot encoding

The first pre-processing in this research is One-hot encoding. One-hot encoding is changing text data (categorical) into numbers. Machine learning algorithms cannot work with categorical data directly. Categorical data must be converted to numbers. This applies because research works with the type of sequence classification and uses deep learning methods such as Long Short-Term Memory Recurrent Neural Networks.

2.5.2 Tokenization and Remove Punctuation

Tokenization is the process of breaking up the flow of text into words, phrases, symbols, or other meaningful elements called tokens. Tokenizing means splitting up text into units that have minimal meaning. This is a mandatory step before all types of processing. This process will divide the text into sentences and sentences into typographic tokens. That means separating punctuation. The feature generated from tokenizing is training data. In this process, padding is also carried out to identify the end of the sentence because the decoder is trained sentence by sentence.

3 RESULTS AND DISCUSSION

3.1 Dataset

Previous research on Zhang 2015, Wang 2018 has shown work well with large-scale datasets [16][36]. From eight large-scale datasets, the AGNEWS dataset was taken for training. AGNews is a classification of topics in four categories of Internet news articles consisting of titles and descriptions classified into four classes: World, Entertainment, Sports, and Business. The dataset is shown in Table 1, with the following content specifications:

Table 1. Dataset Specification

Dataset	Class	Contains
AGNews	4	496,835

3.2 Training Process

AGNews dataset is divided into 80% each for training and 20% for testing. The training dataset used is not used for LSTM testing, and vice versa. From 80% of the training data, 10% is used for the data validation process. The amount of each dataset is randomly split, with an automatic data split.

3.3 Training Models

The hyper-parameters used are the Relu and Tanh activation functions, Adam and RMSProp optimizers will be validated with a learning rate of 0.001 and 0.0001 to minimize errors. The dimensions of word embedding are 300. The structure and hyper-parameters used in LSTM validation with the Glove features can be shown in Table 2.

Table 2. Training Models LSTM with GloVe

Model	Epoch	Neuron	Learning Rate	Optimizer	Loss Function	Activation Function		Dimension
						Hidden	Output	
1	50	128	0.001	Adam	Categorical Cross Entropy	Relu	Softmax	300
2	50	128	0.001	Adam	Categorical Cross Entropy	Tanh	Softmax	300
3	50	128	0.001	RMSProp	Categorical Cross Entropy	Relu	Softmax	300
4	50	128	0.001	RMSProp	Categorical Cross Entropy	Tanh	Softmax	300
5	50	128	0.0001	Adam	Categorical Cross Entropy	Relu	Softmax	300
6	50	128	0.0001	Adam	Categorical Cross Entropy	Tanh	Softmax	300
7	50	128	0.0001	RMSProp	Categorical Cross Entropy	Relu	Softmax	300
8	50	128	0.0001	RMSProp	Categorical Cross Entropy	Tanh	Softmax	300

3.4 LSTM Models

The LSTM sequence classification training process using the word embedding feature Global Vector (GloVe) 300 dimension is trained with hyper-parameter embedding matrix obtained from pre-processing the GloVe feature on input, activation of Relu and Tanh on hidden gate, softmax activation on output gate, optimizer Adam and RMSprop, with dropout 0.5 and epoch 50, have been trained in each of 8 models with tuning Learning rates of 0.001 and 0.0001. The hyperparameter learning rate controls the rate or speed at which the learning model. Specifically, this controls the number of divided errors whose model weights are updated with each time they are updated, such as at the end of each batch of training examples. The learning rate is perhaps the most important hyperparameter.

3.4.1 Model 1

Table 3 shows the results of the evaluation performance of the LSTM training process that was trained using Relu activation, Adam optimizer with a learning rate of 0.001. The accuracy of the training obtained in model 1 is 95.33. Confusion matrix will be used to calculate the Precision, Recall, and F1-score, the results of which can be seen in Table 3 as a result of the evaluation performance of the test. The results in Table 4 show that the training and testing accuracy values are not much different, which is 95 with an average value of Precision, Recall, and F1-score of 95. To see the comparison of training and testing per epoch in the accuracy curve can be seen in Figure 3 and the curve loss in Figure 4.

Table 3. Performance Evaluation Results of the LSTM Training Process with Relu Activation Parameters, Adam Optimizer with a learning rate of 0.001

Accuracy		**95,33**			
Confusion Matrix	**Label**	**0**	**1**	**2**	**3**
	0	29731	286	200	120
	1	343	21462	1186	237
	2	341	859	20463	105
	3	176	253	113	8609

Table 4. Performance Results Evaluation of the LSTM Testing Process with Relu Activation Parameters, Adam Optimizer with a learning rate of 0.001

Label	**Precision**	**Recall**	**F1-Score**	**Data**
0	97	98	98	30337
1	94	92	93	23228
2	93	94	94	21768
3	95	94	94	9151
Avg	**95**	**95**	**95**	**84484**

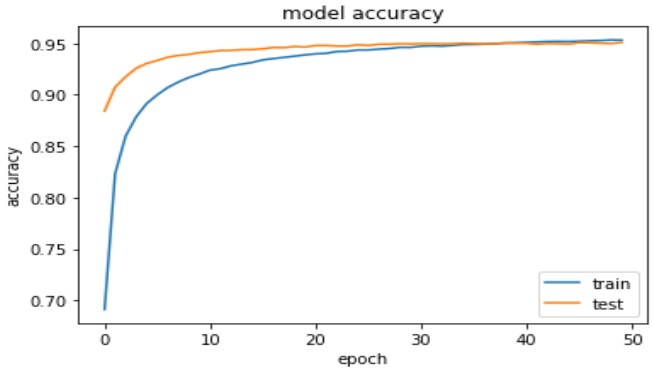

Fig. 3. Comparison Curve of Training and Testing Accuracy of 50 epochs

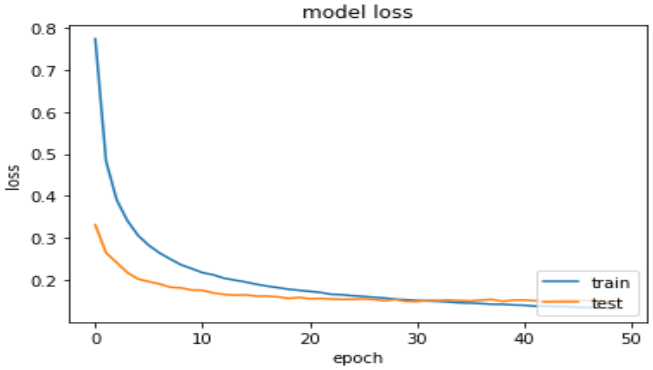

Fig. 4. Comparison of Training Loss and Testing Curves of 50 epochs

3.4.2 Model 2

Table 5 shows the results of the performance evaluation of the LSTM training process that was trained using Tanh activation, Adam optimizer with a learning rate of 0.001. The accuracy of the training obtained in model 2 is 95.34. Confusion matrix will be used to calculate the Precision, Recall, and F1-score, the results of which can be seen in Table 6 as a result of the evaluation performance of the test. Based on the two models above using the same optimizer and learning rate with both Relu and Tanh activation, the resulting value is also not much different. The value of training and testing accuracy, average precision, recall, and f1-score of 95. Figure 5 shows the comparison curve of training and testing accuracy, and Figure 6 shows the Loss curve.

Table 5. Performance Evaluation Results of the LSTM Training Process with Tanh Activation Parameters, Adam Optimizer with a learning rate of 0.001

Accuracy		**95,34**			
Confusion Matrix	**Label**	**0**	**1**	**2**	**3**
	0	29798	267	214	126
	1	356	21139	1413	247
	2	359	755	20425	141
	3	177	245	88	8734

Table 6. Performance Results Evaluation of LSTM Testing Process with Tanh Activation Parameters, Adam Optimizer with a learning rate of 0.001

Label	**Precision**	**Recall**	**F1-Score**	**Data**
0	97	98	98	30405
1	94	91	93	23155
2	92	94	93	21680
3	94	94	94	9244
Avg	**95**	**95**	**95**	**84484**

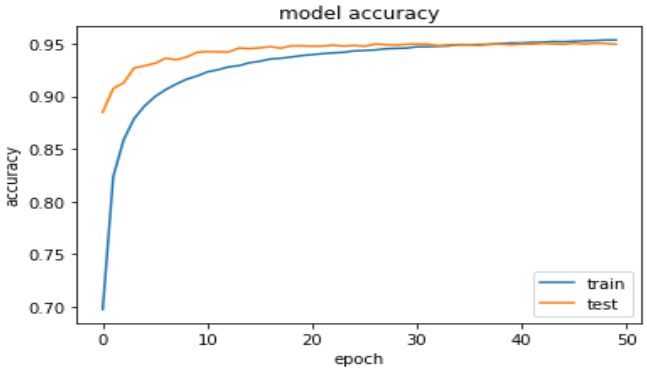

Fig. 5. Comparison of Training Accuracy and Testing Curves of 50 epochs

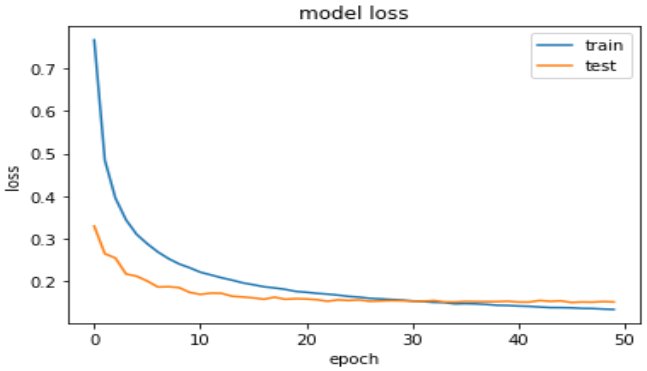

Fig. 6. Comparison of Training Loss and Testing Curves of 50 epochs

3.4.3 Model 3

In model 3 is trained with Relu activation hyperparameter, RMSprop optimizer and learning rate 0.001. The results of the training evaluation performance can be shown in Table 7, while the results of the testing evaluation are shown in Table 8. The accuracy obtained in the training process is 94.25 with an average value of precision, recall, and f1-score of 94. Not much different from the value of testing accuracy which is equal to 94.37. The comparison training curve and testing of accuracy and loss can be seen in Figure 7 and Figure 8.

Table 7. Performance Results of LSTM Training Process Evaluation with Relu Activation Parameters, RMSprop Optimizer with a learning rate of 0.001

Accuracy		**94,25**			
Confusion Matrix	**Label**	**0**	**1**	**2**	**3**
	0	29705	316	187	153
	1	431	21329	1179	314
	2	426	996	20026	176
	3	221	259	98	8668

Table 8. Performance Evaluation Results of the LSTM Testing Process with Relu Activation Parameters, RMSprop Optimizer with a learning rate of 0.001

Label	**Precision**	**Recall**	**F1-Score**	**Data**
0	96	98	97	30361
1	93	92	92	23253
2	93	93	93	21624
3	93	94	93	9246
Avg	**94**	**94**	**94**	**84484**

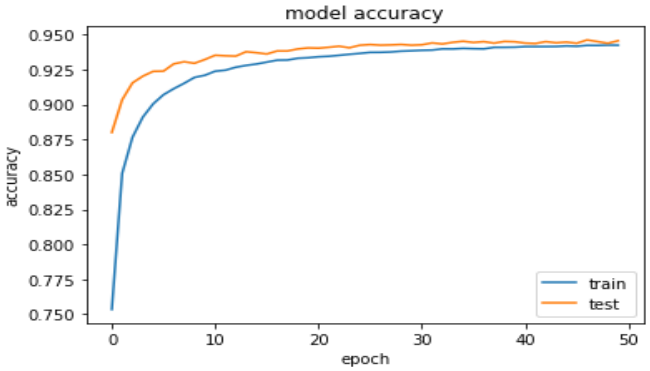

Fig. 7 Comparison of Training Accuracy and Testing Curves of 50 epochs

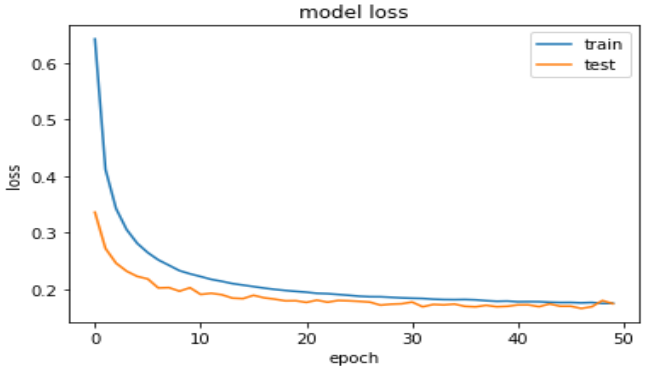

Fig. 8 Comparison of Training Loss and Testing Curves of 50 epochs

3.4.4 Model 4

In model 4 is trained with Tanh activation hyperparameter, RMSprop optimizer and learning rate 0.001. The results of the training evaluation performance can be shown in Table 9, and the results of the testing evaluation are shown in Table 10. The accuracy obtained in the training process is 94.32 with an average value of precision, recall, and f1-score of 94. The testing accuracy is 94.56. The test results in Table 10 show that the macro average of precision is 95, while the recall and f1-score are 94. The accuracy value in this process is 95. The comparison training curve and accuracy testing can be seen in Figure 9 and the loss in Figure 10. Both Adam and RMSprop optimizers trained with a learning rate of 0.001 showed results that are not much different.

Table 9. LSTM Training Process Performance Evaluation Results with Tanh Activation Parameters, RMSprop Optimizer with a learning rate of 0.001

Accuracy		**93,21**			
Confusion Matrix	**Label**	**0**	**1**	**2**	**3**
	0	30083	226	179	104
	1	573	21340	1107	227
	2	481	926	20197	101
	3	298	250	122	8270

Table 10. Performance Results Evaluation of LSTM Testing Process with Tanh Activation Parameters, RMSprop Optimizer with a learning rate of 0.001

Label	**Precision**	**Recall**	**F1-Score**	**Data**
0	96	98	97	30592
1	94	92	93	23247
2	93	93	93	21705
3	95	93	94	8940
Avg	**95**	**94**	**94**	**84484**

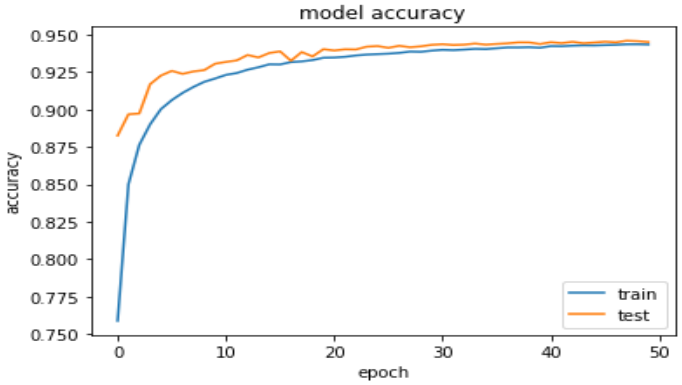

Fig. 9. Comparison of Training Accuracy and Testing Curves of 50 epochs

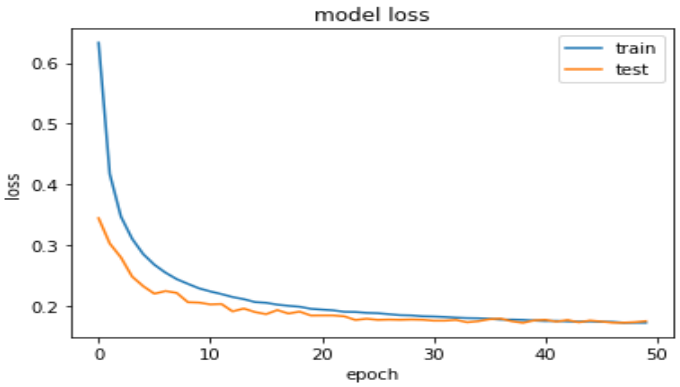

Fig. 10. Comparison of Training Loss and Testing Curves of 50 epochs

3.4.5 Model 5

The LSTM model 5 was trained with the same hyperparameter with a tuning learning rate of 0.0001. Table 11 shows the results of the training evaluation performance and Table 12 shows the performance results of the classification testing evaluation with the activation of Relu, Adam optimizer, and 300-dimensional GloVe word embedding. The accuracy value in the training and testing process for learning rates 0.001 and 0.0001 with the same optimizer, namely Adam gets results that are not much different, both precision, recall, and f1-score of 95. However, the accuracy and loss curves obtained in learning a rate of 0,0001 is better than an accuracy and loss curve with a learning rate of 0.001. It can be seen in Figure 11 and Figure 12.

Table 11. Performance Evaluation Results of the LSTM Training Process with Relu Activation Parameters, Adam Optimizer with a learning rate of 0.0001

Accuracy		**94,58**			
Confusion Matrix	**Label**	**0**	**1**	**2**	**3**
	0	29884	232	235	146
	1	390	21192	1297	247
	2	320	739	20491	94
	3	153	228	120	8716

Table 12. Performance Evaluation Results of the LSTM Testing Process with Relu Activation Parameters, Adam Optimizer with a learning rate of 0.001

Label	**Precision**	**Recall**	**F1-score**	**Data**
0	97	98	98	30497
1	95	92	93	23126
2	93	95	94	21644
3	95	95	95	9217
Avg	**95**	**95**	**95**	**84484**

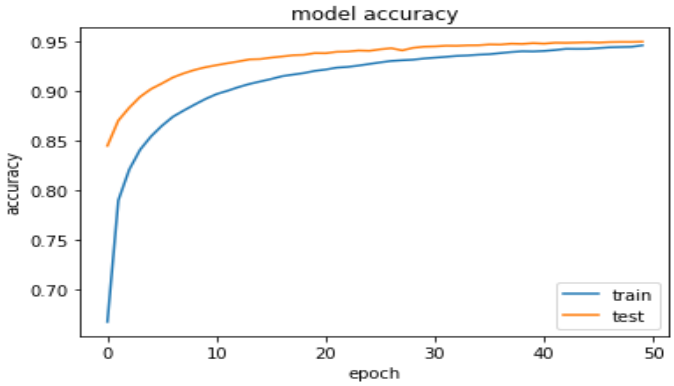

Fig. 11. Comparison of Training Accuracy and Testing Curves of 50 epochs

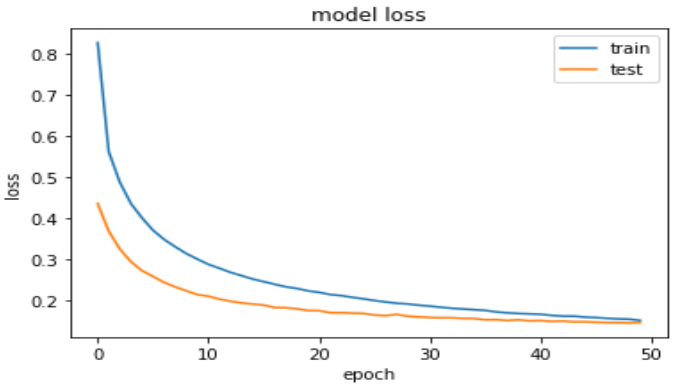

Fig. 12. Training Loss Comparison Curve and Testing 50 epoch

3.4.6 Model 6

In Model 6, training was carried out with the same hyperparameter with Tanh activation, Adam optimizer, and a learning rate of 0.0001. The results of training evacuation performance and confusion matrix are shown in Table 13 with training accuracy of 95. While the results of testing evaluation performance are in Table 14 with an average value of precision, recall, and f1-score of 95. Figure 13 shows a comparison curve of training and testing accuracy for 50 epochs. Although the loss in the validation process continues to decrease, at the 40th epoch the same and slightly greater than the training loss can be seen in Figure 14.

Table 13. Performance Evaluation Results of the LSTM Training Process with Tanh Activation Parameters, Adam Optimizer with a learning rate of 0.0001

Accuracy		**95**			
Confusion Matrix	**Label**	**0**	**1**	**2**	**3**
	0	29940	221	216	113
	1	340	21314	1177	245
	2	309	841	20510	113
	3	172	229	103	8641

Table 14. Performance Results Evaluation of LSTM Testing Process with Tanh Activation Parameters, Adam Optimizer with a learning rate of 0.0001

Label	**Precision**	**Recall**	**F1-score**	**Data**
0	97	98	98	30490
1	94	92	93	23076
2	93	94	94	21773
3	95	94	95	9145
Avg	**95**	**95**	**95**	**84484**

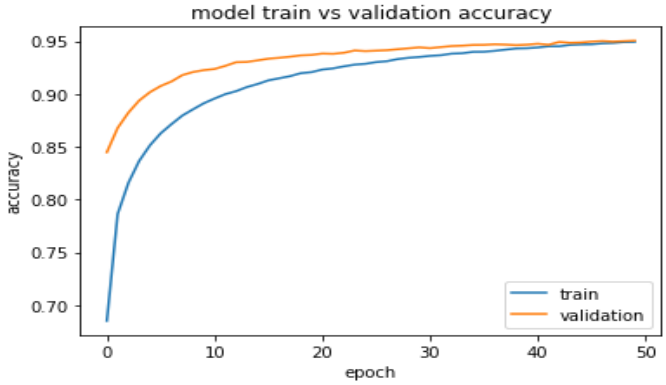

Fig. 13. Comparison Curve of Training and Testing Accuracy of 50 epochs

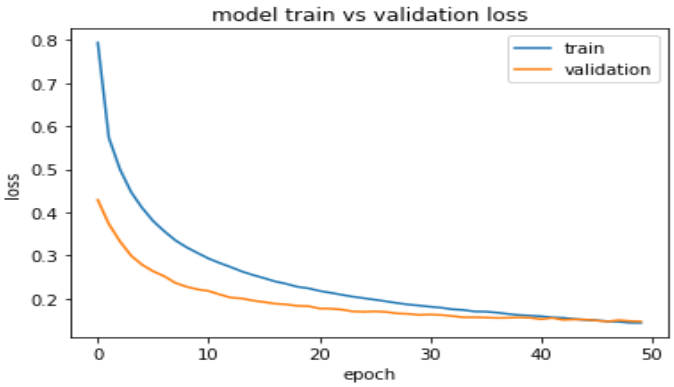

Fig. 14. Curve Comparison of Training and Testing Loss of 50 epochs

3.4.7 Model 7

In model 7, it was trained with Relu activation parameters, RMSprop optimizer, and tuning learning rate 0,0001. Table 15 shows the results of the training evaluation performance and confusion matrix of 50 epochs obtained an accuracy of 93.24. The results of the evaluation performance of precision testing, recall, and f1-score are in Table 16. The accuracy curve resulting from training and testing can be seen in Figure 15 and the loss curve in Figure 16, which shows that the results of the RMSprop optimizer parameter with a tuning learning rate of 0,0001 are more fit than the RMSprop with a learning rate of 0.001 although there is a slight up and down in accuracy and loss.

Table 15. Performance Evaluation Results of the LSTM Training Process with Relu Activation Parameters, RMSprop Optimizer with a learning rate of 0.0001

Accuracy		**93,24**			
Confusion Matrix	**Label**	**0**	**1**	**2**	**3**
	0	29625	305	272	147
	1	458	21251	1293	315
	2	440	954	20192	135
	3	214	264	157	8462

Table 16. Performance Evaluation Results of the LSTM Testing Process with Relu Activation Parameters, RMSprop Optimizer with a learning rate of 0,0001

Label	**Precision**	**Recall**	**F1-score**	**Data**
0	96	98	97	30349
1	93	91	92	23317
2	92	93	93	21721
3	93	93	93	9097
Avg	**94**	**94**	**94**	**84484**

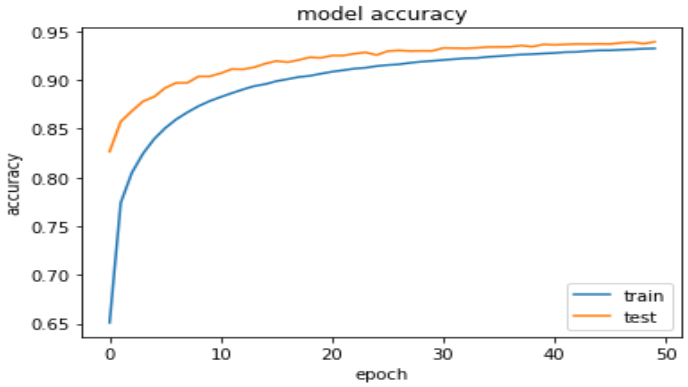

Fig. 15. Comparison of Training Accuracy and Testing Curves of 50 epochs

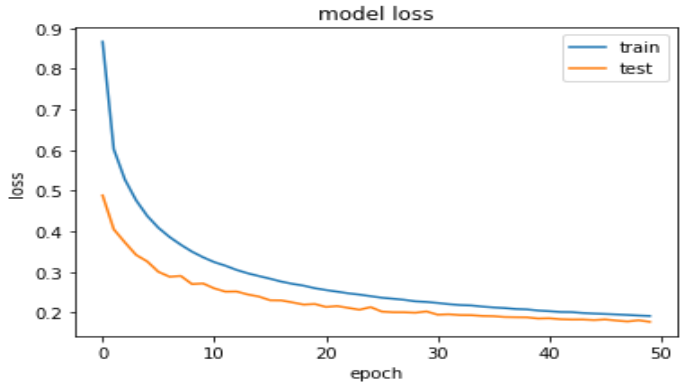

Fig. 16. Curve Comparison of Training and Testing Loss of 50 epochs

3.4.8 Model 8

Model 8 was trained with Tanh activation parameters, RMSprop optimizer, and a learning rate of 0.0001 resulting in training accuracy of 93.21. The results of the training evaluation performance can be seen in Table 17 where there are four class confusion matrix multilabel. The results of the evaluation performance of the test are in Table 18 with an average value of precision, recall, and f1-score of 94. The comparison training curve and testing accuracy model can be seen in Figure 17 with the value of testing accuracy exceeding training accuracy. While the loss model curve decreases with the passage of 50 epochs, where the test loss is smaller

than the training loss can be seen in Figure 18. Table 19 shows a comparison of the testing accuracy of the eight LSTM models using the word embedding GloVe.

Table 17. Performance Evaluation Results of the LSTM Training Process with Tanh Activation Parameters, RMSprop Optimizer with a learning rate of 0.0001

Accuracy		**93,21**			
Confusion Matrix	**Label**	**0**	**1**	**2**	**3**
	0	29738	264	272	154
	1	470	20921	1509	309
	2	406	882	20128	176
	3	251	256	147	8601

Table 18. Performance Results Evaluation of LSTM Testing Process with Tanh Activation Parameters, RMSprop Optimizer with learning rate 0.0001

Label	**Precision**	**Recall**	**F1-score**	**Data**
0	96	98	97	30349
1	94	90	92	23317
2	91	93	92	21721
3	93	93	93	9097
Avg	**94**	**94**	**94**	**84484**

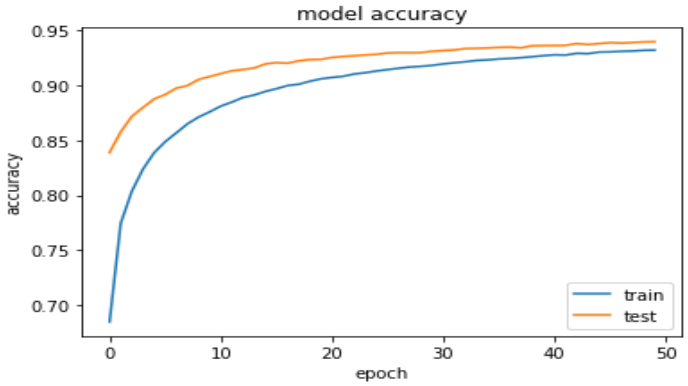

Fig. 17. Comparison Curve of Training and Testing Accuracy of 50 epochs

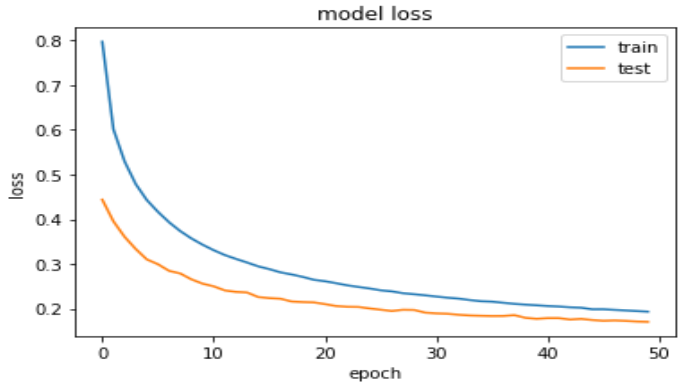

Fig.18. Comparison of Training and Testing Loss Curves in 50 epochs

In the eight of tuning models LSTM using the word embedding Glove feature, the highest test accuracy was 95.17 in model 6 with Tanh activation parameters, Adam optimizer, and a learning rate of 0.0001. While the accuracy and loss model which close to good-fit on models with a learning rate of 0.0001 either with Adam or RMSprop optimizer. Table 20 shows the comparison results of previous works.

Table 19. Accuracy of testing of the eight LSTM models Using

the Word Embedding GloVe Feature

Model	Epoch	Neuron	Lr	Optimizer	Hidden	Accuracy
1	50	128	0.001	Adam	Relu	95.01
2	50	128	0.001	Adam	Tanh	94.81
3	50	128	0.001	RMSProp	Relu	94.37
4	50	128	0.001	RMSProp	Tanh	94.56
5	50	128	0.0001	Adam	Relu	95.03
6	50	128	0.0001	Adam	Tanh	95.17
7	50	128	0.0001	RMSProp	Relu	94.17
8	50	128	0.0001	RMSProp	Tanh	93.97

Table 20. The Comparison of Previous Works

Model	AGNews
Bag-of-words (Zhang et al.,2015)	88.8
Small word CNN (Zhang et al.,2015)	89.13
Large word CNN (Zhang et al.,2015)	91.45
LSTM (Zhang et al.,2015)	86.06
Deep CNN (29 layer) (Conneau et al.,2017)	91.27
SWEM (Shen et al.,2018)	92.24
fastText (Joulin et al.,2016)	92.5
LEAM (Wang et al., 2018)	92.45
LEAM (linear) (Wang et al., 2018)	91.75
GloVe + LSTM	**95.17**

4 CONCLUSION

Text classification using LSTM is done by conducting trial and error experiments. Text classification using LSTM with the Glove feature does hyper-parameter tuning to get the best model. Whereas, the LSTM and hyperparameter structure used from the test results are using embedding of the GloVe features in the input, softmax activation function in the output, Relu and Tanh activation functions, loss categorical cross-entropy function, learning rate 0.001 and 0.0001, with the number epoch 50. The highest accuracy with the Glove feature is on the sixth model of 95.17 with an average precision, recall, and F1-score of 95. It can be concluded that the LSTM evaluation results using the GloVe feature can achieve good performance both in accuracy and the curves.

REFERENCES

[1] L. Li, L. Xiao, W. Jin, H. Zhu, and G. Yang, "Text Classification Based on Word2vec and Convolutional Neural Network," *Neural Information Processing, International Conference on Neural Information Processing (ICONIP), Lecture Notes in Computer Science*, vol. 11305, 2018. DOI: 10.1299/jsmemag.90.823_758

[2] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," *Proceedings of the 2013 conference on Empirical Methods in Natural Language Processing*, pp.1631-1642, 2013, Online

[3] H. Yuan, Y. Wang, X. Feng and S. Sun, "Sentiment Analysis Based on Weighted Word2vec and Att-LSTM," *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, pp. 420-424, 2018. DOI: 10.1145/3297156.3297228

[4] J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and Word2vec for text classification with semantic features," *Proceedings of 2015 IEEE 14th International Conference on Cognitive Informatics and Cognitive Computing, ICCI*CC 2015*, pp. 136-140, 2015. DOI: 10.1109/ICCI-CC.2015.7259377

[5] K. Chen, Z. Zhang, J. Long, and H. Zhang, "Turning from TF-IDF to TF-IGM for term weighting in text classification," *Expert Systems with Applications*, vol. 66, pp. 245-260, 2016. DOI: 10.1016/j.eswa.2016.09.009

[6] R. G. Rossi, A. D. A. Lopes, and S. O. Rezende, "Optimization and label propagation in bipartite heterogeneous networks to improve transductive classification of texts," *Information Processing and Management*, vol. 52, no. 2, pp. 217-257, 2016. DOI: 10.1016/j.ipm.2015.07.004

[7] B. Y. Pratama, and R. Sarno. Personality classification based on Twitter text using Naive Bayes, KNN and SVM. *Proceedings of 2015 International Conference on Data and Software Engineering, ICODSE*, 2016, DOI: 10.1109/ICODSE.2015.7436992

[8] M. Azam, T. Ahmed, F. Sabah, F. and M.I. Hussain, "Feature Extraction based Text Classification using K-Nearest Neighbor Algorithm". *IJCSNS Int. J. Comput. Sci. Netw. Secur*, 18, pp. 95-101, 2018. Online

[9] S. Xu, "Bayesian Naïve Bayes classifiers to text classification," *Journal of Information Science*, vol. 44, no. 1, pp.48-59. 2018. DOI: 10.1177/0165551516677946

[10] L. Jiang, C. Li, S. Wang, and L. Zhang, "Deep feature weighting for naive Bayes and its application to text classification," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 26-39, 2016. DOI: 10.1016/j.engappai.2016.02.002

[11] M. Fanjin, H. Ling, T. Jing, and W. Xinzheng, "The Research of Semantic Kernel in SVM for Chinese Text Classification," *In Proceedings of the 2nd International Conference on Intelligent Information Processing*, pp. 8, 2017. DOI: 10.1145/3144789.3144801

[12] M. Goudjil, M. Koudil, M. Bedda, and N. Ghoggali, "A novel active learning method using SVM for text classification," *International Journal of Automation and Computing*, vol. 15, no.3, pp. 290-298, 2018. DOI: 10.1007/s11633-015-0912-z

[13] A. Onan, S. Korukoğlu, and H. Bulut, "Ensemble of keyword extraction methods and classifiers in text classification," *Expert Systems with Applications*, vol. 57, pp. 232-247, 2016. DOI: 10.1016/j.eswa.2016.03.045

[14] M. Gao, T. Li, and P. Huang, "Text Classification Research Based on Improved Word2vec and CNN," *In International Conference on Service-Oriented Computing*, pp. 126-135. Springer, Cham, 2018. DOI: 10.1007/978-3-030-17642-6_11

[15] K. Kowsari, D.E. Brown, M. Heidarysafa, K.J. Meimandi, M.S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," *In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 364-371, 2017. DOI: 10.1109/ICMLA.2017.0-134

[16] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint*, arXiv:1408.5882, 2014. DOI: 10.3115/v1/D14-1181

[17] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *In Advances in neural information processing systems*, pp. 649-657, 2015. DOI: arXiv:1509.01626v3

[18] D. Shen, G. Wang, W. Wang, M.R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin, "Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms," *arXiv preprint*, 2018. DOI: arXiv:1805.09843

[19] Xu, W., Sun, H., Deng, C., and Tan, Y. Variational autoencoder for semi-supervised text classification. In Thirty-First AAAI Conference on Artificial Intelligence. 2017. Online

[20] R. G. F. Soares, "Effort Estimation via Text Classification and Autoencoders," *Proceedings of the International Joint Conference on Neural Networks*, pp. 1-8, 2018. DOI: 10.1109/IJCNN.2018.8489030

[21] P. Ruangkanokmas, T. Achalakul, and K. Akkarajitsakul, "*Deep Belief Networks with Feature Selection for Sentiment Classification*," *Proceedings - International Conference on Intelligent Systems, Modelling and Simulation, ISMS*, 9-14, 2017. DOI: 10.1109/ISMS.2016.9.

[22] Y. Yan, Y. Wang, WC. Gao, BW. Zhang, C. Yang, and XC. Yin, "LSTM^2: Multi-Label Ranking for Document Classification," *Neural Processing Letters*, vol. 47, no. 1, pp. 117-138, 2018. DOI: 10.1007/s11063-017-9636-0

[23] T. Mikolov, K. Chen, K., G. Corrado, and J. Dean. "Distributed Representations of Words and Phrases and their Compositionality," *Advances in Neural information processing systems*, pp. 3111-3119, 2013. Online

[24] J. Pennington, R. Socher and C. Manning, "Glove: Global vectors for word representation," *In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532-1543, 2014. Online

[25] H. Zen, and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis". *In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4470-4474, 2015. DOI: 10.1109/ICASSP.2015.7178816

[26] I. Sutskever, O. Vinyals, and Q.V. Le, "Sequence to sequence learning with neural networks," *In Advances in neural information processing systems*, pp. 3104-3112, 2014. Online

[27] Li, K., Daniels, J., Liu, C., Herrero-Vinas, P. and Georgiou, P., "Convolutional recurrent neural networks for glucose prediction," *IEEE Journal of Biomedical and Health Informatics*. Vol. 24, no. 2, Febuary 2020 DOI: 10.1109/JBHI.2019.2908488

[28] K. Tseng, C. Ou, A. Huang, R.F. Lin, and X. Guo, "Genetic and Evolutionary Computing," *Proceedings of the Twelfth International Conference on Genetic and Evolutionary Computing*, vol. 834, 2019. DOI: 10.1007/978-981-13-5841-8.

[29] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A C-LSTM neural network for text classification," *arXiv preprint*, 2015. DOI: arXiv:1511.08630

[30] M. Pota, F. Marulli, M. Esposito, G. De Pietro, and H. Fujita, "Multilingual POS tagging by a composite deep architecture based on character-level features and on-the-fly enriched Word Embeddings," *Knowledge-Based Systems*, vol. 164, pp. 309-323, 2019. DOI: 10.1016/j.knosys.2018.11.003

[31] C.C. Chiu, T.N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R.J. Weiss, K. Rao, E. Gonina, and N. Jaitly, "State-of-the-art speech recognition with sequence-to-sequence models," *In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774-4778, 2018. DOI: 10.1109/ICASSP.2018.8462105

[32] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint*, 2013. DOI: arXiv:1308.0850
[33] A. Kumar, and R. Rastogi, "Attentional Recurrent Neural Networks for Sentence Classification," *In Innovations in Infrastructure*, pp. 549-559. *Springer*, Singapore, 2019. DOI: 10.1007/978-981-13-1966-2_49
[34] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint*. 2014. DOI: arXiv:1412.6980
[35] T. Tieleman, T. and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26-31, 2012. Online
[36] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin, "Joint embedding of words and labels for text classification," *arXiv preprint*, 2018. DOI: arXiv:1805.04174